Reinforcement Learning PA1

Ajay S Joshi (CS15B047)

February 21, 2018

1 ϵ - Greedy

1.1 Comparison between different ϵ

- $\epsilon = 0.1$ gives the best results through both average rewards and % optimal actions for 1000 steps.
- $\epsilon = 0.01$ learns slower than $\epsilon = 0.1$, and greedy gets locked to suboptimal actions, due to lack of exploration.

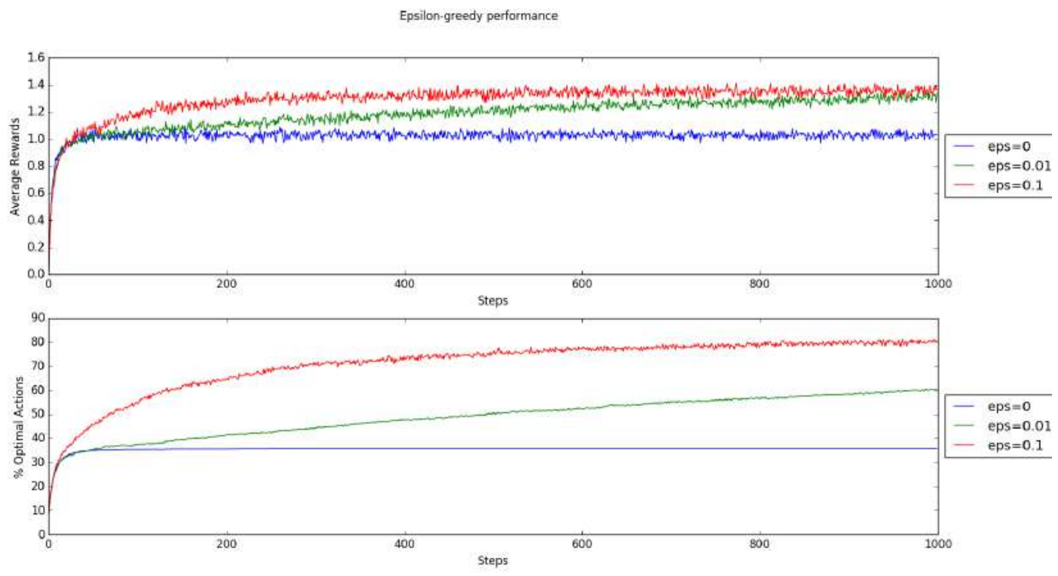

Figure 1: Performance of ϵ-greedy method for different constant ϵ. Performance measured in terms of average rewards and % optimal action at each time step.

- Performance on ϵ = { 1/4, 1/8, 1/16, 1/32, 1/64 } shows that best performance is found on intermediate ϵ, and decreases on either extremes.

