Multi Arm Bandits Basic Algorithms Comparison

Jia Lin Hau

Fall 2021

Abstract

In this project, we explore Multi Arm Bandits (MAB) algorithms and their performance in adversarial domain. According to the empirical result, we obtain a couple important intuitions: (1) Even a little randomization provide difficulty for adversarial to exploit. (2) Decrease exploration weight over-time benefit algorithms to exploit higher reward arms. To our surprise, an algorithms that is not popular UCB-SM, a combination of UCB and softmax outperform most algorithms in both expectation and worst case performance.

1 Introduction

Multi-armed bandits problem have been studied for nearly a century, it was first formally introduced by William R. Thompson in 1933 [3]. The problem was initially interested for the purpose of medical trail, later relate to decision making with uncertainty for multi-levers-operated slot machine ('bandit'). A bandit problem is a sequential game between agent and environment, algorithms in this problem refer to the decision of the agent given the history experience by the agent [1]. The objective of an agent is maximize reward or minimize regret. In Section 2, we will introduce the domain and algorithm we will be comparing in this paper. Section 3, we will explain the benefit of randomize policy in adversarial domain. We provide the empirical result of our experiment in section 4 and concluding remark at section 5.

2 Environment and Algorithms

Environment In this paper we consider adversarial bandit model, an adversary has power to examine the algorithms and choose the rewards according. We define adversarial bandit as a tuple $(\mathfrak{A}, \mathfrak{T}, \Xi, \mathfrak{R})$ where $a \in \mathfrak{A} = 1 : K$ refer to the set of arms, $t \in \mathfrak{T} = 1 : T$ refer to time horizon, Ξ refer to the adversarial constraint, $\mathfrak{R}$ refer to the reward space. In this paper, we consider a specific problem where $K = 4$, $t = 120$, $\mathfrak{R} = \{0, 1\}^{KT}$ Bernoulli reward, and 1000 samples of $\xi \in \mathfrak{R} : \sum_t r_{a,t} = \Xi_a$ as the only possible adversarial options.

We assume every arm has its fixed total reward $\sum_t r_{a,t} = \{24, 30, 40, 60\}$. The optimal action $a^\star$ is known to the adversary but not to the agent. Our objective is to minimize regret which is defined as:

$$\min_{\pi \in \Pi} \max_{\xi \in \Xi} \left\{ \max_{a^\star \in \mathfrak{A}} [\rho(a^\star, \xi)] - \rho(\mathbb{E}_{a \sim \pi}[a], \xi) \right\}$$

where $\rho(a, \xi)$ refer to the sum of reward over the horizon deploying action a given adversary option ξ.

Algorithms In this paper, we consider nine different algorithms within four categories (basic benchmark , UCB benchmark, Randomize algorithm, Softmax variant). Deterministic policy is marked with(*).

- basic benchmark (Explore-Commit*, Uniform Random)
- UCB benchmark (UCB Elimination*, UCB1*)
- Randomize algorithm (Epsilon Greedy, EXP3)
- Softmax variant (UCBSM, ESM, LCBSM)

For Explore-Commit, Epsilon Greedy, UCB Elimination and UCB1 we follow algorithm 1.(1,2,4,5) respectively from "Introduction to MAB" book [2] . For UCBSM , ESM , LCBSM uses softmax function on UCB, expectation and LCB to calculate probability of choosing such action. Uniform Random is just randomly selecting action regarding history and time step. We use a straight-forward EXP3 rather than the lose-based importance-weight EXP3 [1]. We provide our variant of UCBSM and EXP3 as algorithm 1 and 2.

