108026511 李芳妤

Homework 1: Muti-Armed Bandit Problem

Implementation

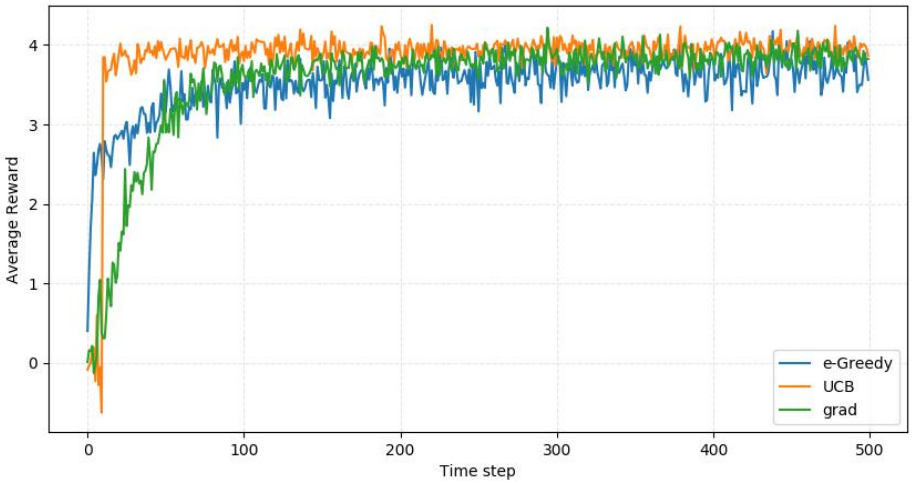
1) In ε-Greedy, how do you select action if the probabilities are equal?
   A:If probabilities of choices are equal, I will use np.random.random() to choose the action randomly. For examples, In my codes, if epsilon is 0.1, the function I use will choose a number from [0,1). if the number is larger than 0.1, the program will choose to do greedy algorithm. Otherwise, it will execute a random action.
2) In UCB, how do you select action when time steps < num of bandits?
   A:I set every bandits should be chosen at least once, then I will follow the equation in the textbook to choose the action.

3) Briefly describe your implementation.
   A:I randomly choose a number from [0,1), if the number is larger than epsilon, the program execute the greedy algorithm(and the greedy algorithm will choose an action that have the most average rewards). Otherwise, the program will randomly select a interger from (0, self._nb).After that the program update self._Q[action] and self._action_N[action].

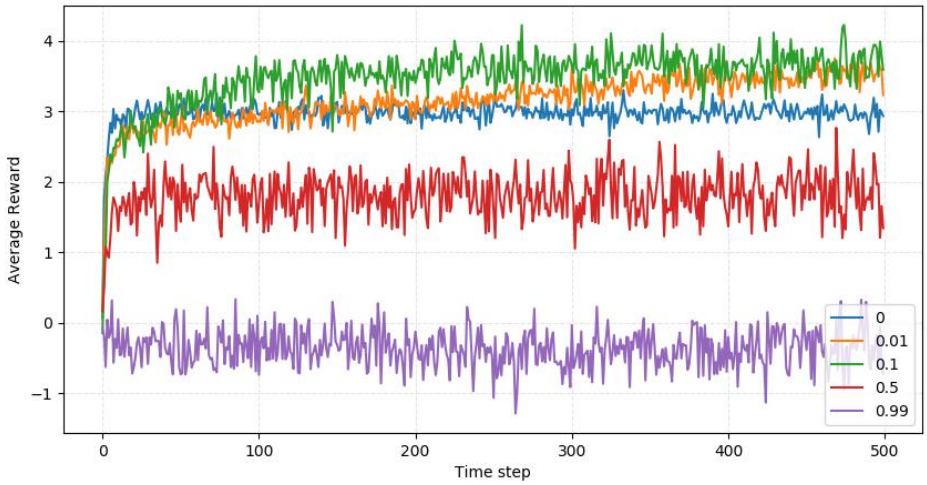
   If the sum of times I choose the arms is smaller than the number of bandits, then the program randomly choose the number between [0,self._nb) and map into the action, if the action has operated then the program choose another action which hasn't been operated and operate it. After that, we follow the equation in the textbook to choose the action. And the program update self._Q[action] and self._action_N[action].

- Experiments and Analysis

1) Plot the average reward curves of different methods into a figure.



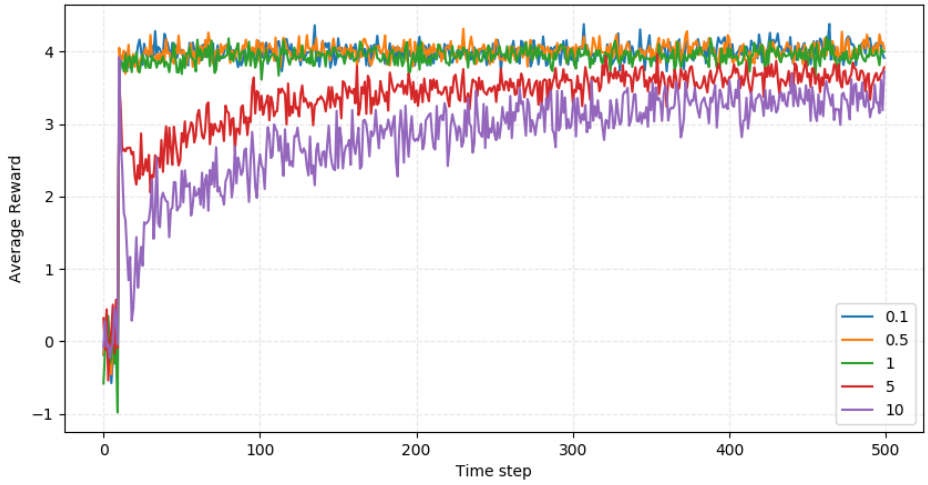
2) Vary epsilon value with 0, 0.01, 0.1, 0.5 and 0.99. What happens? Why? Please plot it.