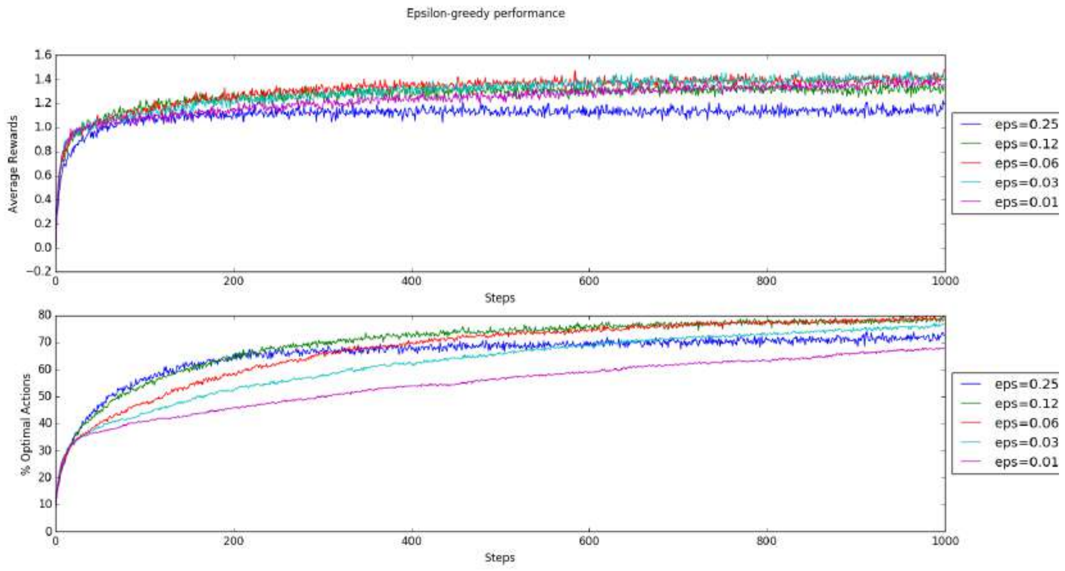

Figure 2: Variation of performance with varying ϵ

1.2 Comparison of constant and decaying ϵ

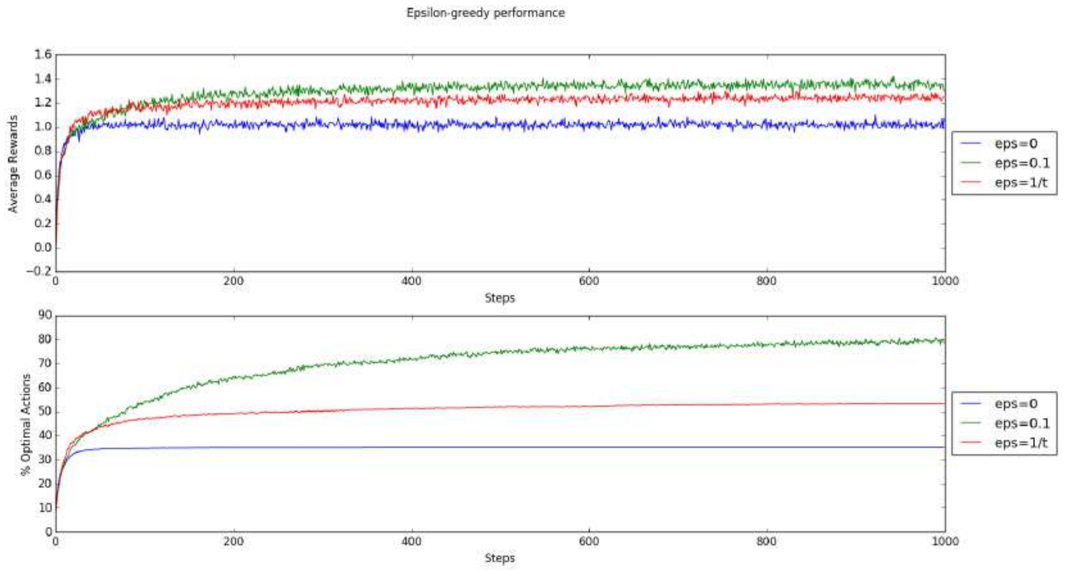

Figure 3: Comparison of ϵ-greedy method for constant ϵ v/s a decaying schedule of 1/t for ϵ

- Though asymptotically, decaying schedule of $\epsilon = \frac{1}{t}$ performs better (logarithmic regret) than constant ϵ (linear regret), $\epsilon = 0.1$ performs better than a decaying schedule $\epsilon = \frac{1}{t}$ for limited steps (1000).

2 Softmax

Actions are sampled from Gibbs Distribution :

$$\pi_t(a) = \frac{\exp(Q_t(a)/\tau)}{\sum_{i=1}^{i=k} \exp(Q_t(i)/\tau)} \tag{1}$$

2.1 Variation of performance with variation in τ

- Here, performance is measured on 5 constant temperature values (0.1,0.3,0.5,0.7,0.9).
- Highest performance is achieved on intermediate $\tau = 0.3$, and performance decreases on both lower and higher τ, due to lower and higher exploration than required.
- Performance of softmax with constant τ is similar to ϵ-greedy with constant ϵ.

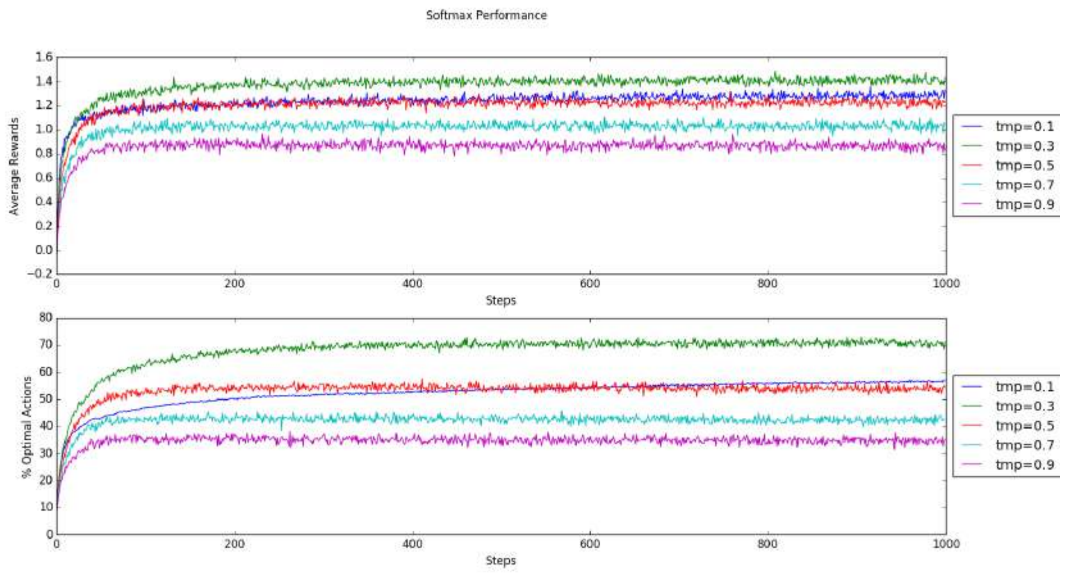

Figure 4: Variation of performance with variation in τ

2.2 Comparison of softmax with constant and cooled τ

- τ is cooled with time as $\frac{1}{\log(t)}$, and gives better performance than any constant τ.

