Indian Institute of Technology, Madras
CS6700: Reinforcement Learning
Reinforcement Learning Assignment-I Report

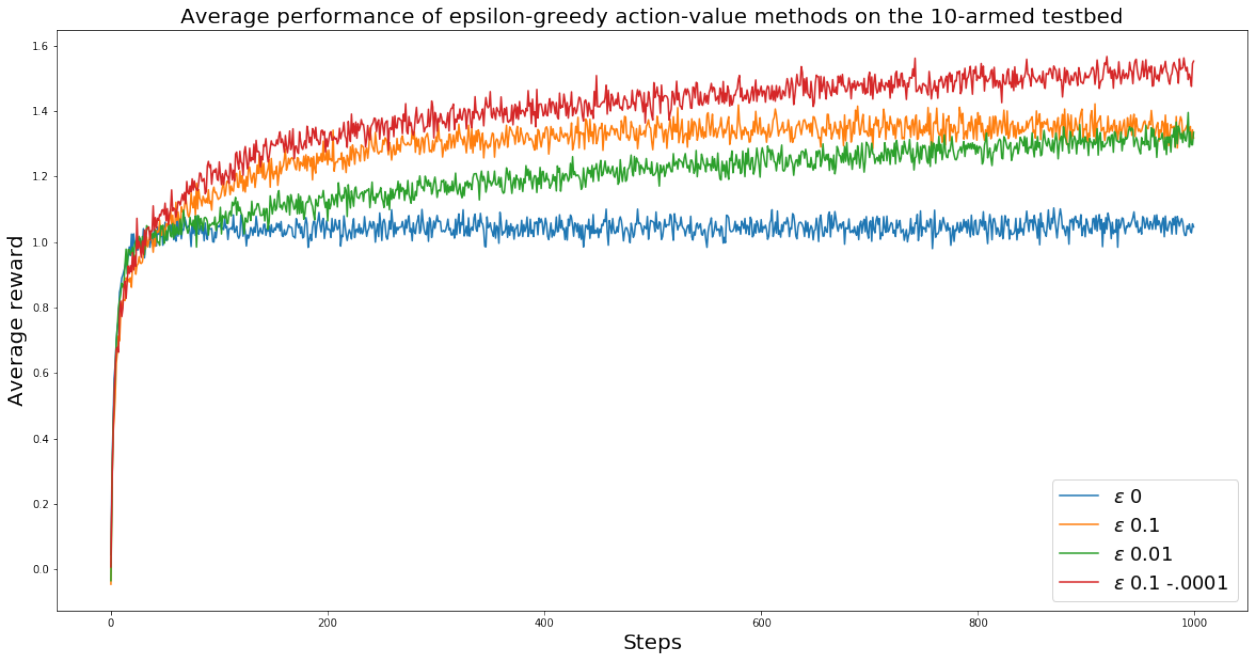
Rajan Kumar Soni - CS18S038

January 2020

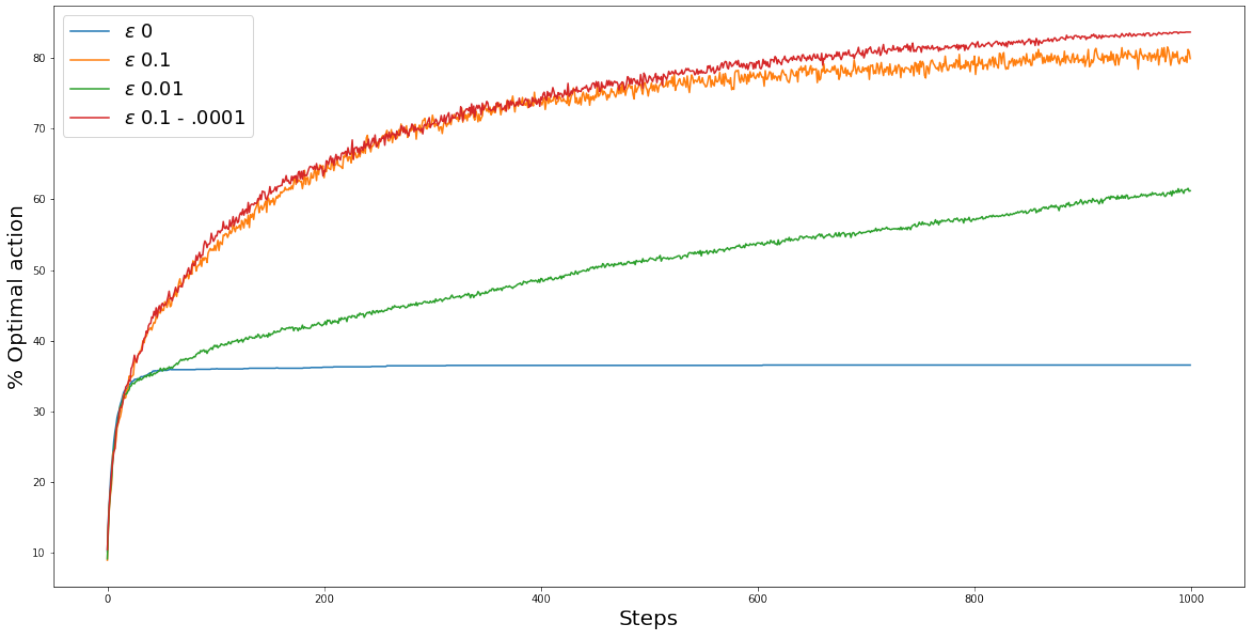
Contents

1 Implementation of ϵ-greedy and related plots

1.1 Related plots



(a) Average reward per step



(b) percent Optimal action per step

1.2 Inference from Observation

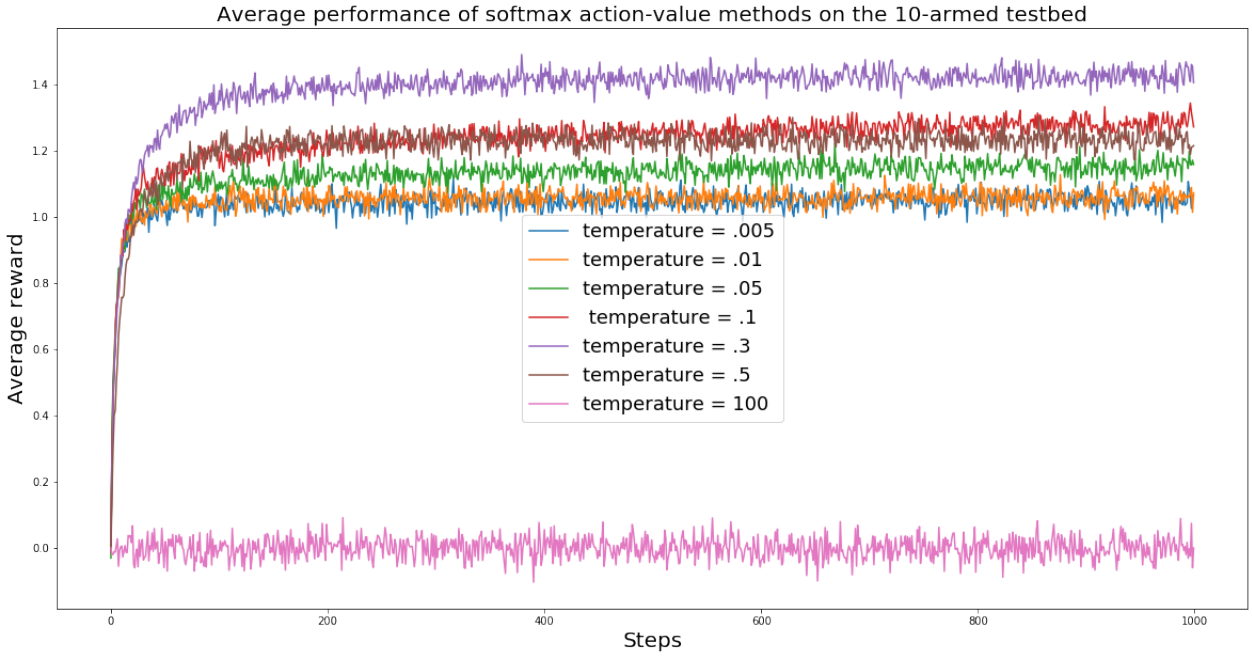
Referring to the figure (a) and (b) above:

- I have plotted the curve for different values of $\epsilon = 0.0$, .1, .01 and decaying epsilon .Above plots are averages over 2000 runs with different bandit problems
- For $\epsilon = 0$, we are fully greedy towards best action at that moment, So not wasting any time for exploration. The greedy method improved slightly faster than the other methods at the very beginning, but then leveled off at a lower level. It achieved a reward-per-step of only about 1, compared with the best possible of about 1.5 on this test-bed. It performs bad in long run as it got stuck in performing sub optimal action.Although it is good for stationary distribution with reward variance is zero.
