EE551000 System Theory

Homework 1 Report: Multi-Armed Bandit

周柏宇 105061110

Implementation

✓ In ε-Greedy, how do you select action if the probabilities are equal?

I just use np.argmax() to select the action. As documentation of np.argmax() says, "In case of multiple occurrences of the maximum values, the indices corresponding to the first occurrence are returned." In terms of numerical computation, it is very unlikely to have float numbers being exactly the same. Therefore, any ties breaking solution should not affect too much of the algorithm performance.

✓ In UCB, how do you select action when time steps < number of bandits?

According to textbook page28, it says as follows, "If $N_t(a) = 0$, then a is considered to be a maximizing action." It means that when time steps< number of bandits, we will immediately choose these actions that have not been tried before.

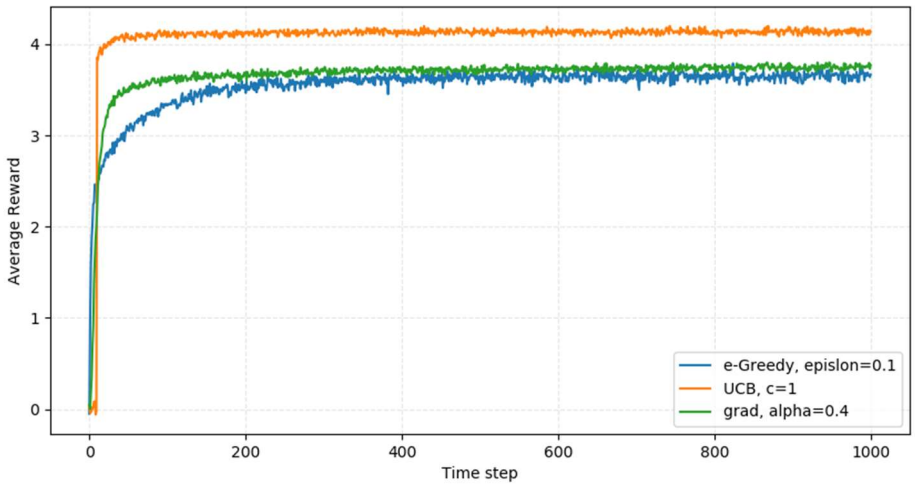
✓ Briefly describe your implementation.

In every experiment, we initialize a new environment, i.e. a new reward model parameter. Every time step in an experiment we do things as follows: first choose an action, then we get an immediate reward from the environment. Based on the reward, we will update our Q table for ε-Greedy method and UCB or H table for gradient bandit algorithms. Finally we store the reward for this time step so that we can calculate the average rewards over all experiments.

We use Incremental Implementation to update Q table in ε-Greedy method and UCB and the $\bar{R}_t$ in gradient bandit algorithms to improve memory efficiency.

Experiments and Analysis

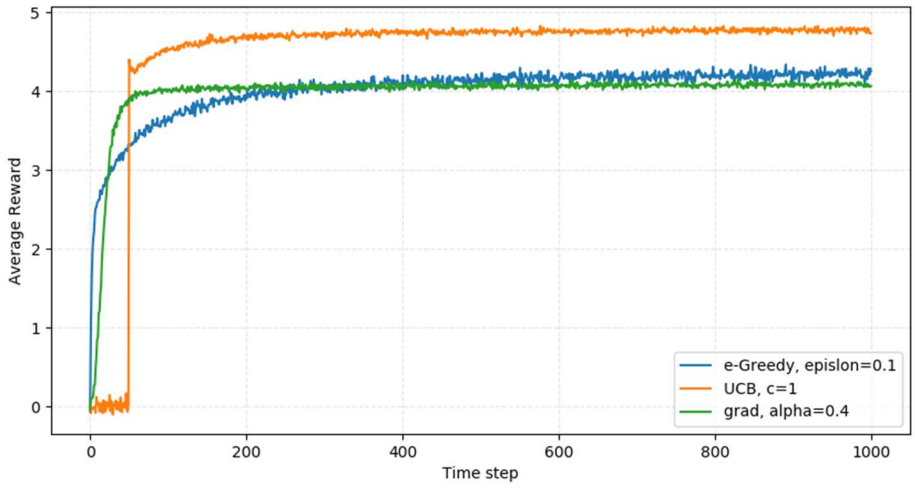
✓ Plot the average reward curves of different methods into a figure.