- This happens as the exploration rate should decrease with the increase in exploration performed, which is captured in the cooling schedule.

($\tau = \frac{1}{t}$ gave overflows, so $\frac{1}{log(t)}$ is chosen).

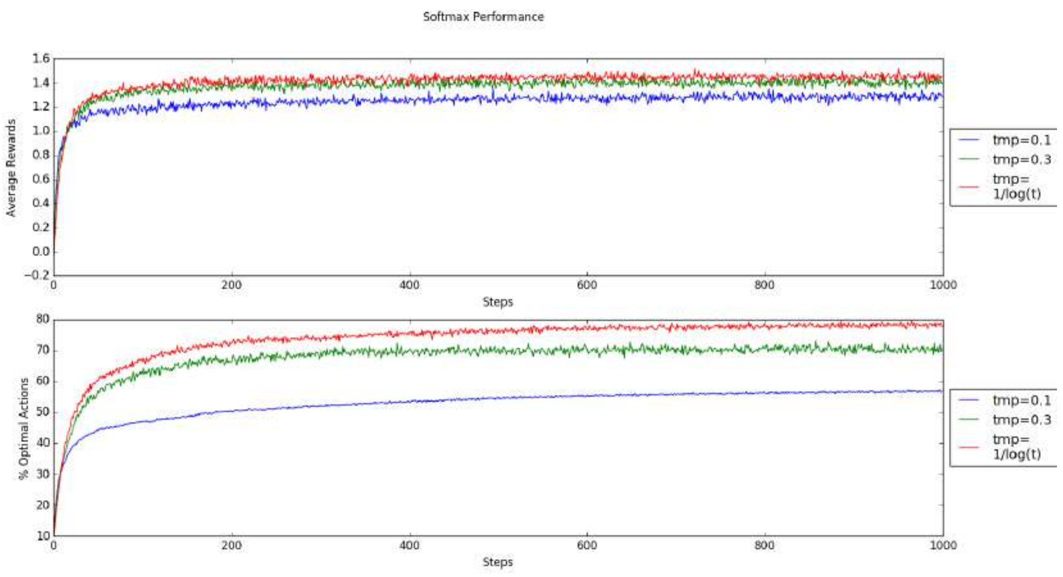

Figure 5: Performance comparison between constant and decaying τ

3 UCB1

3.1 Variation of UCB performance with variation in c

Performance for intermediate c is best, decreasing towards either sides. c = 1 performs best for 1000 steps.

3.2 Comparing UCB1 with ϵ-greedy and softmax

- UCB1 is compared with ϵ-greedy and softmax using parameters (ϵ, τ, c) which perform better than other parameters in 1000 step setting.
- UCB1 performs better than softmax and ϵ, in terms of average rewards and % optimal actions chosen.
- The initial k steps have a bad performance, as all arms are tried once in random order in this period.
- UCB1 performs better than ϵ-greedy and softmax, as exploration in UCB takes into account uncertainty in the arm value, unlike ϵ-greedy, which explores randomly or softmax, which does not take into account uncertainty of current arm value estimates.