- For $\epsilon = .01$, As it chooses the optimal action at that moment with .991 probability. As it grows slow then greedy because it is also exploring to find the optimal action. Although if it finds the optimal action still it is exploring with probability .009.
- For $\epsilon = .1$, As it grows faster then ϵ =.01 and slower then greedy because it explores more then ϵ =.01 so it found faster optimal action. found the optimal action earlier, but it never selected that action more than 91% of the time. It is good if we have less number of steps and high variance in rewards.

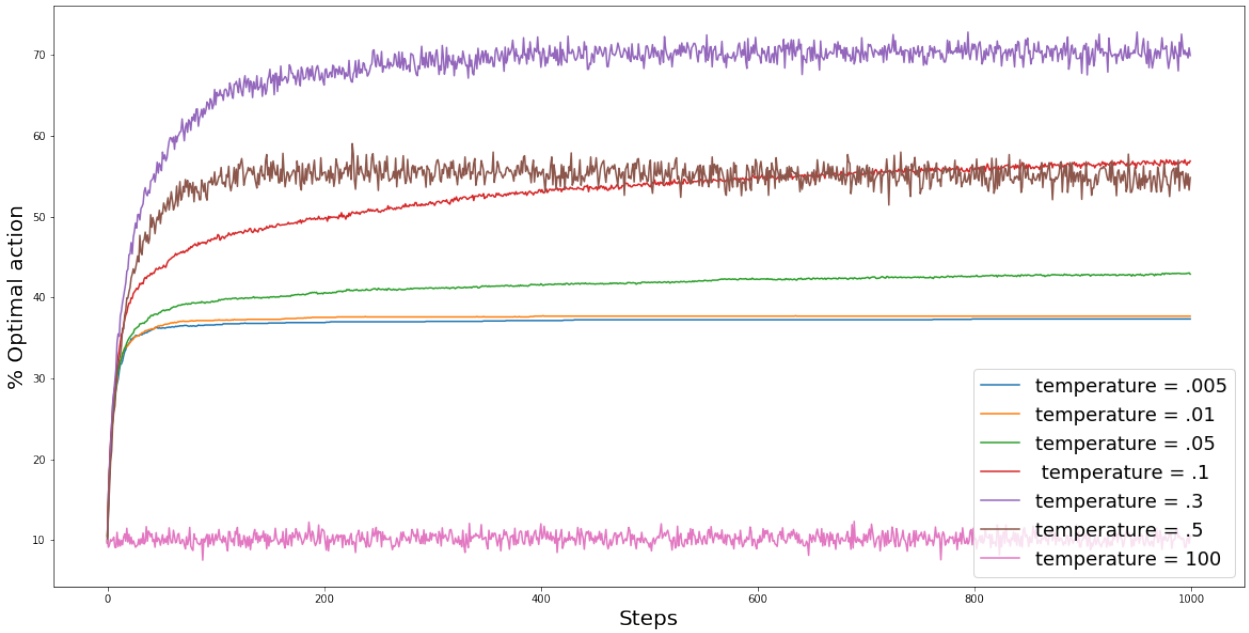
- For $\epsilon = 0$, the greedy method found the optimal action in only approximately one-third of the tasks. In the other two-thirds, its initial samples of the optimal action were disappointing, and it never returned to it.
- While the other two plot , keep exploring so founding of optimal action keep on increasing, but the still not be able to pick 100% optimal action because they will still be exploring.
- For variable $\epsilon 0.1 - .0001$, As we can see in the figure, It is better then all others because it is keep decreasing or dcaying exploration part. In the start it focuses more on exploration but as the time passes exploration keep decreasing so % optimal action keep on incresing.