Algorithm 1: UCBSM

Input: T, K, c, β

Initialize $R_i, N_i = 0, \forall i \in 1..K$;

for $t = 1...T$ **do**

  $W_{ti} = \frac{R_i}{N_i} + \sqrt{c \frac{\log(t)}{N_i}}, \forall i \in 1..K$;

  $P_{ti} = \frac{\exp \beta W_{ti}}{\sum_j \exp \beta W_{tj}}, \forall i \in 1..K$;

  Sample $a \sim P_t$ and observer reward X_t ;

  $R_a = R_a + X_t$;

  $N_a = N_a + 1$;

Hyper-parameter c for UCBSM determine the factor for the confidence level, small c will have low confidence. Hyper-parameter β is for the softmax function, higher β will

Algorithm 2: EXP3

Input: T, K, η, α

$W_{1i} = 1, \forall i \in 1..K$;

for $t = 1...T$ **do**

  $P_{ti} = \frac{\alpha}{K} + (1-\alpha) * \frac{W_{ti}}{\sum_j W_{tj}}, \forall i \in 1..K$;

  Sample $a \sim P_t$ and observer reward X_t ;

  $W_{ta} = W_{ta} * \exp \eta \frac{X_t}{P_{ta}}$;

allocate more probability to the biggest value, while smaller β will distribute probability more evenly between arms.

Hyper-parameter η for EXP3 use for the weights update, bigger η will result a drastic change of weights and probability. Hyper-parameter α is the randomize exploration probability the ideally α_t is decreasing which will reduce exploration over time. However, in our experiment we could not find a good function for α_t that yield better result than $\alpha = 0.2$, thus we will use it as constant.

3 Experiment and Empirical Result

Algorithms and their respective hyper-parameter selection		
Uniform Random		
Explore Commit	m = 11	
UCB Elimination	c = 0.07	
UCB1	c = 0.1	
Epsilon Greedy	$\epsilon = 0.15$	
Expected Reward Soft Max	$\beta = 12$	
EXP3	$\alpha = 0.2$	$\eta = 0.95$
UCBSM	$c = 0.12$	$\beta = 125$
LCBSM	$c = 0.2$	$\beta = 7.25$

To provide a fair hyper parameter selection for each algorithm, we provide equal number of options for each hyperparameter, evaluate each hyper parameter options in 1000 randomly generated test case and pick the hyperparameter that perform best on average.

3.1 Random Set Adversarial

In this section of the experiment, we randomly generate 1000ξ for our adversarial option set Ξ and evaluate each algorithms with their performance on these adversarial option sets.

From figure 1 we can see that UCB1 and UCBSM outperform all other algorithms on average in the 1000 random samples adversary set. However, when we looked at the worst

10% quantile in figure 2, UCB1 perform very badly because it does not randomize. UCBSM still outperform all other algorithms, followed closely by EXP3. The most relevant measure to our objective is the worst case measure figure 3, because we assume the adversarial has the option to pick the worst possible case $\xi \in \Xi$ given an algorithm. In the worst case measure, EXP3 outperform all other algorithms followed closely by UCBSM. Another notable scenario we can see from the plot is, all the deterministic algorithm worst case perform very poorly even in this randomly generated adversary option set, we will exploit deterministic algorithm further in next subsection (greedy adversarial).

3.2 Greedy Adversarial

Previously we explain the adversarial option is a set of randomly generated samples. In this section, we consider a smarter adversary. Given an algorithm, the adversary will see what the algorithm pick and then give that specific arm a reward of zero and other arm a reward of one unless it run out of budget $\sum_t r_{a,t} = \{24, 30, 40, 60\}$. For deterministic algorithm, this greedy adversary is efficient. For each deterministic algorithm, we change it's parameters and produce a number of non repetitive test case ξ as our adversary option set Ξ. For each randomize algorithms, we use the exact parameter that we will be using to evaluate the policy and generate multiple test cases and put it into the set Ξ. In this section, all adversary option is a greedy allocation of an instance. Deterministic algorithm will usually obtain no reward in the worst case. This implies randomizing is very important in a adversarial domain.