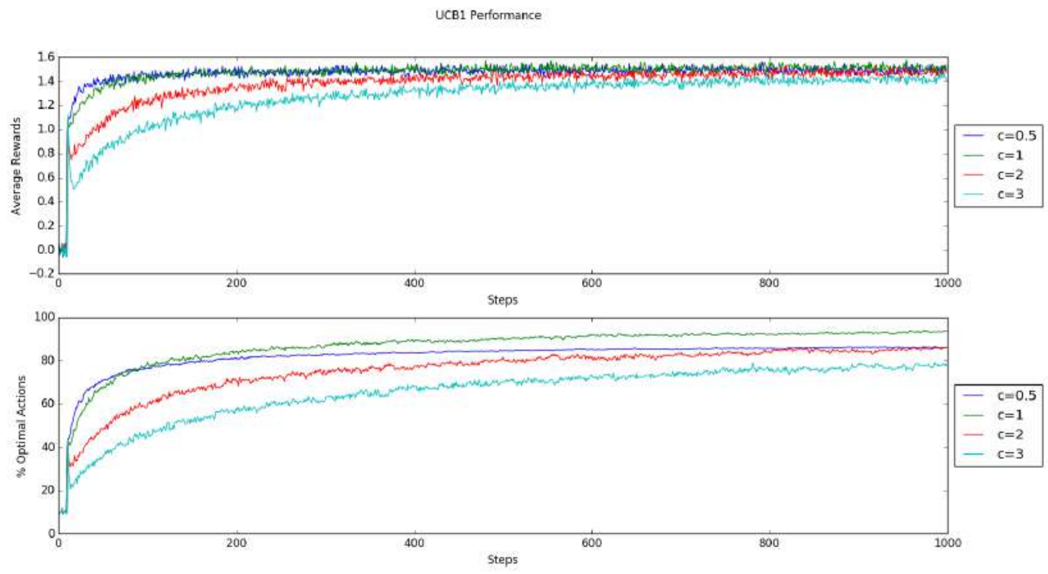

Figure 6: Performance variation for variation in c

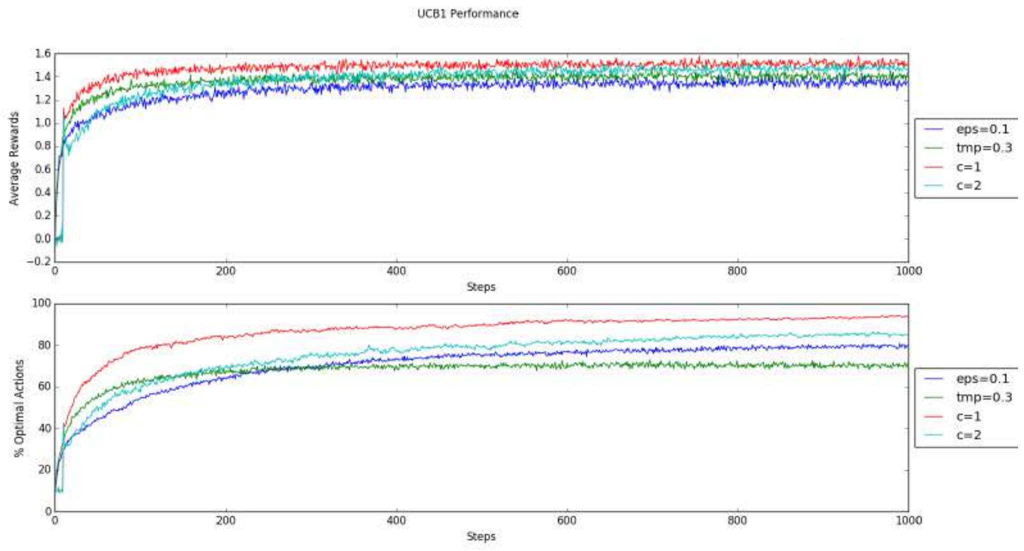

Figure 7: Comparison between UCB1, Softmax, and ϵ - greedy methods

3.3 Performance of UCB1 from quality of best final arm

UCB1 is assessed by % times the best arm in the end (considering confidence bounds) is not the true optimal arm. Performance shows that intermediate c value is best, as smaller values correspond to lesser exploration, and larger values perform worse, maybe due to their wider confidence bounds.

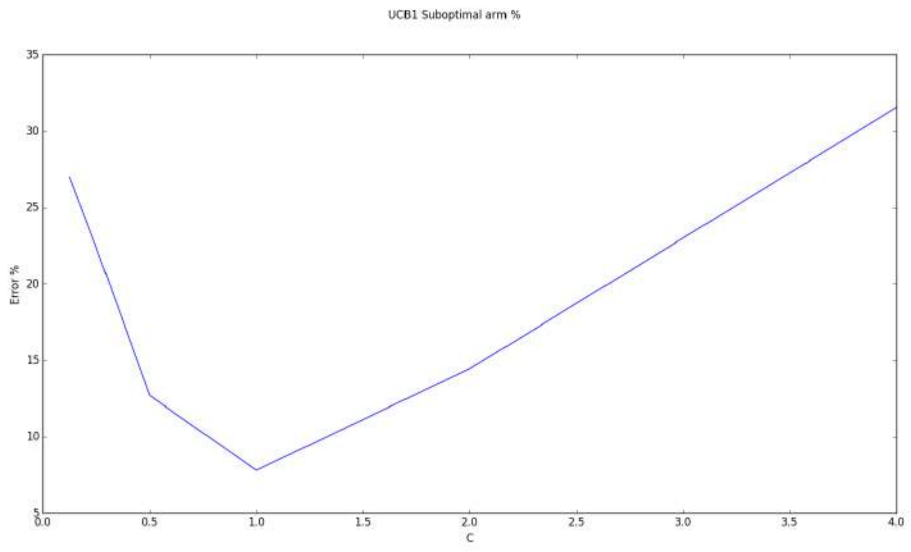

Figure 8: % times final best arm is not optimal, for different c

4 Median Elimination Algorithm(MEA)

4.1 Performance comparison with other algorithms

- 1. Average rewards
  2. % Optimal Action
  3. Optimality of final arm returned

  are the metrics used for comparing MEA with other algorithms.

