Probability and Mathematical Statistics: Final Project #1

Due on December 21, 2022 at 11:59am

⓪

Name: **Zhu Zhelin,Guo Junxuan**
Student ID: 2021533077 2021533081

Contents

*

Abstract

The bandit problem is a problem of decision making in an uncertain environment, in which an agent repeatedly chooses among several options, or "arms", in order to maximize its total reward, while balancing the exploration and exploitation of the rewards of each arm. After one month's exploration, we try hard to work on bandit problems and eventually figure out the answer for all problems in both parts.

In part one, we start from the independent bandit problems,using the ϵ-greedy ,UCB algorithm and TS algorithm with different parameters respectively,to analyze the peformance,we introduce the regret,and find that the ϵ-greedy algorithm perform worst, UCB algorithm and TS algorithm achieves better result ,and when the prior distribution can provide some information ,the TS algorithm may perform better than UCB algorithm.Then we discuss the parameter in each algorithm, explore their role in exploration and exploitation,then without loss of generation,we discuss the dependent situation,considering to use exp3 algorithm to solve the adversarial bandit problem.

In part two,we consider that the reward will decay in γ^{t-1} way,under this circumstance,the intuition optimal solution may not work,so that we turn to the theory optimal solution first,finall solve this equation.

Keywords: bandit,ϵ-greedy,UCB,TS,exp3

Part 1 (Project : Performance Evaluation of Bandit Algorithms)

1.1:Problem 1

the theoretically maximized expectation of aggregate rewards over N time slots is each time we choose the arm with the greatest oracle value=$5000 \cdot 0.7 = 3500$

1.2:Problem 2

1. with epsilon-greedy,when $\epsilon = 0.1$,the probability of Θ is 0.6969,0.432,0.473 respectively,the total reward is 3406;when $\epsilon = 0.5$,the probability is 0.707,0.5210,0.395189,the total reward is 3111; when $\epsilon = 0.9$,the probability is 0.71302999 0.49331636 0.40602007,the total reward is 2761.
2. using UCB algorithm when c=1,the total reward 3453,when c=5,the total reward is 3132 c=10,the total reward is 2885
3. using Thompson algorithm ,when $(\alpha_1 = 1, \beta_1 = 1, \alpha_2 = 1, \beta_2 = 1, \alpha_3 = 1, \beta_3 = 1)$,the result is 3496;when$(\alpha_1 = 601, \beta_1 = 401, \alpha_2 = 401, \beta_2 = 301, \alpha_3 = 2, \alpha_4 = 3)$ the result is 3491

1.3:Problem 3

running N=5000,and using 200 times to simulate the expectation,we can get the answer

```
Results of 200 independent trials
Greedy 3411.695 epsilon=0.1
Greedy 3084.085 epsilon=0.5
Greedy 2744.025 epsilon=0.9
UCB 3409.815 c=1
UCB 2978.295 c=5
UCB 2826.165 c=10
TS 3484.48 a1, a2, a3, b1, b2, b3=1
TS 3494.445 a1=601, a2=401, a3=2, b1=401, b2=601, b3=3
```

1.4:Problem 4

To compute the gap between the algorithm and orcale value,and use the graph to plot.We can first see the comparision of total reward,and compare it to the oracle,ϵ greedy algorthim with $\epsilon = 0.05$,ucb algorithm with c=0.3,TS algorithm with $(\alpha_1 = 1, \beta_1 = 1, \alpha_2 = 1, \beta_2 = 1, \alpha_3 = 1, \beta_3 = 1)$ we can get the total reward 3426.2,3483.42,3483.5 respectively,and the regret is 73.3,16.58,16.5.