From figure 4 we can see that our UCBSM still outperform all algorithms and followed closely by EXP3. Even in expectation, all other algorithms does not deviate much from the result of just randomly selecting arms. However, when we look at worst case among these greedy generated test case figure 5, we see that EXP3 is the only algorithm that perform better than the uniform random benchmark. We can also see that, all deterministic algorithm were exploit by the adversarial and get exactly no reward at all.

4 Conclusion and Future Work

As conclusion, randomization is very important in adversarial domain. Even slightly randomization can be very beneficial, we can see it in the comparison among Explore-Commit, against Epsilon Greedy in the worst case figure 5. UCBSM and EXP3 the best performance algorithms, try to adjust the probability to balance out exploration and exploitation. When it first started, each arm has equal probability of being chosen. After some realization of reward given arm, the randomization will provide higher probability to visit arm that has higher estimate reward while still have some low probability to explore and try out other arm based on their estimate reward. UCBSM perform very well overall regarding the measure, and EXP3 perform well specifically when adversarial has much control (eg: worst case).

Future Work Due to time constraint, I was not able to implement all the algorithms that I wanted to. It would be interesting to see how Gitten's Index and softmax version of Gittin's Index perform. There are also a lot of UCB and EXP3/4 variant that would be interesting to implement and compare their performance.

Another direction of future work includes optimizing the adversary to provide a near optimal to the worst test case given randomize algorithm. Given an algorithm, the adversary could solve a MDP and provide the worst reactive test case.

References

[1] Tor Lattimore. *Bandit algorithms / Tor Lattimore (deepMind) and Csaba Szepesvari (University of Alberta).* Cambridge University Press, Cambridge, United Kingdom ;, 2020 - 2020.

[2] Aleksandrs Slivkins. Introduction to multi-armed bandits, 2021.

[3] William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

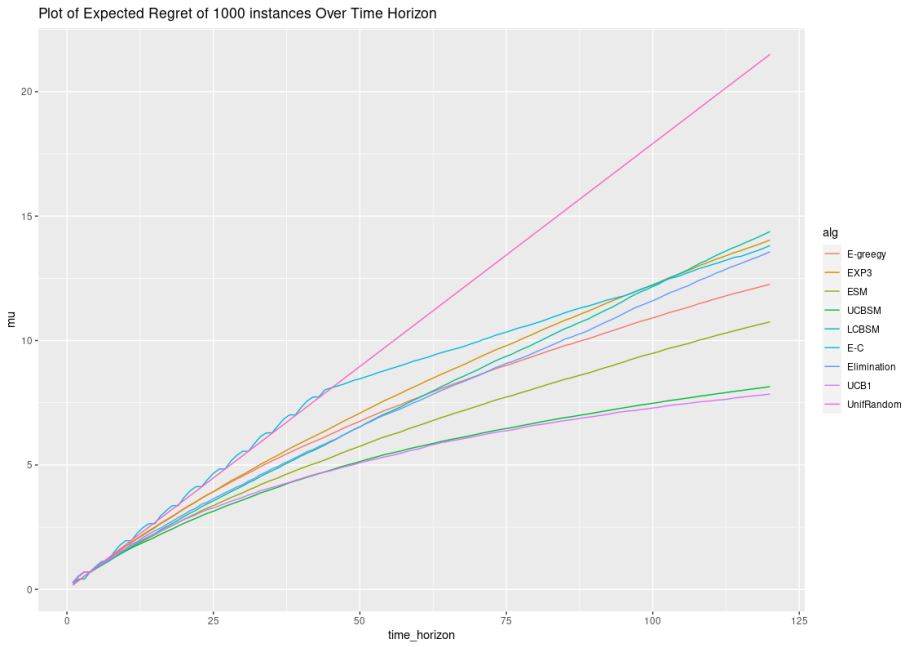

Figure 1: Mean Regret over Time horizon plot (Sampling Adversary Set)