- In median elimination, a particular set of arms is maintained at each stage and each arm is pulled a specific number of times, which increases as number of remaining arms decrease.

- The number of stages depends only on number of arms. Thus, the average reward for each stage is constant, corresponding roughly to average of the remaining arms, and % optimal actions is also constant corresponding to number of arms left.

4.2 Performance comparison using optimality of final arm returned

- Upper graph shows % times final arm returned is suboptimal for different values of (ϵ,δ) and lower graph denotes error for different algorithms.

- The difference in % error is explained by higher number of iterations in MEA vs 1000 iterations in other algorithms. When the other algorithms are run for similar iterations as MEA (for particular ϵ,δ), similar range performance is obtained.

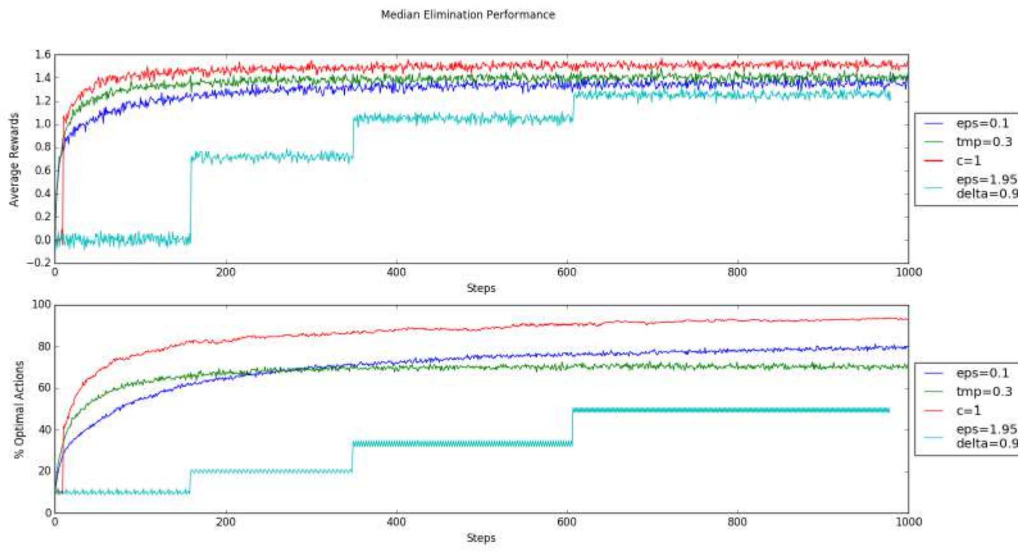

Figure 9: Comparison with other algorithms using average reward and % optimal action for 1000 steps

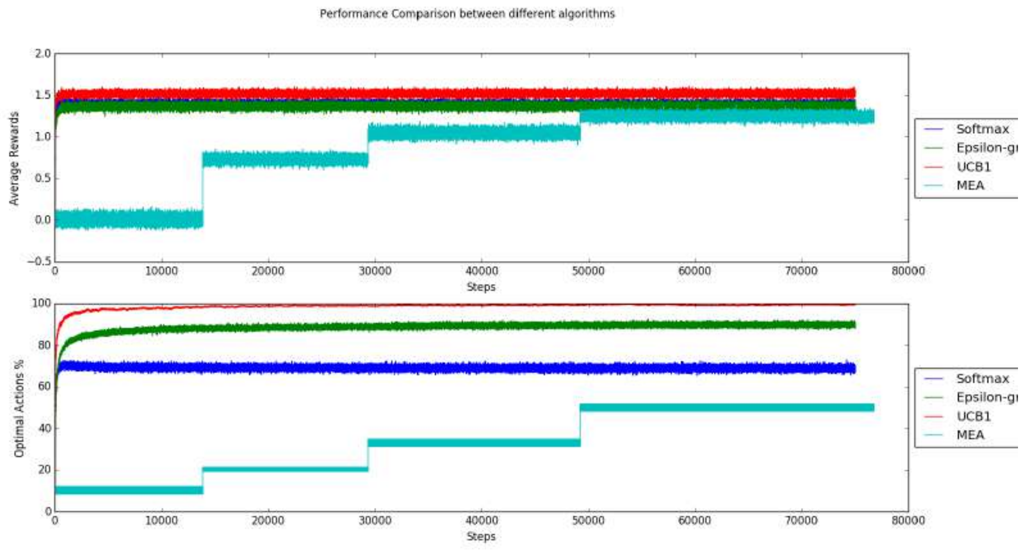

Figure 10: Comparison with other algorithms using average reward and % optimal action for 75000 steps

- Performance of MEA is better than other algorithms in terms of % suboptimal arm returned is because it tries to get the best arm in a set of arms by exploring better than other algorithms. No notion of regret is considered, and all arms are tried equally in a given set of arms.

