CS - 6700 Reinforcement Learning

Assignment -1

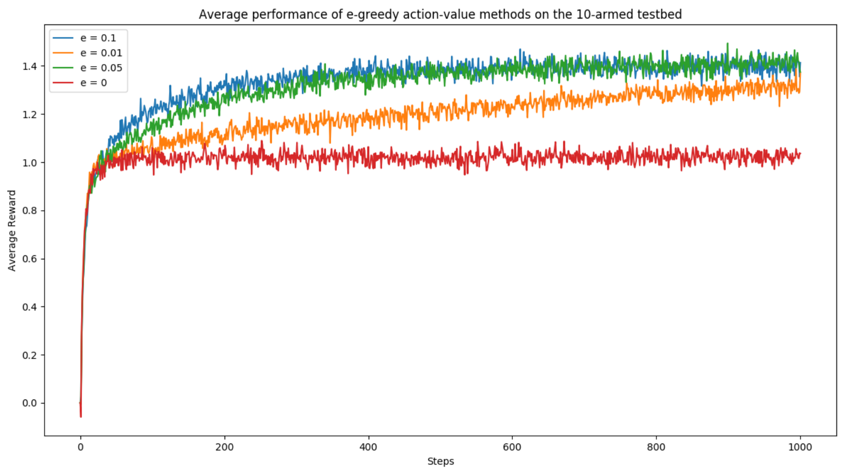
Mohammed Khandwawala EE16B117

Contents

1 10 Arm Testbench

2000 10-Arm bandits are evaluated. Reward ($q^*(a)$)for each arm is picked from normal distribution with mean 0 and standard deviation 1. When any learning method is applied actual reward for action A at time t is selected from a normal distribution with mean $q^*(A_t)$ and standard deviation 1. Each bandit is iterated for 1000 time steps. Evaluation of bandit algorithms is done on the same testbench.

2 Problem1 : ϵ - greedy



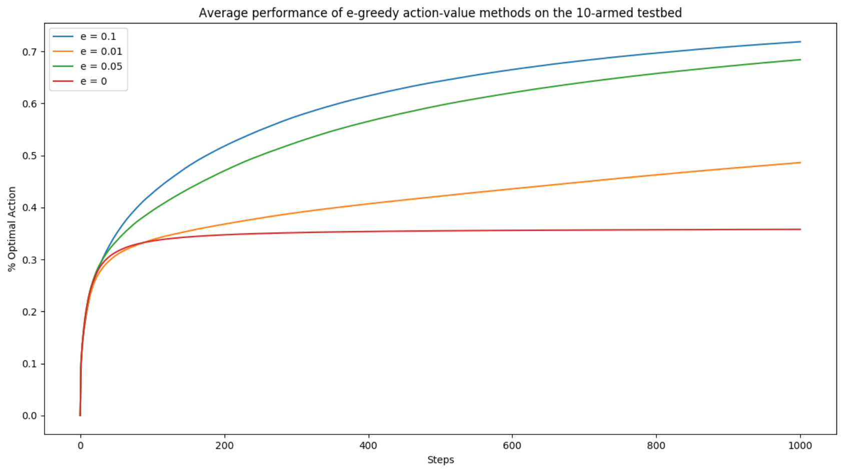
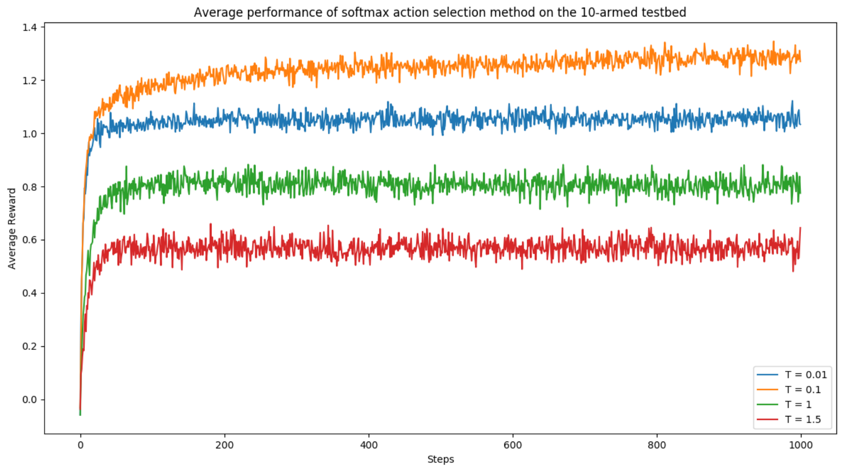


