Empirical Evaluation of Multi-Armed Bandit Algorithms

Wenzhe Li*
Department of Computer Science
Texas A&M University
College Station, TX 77843
liwenzhe@cse.tamu.edu

Abstract

The goal of multi-armed bandit problem is to maximize the total rewards in a series of trials by pulling arms sequentially. Multi-armed bandit problems provide general framework for exploration and exploitation task. Although it is well studied problem and many well proved algorithms have been proposed during the last decades, it still remains that there is lack of empirical evaluations in practical situations. Even though many of these algorithms have nice theoretical bounds, it does not necessarily means that one algorithm outperforms another. In this paper, we give empirical evaluations for well known MAB algorithms and analyze them in the different settings. In the second part of paper, we slightly extend the multi-armed bandit framework and considering how these algorithms perform when we have some knowledge about portion of arms. For e-commerce application, there is always the case that new items are added to the category which greatly fit to this dynamic multi-armed bandit case. We found out that ϵ greedy policy performs well for both of these cases.

1 Introduction

Multi-armed bandit problem is well studied topics and extensively used to model the exploration and exploitation trade-off. It can help the agent discover new knowledge to improve the future decision. We should notice that the strategy of multi-armed bandit problem is trying to improve the long-term performance by continually discovering new knowledge, while pure exploration is for maximizing the short-term rewards.

Multi-armed bandit is shown to have great applications. Search engine ranking [8,9] is one good example. In order to improve the user experience, it is critical for search engine to understand user's interest and rank the search results based on that. It tends to put the most relevant results on the top of the page. However, in order to broadly understand user's preference and improve user's long term satisfaction, search engine should also be able to discover new topics for the user, which is under the multi-armed bandit setting. Product recommendation or online advertisement is another rich area that can be modeled as exploration and exploitation trade off.

During the last decades, many of well studied MAB algorithms have been proposed by the researchers. Their regret bounds are well understood and much research has been done to understand these algorithms under the PAC-framework [3]. While these theoretical work proved that these algorithms have nice performance, very little empirical evaluation has been done for these algorithms. As far as we know, there are only two papers that deal with evaluation of multi-armed bandit problems. In [4], it provides evaluation result for several algorithms, but not comprehensive, also does

*personal webpage: www.students.cse.tamu.edu/liwenzhe

not account for different parameters which is shown to be important. [5] is the most recent work, which evaluates six different multi-armed bandit algorithms using different variance and different number of bandit arms. It is interesting to see that the variance of the reward is the most important factor that directly impact the MAB algorithm performance. In this paper, we further extend this evaluation to more comprehensive and general cases. We assume the variance of these rewards does not have the identical value, instead we set them randomly from $[0, 1]$. In this case, we can look at the performance of these algorithms in more general view.

In the second part of paper, we slightly extend the multi-armed bandit framework and considering how these algorithms perform when we have some knowledge about portion of arms. We model the problem by periodically adding new arms into the existing arms, where we have some knowledge about those existing ones. We will show that ϵ-greedy policy performs well in both situations, which is consistent with the work done in [4,5].

The paper is organized as follows, In section 2, we will give a formal definition of multi-armed bandit problem as well as some notations that frequently used in the following sections. In the third section, we briefly discuss some well known multi-armed bandit algorithms. Including ϵ-greedy policy, ϵ_n-greedy policy, UCB1, Softmax, EXP3, and Pursuit. In the fourth section, we give comprehensive experiment results for the performance of all these algorithms, and analyze their performances. Finally, we will provide some potential ideas on how to apply multi-armed bandit problems to practical applications.

2 Multi-armed bandit problem

As defind in [4], in its most basic formulation, a K-armed bandit problem is defined by random variables $X_{i,n}$ for $1 \leq i \leq K$ and $n \geq 1$, where each i is the index of a gambling machine. (i.e., the "arm" of a bandit). Successive plays of machine i yields rewards $X_{i,1}, X_{i,2}, ..$, which are independent and identically distributed to an unknown law with unknown expectation μ_i.

In the simplest formulation, a bandit problem consists of a set of K probability distributions $(D_1, ..D_K)$, with associated expected values $(\mu_1, \mu_2, ...\mu_K)$ and variances $(\sigma_1^2, \sigma_2^2, ...\sigma_K^2)$. Multi-armed bandit problem is a sequential problem that each time we choose to pull one of those and get a reward based on certain distribution.

Formally, we can write down it as: at each trial, $t = 1, 2, ...$, player selects an arm with an index $j(t)$, and receives a reward $r(t) \sim D_{j(t)}$. The goal is to maximize the total rewards (or minimize the total regret) for the long run.

The total expected regret, defined for any fixed turn T as:

$$R_T = T\mu^* - \sum_{t=1}^{T} \mu_{j(t)}$$

where $\mu^* = max_{i=1,...k}\mu_i$, is the expected reward of the best arm. Alternatively, we can express the total expected regret as

$$R_T = T\mu^* - \mu_k \sum_{k=1}^{K}(T_k(T))$$

where $T_k(T)$ is the number of times arm k has been pulled off during the first T trials.