Algorithm	Error %
ϵ - Greedy (ϵ=0.1)	1.4
Softmax (τ = 0.3)	0.6
UCB1 (c = 1)	0.65
MEA (ϵ=0.3, δ=0.2)	0.35

Table 1: Error % for different algorithms

Number of time median is found per stage	Time %
Once	216.60793900489807
Twice	223.7760488986969

Table 2: Time required for MEA execution

- Increasing ϵ increases the error due to increase in number of iterations, but no specific trend is observed for δ.
- For the other algorithms, error reduces generally, as their exploration increases, as reliable estimates of arms are obtained due to exploration.

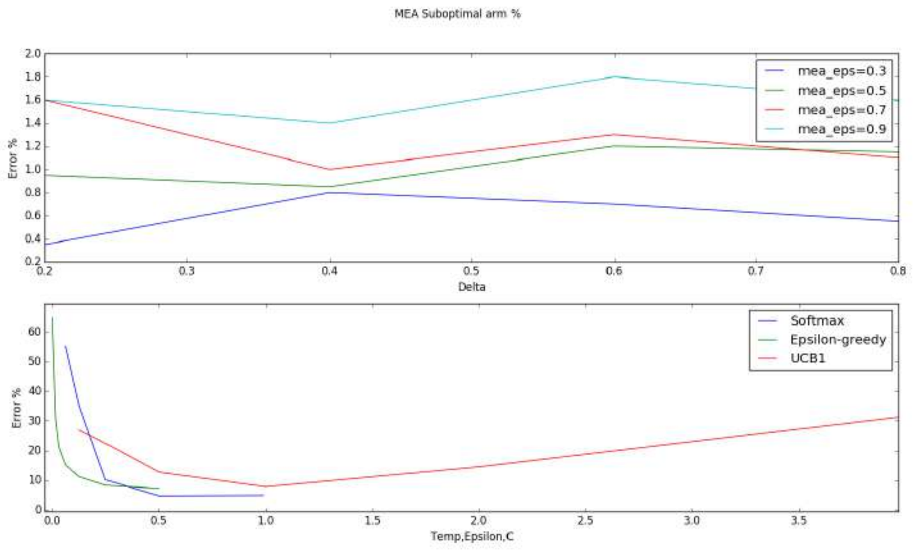

Figure 11: Suboptimal arm % for different algorithms

4.3 Determining Median as RDS

Median is found using np.median() method. MEA run with k = 1000, ϵ=0.75 and δ = 0.9 .Refer to Table 2 for comparison.
Thus, at experimented k,ϵ,δ, finding median is not the RDS, as time does not double with doubling of number of calculation of median, but at higher k and comparatively higher ϵ, δ, median finding can be RDS, if calculated in O(klog k), using sorting.

Way to make median finding faster is to use Quickselect algorithm, which has expected running time of O(k).

5 1000 Arm Bandits

5.1 Comparison among different methods

- In terms of average rewards, behavior is same as in the 10 arm case, i.e. UCB1 performs better than the other two.
- But, in terms of % optimal actions, UCB1 outperforms others by a huge margin of around 40-50 %.