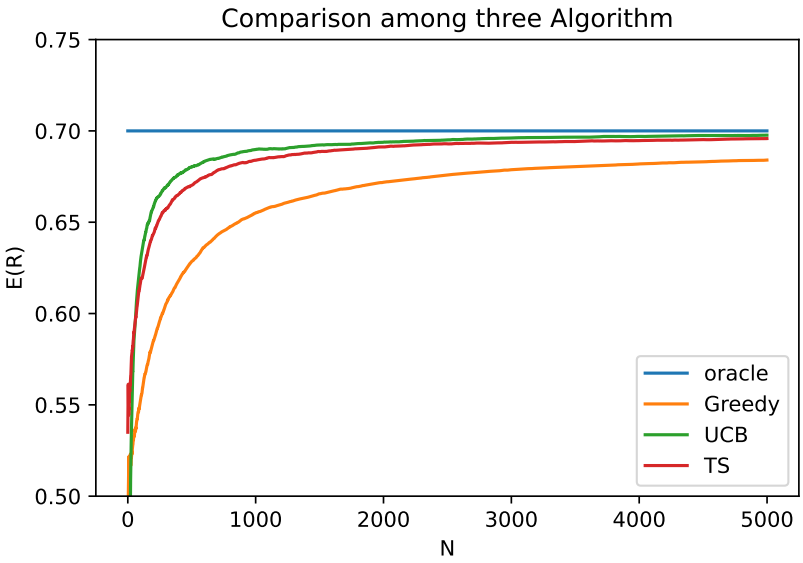


Figure 1: TS with parameters doesn't provide information well

Figure one compares the total reward between three algorithms,to make it more intuitive We can introduce the concept of regret,it is mostly used in bandit problem,defined by the gap between optimal choice ,we define it as G_T

$$G_T = \sum_{t=1}^{T} r_{opt} - r_{I(t)} = T * r^* - \sum r_{I(t)}$$

it is a function of T,using Asymptotic time complexity,we can get the gap,and find the ϵ-greedy algorithm with linear regret,while the UCB and TS algorithm with logarithmic asymptotic total regret,using the graph,we can see it intuitively.

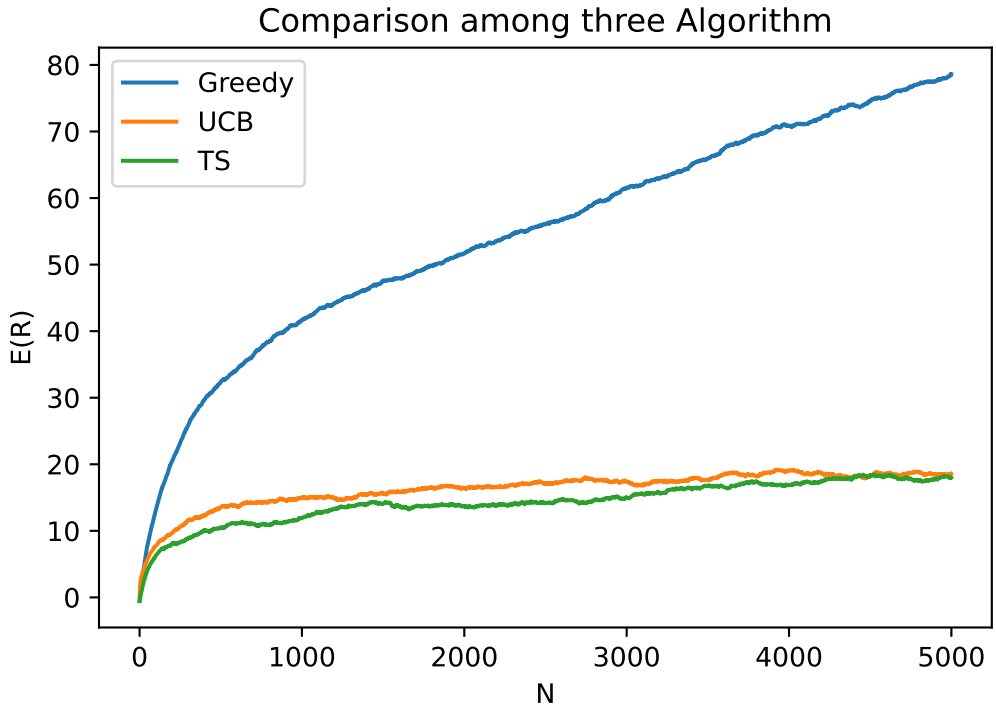


Figure 2: TS with parameters provides the information

from this picture,we can see the TS and UCB perform almost same,and the greedy algorithm perform worse than those two algorithm,with better prior parameter(which gives the information about the oracle probability),then TS algorithm perform well ,when the prior information cannot provide the information about oracle probability well,it performs worse than UCB algorithm,so when choosing $(\alpha_1 = 1, \beta_1 = 1, \alpha_2 = 1, \beta_2 = 1, \alpha_3 = 1, \beta_3 = 1)$,the UCB algorithm is best ,when choosing $(\alpha_1 = 601, \beta_1 = 401, \alpha_2 = 401, \beta_2 = 301, \alpha_3 = 2, \beta_3 = 3)$,the TS algorithm is best
After the experiment,we can discuss the impact of parameters.