In order to compare different strategies, we use the total regret as evaluation criteria in this paper. Some other criterias [5] also can be used for evaluation purpose.

3 Multi-bandit Algorithms

In this section, we will briefly introduce several well-known bandit algorithms that we experiments in this work. These are ϵ-greedy policy, ϵ_n greedy policy, UCB1, Softmax, EXP3, and Pursuit, respectively.

3.1 ϵ-greedy strategy.

ϵ-greedy is the most simple strategy that is widely used and proves to be very effective one. It has obvious generalizations for sequential decision problems. At each trial $t = 1, 2...$ algorithm selecs the arm with the highest empirical mean with probability $1 - \epsilon$, and selects a random arm with probability ϵ. The process can be represented as :

$$p_i(t+1) = \begin{cases} p_i(t+1), & \text{if } i = argmax_{j=1,...K}\hat{\mu_j}(t) \\ \epsilon/k, & \text{otherwise} \end{cases}$$

Here, we use fixed ϵ value. Actually, there are many different versions of ϵ-greedy strategies, including ϵ-decreasing strategy, ϵ-increasing strategy, etc.

3.2 ϵ_n-greedy strategy

Proposed by [2], is modified version of ϵ-greedy algorithm, where the ϵ values change over time instead of making it fixed. During the initialization phase, we define sequence of ϵ_n as for $i = 1, 2...n$

$$\epsilon_n = \text{min}\{1, \frac{cK}{d^2 n}\}$$

where $c > 0$ and $0 < d < 1$. The pulling phase is same as in ϵ-greedy algorithm.

3.3 UCB1 strategy

The UCB family of algorithms has been proposed by [2], as a simpler, more elegant implementation of the idea. An extension of UCB-style algorithms to sequential, tree-based planning was developed by [10]. In this paper, we give simplest algorithm, called UCB1. The algorithm maintains both the number of pulls and empirical mean for each arm. At first, each arm is pulled once. After that, at time $t = K + 1, ...$, arm $j(t)$ is picked by the formula:

$$j(t) = \text{argmax}_{i=1,...k}(\hat{\mu_i} + \sqrt{(\frac{2 \ln t}{n_i})})$$

where n_i is the number of times arm i has been pulled. And it shows that after any number n of plays, the expected regret is at most

$$[8 \sum_{i:\mu_i < \mu^*} (\frac{\ln n}{\Delta_i})] + (1 + \frac{\pi^2}{3})(\sum_{j=1}^{K} \delta_j)$$

3.4 Pursuit stategy

Pursuit algorithms [11] maintain an explicit policy over the arms, whose updates informed by the empirical means. At each time t, it updates the probabilities as

$$p_i(t+1) = \begin{cases} p_i(t+1) + \beta(1 - p_i(t)), & \text{if } i = argmax_{j=1,...K}\hat{\mu_j}(t) \\ p_i(t+1) + \beta(0 - p_i(t)), & \text{otherwise} \end{cases}$$

where $\beta \in (0, 1)$ is a learning rate. [12] provides PAC-bounds for different forms of pursuit algorithm.

3.5 Softmax strategy

It picks each arm with a probability that is proportional to its average reward. Arms with greater empirical means are therefor picked with higher probability. Softmax selects an arm using a Boltzmann distribution. The probability update formula is as follows.

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

where γ is a temperature parameter, controlling the randomness of the choice. When $\gamma = 0$, Boltzmann Exploration acts like pure greedy. As γ tends to infinity, the algorithms picks arms uniformly at random.

3.6 Exp3 strategy

The last one we experiment in this paper is Exp3, which stands for "exponential weight algorithm for exploration and exploitation." It also choose the arm based on probabilities. For each time $t = 1, 2...$, the probability for choosing arm i represented as

$$p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_i(t)} + \frac{\gamma}{K}, i = 1....K$$

After pulling the ith arm, change the estimated reward $\hat{x_i}$ to $x_i(t)/p_i(t)$.

4 Experiment

Experimentations are consists of three parts. In the first part, we will experiment on each individual strategy by changing parameters, and see how each parameter change the performance of algorithms. In the second part, we compare all these algorithms by changing number of arms and reward distributions. Comprehensive evaluations are made in this part. Finally, we will give preliminary experiments result for dynamic multi-armed bandit settings, where the number of arms can change periodically.

4.1 Experiment Setup

All the distributions for arms are randomly generated. Mean for each arm is randomly generated uniformly from $(0, 1)$. In terms of getting reward from each distribution, we assume that the distribution is gaussian distribution. We set the number of trials equals to 1000, and repeat this procedure for 1000 times. In terms of the number of arms, we set it as $5, 10, 20, 30$ respectively (Initially, we decided to test large K values, but it takes too long time, we will include the results for larger K values in the later version of paper).

