DBA 5101：

Determining the Best Combination

Ke Ma, A0212524U
Mingzhe Xu, A0232022A
Nanhai Zhong, A0231953E
Xiao Liang, A0232007X

1. Introduction

1.1 Background

This project is based on a case study of Obama's fundraising campaign. In this project, we are required to find the best combination of "media" section and call-to-action "Button" from 24 combinations. We aim to find the one that results in the highest sign-up rate. Due to the budget limit, we are supposed to find the best combination/arm in the least amount of tries as possible less than one million.

1.2 Problem Statement

The multi-armed bandit problem is used in reinforcement learning. In this problem, an arm is chosen from 24 different combinations/arms and upon "pull" the arm, the learner receives a reward based on an invisible distribution of that arm. Through the algorithm designed, the best arm is identified and we will keep pulling this arm to get maximum expectation of the net rewards.

1.3 Exploration vs Exploitation

Exploration allows the learner to advance its current information about each step and consider more long-term benefits. Improving the estimation of combinations' reward to enable learners to make more informed decisions in the following steps. Exploitation, choosing the greedy action to get the maximum reward, exploits the learners' decided strategy. But being greedy with reward, it may lead to local optimal strategy. To avoid this problem, we adopt several algorithms and extend the exploration pulls to 1000 times. We expect that comparison between different algorithm and exploration section help to prompt robustness of the model.

2 Candidate Algorithms

2.1 E-greedy

Epsilon-Greedy is an algorithm to balance exploration and exploitation by choosing between exploration and exploitation randomly. The epsilon refers to the probability of choosing to explore. If the epsilon is larger than a certain threshold, a random arm will be selected. Otherwise, the arm that gives the largest return will be selected to continue the experiment. Normally, this algorithm exploits most of the time with a small chance of exploring.

2.2 Adaptive e-greedy

This method is based on the classic e-greedy but allows the value of E to vary in a controlled way throughout the execution. Changing the value of E is accomplished by performing an adaptive action triggered throughout the process.

2.3 UCB1

UCB – Upper Confidence Bound Algorithm, is an algorithm for the multi-armed bandit that achieves regret that grows only logarithmically with the number of actions taken. It is also dead-simple to implement, so good for constrained devices. The previous algorithms only care about the return, and do not care how many times each arm is pulled down, which meant that these algorithms would no longer select the very low arm with the initial return, even if the return of this arm is tested only once. Using the UCB1 algorithm will not only focus on returns, but also the number of times each arm is explored.

Let $n_a(t)$ denote the number of times arm a has been chosen; Let $\hat{\mu}_a(t)$ be the estimated mean of arm a at round t; Donate the UCB index of arm *a* as following:

$$\hat{\mu}_a(t) + \sqrt{\frac{2\ln(t)}{n_a(t)}}$$

3 Model Analysis

3.1 Overall Strategy

Specifically, we run each of the algorithm 1000 times (e-greedy algorithm 2000 times for e = 0.1 and e = 0.01 respectively), calculate the regret, net rewards (sum of rewards from all pulls), average reward per pull, compare these indicators and best arm picked by each algorithm. Finally, the best algorithm is identified and we exploit on the best arm picked by that algorithm for the rest of the budget.

3.2 Algorithm Comparison

a. Differences in principles

For all the three methods mentioned above, they all take the possibility of choosing the locally optimal arm into account and deal with this mistake by keep exploring with the introduction of probability or UCB index. However, the probability in e-greedy is constant, while the probability in adaptive e-greedy and the UCB index are both changing with the times of pulls. It means that when information becomes more, e-greedy doesn't change the strategy of explore and exploit. But for the other two methods, with the increase of times of pulls, they pay more attention to exploitation and less attention to exploration.

b. Dealing with algorithm randomness of finding the best arm

Among those three methods, adaptive e-greedy has the most randomness in finding the best arm and UCB has the least.

If we use the same algorithm to solve the same multi-armed bandit problem for several times, it is very likely for adaptive e-greedy to gain different results on which one is the best arm. The reason is the correlation between the result and the arm selected in the first few rounds. At the beginning, the probabilities of choosing each arm are very similar. However, with the increase of times of pulls, the probability of choosing the current best arm is increasing. It means that if we get stuck at a locally best arm, the probability of correcting the mistake is getting lower and lower.

