# Homework 1: Multi-Armed Bandit Problem

109061515 劉維仁

## Implementation

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

Randomly select one of actions.

1. In UCB, how do you select action when time steps < num of bandits?

Randomly select one of bandits have not used before.

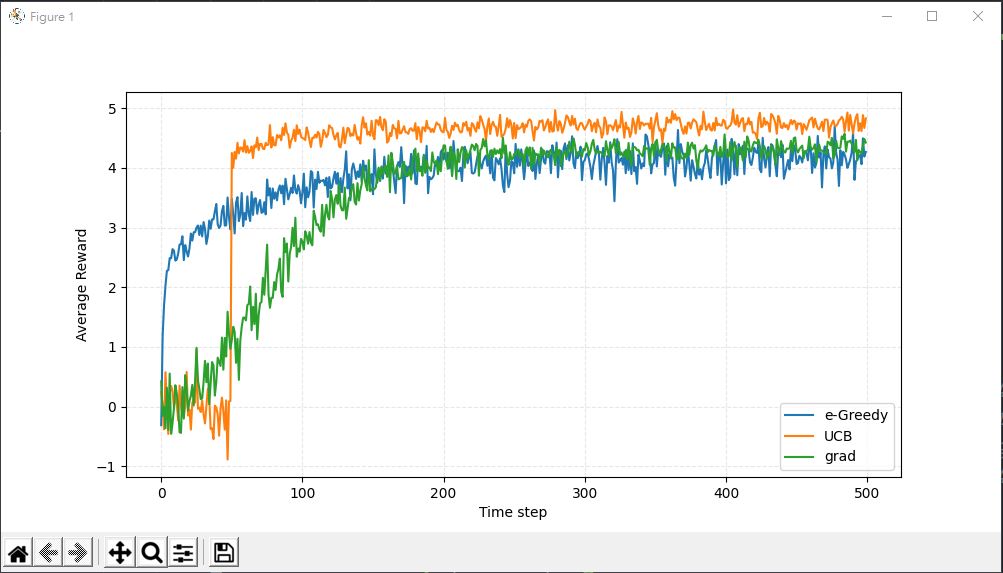
1. Briefly describe your implementation.

先隨機取 的數，如果取出的數大於 ，則使用 greedy action selection. 否則會隨機選一個 bandit。

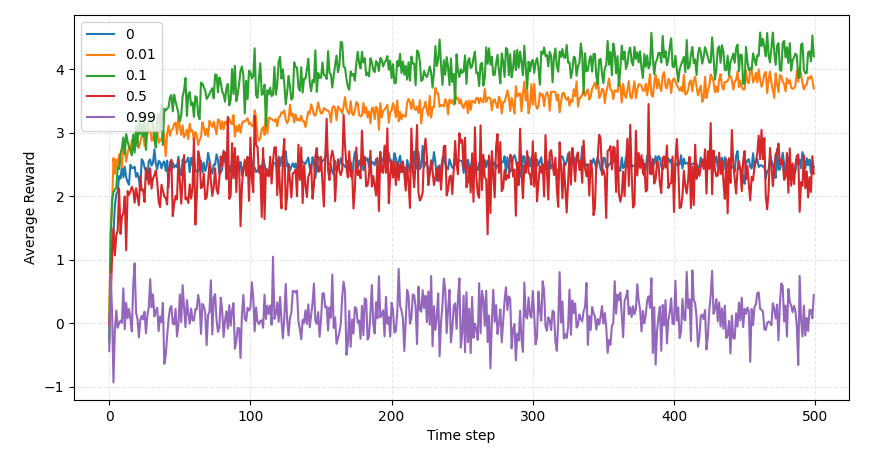
在 UCB 中，如果 timestep 小於 number of bandit，則同樣會隨機選出 bandit。但倘落已被選過，則會優先選擇其他尚未被選過的