4.2 Evaluation of each algorithm by different parameters

Figure 1 and figure 2 shows how each bandit algorithm performs with different parameters. In figure 1, we set the number of arms as 10, variance as 0.1, and in figure 2, we set the number of arms as 10, variance as 1. Since UCB1 does not require any parameter, we exclude UCB1 in this part. The only difference between figure 1 and figure 2 is with different variance. It is obvious to see that the regret in figure 2 is higher than in figure 1. At least, in these two figures, we can notice that the result is consistent with each other. As the parameter values decrease, the performance tends to improve. For ϵ-greedy policy, as the ϵ value decrease, it tends to always pull the optimal arms. On the contrary, due to the large ϵ value, even after sufficient number of trials, the gambler still tries to choose random arm with probability ϵ. We can clearly recognize these cases in figure 1 and figure 2. For pursuit algorithm, the learning rate also has some impact on the performance. As the learning rate decrease, the regret becomes smaller. However, when we choose the learning rate as slightly large value, the regret decrease quickly. This implies that if we gradually decrease the learning rate, pursuit algorithm is likely to perform better.

4.3 Comparison of different algorithms

In this section,we experimented on two different cases. In the first case, we assume all the arms have identical variance of rewards, while in the second setting, we assume both mean and variance of rewards are randomly generated. [5] does not consider this case. By choosing random variance, we can consider their performance in more general setting.

4.3.1 Fixed variance

From figure 3 to figure 6 shows the experiment results for fixed variance case. Each figure is corresponding to fixed number of arms, where they have different variances. We experimented on four different K values, $5, 10, 20, 30$. Our observations can be summarized as follows:

- UCB1 algorithm tends to perform poorly compared to other algorithms. And also shows that it has unstable behaviors.
- Softmax and ϵ-greedy algorithm performs very similarly. The corresponding lines almost overlap. They both performs very well.
- The performance of Exp3 is worst. The expected regret is always higher than others.
- Softmax tends to reach the minimum per trial regret more quickly.
- Pursuit, ϵ-greedy, softmax, ϵ_n-greedy performs well.

4.3.2 Random variances

Instead of using identical variance for each arm, in this experiment, we use the random variance for each arm. We randomly generate each variance from $(0, 1)$. The experiment results are shown in figure 8.

As we expected, the overall performance looks worse than when we set the variance as identical value. However, the performance behaviors of these algorithms looks similar to the previous case. And, we can still easily find that performance of ϵ-greedy, ϵ_n-greedy, Softmax are better than the other three algorithms.

4.4 Preliminary experiment results on dynamic bandit problem

Consider one simple example, assume a person just arrive in a new town, and there are 10 restaurants in the town. And he tries to decide which one is best. After several trials, he will get some idea about the quality of each restaurant. After one year, there are another 10 restaurants newly built. In this case, in order to get information about the new ones, he also tries to visit each one. However, he has some knowledge about the service of the old 10 restaurants. The thing he should do is to balance between existing knowledge and completely new areas. Actually, we can formulate the degree of knowledge in a proper way to better understand the new areas.

In order to experiments for this case. First we run the algorithm as we did before, for 1000 trials. After finishing 1000 trials we add the same amount of arms and continue for another 1000 trials. Finally, we record the expected regret for each algorithm to see how they adapt to the dynamic case.

Because of the limited time, we only run for the UCB1 and ϵ-greedy policy. The results are shown in figure 7. It clearly shows that ϵ-greedy algorithm performs much stable and better than UCB1 algorithm. More experiments for this case, will be included in the future version of this paper.

5 Applications

Even though multi-armed bandit problem has been extensively studied, there still exists a gap between theory and practice. It means that we should also pay attention to the applications that can benefit from these beautiful theories. In this section, we give some potential ideas on how multi-armed bandit problem can be applied to practice.

In fact, the most widely known application is clinical trials, where each patient arrives sequentially and receive a treatment. The goal is to maximize the treatment effectiveness to make more patients get cured. Another well-known application is online advertisement. Where the system can monitor the user's behavior such as clicking, to decide optimal strategy for advertising.

Besides, there are several other potential applications that we can look for.

- Online product recommendation. In order to fully understand customers interest, it is very necessarily to find out new items that the customers might be interested in, instead of only recommending items that we already know that the customer are interested in. Most of recommendations system are based on collaborative filtering and content based recommendation strategy. It will be beneficial if we can appropriately combines bandit strategies for recommendation.
- In machine learning domain, feature subset selection might be another related topics. Since we can simply extend the multi-armed bandit to subset selection setting, it is likely that

we can benefit by using bandit strategies. As we know, subset selection by wrapper is combinatorial problem. By using bandit strategy, it is likely that we can reduce the search space and only search through portions of regions. Similar work has been in [7], where they use the bandit strategy to reduce the complexity for boosting.

- Multi-armed bandit can be applied to adaptive user interface. Intelligent interface design is an very active research in human computer interaction. It tries to solve the problem on how to dynamically adapt the interface to the specific user to improve his experience. This problem actually is similar to any recommendation system. Adaption may include dynamically arranging the interface structure like UI components, or display or hide some functionalities based on user's preference.