But, for e-greedy, since the *e* is constant, the probability of correcting the mistake is small but won't change. In addition, by setting a higher, we could get higher chance to detect the globally best arm rather than the local best arm, which unfortunately may cause some loss.
For UCB, it plays each arm once at beginning, which eliminate the uncertainty of choosing in the first few rounds. In the following process, the index is related to the average rewards of each arm. That means not only the highest average reward but also the others are all been considered when making decisions. Therefore, in most cases, UCB always finds the same best arm.

4 Results

pull function input:

pull ('user7','XTkjktIc', arm)

pull function output:

e.g., {'Arm': '17', 'NetReward': 455, 'Pull': 30, 'Reward': 22}.

	UCB1	E-greedy e = 0.1	E-greedy e = 0.01	Adaptive E-greedy
number of pulls	1000	1000	1000	1000
best arm	17	17	21	11
average rewards of the best arm	24.50	24.51	23.21	18.33
total rewards	24210	22482	19502	17478
regret	286.34	2032.91	3708.69	849.76

Now, we have already used 4000 chances and the total rewards are 83672. Then, we are going to choose the best arm from the current results.

Firstly, the best arms in UCB1 and e-greedy when are both Arm 17, while the best arm is Arm 21 in e-greedy when and is Arm 11 in Adaptive e-greedy. Considering the average rewards of each best arm at the same time, we could find that Arm 17 has similar results in the two methods and is better than Arm 21 and Arm 11.

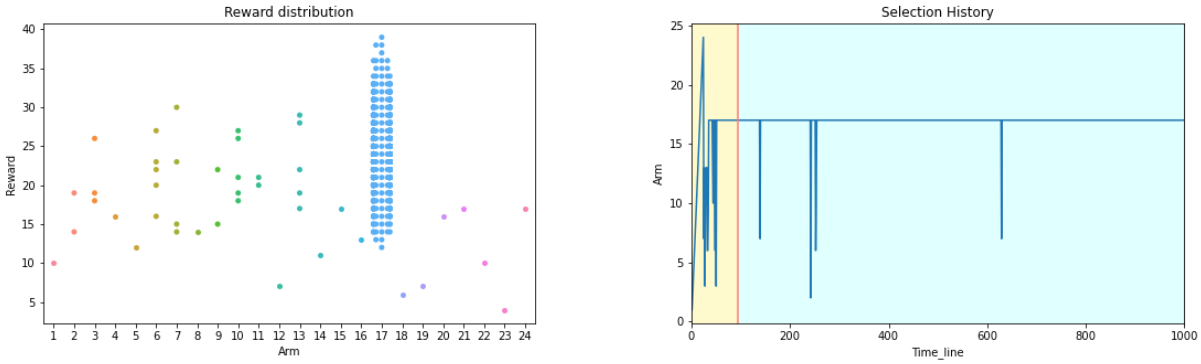


Figure 4.1 **UCB1 algorithm results** Plot on the left shows the distribution of all pull rewards through arms, bule points aggregated in arm 17 shows that arm 17 is pulled mostly and the mean rewards also is at the top of all arms. Plot on the right displays the history of all pulls in time series (1 – 1000 times). Yellow areas denote the exploratory stage while blue area denotes exploitation stage. Arm 17 was chosen the most while other arms are also chosen seldomly by algorithm.

Secondly, comparing the total rewards among each method, we could know that UCB1 and e-greedy have higher rewards than the others, which means they performs better in this case.