## Experiments and Analysis

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

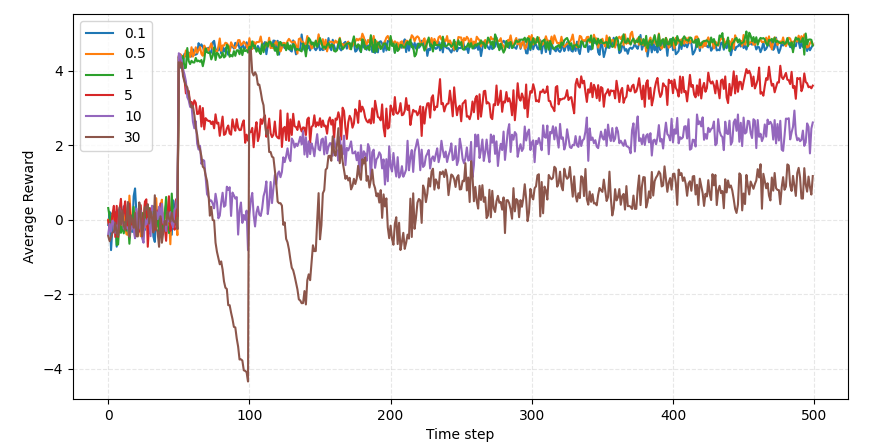


1. Vary 𝜀 value with 0, 0.01, 0.1, 0.5 and 0.99. What happens? Why? Please plot it.



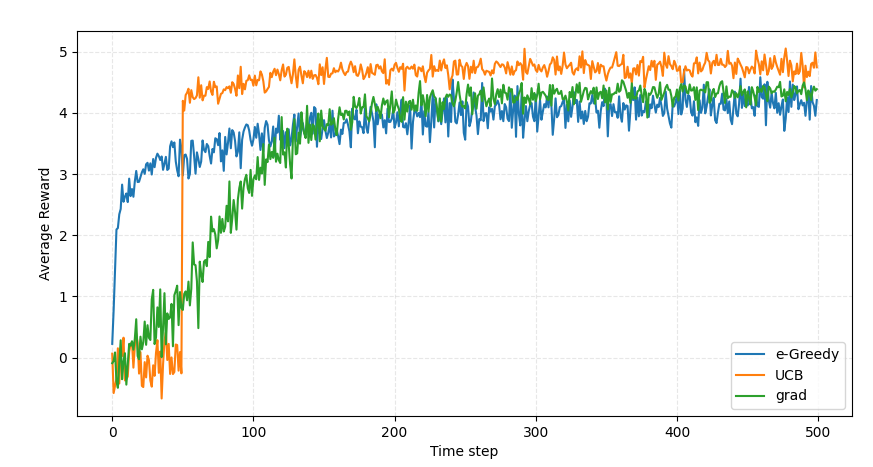
在 -greedy selection 中， 代表選擇 exploit 或 explore 的閥值。因此當其為0時，average reward 不會上升。而當增加為 0.01 則有更多機會去explore，也因此使 average reward 上升。0.1 同理。但當 太高，則會浪費太多時間，反而使 average reward 下降，如 0.5/0.99

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

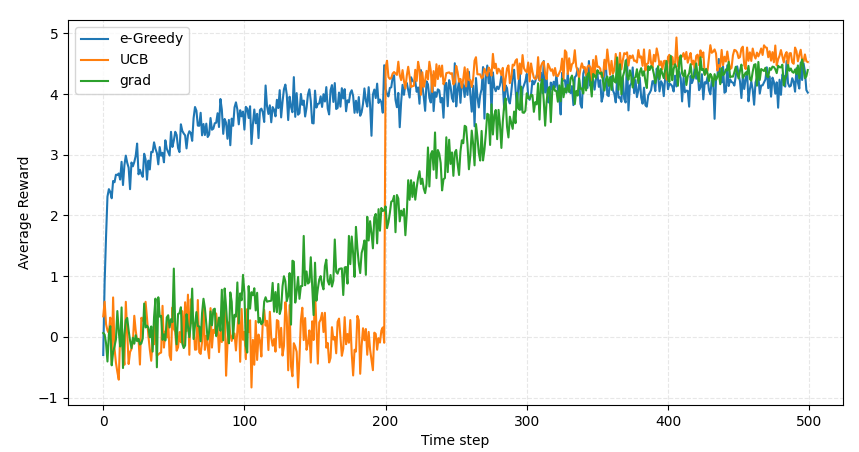


在程式剛開始時，所有 bandit 都會經過一次 explore，因此 c = 0.1, 0.5, 1 結果相差不多。而當 c = 5 或 10甚至更高時，程式會浪費時間在探索，因此 average reward 會表現較差，收斂也更久。

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



Bantit = 50



Bandit = 200

可以觀察到當 numbers of bandit 很大時，由於 UCB 需要先歷遍，因此在一開始的 average reward 表現很差，並浪費很多時間。

當 bandit = 200 時，因為 action 變得更多了，因此在梯度下降的過程需要花更多的時間，也因此使 average reward 上升的比 bandit = 50 還慢。