2 Implementation of soft-max algorithm and related plots

2.1 Related plots



(a) Average reward per step



(b) percent Optimal action per step

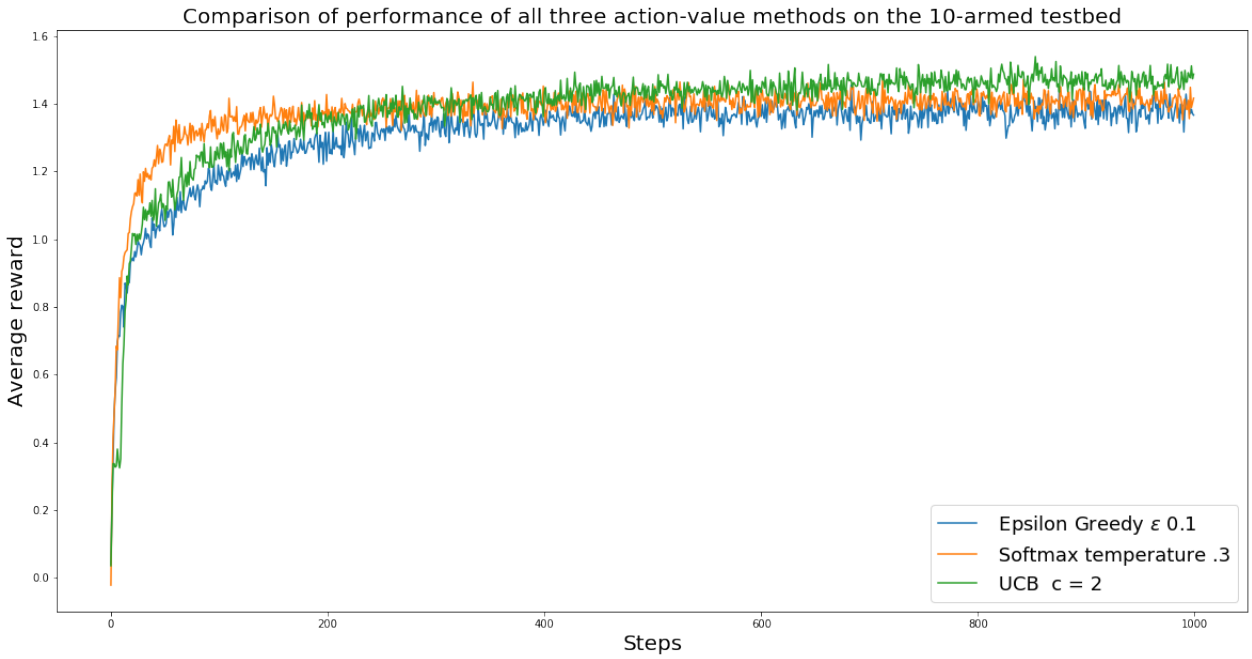
2.2 Inference from Observation

Referring to the figure (a) and (b) above:

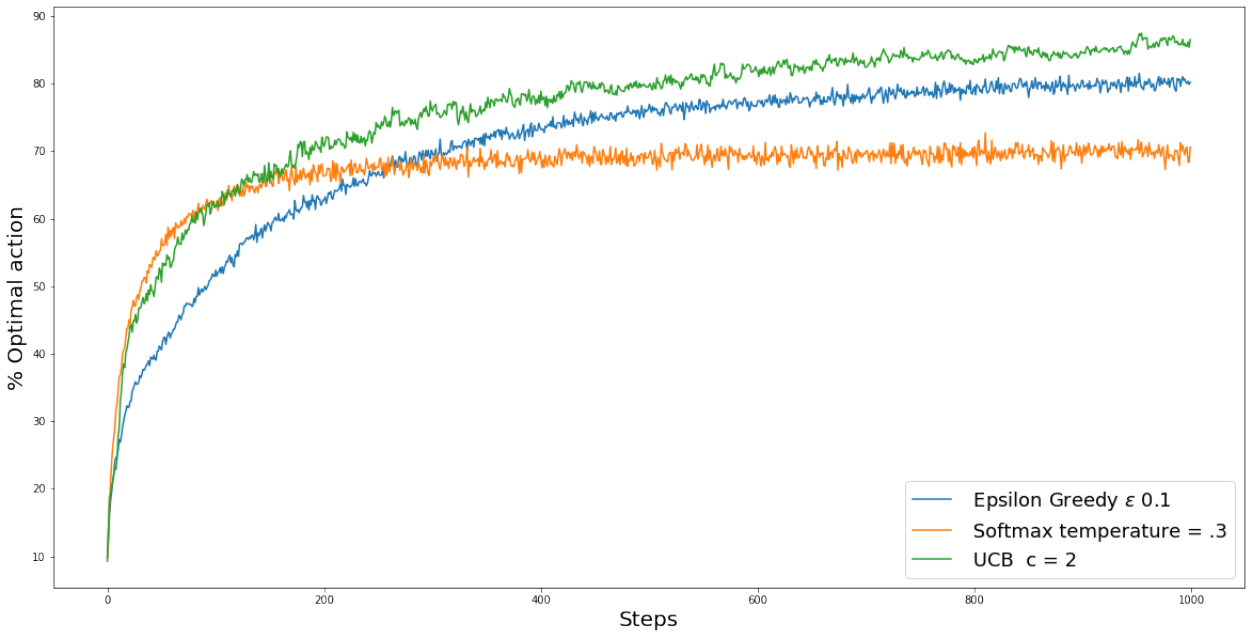
- We can see plots for different values of temperature **.001, .005, .01, .05 , .1, .5**. Here soft-max algorithms selects an arm using Gibbs Distribution.
- In this algorithm, temperature parameter controls the distribution of the actions.
- for temperature close to 0, in above plot When $t = .001$ or $t = .005$. The arm having high estimate will have very high probability and others will get get approx zero probability.We can see it is behaving like greedy one.
- And as t tends to infinity, the algorithms picks arms uniformly at random; what ever mean estimate for each arm you have, all arms value will become close to zero. Since our rewards are taken from standard normal distribution, so when $t = 1$, it will pick arms uniformly at random, hence will perform poorly as it will not be pulling optimal arm more times.
- We can observe from the above plot for temperature values near to zero it performs mostly exploitation, for 0 <=temperature <= 1 we can tune it for like ϵ-greedy by balancing the exploit and explore dilemma but for temperature > 1 it started to behave randomly at every step it chose action uniformly.At temperature = infinity updating the action reward estimate does not effect the probability distribution of actions
- In above graph, For temperature = 100 soft-max action selection becomes proper random and because rewards means and reward itself coming from Gaussian distribution, so we can see only 10% optimal action chosen.