6 Conclusion

In this paper, we give a comprehensive evaluation of different multi-armed bandit algorithms, namely, ϵ-greedy policy, $\epsilon_n$greedy policy, UCB1, Softmax, EXP3, and Pursuit. We tested each algorithm on different cases and also give a comparison of these six algorithms. From our experimentations, we can find that the most simplest algorithm - ϵ-greedy performs very well and is stable. For summary, we list our contributions as follows:

- Provide comprehensive evaluation for each bandit algorithm.
- Provide comparison results for six bandit algorithms in the different settings. Also, it is the first paper that evaluate the performance based on random variance for each arm.
- Evaluate bandit algorithms for dynamic multi-armed bandit problems, where the number of arms can change periodically.
- Show potential applications that can be greatly benefited from bandit algorithms.

References

[1] Auer, P. & Cesa-Bianchi, N. & Robert E.Schapire, Y. (2003) The non-stochastic multi-armed bandit problem *SIAM J. Comput.* pp.48-77, 2002

[2] Auer, P. & Cesa-Bianchi, N. & Fisher, P. (2002) Finite-time analysis of the multi-armed bandit problem. *Machine Learning*. pp.235-256, 2002

[3] Even-Dar, E. & Mannor, S. & Mansour, Y. (2002) PAC bounds for Multi-armed Bandit and Markov Decision Processes. *Computational Learning Theory*, Vol 2375, pp.193-209

[4] Vermorel, J. & Mohri, M. (2005) Multi-armed Bandit Algorithms and Empirical Evaluation *Machine Learning, ECML 2005*. Vol 3270 pp. 437-448

[5] Kuleshov, V. & Precup, D. (2010). Algorithms for the multi-armed bandit problem. *Journal of Machine Learning*. 2010

[6] Li,L.H. & Chu, W. & Langford, J. & E.Shapire, R.E. A Contextual-Bandit Approach to Personalized News Article Recommendation. *WWW*, Raleigh, North Carolina, USA

[7] Busa-Fekete, R. & Kegl, B. Fast Boosting using Adversarial bandits *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

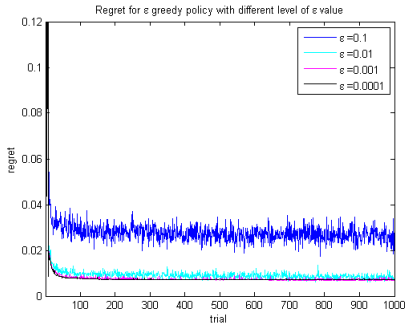
[8] Radlinski, F. & Kleinberg, R. & Joachims, T. (2008) Learning Diverse Rankings with Multi-Armed Bandits. *Proceedings of the 25th International Conference on Machine Learning*. Helsinki, Finland, 2008

[9] Slivkins, A. & Radlinski, F. & Gollapudi, S. (2010). Learning optimally diverse rankings over large document collections. *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

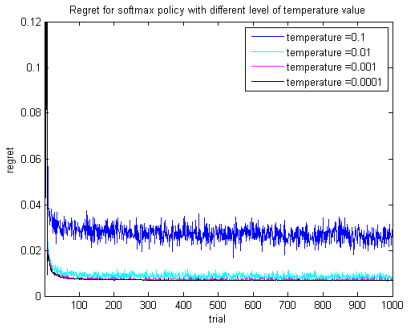
[10] Kocsis, L. & Szepesvari, C. (2006) Bandit based monte-carlo planning. *In Proceedings of ECML*, pp. 282-293, 2006.

[11] Thathachar, Sastry, P.S. (1985) A class of rapidly converging algorithms for learning automata. *IEEE transactions on Systems, Man and Cybernetics*. pp. 168-175.

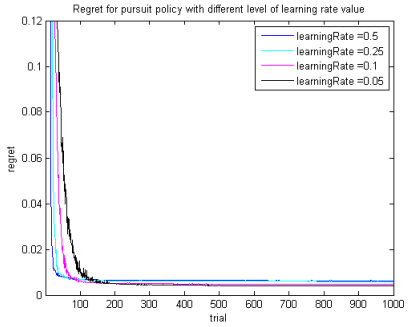
[12] Rajaraman, R. & Sastry, P.S. (1996) Finite time analysis of the pursuit algorithm for learning automata. *IEEE Transactions on Systems, Man, and cybernetics, -Part B: Cybernetics*. pp.590-598,1996