All results in this report is the average of 2000 random experiments. Without specification, the number of bandits=10 and the action value are initialized with 0.

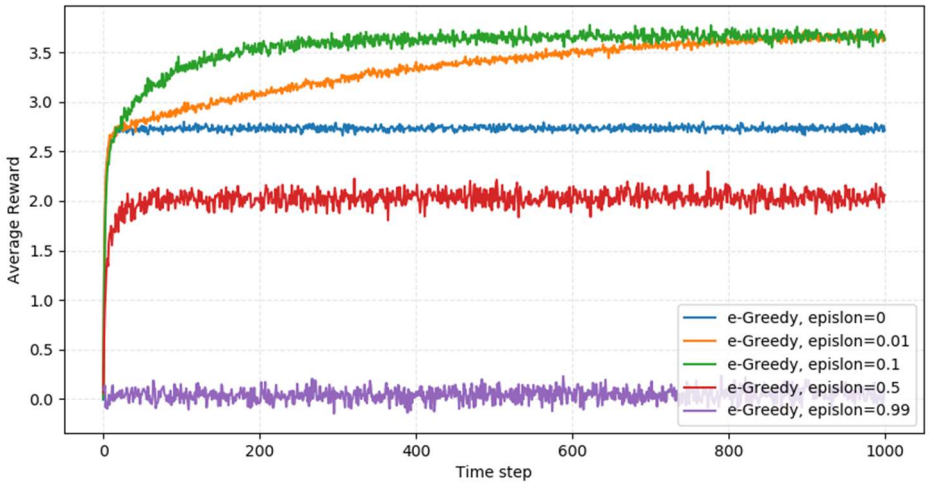
Qualitatively speaking, UCB is better than ϵ-Greedy because it adopts a more intelligent way of exploring actions; it selects actions that are less selected rather than randomly selects an action. The action chosen using UCB still strongly depends on the action value, which guarantees the rewards.

The gradient bandit algorithms are different from the action-value methods, as a result we cannot give a straight forward explanation on the performance. In this case, the gradient bandit algorithms are better than ϵ-Greedy methods, but in the below figure with number of bandits=50, the conclusion is opposite. One thing is certain that the gradient bandit algorithms take a lot more computation than the other two methods.



We can further observe the behavior of UCB from the above figure. In early stage, UCB selects action that have never been chosen, which reflects on the average rewards in the first 50 time steps. Once all actions have been tried, UCB starts to consider the action values. Consequently, the average rewards increase.

✓ Vary ε value with 0, 0.01, 0.1, 0.5 and 0.99. What happens? Why? Please plot it.

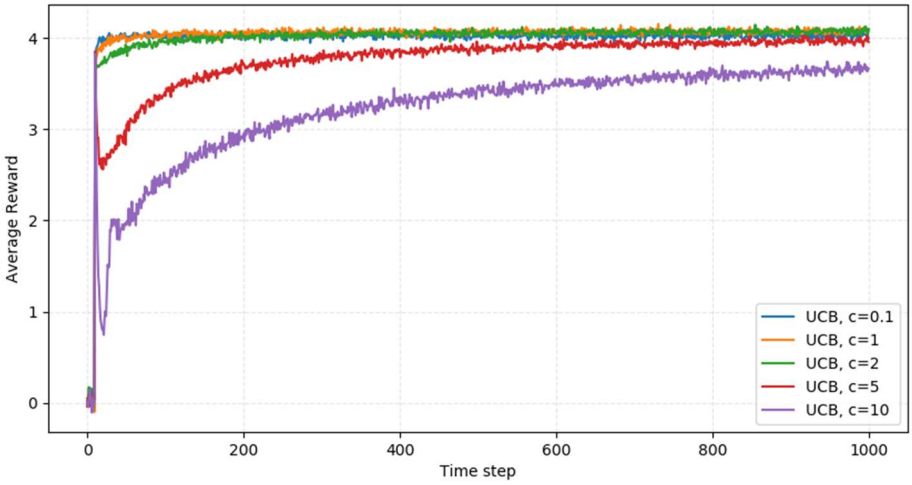


Starting from the worst-performed line, epsilon=0.99, basically, this policy is just a random guess, so the average reward should be close to the average true means (generated uniformly from [-5, 5]), which is zero as illustrated. The policy with epsilon=0.5 randomly selects an action half of the time, thus it is a lot better than the purple one. In general, these two lines have relatively large variance.