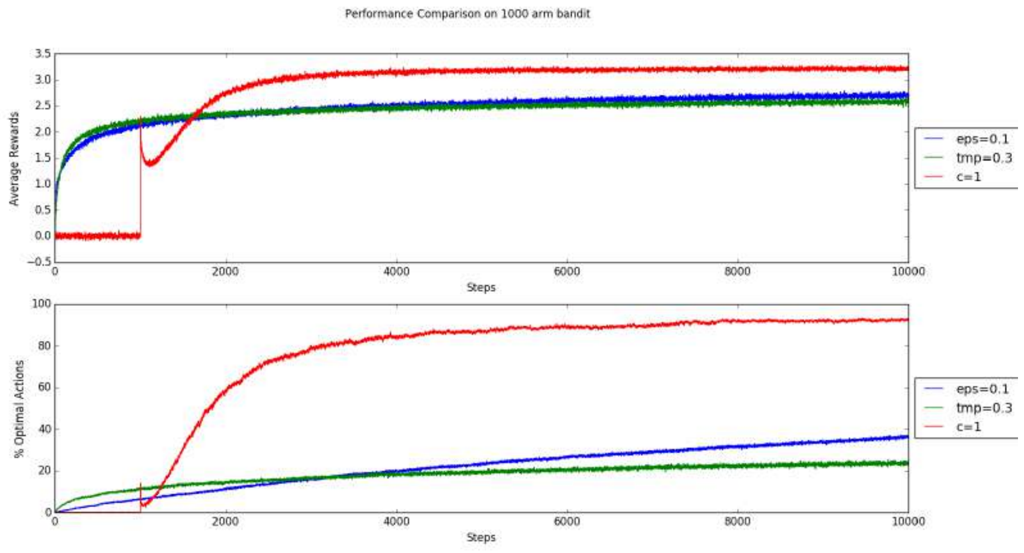

Figure 12: Comparison between UCB1, Softmax, and ϵ - greedy methods with 1000 arms for 10000 steps

5.2 Individual method comparisons

- Performance of all methods (except MEA) in 10 arm and 1000 arm case is compared by running both for 10000 steps.
- The average maximum reward that can be achieved is lower for the 10 armed than the 1000 arm testbed. Thus, all methods show difference in average rewards, with 1000 arm case having higher average rewards.
- But, the % optimal action is low for 1000 arm case compared to 10 arm case, as in 10 arm case, reliable estimates of arm values are formed compared to 1000 arm case, where number of steps are inadequate to form reliable estimates of arm values.

- Only in the case of UCB1, % optimal actions are comparable, thus indicating that UCB can make identify the best arm in lesser steps than the other methods.
- For MEA, same number of steps are used by changing ϵ,δ as 10 arms : (0.15,0.4) and 1000 arms : (1.8,0.9). Thus, we observe that computation required for more arms increases, giving a lot worse (ϵ,δ) guarantees for same steps.

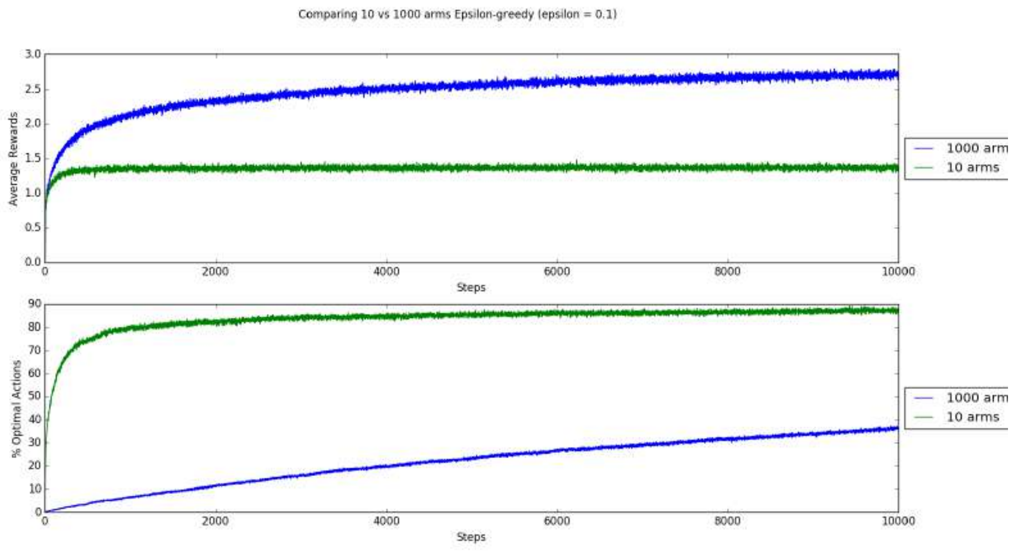

Figure 13: Comparison between Epsilon-greedy for 10 and 1000 arms run for 10000 steps each