(i) In the ϵ-greedy algorithm ,the ϵ represent the willing to explore,when ϵ is large,it is more likely to explore,otherwise,it will choose to use the optimal choice among explored instead of exploring.In practical ,we won't choose ϵ too small or too big,too small will result in local optimal solution,too large will result in too much waste in exploration.We can also improve this algorithm to $\epsilon - t$ algorithm,which change $\epsilon_t = \min\{1, \frac{c|A|}{d^2 t}\}$ where A represent the action,it will have logarithmic asymptotic total regret,while the origin has linear regret.

(ii) In UCB algorithm,c is important,it defines the width of upper bound ,when c is large ,the user is more likely to be an optimistic explorer,who tend to explore more,otherwise,more likely to stick to experience.

(iii) In TS algorithm,α_j, β_j is the prior "guess" about the oracle probability,if you guess well,which means the prior probability is positively relate to the real probability,it will work well,if it doesn't ,it will work worse.

After theory analysis,we will show our experiment and point out the best parameter.

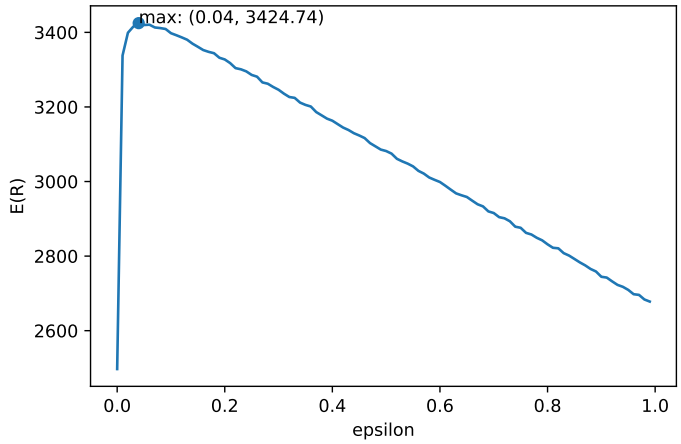


Figure 3: the epsilon algorithm with ϵ=0.04 best

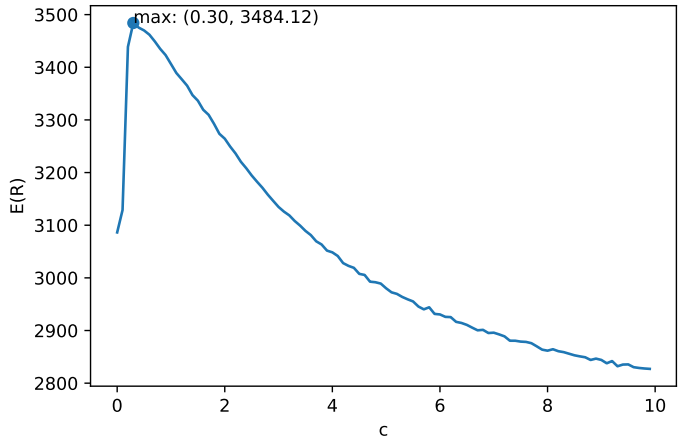


Figure 4: the UCB algorithm with c=0.3 best

At the end of this problem,I want to discuss the pros and cons of UCB and TS algorithm. The UCB algorithm,which doesn't rely on the prior distribution,may perform worse than TS algorithm which gives positively related prior probability distribution,but it is more robust,so to choose between the two algorithm,it depends on the request.

1.5:Problem 5

In bandit algorithms, the exploration-exploitation trade-off is a fundamental concept that is central to the problem being solved. The basic idea is that the agent must balance the need to explore the different options (arms) in order to gather information about which arm is the best, with the need to exploit the current best-known arm in order to obtain the highest rewards.