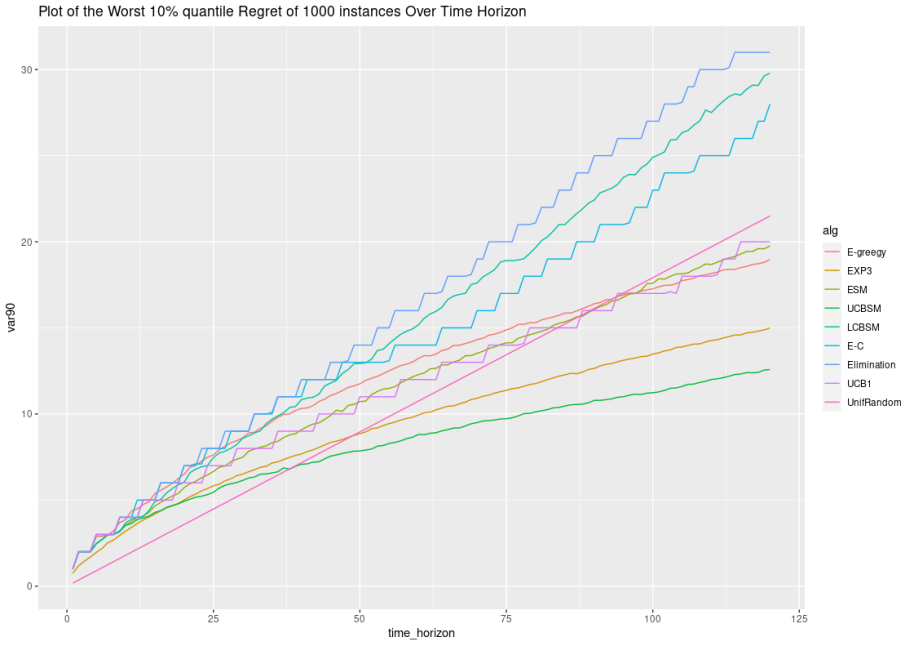

Figure 2: Worst 10% quantile over Time horizon plot (Sampling Adversary Set)

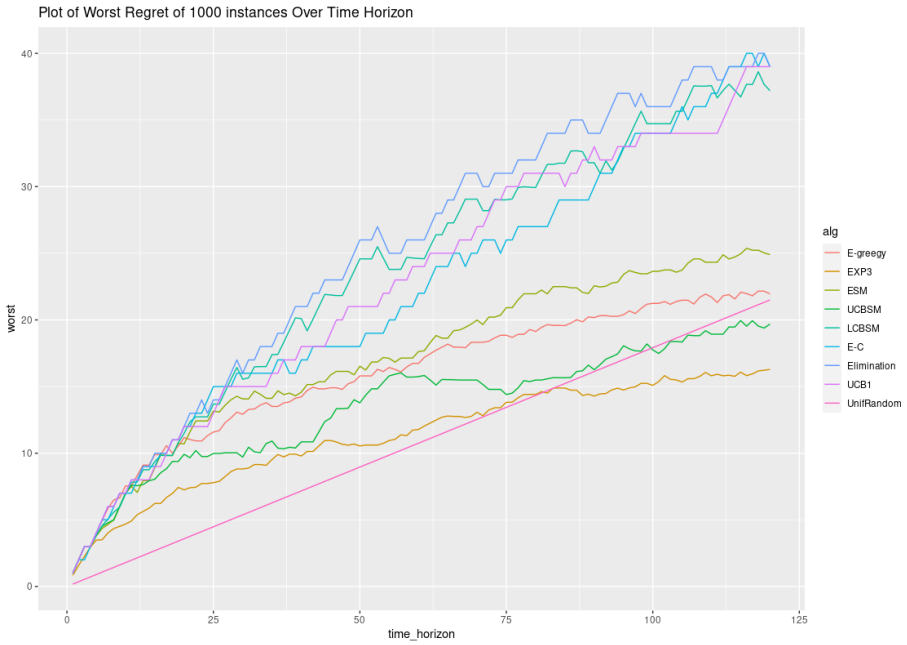

Figure 3: Worst Case Regret over Time horizon plot (Sampling Adversary Set)