The blue line is the Greedy Action Selection policy, it is outperformed by ϵ-Greedy with small ϵ as illustrated in our experiments and figure 2.2 in the textbook. The epsilon=0.01 line and epsilon=0.1 are better than epsilon=0.5 line and epsilon=0.99 line because most of the time they optimize the action values.

It may take orange line longer time steps to find the optimal action since it has lesser chance to explore. In the long run, both of them will find the optimal action, and orange line has 0.99 chance to stick to the optimal action while green line has only 0.9 chance. As a result, the orange line will pass green line eventually.

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

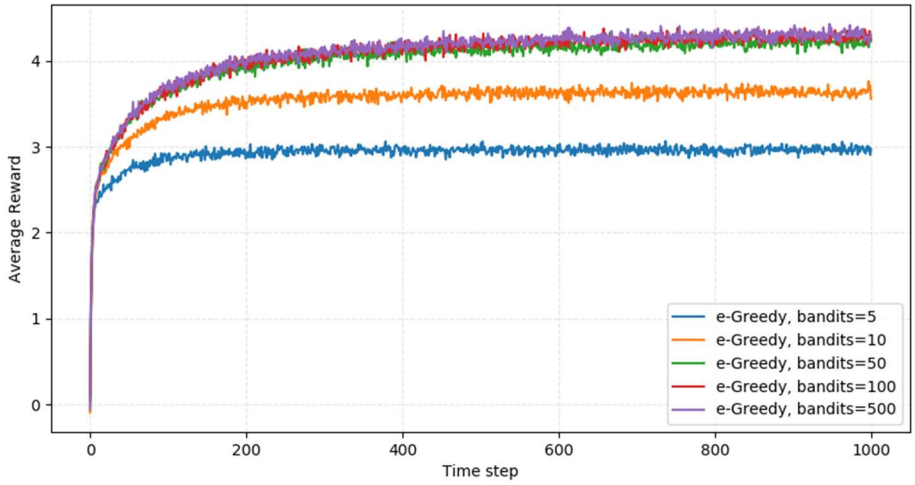


In UCB, we select the action according to $A_t = \underset{a}{\text{argmax}}[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}]$. The parameter c serves as a confidence level, multiplied by the uncertainty of the action value estimation. As we can see, it encourages the exploration of less selected actions, but large c may deviate the objective function from action value, causing suboptimal action selections.

We can perform a rough calculation to elaborate our conclusions. In this simulation, t=1000 and number of bandits=10. We assume that $N_t(a)$ is uniform across all actions, thus $N_t(a) = \frac{1000}{10} = 100$. Now, consider the right side term $\sqrt{\frac{\ln}{N_t(a)}} \approx 0.17$, if we time 0.17 with c=0.1, 1, 2 and 5, it has relatively small impact compared to $Q_t(a)$. As we expected, they all performs roughly the same, but when c=10 it performs significantly worse because the right side term affects the whole objective too much.

Another observation we have is that there's a peak in c=10 line at the early stage, actually it is when time steps=number of bandits+1, because it is the first time that all actions are chosen fairly, that is, all actions have been tried once and the right side terms are equal. With this information, the action selection is similar to Greedy method, in return we get a near optimal solution. But the right side term has so much influence that it will prevent us from selecting the same action next time. Thus, the average rewards plummet.

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



Since the rewards are generated from $\mathcal{N}(\mu_n, 0)$, where $\mu_n \sim \mathcal{U}[-5, 5]$, n=1, …, number of bandits. As a result, the more bandits we have, the greater chance we have to obtain a Gaussian distribution reward model with a higher mean. Using any action selection method, we are expect to have higher average rewards.