Figure 1: Average Performance of ϵ-greedy algorithm over 2000 10-Armed Bandit

- The above plots are the comparison of the ϵ-greedy for different values of ϵ (0 , 0.1 , 0.05 ,0.01) on the 10-Armed Bandits. Above plots are averages over 2000 different bandit problems.Estimated reward is update for the arm every time it is pulled. The first graph shows the average reward vs iterations. As the iterations increased , reward increased. The greedy method ($\epsilon = 0$) improved slightly faster than the other ϵ values , but then convergence value was lower than the optimal. The greedy method(ϵ) will perform even worse in the long run because it will often get stuck pulling suboptimal actions. For $\epsilon > 0$ (0.1, 0.05, 0.05) performed better than greedy. After 1000 iteration $\epsilon = 0.1$ has the highest reward arm, almost same as $epsilon = 0.05$. For $\epsilon = 0.1$ reward grows faster than both the other values. For $\epsilon > 0$ give better reward as they spent more iteration in exploring than $epsilon = 0$.
- The second graph shows that the greedy method ($\epsilon = 0$) found the optimal action in only approximately one-third of the tasks. In the other two-thirds, its initial samples of the optimal arm were suboptimal, so it never returned to it. The ϵ-greedy methods performed better because they continued to explore and to improve their chances of recognizing the optimal arm. The $\epsilon = 0.1$ method explored more, and found the optimal action earlier, hence performed better in initial stages. The $\epsilon = 0.05$ method improved more slowly, but eventually it will perform better than the $\epsilon = 0.1$ method on both performance measures.

3 Problem 2 : Action Selection Using Softmax



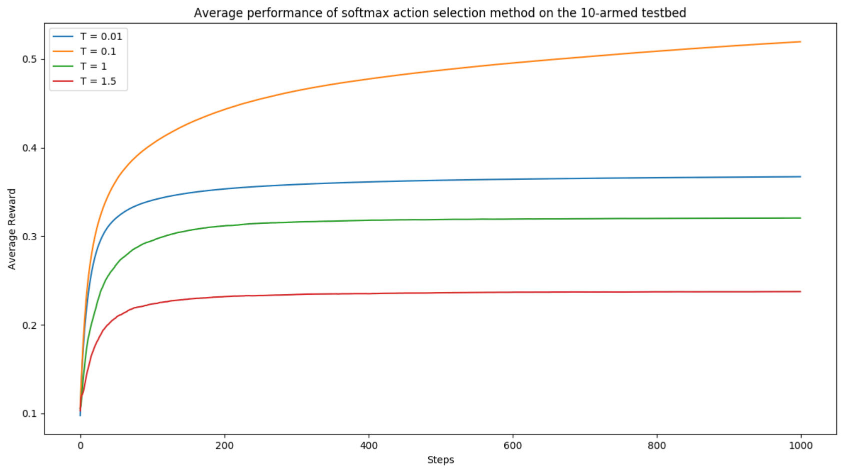


Figure 2: Average Performance of selecting action using Softmax over 2000 10-Armed Bandit

- In this section we change the exploration policy , arms are picked by probability given by softmax function as shown