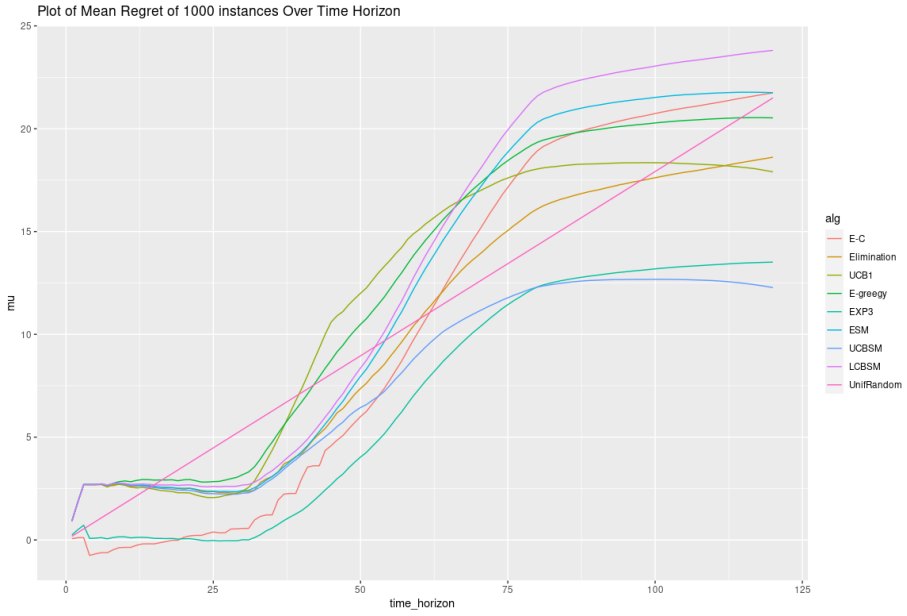

Figure 4: Mean Regret over Time horizon plot (Greedy Adversary Set)

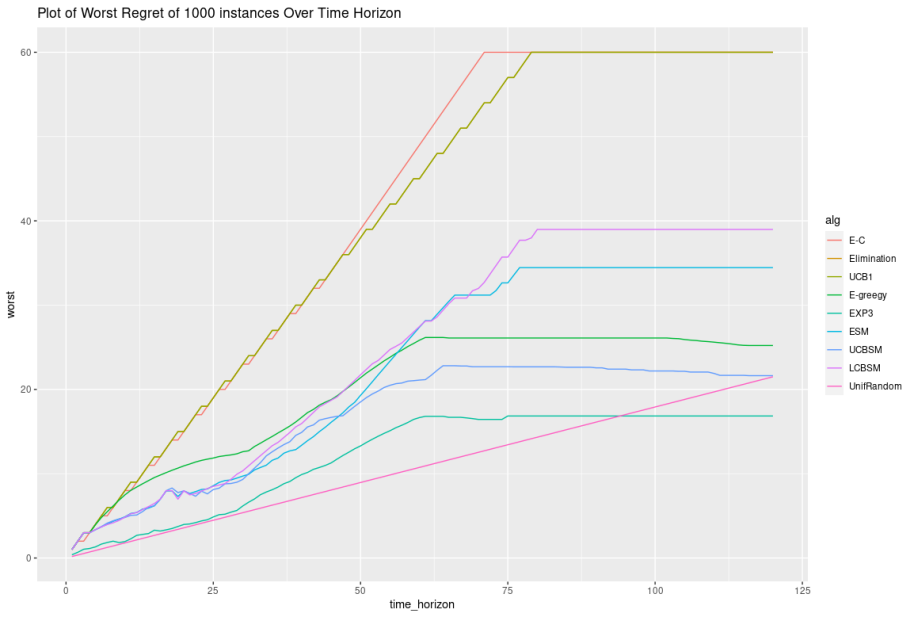

Figure 5: Worst Case Regret over Time horizon plot (Greedy Adversary Set)