3 Implementation of UCB1 algorithm and comparing with ϵ-greedy and soft-max and related plots

3.1 Related plots



(a) Average reward per step



(b) percent Optimal action per step

3.2 Inference from Observation

Referring to the figure (a) and (b) above:

- Formula used for Updating the mean estimate for every arm at time t: $A_t = argmax[\mathbf{Q}_t(a) + c\sqrt{\frac{\ln(t)}{N_t(a}}]$.
- UCB1 algorithm is a simpler, more elegant implementation of the idea of optimism in the face of uncertainty. It uses the fact that there is always uncertainty about the accuracy of the action- value estimates, hence exploration is needed. Thus it would be to better to select non optimal arms according to their potential for actually being optimal, by taking into account both how close their mean estimates are to being maximal and the uncertainties in those estimates.
- $c > 0$ controls the degree of exploration.The idea of this upper confidence bound (UCB) action selection is that the square-root term is a measure of the uncertainty or variance in the estimate of a's value. The quantity being max'ed over is thus a sort of upper bound on the possible true value of action a,with c determining the confidence level. Each time a is selected the uncertainty is presumably reduced: $N_t(a)$ increments, and, as it appears in the denominator, the uncertainty term decreases. On the other hand, each time an action other than a is selected, t increases but $N_t(a)$ does not; because t appears in the numerator, the uncertainty estimate increases.
- UCB1 guarantees that all arm will be selected and as the time passes arms having higher estimate will be selected more and the arms having lower estimates selected less.

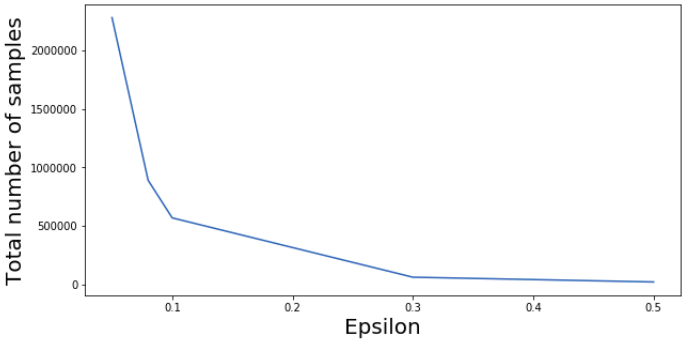
Refering to figure (a) and (b)

- We can see on initial steps UCB1 more focused on exploration after that as its starts focusing the arms having high estimate confidence interval.UCB1 is self balancing because there is no parameter required to tune exploration although we can control the degree of exploration , while in the case of softmax and ϵ-greedy exploration is at constant rate.

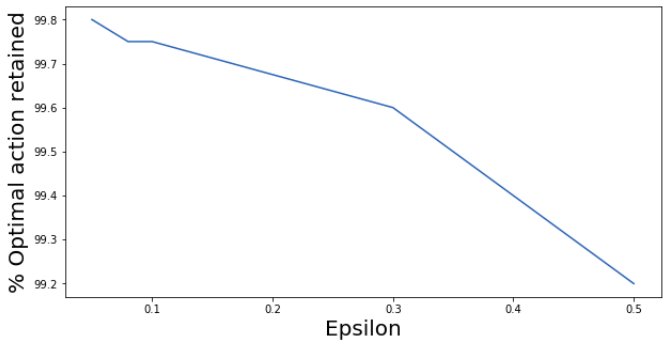
4 Implementation of Median algorithm and comparing with ϵ-greedy, soft-max and UCB1 and related plots