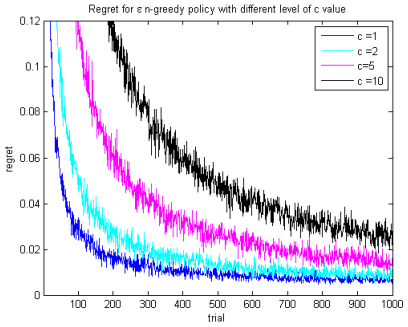
(a) ϵ-greedy



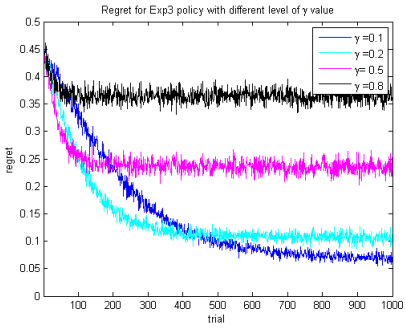
(b) Softmax



(c) Pursuit

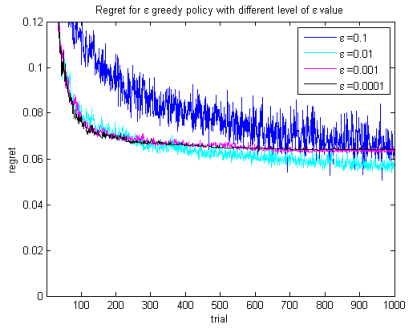


(d) ϵ_n-greedy

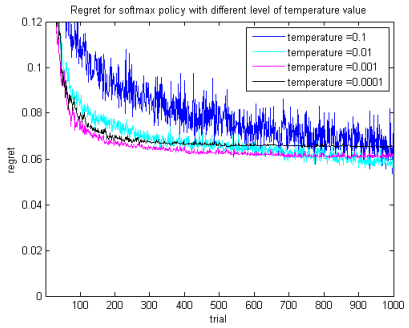


(e) Exp3

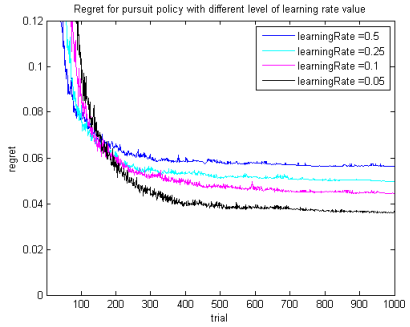
Figure 1: Performance of each bandit algorithm with different parameters (K=10, variance = 0.1)



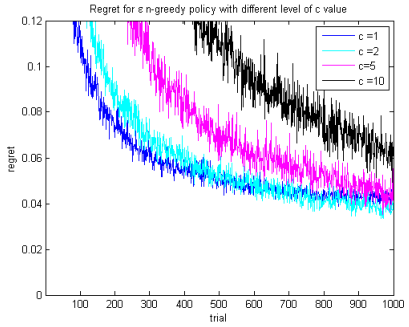
(a) ϵ-greedy



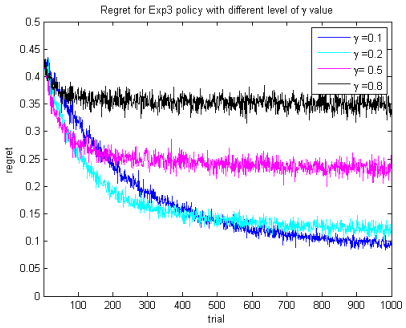
(b) Softmax



(c) Pursuit

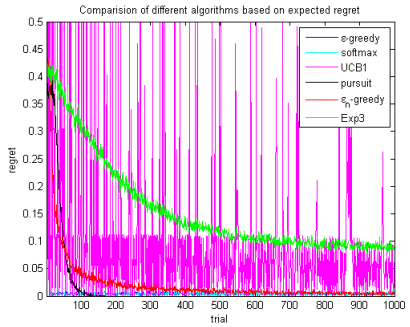


(d) ϵ_n-greedy

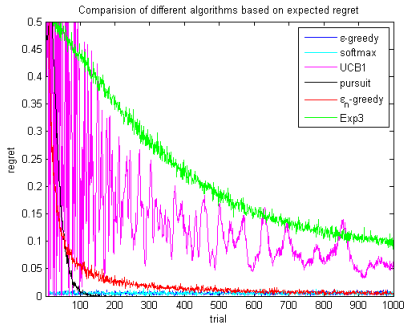


(e) Exp3

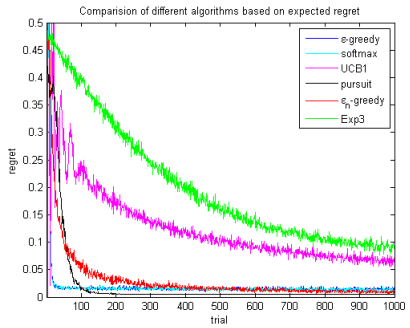
Figure 2: Performance of each bandit algorithm with different parameters (K=10,variance = 1)