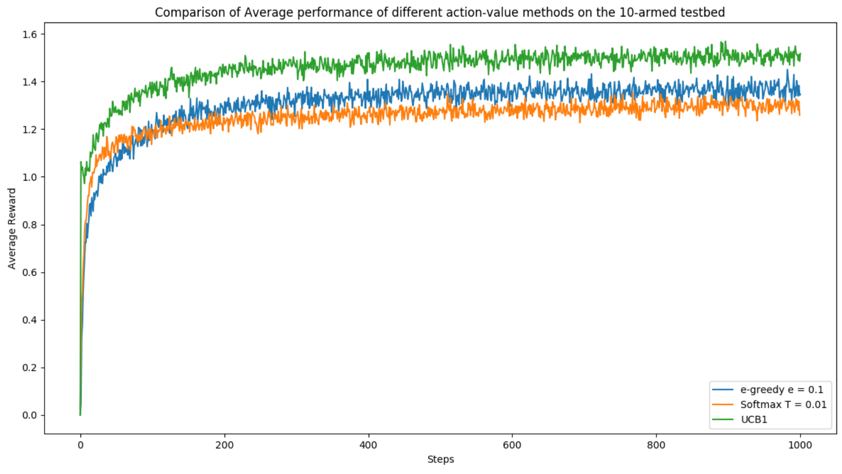
$$p(a_i) = \frac{e^{\frac{Q_t(a_i)}{T}}}{\sum e^{\frac{Q_t(a_j)}{T}}} \qquad j = 1, 2, ..10$$

  $Q_t(a_i)$ denotes empirical reward for pulling arm i at time t.Parameter T in the above equation is called the temperature. This parameter controls the probability distribution , for a very high value of temperature it will select the arm with best empirical reward (Greedy) and for very low values it will choose arm uniformly at random.

- At T $\rightarrow$ 0 it is purely exploitation and as T increases it becomes more explorative as can be seen from the plot above increasing the T from 0.01 , 0.1 , 1 and 5. We get to see the best average reward for T $=$ 0.1. T $=$ 0.1 has the optimal exploration-exploitation trade off.

- From the second plot it clear the algorithm for T $=$ 0.1 picks the optimal arm more frequently than any other value which is close to 55% at the end of 2000 iterations and increasing. Where as for other values of T the optimal arm is selected at most 35% of the times.

4 Problem 3 : UCB1 Algorithm on 10-Armed Bandit



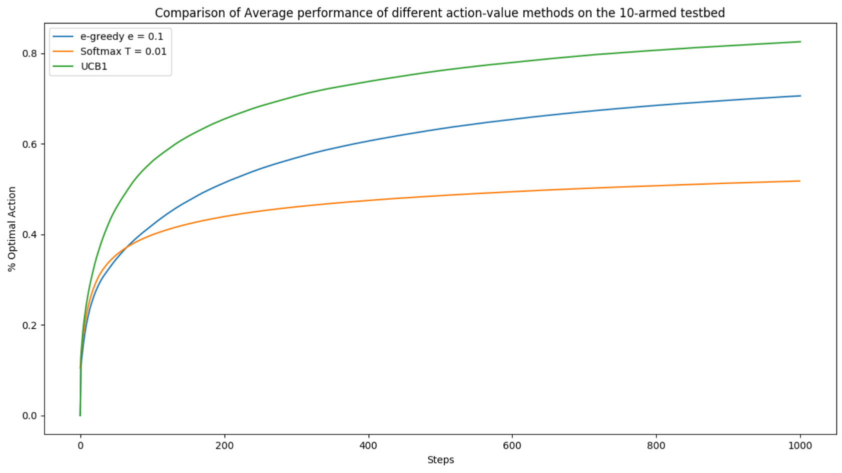


Figure 3: Average Performance of selecting action using Softmax over 2000 10-Armed Bandit