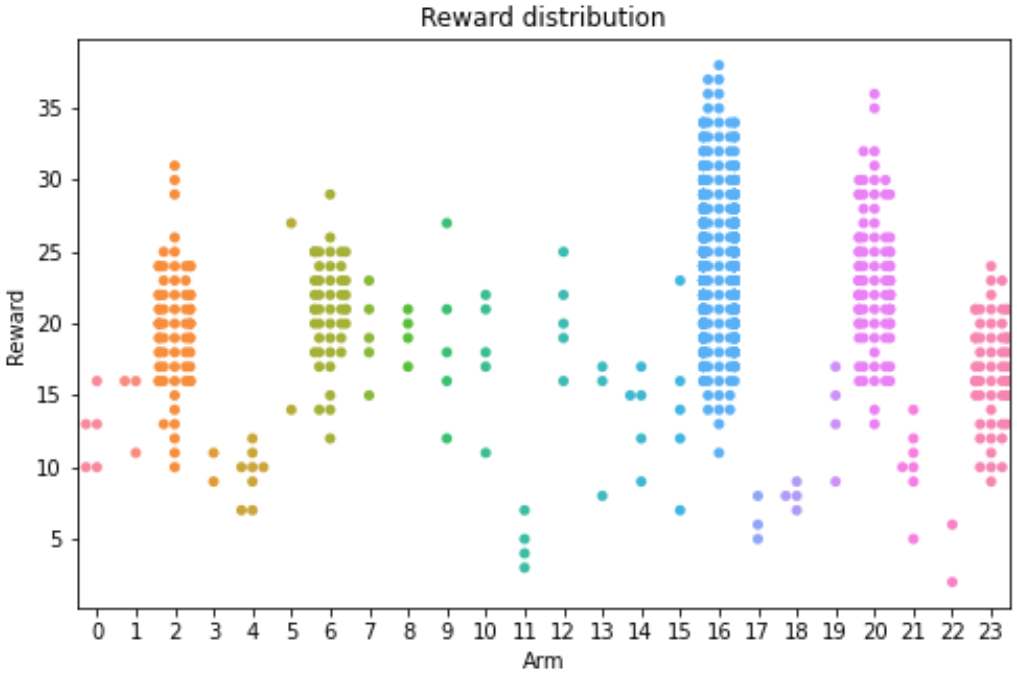
In summary, we select Arm 17 as the best arm and keep pulling Arm 17 to complete the remaining 996,000 attempts.