In the exploration phase, the agent selects arms that it has less knowledge about in order to learn more about them. This is useful when the agent does not have prior knowledge about which arm is the best, or when the environment changes over time. It allows the agent to gather information about different arms and estimate the expected rewards of each arm.
In the exploitation phase, the agent selects the arm that it currently believes to be the best in order to obtain the highest rewards. This is useful when the agent has a good estimate of which arm is the best, based on the information it has gathered through exploration.
The trade-off between exploration and exploitation is a balancing act, as the agent must weigh the short-term rewards of exploiting the best arm against the long-term rewards of exploring other options. If the agent spends too much time exploring, it will miss out on the rewards that could have been obtained by exploiting the best arm. If the agent spends too much time exploiting the best arm, it will miss out on the potential rewards that could be obtained by exploring other options.
Different bandit algorithms have different ways of balancing exploration and exploitation. For example, the epsilon-greedy algorithm uses a fixed probability of exploring, while the UCB algorithm uses confidence intervals to determine which arms to explore. Thompson Sampling is a Bayesian approach that samples from a probability distribution to determine which arm to select.
Ultimately, the choice of bandit algorithm and the specific approach to balancing exploration and exploitation will depend on the specific problem being solved and the constraints of the application.
To make it more specifically

(i) In ϵ greedy algorithm,the ϵ controls the exploration rate.

(ii) UCB (Upper Confidence Bound) algorithm is an exploration-exploitation algorithm that uses a different approach to balance exploration and exploitation. It works by maintaining a running estimate of the expected reward and the number of times each arm has been played, and using these estimates to compute a confidence interval for each arm. The algorithm selects the arm with the highest upper confidence bound at each step, which favors the arms that have been played less in order to explore more.

(iii) Thompson Sampling is a Bayesian approach for the multi-armed bandit problem. It works by maintaining a probability distribution over the rewards of each arm, and at each step, sampling from these distributions to select the arm with the highest expected reward. The algorithm updates the probability distributions based on the rewards received, and uses them to balance exploration and exploitation.

1.6:Problem 6

In bandit problems, we should be more concerned about whether the random bandit model can be used rather than whether the model is correct. The correctness of the model is usually represented by the prediction results, and the usability of the model is usually represented by the accuracy of the model's prediction results. Adversarial bandits do not assume how rewards are generated. We usually call the environment of adversarial bandits as the opponent. The goal is to use it to describe whether a strategy can be well opposed to the optimal action.
You can imagine playing a game with a friend, the game flow is as follows:
You tell your friend what action you are going to choose, the action is 1 or 2. Your friend secretly chooses reward $x_1 \in \{0, 1\}$ and $x_2 \in \{0, 1\}$ You use your strategy to choose action A_1 or A_2, and then get reward x_A The regret is equal to $R = max\{x_1, x_2\} - x_A$
Therefore, we can see that the difference between adversarial bandit and stochastic bandit is that the former's rewards are randomly given by the opponent (you can imagine a casino where the gambling machine can be manipulated by human).We can use exp3 algorithm to solve this problem. The algorithm can be described as follows:

1.Initialize the probability distribution over the actions $p_i = 1/K$ where K is the number of actions
2.For each round t = 1, 2, ... , T:

a Sample an action A_t according to the probability distribution p_t

b Observe the reward x_t for the action A_t

c Update the probability distribution p_{t+1} using the following rule:

$$p_{t+1,i} = \frac{p_{t,i} e^{\gamma x_{t,i}/p_{t,i}}}{\sum_{j=1}^{K} p_{t,j} e^{\gamma x_{t,j}/p_{t,j}}}$$

where γ is a parameter that controls the exploration-exploitation tradeoff.