- In this section we will compare Upper Confidence Bound 1 (UCB1) algorithms with ϵ-greedy and softmax on the 10-Armed Bandit testbed.
- In UCB1 we select the arm at time t as

$$A_t = \underset{a}{\operatorname{argmax}}(Q_t(a) + \sqrt{\frac{ln(t)}{N_t(a)}})$$

  where N t (a) denotes the number of times that action a has been selected prior to time t, Q_t(a) is the mean estimate of arm a. $\sqrt{\frac{ln(t)}{N_t(a)}}$, expression corrects for the uncertainty in our current estimate of the average reward. $N_t(a)$ is in the denominator of the term , as the number of times we take the action the uncertainty around the average reward of that particular action reduces. UCB1 guarantees all arms are selected once and the estimate of an arm does not increase with $N_t(a)$ is eventually not be selected.
- From the plots comparing UCB1 with ϵ-greedy and softmax with ϵ value of 0.1 and temperature value of 0.1 respectively. From the plot it is clear that UCB1 performs better than both softmax and ϵ-greedy. Unlike in ϵ greedy and softmax there is no parameter to tune in case of UCB1 , so the best values of temperature and ϵ parameter are selected from previous experiments.

5 Problem 4 : Median Elimination Algorithm

5.1 Problem 4a : Median Elimination Algorithm and comparison with other Bandit Algorithms

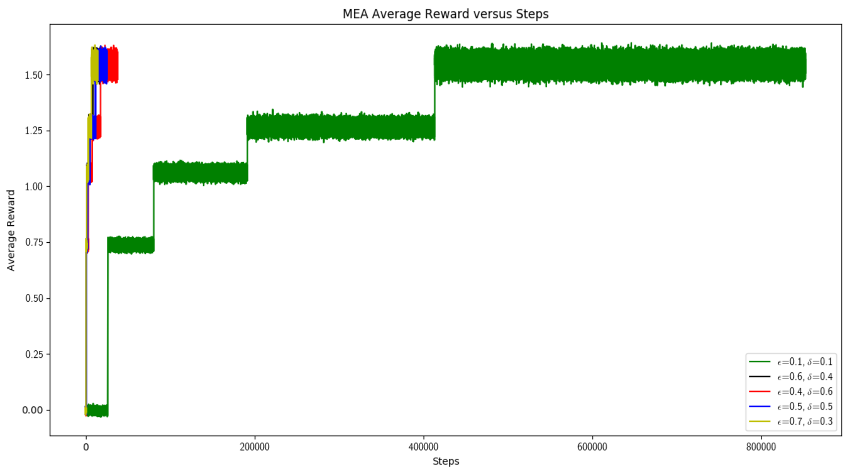


Figure 4: Average Performance of selecting action using Median Elimination over 2000 10-Armed Bandit , zoomed out

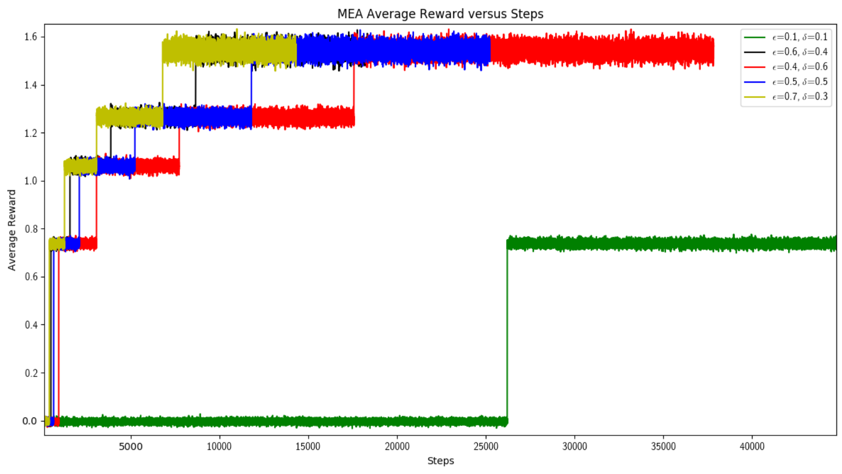


Figure 5: Average Performance of selecting action using Median Elimination over 2000 10-Armed Bandit , zoomed in