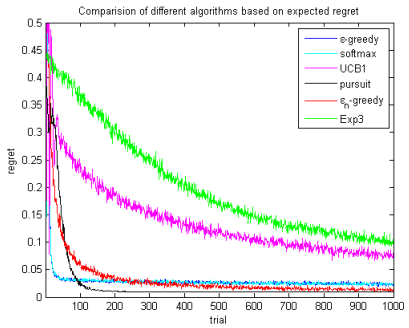
(a) k=5, variance=0.001



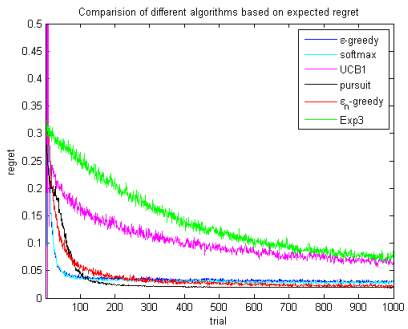
(b) k=5, variance=0.01



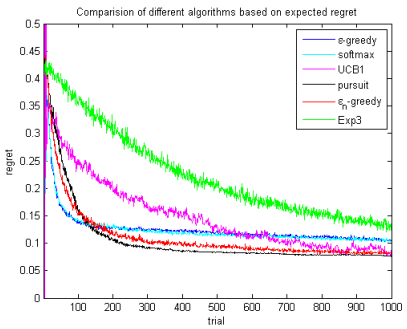
(c) k=5, variance=0.1



(d) k=5, variance=0.2

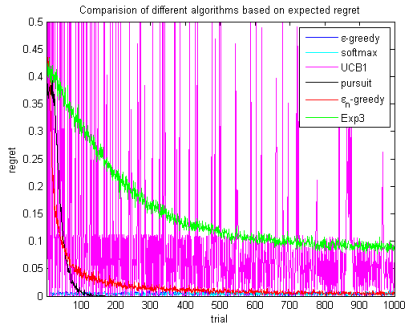


(e) k=5, variance=0.5

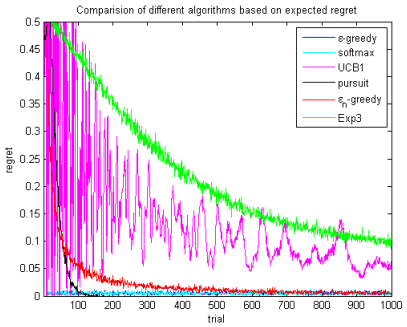


(f) k=5, variance=0.8

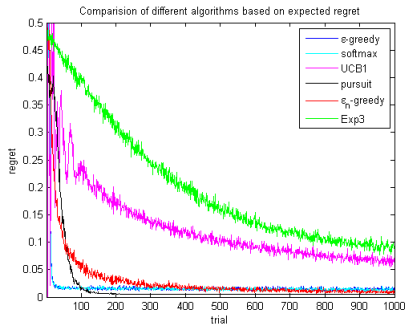
Figure 3: Comparison of different algorithms with K =2. For ϵ-greedy, set ϵ=0.01, for softmax, set learning rate as 0.01, for pursuit, set temperature as 0.05, for ϵ_n-greedy, set $c = 1$, for Exp3, set $\gamma = 0.1$



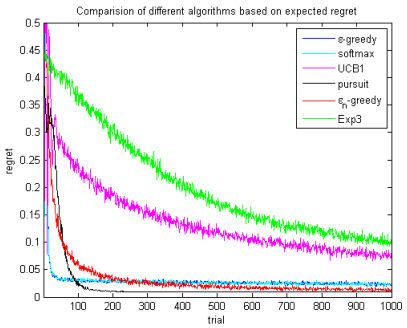
(a) k=10, variance=0.001



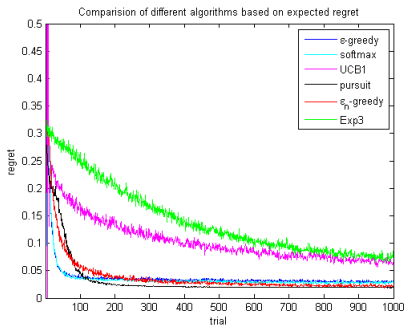
(b) k=10, variance=0.01



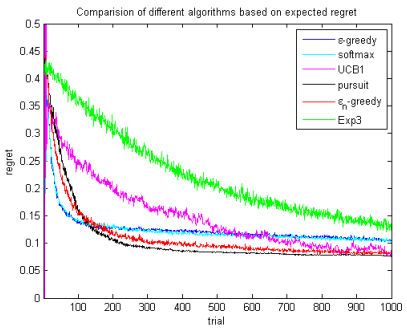
(c) k=10, variance=0.1



(d) k=10, variance=0.2

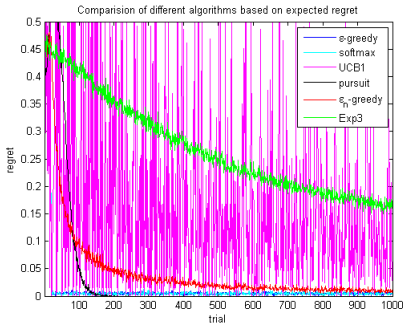


(e) k=10, variance=0.5

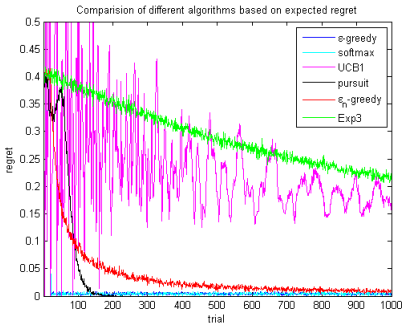


(f) k=10, variance=0.8

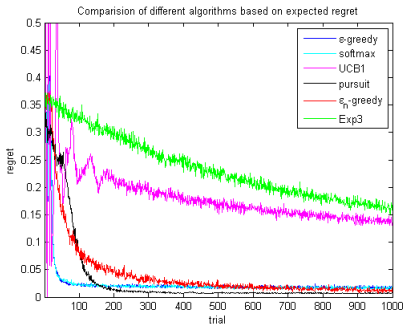
Figure 4: Comparison of different algorithms with K =10. For ϵ-greedy, set ϵ=0.01, for softmax, set learning rate as 0.01, for pursuit, set temperature as 0.05, for ϵ_n-greedy, set $c = 1$, for Exp3, set $\gamma = 0.1$