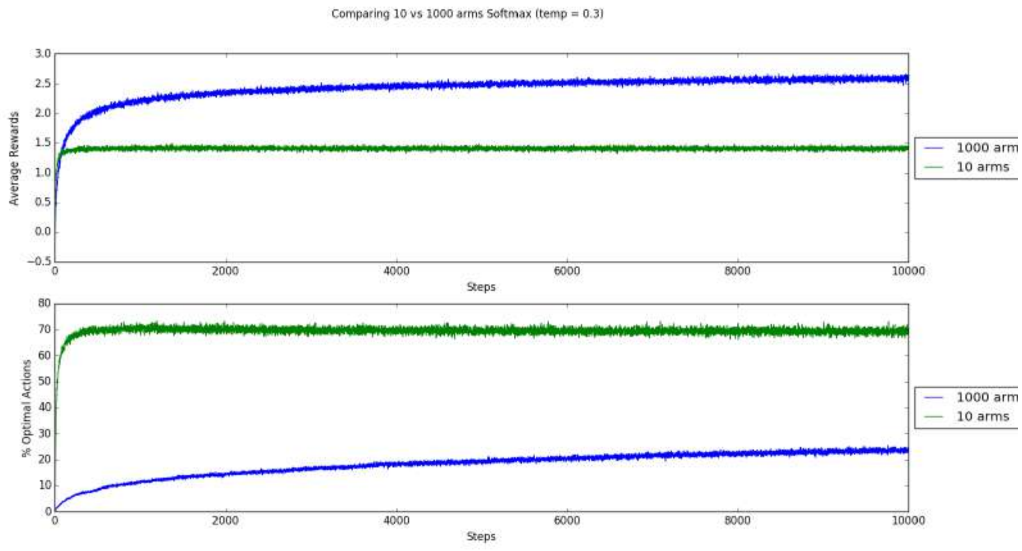

Figure 14: Comparison between Softmax for 10 and 1000 arms run for 10000 steps each

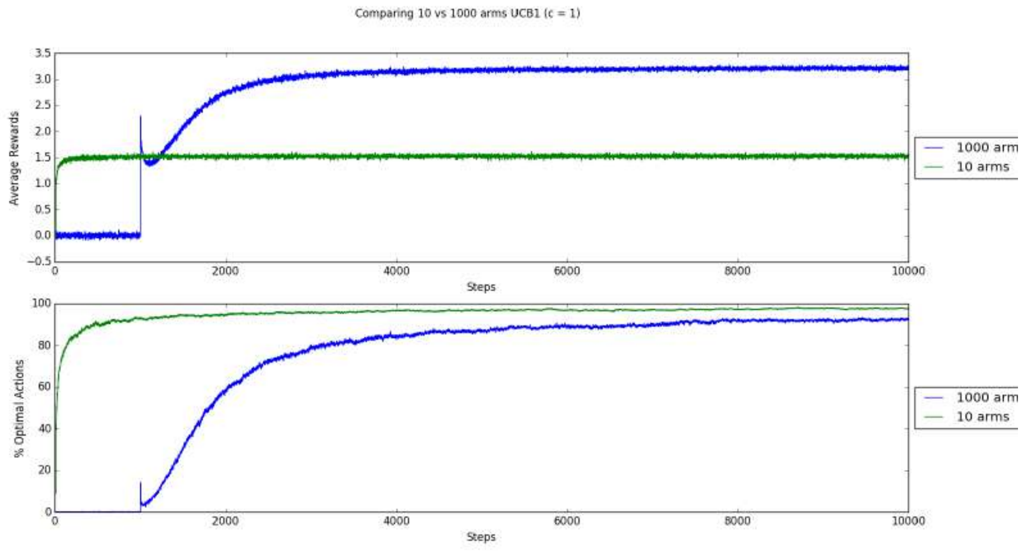

Figure 15: Comparison between UCB1 for 10 and 1000 arms run for 10000 steps each

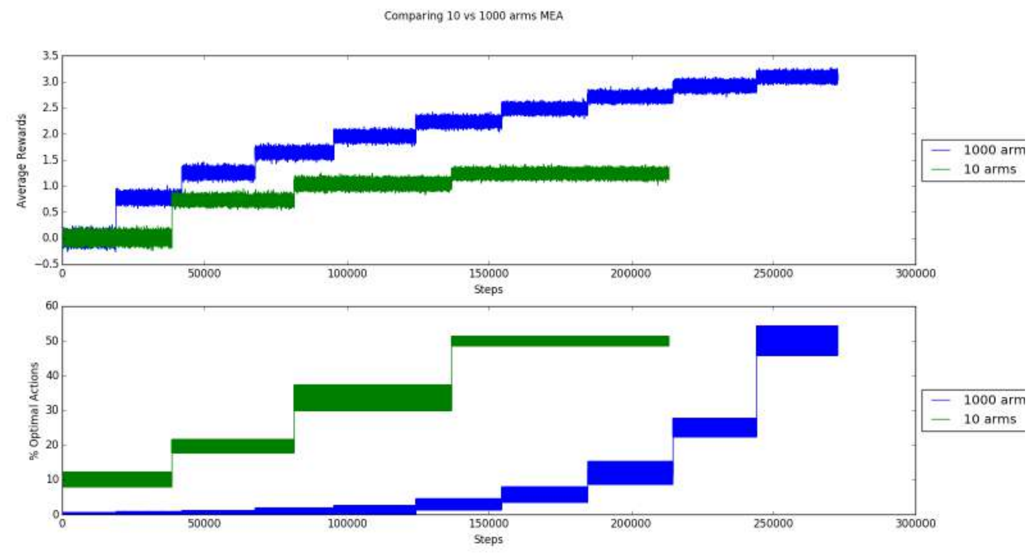

Figure 16: Comparison between MEA for 10 and 1000 arms run for approximately 200000 steps each