The pseudo code of the Exp3 algorithm can be written as

Initialize: $p_i = 1/K$ for all $i \in \{1, 2, ..., K\}$
for $t = 1$ to T **do**
    Sample action $A_t \sim p_t$
    Observe reward x_t for action A_t
    for $i = 1$ to K **do**
        $p_{t+1,i} = \frac{p_{t,i} e^{\gamma x_{t,i}/p_{t,i}}}{\sum_{j=1}^{K} p_{t,j} e^{\gamma x_{t,j}/p_{t,j}}}$
    end for
end for

Algorithm 1: Exp3

the total regret is proven to be sublinear,or more specifically ,$O(\sqrt{nk})$

Part 2 (Performance Evaluation of Bandit Algorithms #2)

In the case where the reward distributions of the arms are dependent, it may be possible to design an algorithm that takes this dependency into account and uses it to obtain a better result. Here are a few possible approaches:
Modify the exploration-exploitation trade-off: In the independent case, the exploration-exploitation trade-off is often controlled by the "exploration rate", which determines the probability of selecting a random arm rather than the arm with the highest expected reward. In the dependent case, you could modify this exploration rate based on the current state of the system. For example, if you have just pulled an arm that had a high reward, you might want to increase the exploration rate to try and find another arm that is dependent on that arm and has an even higher reward.
Use model-based learning: One way to exploit information about the dependency between the arms is to build a model of the system and use it to predict the reward distribution of each arm. You could then use this model to guide your arm selection and optimize your expected reward.
Use online learning algorithms: There are various online learning algorithms that can be used to learn the dependency between the arms in a bandit problem. These algorithms typically update their estimates of the reward distributions of the arms based on the outcomes of past actions, and can be designed to take into account the dependency between the arms.
It's worth noting that designing an algorithm to solve the dependent bandit problem is generally more challenging than the independent case, and may require a more advanced and sophisticated approach.

2.1:Problem 1

We use python to make a simulation of this algorithm. In this simulation, we have two arms: the first one have the possibility of 0.7 to win the other have the possibility of 0.5 to win. We simulate different consequence with different k and plot a function image of $E(R)$ with respect to k ($\alpha_1 = \alpha_2 = \beta_1 = \beta_2 = 1$)

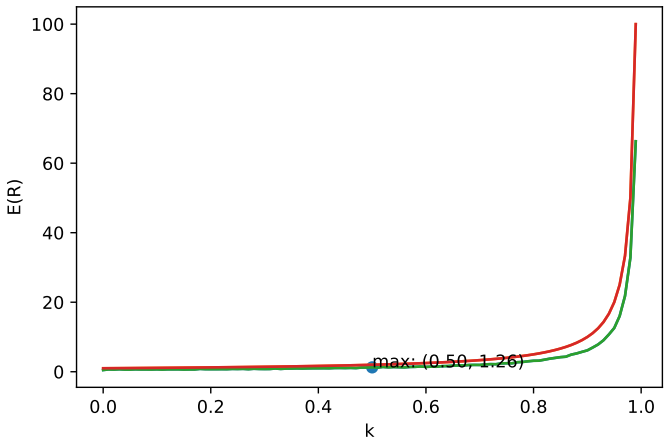


Figure 5: the simulation of E(R) when pull the largest θ_i
(Red line means the largest R, Green line means E(R) using TS)

2.2:Problem 2

We try the other two algorithm in Problem 1.2 and surprisingly found that The Greedy algorithm's reward is similar to TS algorithm after we make some revision(we revise the epsilon and let it be a function about t($\epsilon = 0.1^{t-1}$). And when γ is small, it may be better than Ts.
So we infer that the reason when Ts is not the best Algorithm is because this method is too slow to update. So we attempt to make Ts faster too, we let every trying add $\frac{1}{\gamma^2}$ instead 1.($\alpha_1 = \alpha_2 = \beta_1 = \beta_2 = 1$),however, it does not help