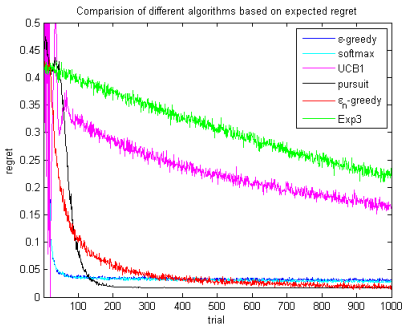
(a) k=20, variance=0.001



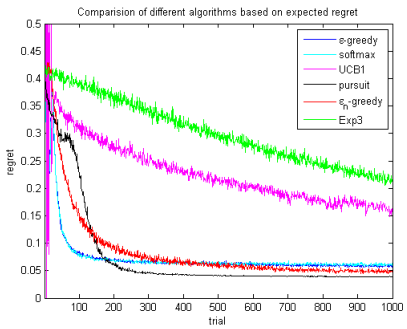
(b) k=20, variance=0.01



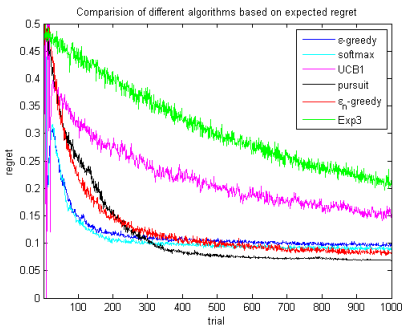
(c) k=20, variance=0.1



(d) k=20, variance=0.2

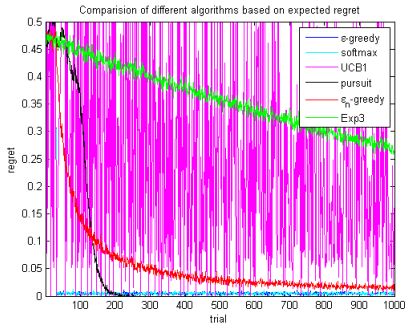


(e) k=20, variance=0.5

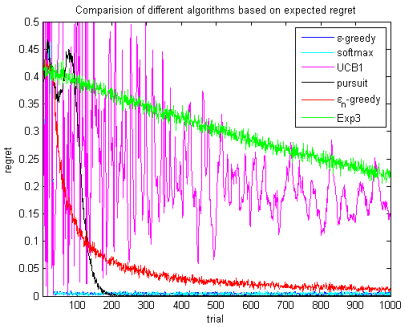


(f) k=20, variance=0.8

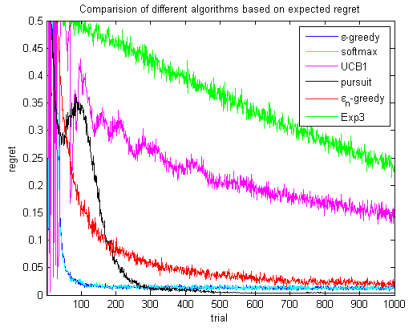
Figure 5: Comparison of different algorithms with K =20. For ϵ-greedy, set ϵ=0.01, for softmax, set learning rate as 0.01, for pursuit, set temperature as 0.05, for ϵ_n-greedy, set $c = 1$, for Exp3, set $\gamma = 0.1$