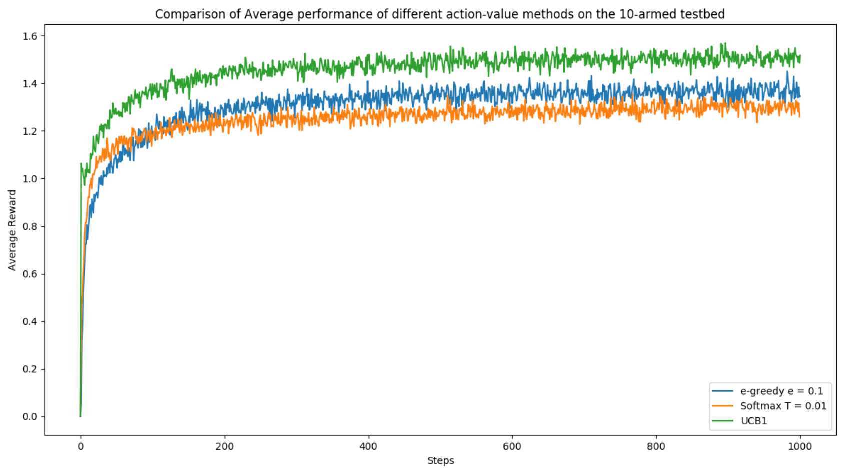
- Median Elimination is PAC algorithm. It eliminates 50% arms in every iteration. If there are k arms in k step it will return optimal arm with (1 - δ) probability and the reward of the arm is in ϵ range of optimal value. Every arm is sampled $\frac{1}{(\frac{\epsilon}{2})^2} \times log(\frac{3}{\delta})$. To get an empirical estimate for the arm reward. The arms whose empirical estimate is less than the median are eliminated.

ϵ and δ	steps	reward	% optimal Arm
$\epsilon = 0.1, \delta = 0.1$	852521	1.54744	99.85%
$\epsilon = 0.4, \delta = 0.6$	37843	1.54744	99.35%
$\epsilon = 0.5, \delta = 0.5$	25222	1.54744	99.15%
$\epsilon = 0.6, \delta = 0.4$	18368	1.54744	98.85%
$\epsilon = 0.7, \delta = 0.3$	14303	1.54744	99.00%
$\epsilon = 0.9, \delta = 0.9$	6780	1.54744	98.15%

- The above table shows the performance of median elimination algorithm for different values of epsilon delta. It can be noted the performance of the algorithm does not change significantly (for this bandit problem where rewards are sampled from Gaussian with variance 1). With the value ϵ and δ 0.9 number of steps have dropped down by two order of magnitude.

  The table below shows if effect of ϵ and *delta* when the variance of the distribution from which reward is picked when the arm is pulled is set to 100. Unlike for the case of unit variance we can see the the effect of loosening the PAC bounds on the reward.

ϵ and δ	steps	reward	% optimal Arm
$\epsilon = 0.1, \delta = 0.1$	852521	1.4788	89.34%
$\epsilon = 0.9, \delta = 0.9$	6780	0.5285	64.28%
$\epsilon = 0.4, \delta = 0.6$	37838	0.9427	72.57%
$\epsilon = 0.5, \delta = 0.5$	25222	0.8813	64.52%
$\epsilon = 0.6, \delta = 0.4$	18368	0.6955	64.372%
$\epsilon = 0.7, \delta = 0.3$	14303	0.6995	64.55%



- From the above plot it is clear that UCB1 is better than ϵ-greedy and softmax. Arguments from previous section (Comparing UCB1 with other algorithm). While UCB1 is action selection algorithm , MEA is action elimination which guarantees (ϵ,δ)-PAC.
- The number of steps required in MEA are very high, even with lower bonds ($\epsilon = 0.9$, $\delta = 0.9$) it took 6780 steps , which was achieved in 1000 steps with same reward by UCB1. This was the observation for 10 arms.
- For the case k = 1000 or 10000. A combination of MEA and UCB1 can be used. MEA can be reduced the number of arms and then MEA can be used on reduced number of arms.