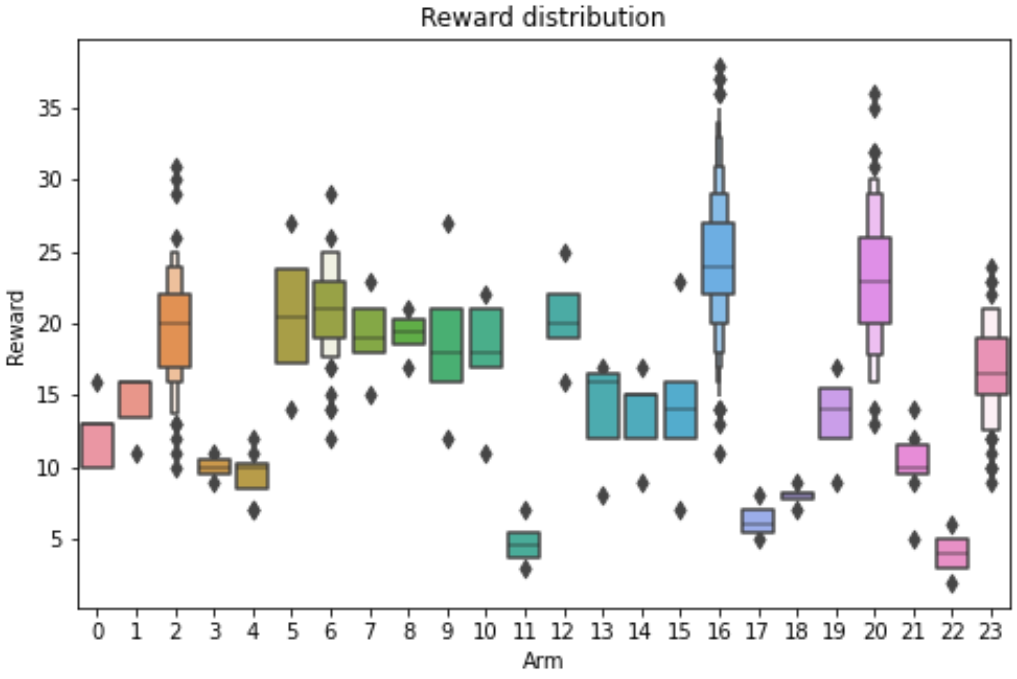
Reference

1. https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/

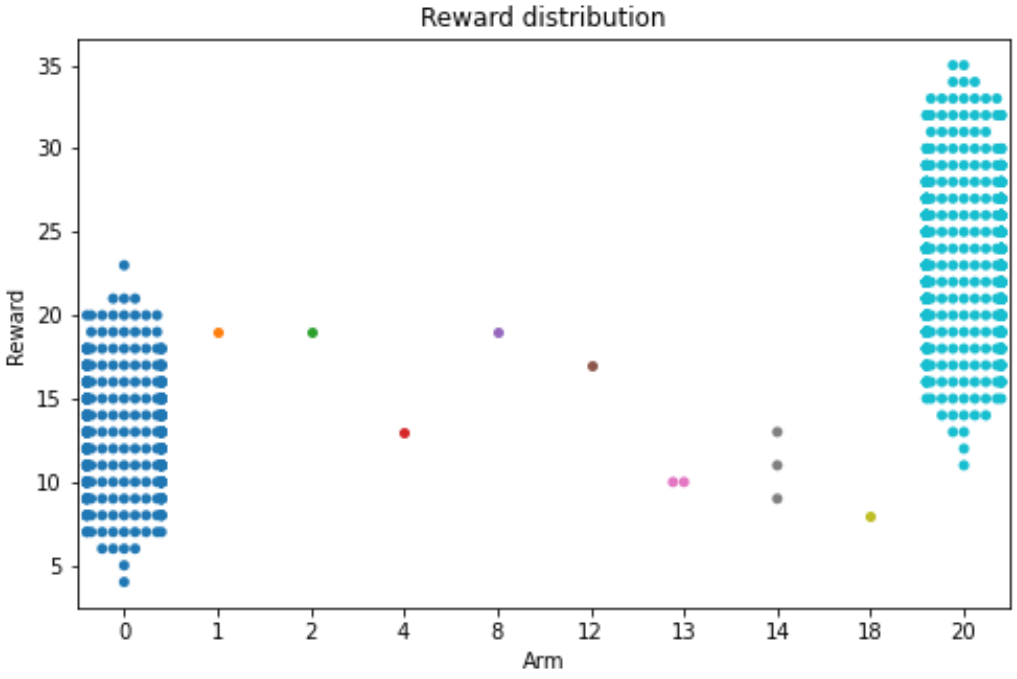
Appendix I: E-greedy algorithm results