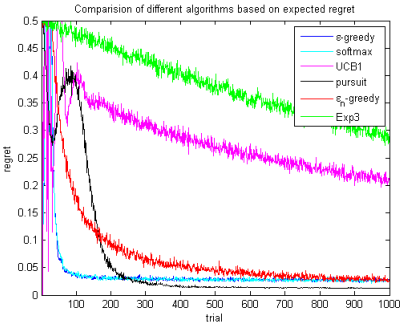
(a) k=30, variance=0.001



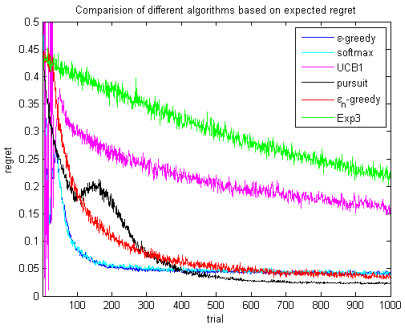
(b) k=30, variance=0.01



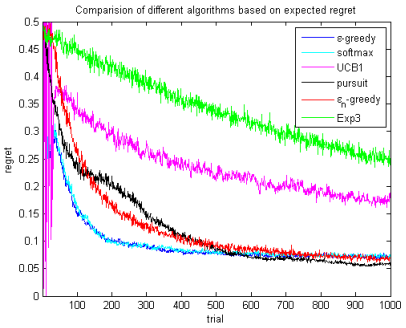
(c) k=30, variance=0.1



(d) k=30, variance=0.2

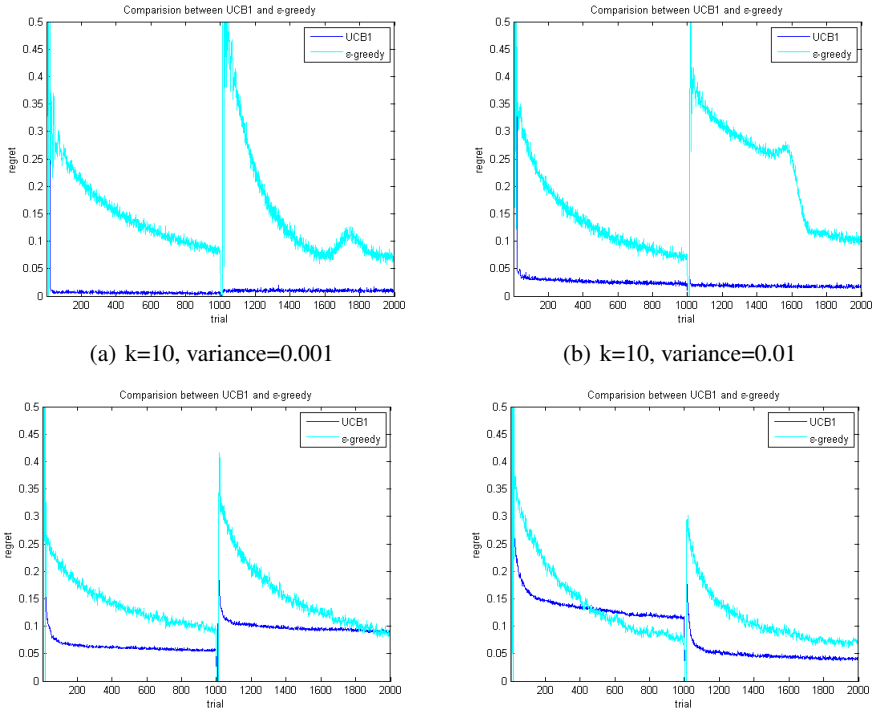


(e) k=30, variance=0.5



(f) k=30, variance=0.8

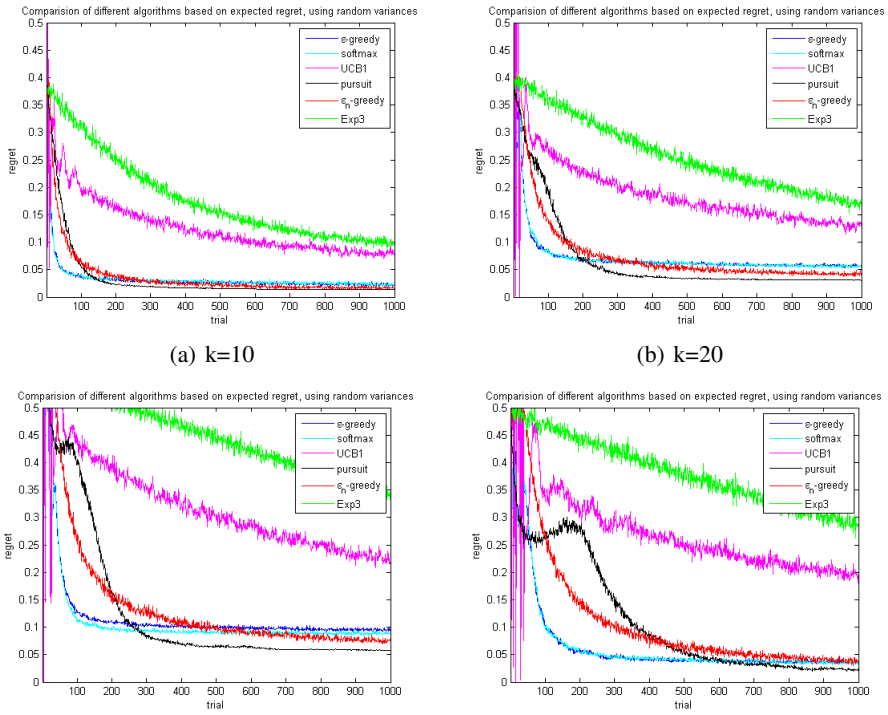
Figure 6: Comparison of different algorithms with K =30. For ϵ-greedy, set ϵ=0.01, for softmax, set learning rate as 0.01, for pursuit, set temperature as 0.05, for ϵ_n-greedy, set $c = 1$, for Exp3, set $\gamma = 0.1$



(a) k=10, variance=0.001 (b) k=10, variance=0.01

(c) k=10, variance=0.1 (d) k=10, variance=0.2

Figure 7: Comparison between UCB1 and ϵ-greedy algorithms for dynamic multi-bandit problem, where after 1000 trials, another set of arms are added into existing arms. Then the process repeat another 1000 trials



(a) k=10 (b) k=20

(c) k=30 (d) k=40

Figure 8: Comparison between different algorithms for different K values, with random variances