5.2 Problem 4b : Computational Cost of Median and Rate Determining Step

- In the case when k (number of arms) are 10 , finding median is not the rate determining step also each step the number of arms get halved. For k = 1000 findinag median is

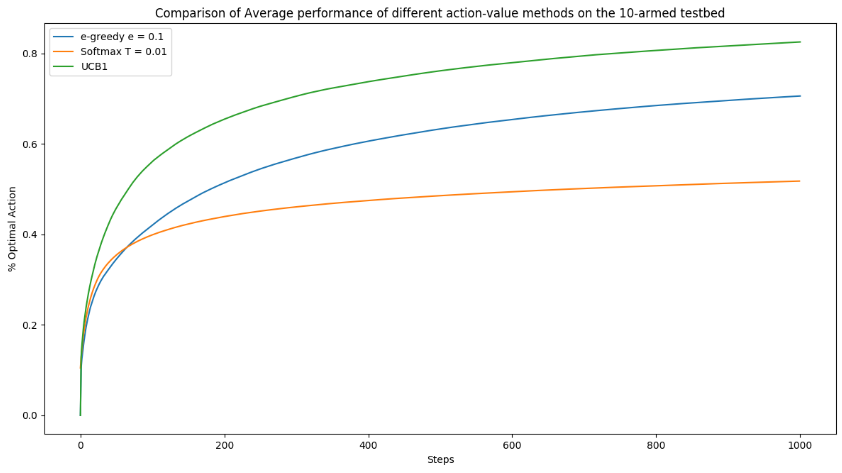


Figure 6: Comparison of Average Performance of various bandit algorithms over 2000 10-Armed Bandit

the rate determining step the computational cost of computing median is O(klog(k)) in naive implementation (sort and pick the middle). For faster performance **numpy** implementation of median can be used which uses the following algorithm or some variation of it. **Binmedian** is an algorithms to compute the median. Binmedian has O(n) average running time . This algorithm is highly competitive with the use of standard algorithm i.e. quickselect when computing the median of a single data set.

6 1000 Arm Bandit

All four algorithm discussed before are now run on the same 2000 k-Arm Bandit test with value of k = 1000. unlike previous case in this case , algorithms will not converge in 1000 iteration , so UCB1 , ϵ-greedy and softmax are allowed to run for 10000 iterations.

- Below is the plot comparing UCB1 , ϵ - greedy and Softmax. Same parameter values for these algorithms are used as used while comparing for 10 arms. The Regret of UCB is very high compared to the other two. But eventually after running for 10000 iteration the average reward is higher in UCB (UCB - 3.21 , ϵ-greedy - 2.63 , Softmax - 1.77).

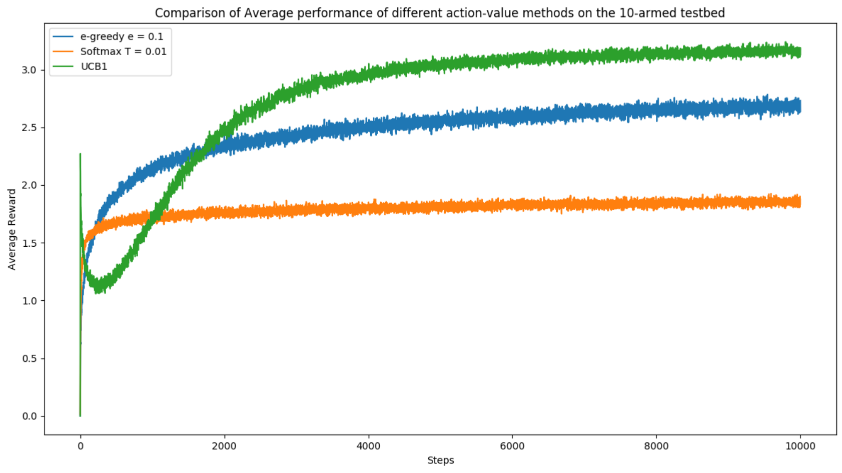


Figure 7: Average reward of various bandit algorithm over 2000 - 1000 Arm Bandit Problems

- From the table it can seen that Median Elimination also reaches similar reward but the number of steps required are very high. Even for loose bound on epsilon-delta (0.9 each) reaches the reward close to UCB.