4.1 Related plots

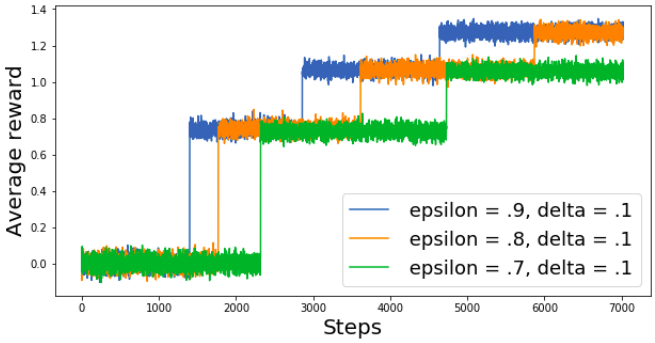
for delta = .1



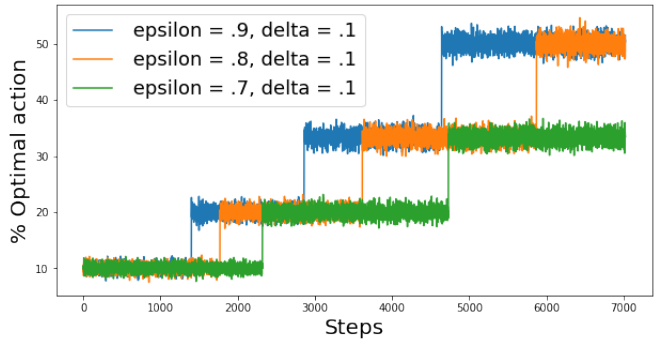
(a) Total number of samples vs epsilon for median Elimination



(b) percent Optimal retained vs epsilon for median Elimination



(c) Average reward per step for median Elimination

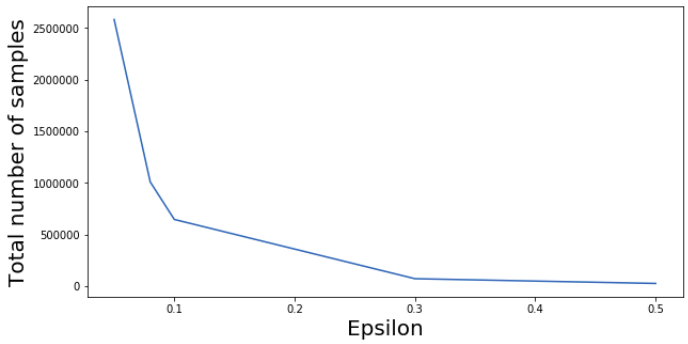


(d) % optimal action per step for median Elimination

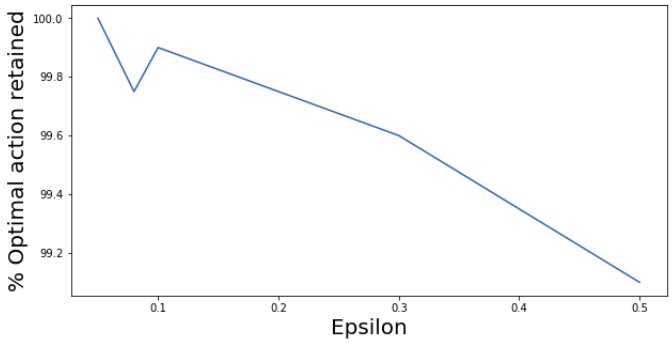
for delta = .05

4.2 Inference from Observation

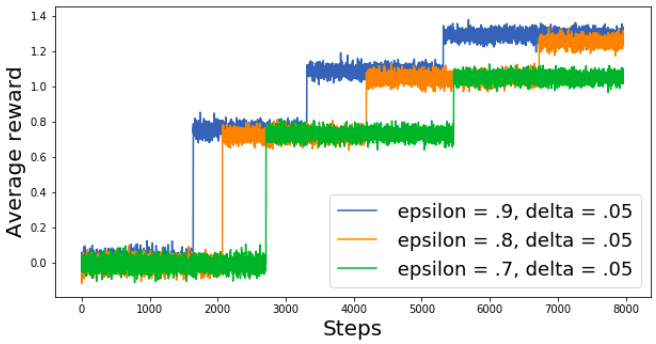
- The Median Elimination algorithm is an (ϵ, δ)-PAC algorithm. In every round it eliminates half of the bad arms, thus in $\log(k)$ steps it will return an arm within range of optimal arm with at least (1 δ) probability.
- The sample complexity of the Median Elimination is $\mathbf{O}(k \log(\frac{3}{\delta})/\epsilon^2)$, where k is the number of arms
- From the above results we can see that as the epsilon and delta is decreasing Number of samples required to satisfy (ϵ, δ)-PAC also increasing. For given delta as the epsilon value



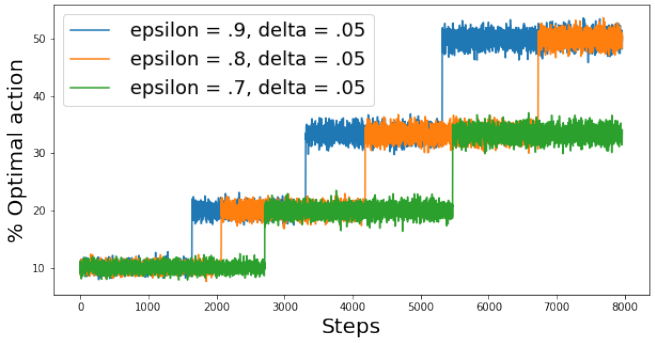
(a) Total number of samples vs epsilon for median Elimination



(b) percent Optimal action retained vs epsilon for median Elimination



(c) Average reward per step for median Elimination



(d) percent Optimal action retained per step for median Elimination

is decreasing percentage of optimal arm retained also got increased and for that number of sample also got increased.