When epsilon is 0, it only choose to exploit, so the average reward stop rise.
When epsilon increases to 0.01, we have some probability to explore for better action, so the average reward will also increases.
However, when epsilon increases to 0.1, we have more probability to explore for better action than epsilon is 0.01 , so the average reward will be better than epsilon is 0.01. Hence I consider 0.01 might be suitable for ten bandits.
Now we add epsilon to 0.5, though we have more probability to explore other actions, but this will waste too much time on exploration, so the average reward will lower than smaller epsilon.

Then we add epsilon to 0.99, its means that we have probability 0.99 to explore another action, this will waste the most of the time, so the average reward is the lowest.

3) Vary the parameter c in UCB. What happens? Why? Please plot it.

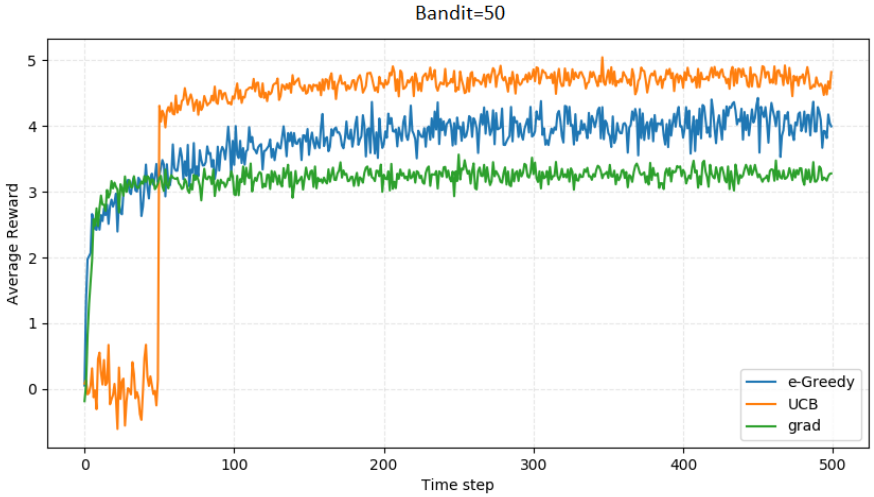


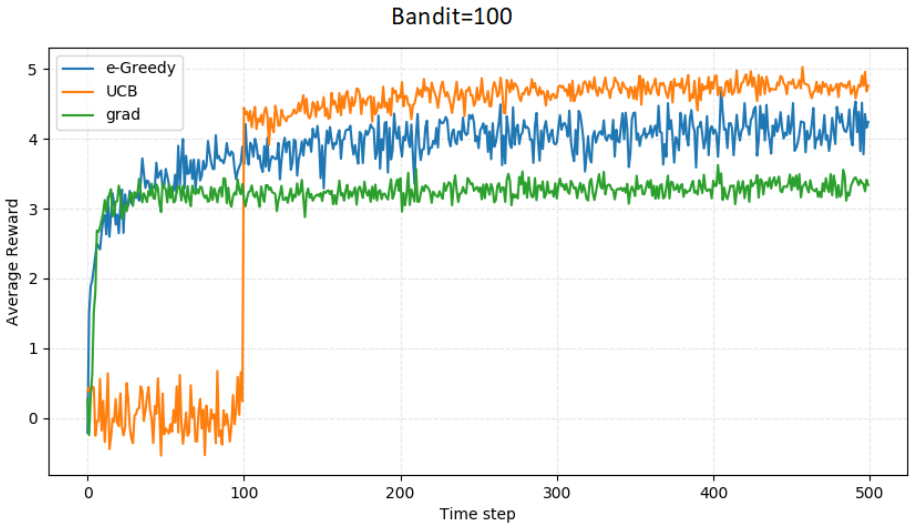
The average reward performs similar when c is 0.1, 0.5, 1. Because in the early stage the program would explore every bandits once, so the average reward in that period is lower for all c. After that, every action has been execute for at least once, now the program will choose action that will satisfy np.argmax(self._Q + self._c * np.sqrt(np.log(t) / self._action_N)) .

And when c=5 , the program will waste some time to exploration, so the average reward will lower than when c=1, 0.5, 0.1

So when c=10 , the program will waste a lot of time to exploration, so the average reward is the lowest.

4) Vary the number of bandits. What happens if the number of bandits is large?Please plot it.





In the early period, the average reward of UCB is not ideal, because UCB should exploration every action once, this will waste some time, so the average reward is lower in the early period.

And 100 actions cost more time than 50 actions, so the average reward of 50 actions rise earlier than the average reward of 100 actions.

The average rewards of another two algorithms are similar when the number of bandits is different.