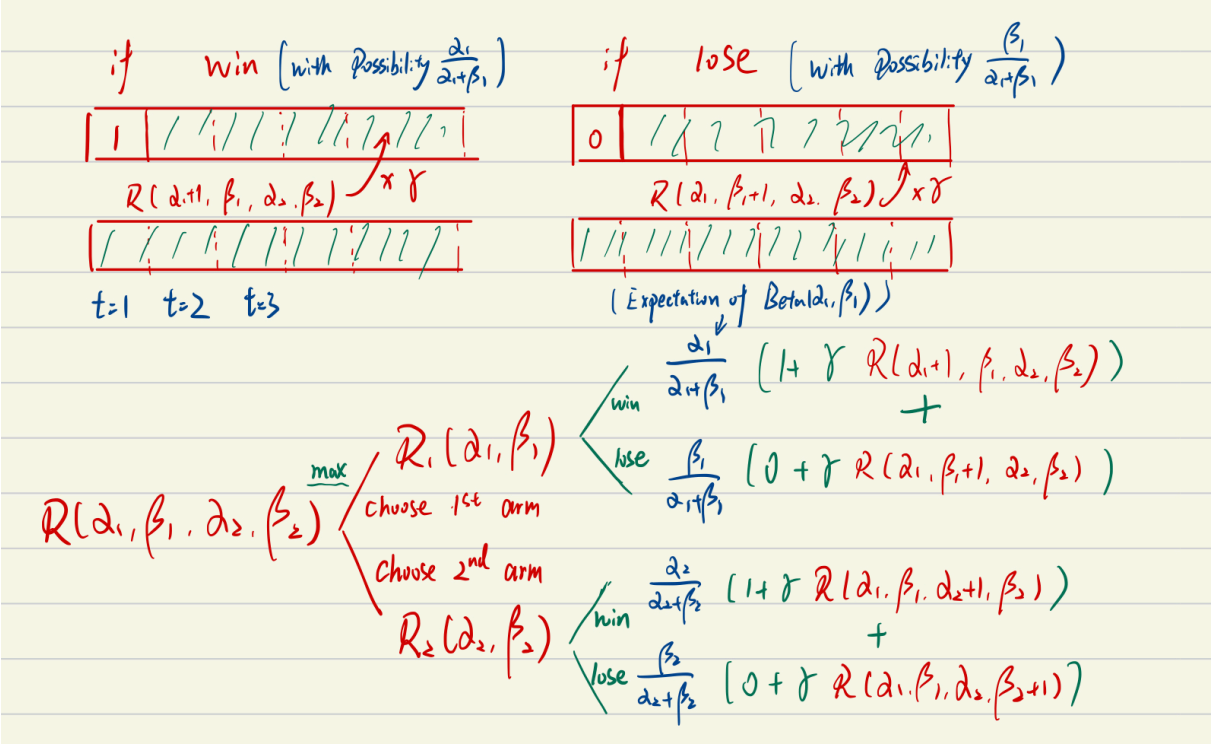
```
Greedy Algorithm:
E(R) is  6.759499825217127  when k= 0.9  and epsilon=0.1
Ts:
E(R) is  6.300978744730314  when k= 0.9 when a1, b1, a2, b2=1
```

```
Greedy Algorithm:
E(R) is  0.755959773091473  when k= 0.1  and epsilon=0.1
Ts:
E(R) is  0.6703302083715292  when k= 0.1 when a1, b1, a2, b2=1
```

```
Greedy Algorithm:
E(R) is  1.369380654859126  when k= 0.5  and epsilon=0.1
Ts:
E(R) is  1.2116922615769317  when k= 0.5 when a1, b1, a2, b2=1
```

2.3:Problem 3

$R(\alpha_1, \beta_1, \alpha_2, \beta_2)$ means the biggest total reward with the prior possibility of $\beta(\alpha_1, \beta_1)$ and $\beta(\alpha_2, \beta_2)$
And $R_1(\alpha_1, \beta_1)$ means the rewards if we choose the first arms
So it's obvious that $R(\alpha_1, \beta_1, \alpha_2, \beta_2) = max(R_1(\alpha_1, \beta_1), R_2(\alpha_2, \beta_2))$since our objective is to make the reward biggest
then we use a story to prove the equation is right



2.4:Problem 4

To solve it approximately,we can use depth-first search method to iterate,when α or β equal to limit ,we can assume the probability will not be updated ,then using the $\frac{1}{1-\gamma}$ $\max\{\frac{\alpha_1}{\alpha_1+\beta_1}, \frac{\alpha_2}{\alpha_2+\beta_2}\}$ to represent the reward, then return the total reward.

2.5:Problem 5

To solve this problem,we can use depth-first search or dynamic programming to solve this problem,in depth-first search,to find the greatest value ,we can use the dfs to predict the expected value,then each time choose the one with greater reward,after choosing the arm,we try to make a trial,using modified Thompson algorithm to update.After each trial,if win,$\alpha+ = 1/(k^2)$,if lose $\beta+ = 1/(k^2)$,the experiment shows that it will be greater.