Observation for different values of ϵ and δ			
ϵ	δ	Total number of samples	% optimal arm retained
.05	.1	2277388	99.8
.08	.1	889601	99.75
.1	.1	569342	99.75
.3	.1	63251	99.6
.5	.1	22767	99.2
.05	.05	2582629	99.85
.08	.05	1008826	99.9
.1	.05	645644	99.6
.3	.05	71730	99.4
.5	.05	25820	99.1

- Also, we can observe that % optimal arm retained is relatively little less when ϵ is large, because since we are taking less samples probability of eliminating optimal arm is more since ϵ is large. For example for $\epsilon = 0.05$ $\delta = 0.05$, total samples taken is 2582629 and percentage

optimal arm retained is 99.85 while on other hand when error is allowed greater then .05 but less then .5 percentage optimal arm retained is 99.1. This is because in first case we were more restricted that is it took more sample and result got improved. Similar trend we can see keeping epsilon value constant for delta = .1 and delta = .05, as we want more confidence so number of samples increases and because samples increases in such amount that percentage optimal arm retained also increases.

- Thus, with these empirical observation, we can conclude that Median elimination algorithm is indeed (ϵ, δ)-PAC algorithm with sample complexity of $\mathbf{O}(k \log(\frac{3}{\delta})/\epsilon^2)$.

- If we talk about the complexity of calculating Median. If we use sort algorithm it would be $O(k \log(k)$ where k is the number of arms. We can use Deterministic linear time median-of-medians algorithm or randomized Monte Carlo for O(k) complexity.
- we can see from the graph regret is maximum for Median Elimination for high bouning error .9 and low confidence .1. So for limited number of time steps me Median Elimination is not good.

Is finding median the rate-determining-step ?

As we know in median elimination algorithm we can divide the algorithm in two parts sampling and finding median.We know that sampling complexity depend on epsilon and delta, it also depends on the type of distribution rewards while finding median complexity depends on number of arms.

From the below table we can observe that,for given value of epsilon, delta as the number of arm increases ,rate of increases of Time taken by algorithm is more then rate of increase of Total time taken by finding median. So from here we conclude is not the rate determining step. But as the arm grows ratio (Time taken by algorithm)/(Total time taken by finding median) decreases. From here we can say that for small number of arms it is not rate determining but for large number of arms it an be.

Observation for rate determining step(time interval in second)				
ϵ	δ	Time taken by algorithm	Total time taken by finding median	number of arms
.05	.05	0.126398	0.000162	10
.08	.05	0.047517	0.00012	10
.1	.05	0.0300	0.000132	10
.3	.05	0.003837	6.1e-05	10
.5	.05	0.00163 4.406e-05	10	
.05	.05	23.202	0.0011	1000
.08	.05	9.112599	0.0012	1000
.1	.05	5.9413	0.0010	1000
.3	.05	0.696	0.00093	1000
.5	.05	0.268846	0.00085	1000

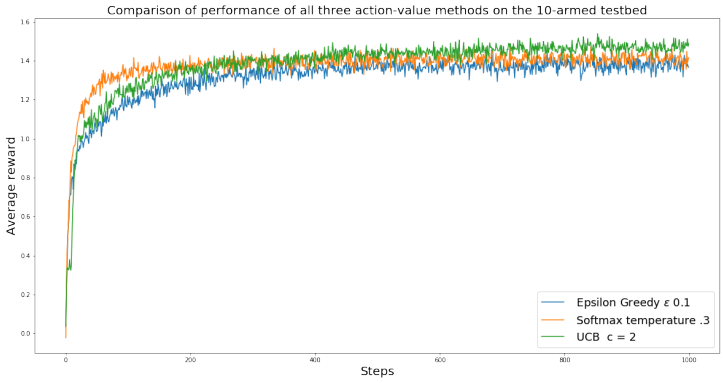
Compare MEA with other algorithm ?
MEA is an action elimina- tion algorithm which gaurentees (ϵ, δ)-PAC, on other hand ϵ-greedy, Softmax and UCB1 are algorithm who selects arm..

If the number of time steps is very large, then we can first use MEA to find out optimal arm with (ϵ, δ)-PAC and pull that arm for remaining time steps, but if time steps are less than the sample complexity of MEA for (ϵ, δ)-PAC, then we would do better using action selection algorithm like UCB1. We can see from the graph above in 1000 steps UCB1is selection optimal arm about 90% but Median elimination take more than 3000 steps even for very bad PAC Bound condition.

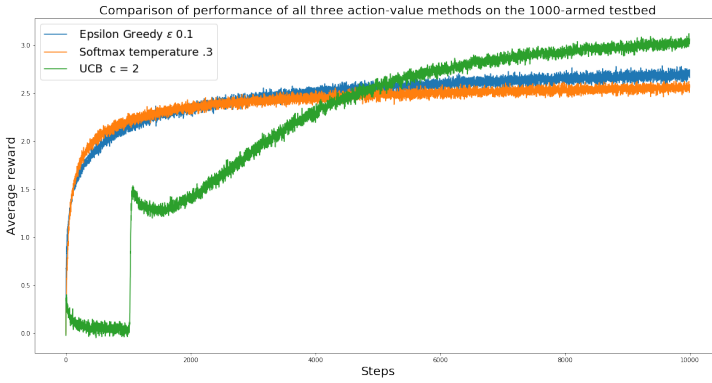
We could also use both at the same time ,action elimination and action selection algorithm, when the number of arms is very large like 100000 or 10000000, in these case we can use Median elimination algorithm for first few number of steps eliminate the big number of bad arms as it will remove half of the arms in every round, then use action elimination UCB1 algorithm on the remaining arms for the remaining time steps.