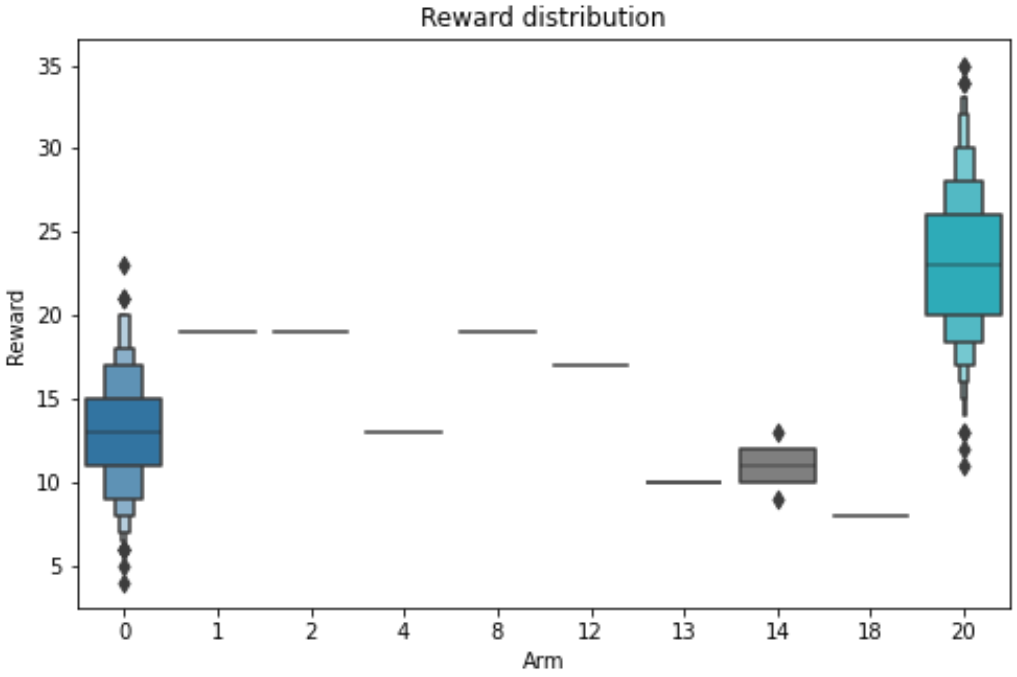
e = 0.1



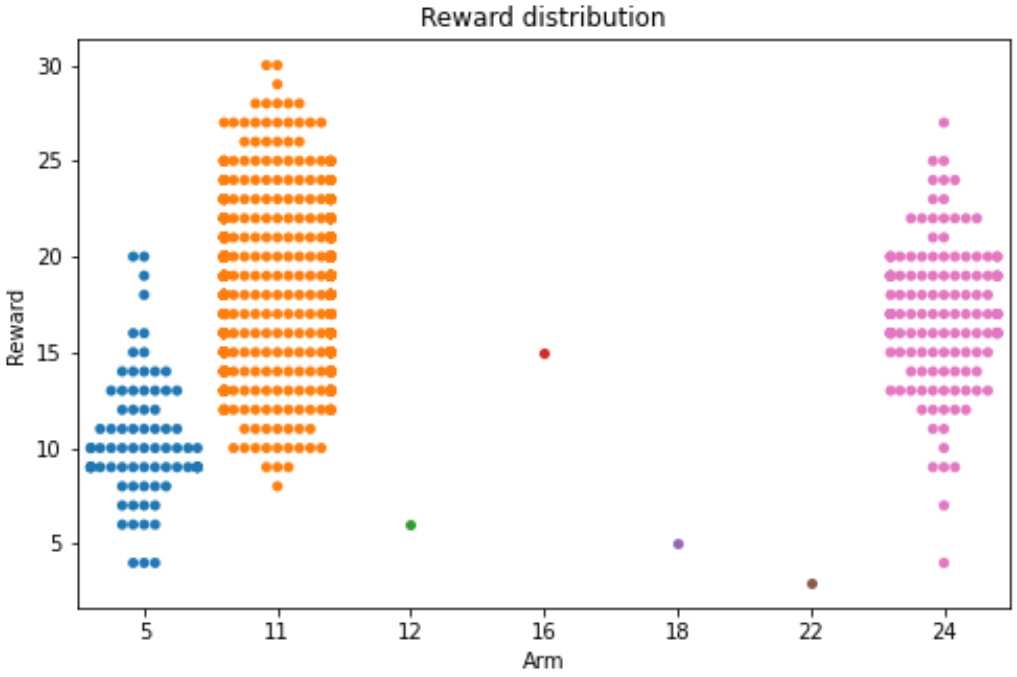

Reward distribution
Reward
Arm
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
5 10 15 20 25 30 35