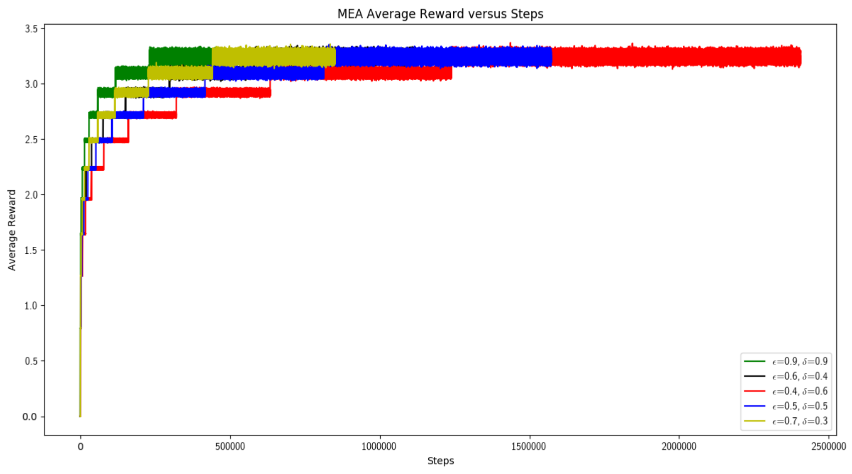


Figure 8: Median Elimination Algorithm average reward of 2000 - 1000 Armed Badits for different values of ϵ and *delta*

ϵ and δ	steps	reward	% optimal Arm
$\epsilon = 0.9, \delta = 0.9$	451993	3.255	99.3%
$\epsilon = 0.4, \delta = 0.6$	2404940	3.288	99.69%
$\epsilon = 0.5, \delta = 0.5$	1572740	3.250	99.64%
$\epsilon = 0.6, \delta = 0.4$	1120720	3.230	99.60%
$\epsilon = 0.7, \delta = 0.3$	850419	3.255	99.41%

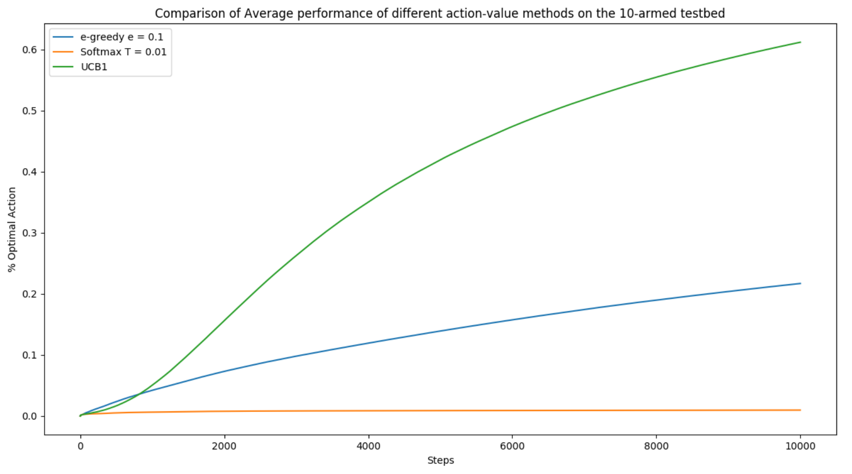


Figure 9: % Optimal Action taken of various bandit algorithm over 2000 1000-Armed Bandit

- As far percentage of optimal arm pulled for UCB1 is 62% at 10000 iteration and increasing where as for softmax it is 3 % and flat.
- Hence, we can observe that selection of parameter (ϵ t) is crucial for ϵ-greedy softmax algorithm to work well, when number of arm increases/decreases. While UCB1 will adjust it's upper confidence bound for each arm based on $N_t(a)$, $Q_t(a)$ t and will do enough

exploration, when number of arms increases in initial steps, and later will lock on to pulling optimal arm more number of times.