5 Comparison of the above four algorithm as the number of arm grows

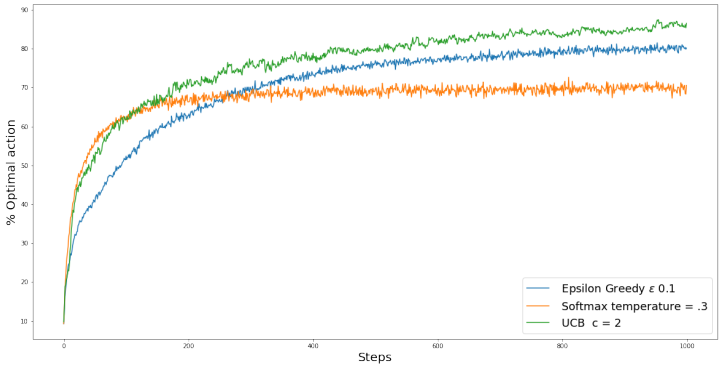
5.1 Related plots



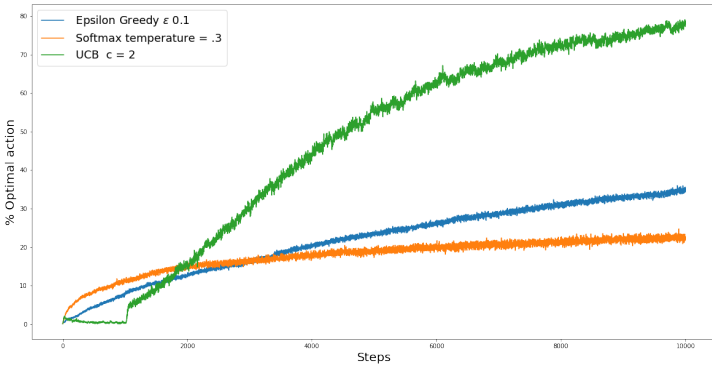
(a) Average reward per step for 10 arm



(b) Average reward per step for 10 arm



(c) percent Optimal action per step for 1000 arm



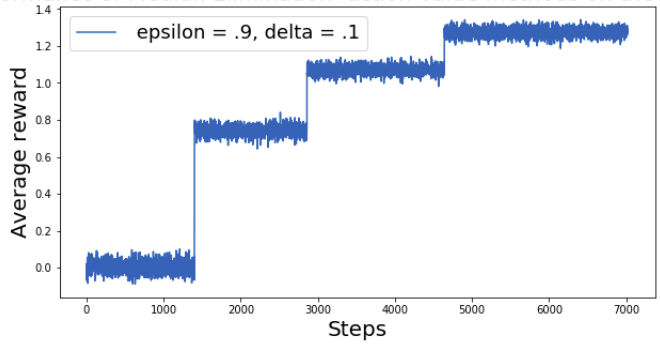
(d) percent Optimal action per step for 1000 arm

5.2 Inference from Observation

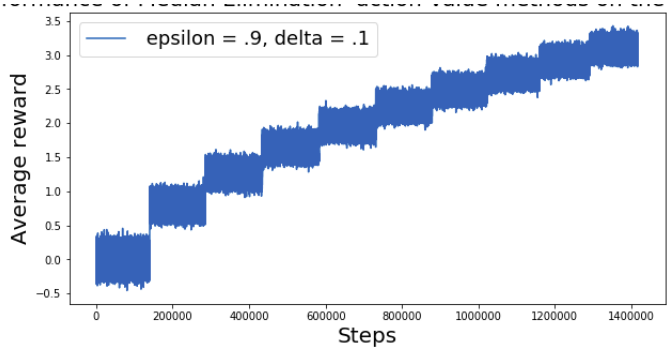
- Here we are Comparing the average performance of UCB1, Softmax and $\epsilon-$greedy algorithm on the 10 armed testbed and 1000-armed testbed over 2000 different bandit problems for time steps 1000 and 10000 repectively.
- Thus, we run all three algorithms for 10000 100000 time steps. From the above graph We observe that Softmax and epsilon greedy algorithm performs better in initial steps. As we know that tuning the temperature parameter we can make softmax to behave like epsilon greedy.So we can compare only Softmax and UCB1. As in the case of Softmax, it will try to pull arm which highest mean estimates with more probability and explore other arms with less probability. But UCB1 does not perform well in intial steps as it will do more exploration selects arms randomly which are not pulled enough time ($N_t(a)$ is low) and who's variance is high.

Figure 1: Comparison between 10 arm and 1000 arm performance for Median Elimination

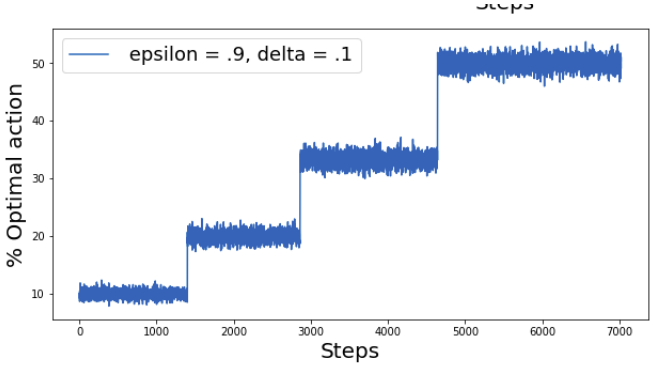
(a) Average reward per step for 10 arms



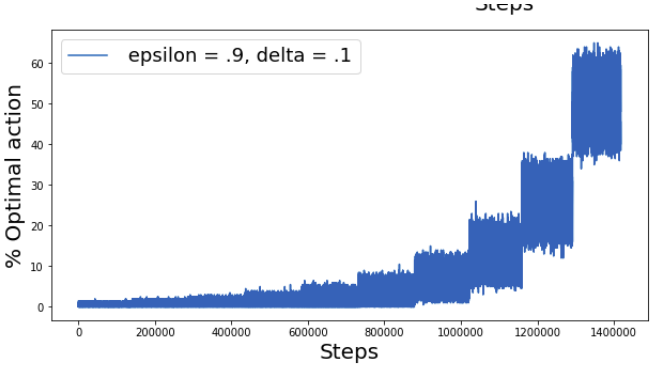
(b) Average reward per step for 1000 arms