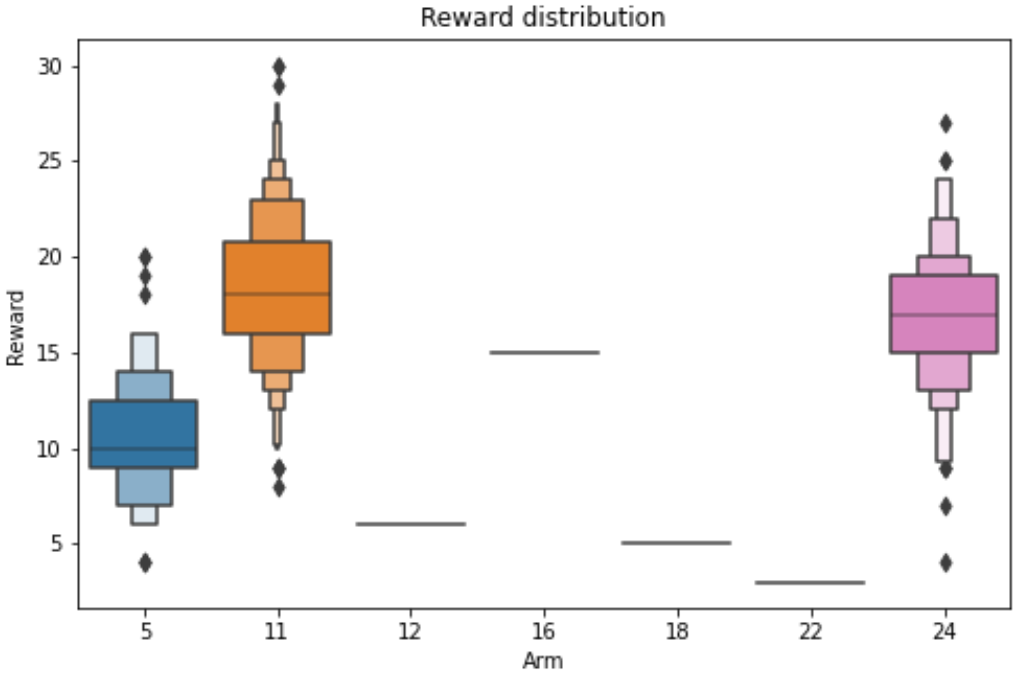
e = 0.01


Reward distribution
Reward
Arm
0 1 2 4 8 12 13 14 18 20
5 10 15 20 25 30 35

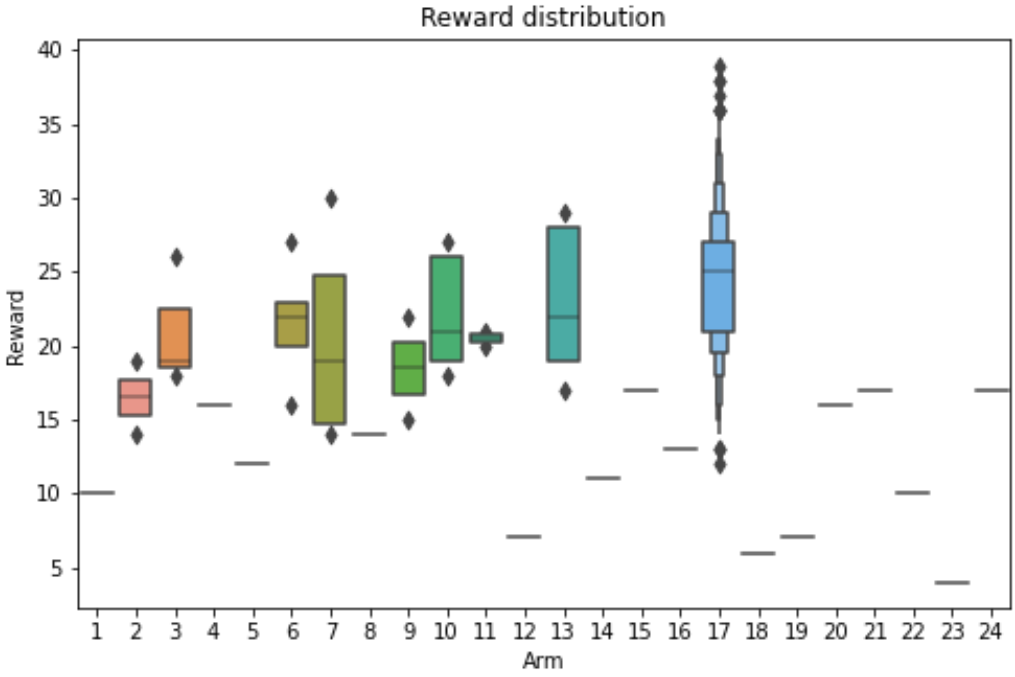

Reward distribution
Reward
35
30
25
20
15
10
5
0
1
2
4
8
12
13
14
18
20
Arm

Appendix II: Adaptive e-greedy





Appendix III: UCB1


Reward distribution
Reward
Arm
40
35
30
25
20
15
10
5
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24