(c) percent Optimal action per step for 10 arms



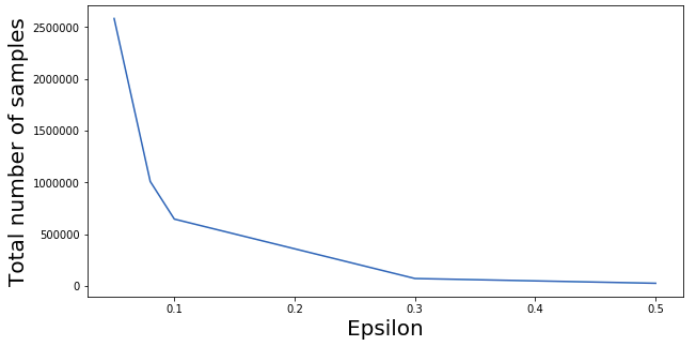
(d) percent Optimal action per step for 1000 arms



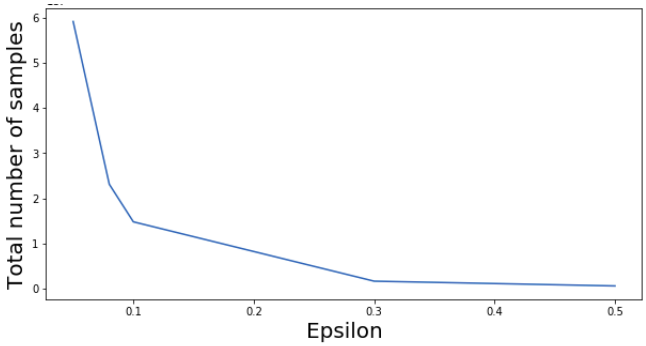
- UCB1 explores more in initial time steps.In above graph Spike can be seen in UCB1,WE can see in case of UCB1, As I have initialised all expected reward to zero.At t=1, it first randomly select actio a. It $N_1(a)$ got incresed by 1 $c\sqrt{\frac{\ln(t)}{N_t(a}}]$ and numerator 't' also got increased making its variance reduced but for other arms only numerator got increased making their variance higher , so in nest time step some other arm got selected having high variance. That is why we see a jump in Average reward after 10th step in case of 10 arm and after 1000th step in case of 1000 arm.
- Now, we run Median Elimination algorithm for 1000 armed-bandit case. And similar to earlier case when epsilon is large, total samples required would be comparitively less and when epsilon is small more total samples would be required to achieve (ϵ, δ)-PAC.
- As we can see from above graph as we increase number of arms for given alpha and delta regret also increases for Median Elimination.
- But now as number of arms are 1000 it's sample complexity would also be increase by 100x compared to earlier case when there we 10 arms, which can be observed from below table and graphs.
- From above discussion we concluded that Median Elimination has maximum regret.

Figure 2: Comparison between 10 arm and 1000 arm for Median Elimination for given $\delta = .05$

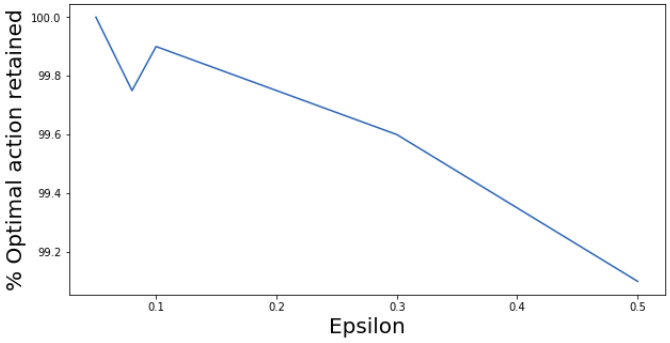
(a) Total number of samples vs epsilon for 10 arms



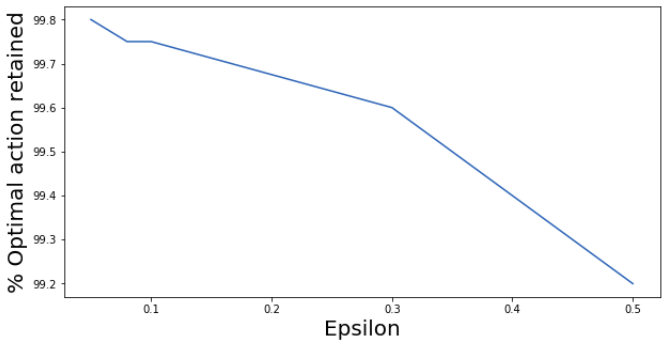
(b) Total number of samples vs epsilon for 1000 arms



(c) percent Optimal retained action vs epsilon for 10 arms



(d) percent Optimal retained action vs epsilon for 1000 arms



Observation for different values of ϵ and δ						
ϵ	δ	number of arms	Total number of samples	% optimal arm retained	Time taken by algorithm	Total time taken by finding median
.05	.05	10	591410	99.7	2.07	.00014
.08	.05	10	231010	99.3	.799	.00015
.1	.05	10	147840	99.7	.52	.00014
.3	.05	10	16420	98.4	.057	.000075
.5	.05	10	5910	97.3	.02	.000070
.05	.05	1000	59141000	99.6	205.2	.00098
.08	.05	1000	23101000	99.4	79.12	.00099
.1	.05	1000	14784000	99.5	51.2	.00090
.3	.05	1000	1642000	98.4	5.56	.0007
.5	.05	1000	591000	97.8	2.12	.00099

- As we can see from the above table Time taken by the whole algorithm is much more than the time taken by median finding. Still here sample complexity is dominating here. But arms much much bigger median finding might dominate.

6 References

1 http://jmlr.csail.mit.edu/papers/volume7/evendar06a/evendar06a.pdf

2 https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf