EE551000 System Theory
Homework 1: Multi-Armed Bandit

謝昉澂 109061589

Implementation

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

在ε-Greedy 中，我呼叫 numpy 的方法產生隨機變數 X，其中 X 的機率分佈，是在 0 和 1 之間的 uniform distribution，若 X> ε就選擇 Q 值最大的動作，否則從所有的動作裡面以相等的機率挑選一個動作。一個數列裡面的每個元素，以什麼樣的機率做挑選，可以由 numpy 的方法完成，因此相同機率也沒有問題。

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

在 steps<num 的時候有 num 減 steps 個動作沒被選過，我以相同的機率分佈從沒有被選過的動作裡面挑選一個。

3. Briefly describe your implementation

a) EpislonGreedy 裡面我更改了 act 函數還有 update 函數，act 函數根據 EpislonGreedy 選擇動作，update 則根據 Incremental 的形式更新 Q-table(如下)。

$$Q_{n+1} \quad = \quad Q_n + \frac{1}{n}\Big[R_n - Q_n\Big]$$

b) UCB 裡面我更改了 act 函數還有 update 函數，act 函數會根據以下演算法選擇動作

$$A_t \doteq \underset{a}{\operatorname{argmax}}\left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}\right],$$

If $N_t(a) = 0$, then a is considered to be a maximizing action.

若有多個 maximizing 的動作，則從中以等機率選擇，update 則跟 EpislonGreedy

的 update 方法一樣。

c) Gradient 裡面我將 Q-table，換成 H-table，代表 preference，在 act 函數裡面，每個動作會根據自己對應的機率被選擇，每個動作對應的機率可從 pi 函數被決定，如下式:

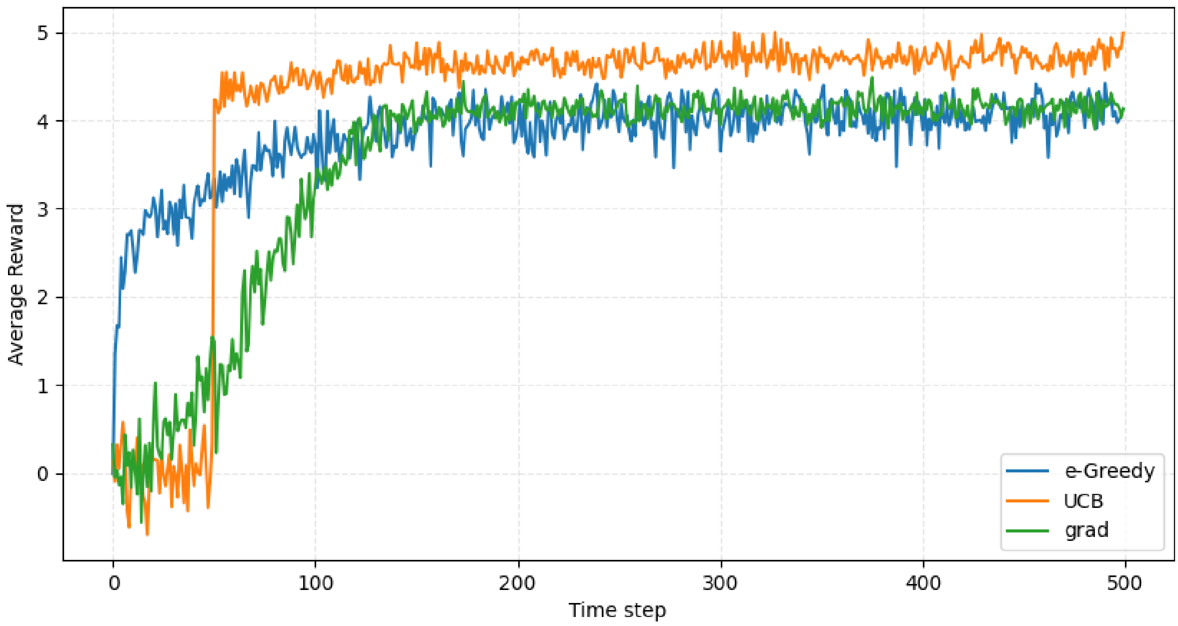
$$\Pr\{A_t = a\} \doteq \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \doteq \pi_t(a),$$

Update 函數則是實作以下演算法

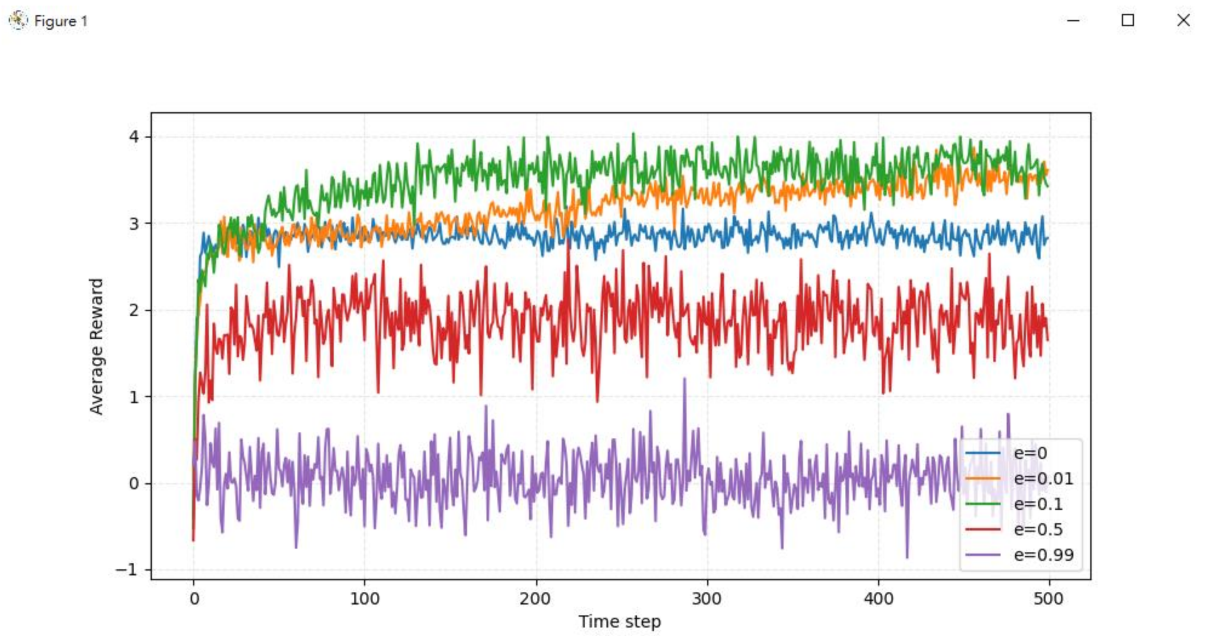
$$\begin{aligned} H_{t+1}(A_t) &\doteq H_t(A_t) + \alpha\big(R_t - \bar{R}_t\big)\big(1 - \pi_t(A_t)\big), && \text{and} \\ H_{t+1}(a) &\doteq H_t(a) - \alpha\big(R_t - \bar{R}_t\big)\pi_t(a), && \text{for all } a \neq A_t, \end{aligned}$$

Experiments and Analysis

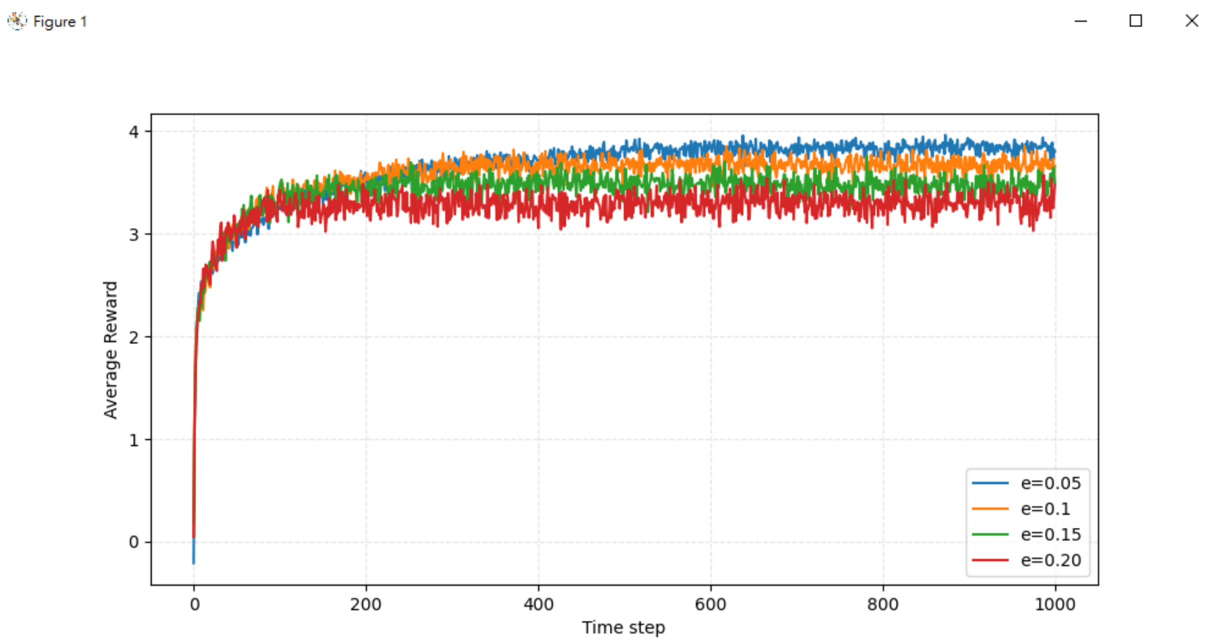
1.Plot the average reward curves of different methods into a figure.



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

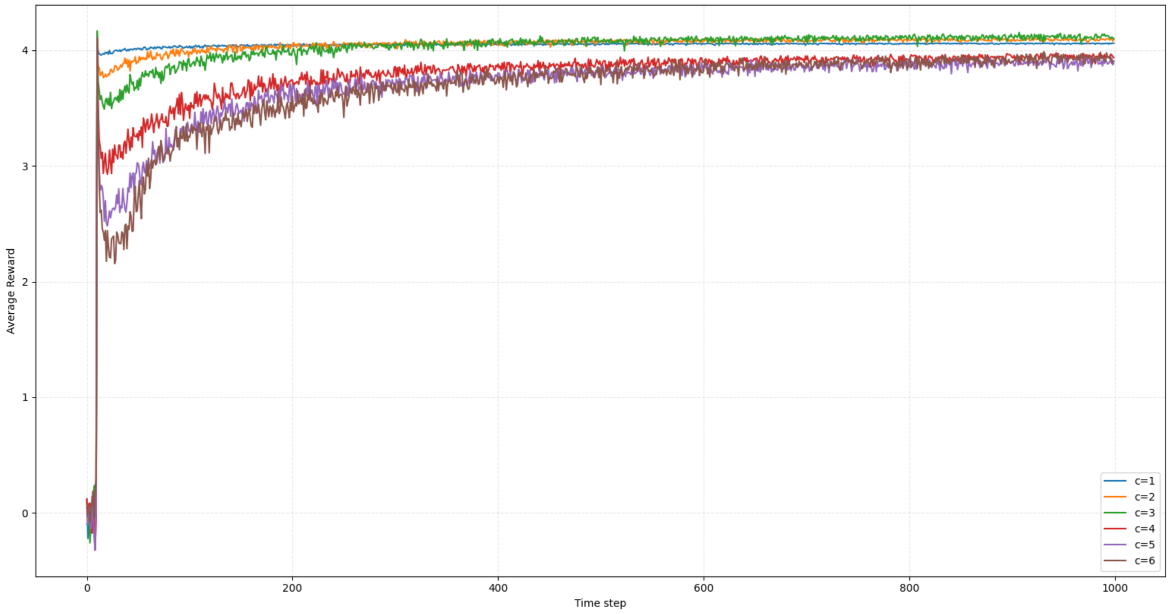


從上圖可以發現，ε如果太大，動作大部分都是隨機選擇，因此沒有辦法得到好的 reward，而當ε夠小的時候，可以根據 Q 值去選擇比較好的動作，然而當$\varepsilon=0$，沒有辦法探索是否有更好的動作可以選擇，當ε稍微大於 0 的時候，ε越大探索速度越快，但是最後收斂的結果也會因ε越大越容易隨機選擇所以 reward 越少，如下圖。



3.Vary the parameter c in UCB. What happens? Why?

Please plot it.



C 值越大初期越喜歡探索，因此初期 reward 會較差，隨著時間增加，每個動作都被選了很多次，最後都會選擇最大的 Q 值當作動作(如下公式)，若 c 值太小則會沒有足夠的探索變成 greedy selection。

$$A_t \doteq \underset{a}{\arg\max}\left[Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}}\right],$$

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

a)ε-greedy

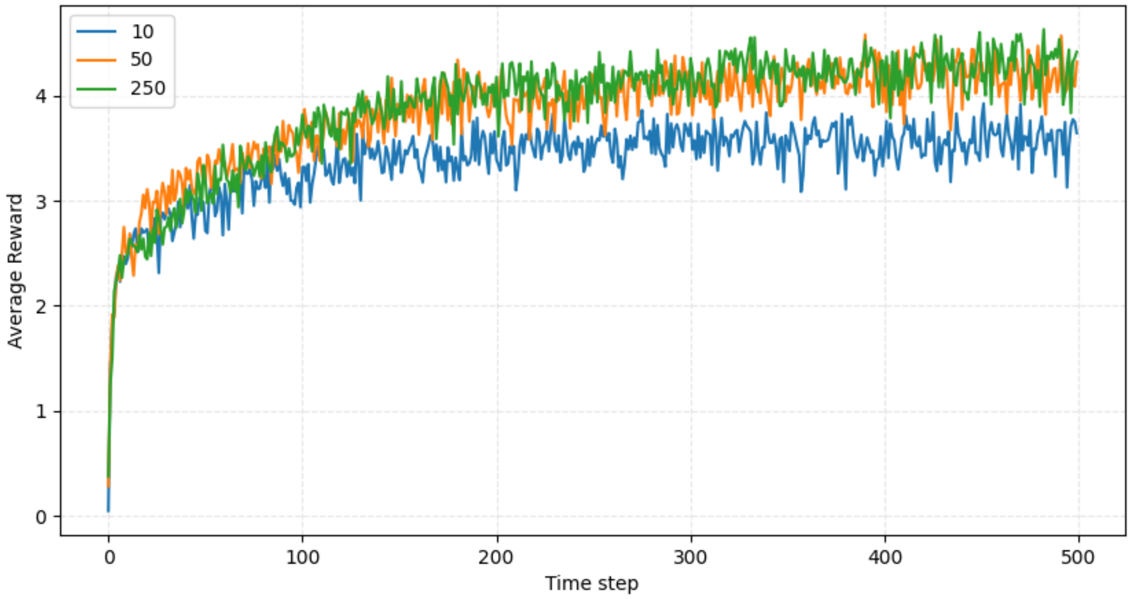
如下圖，在ε-greedy 的情況下沒有什麼改變，但是因為每個 bandit 的獎勵是在-5 和 5 之間的高斯分佈，在 bandit 較多的時候比較可能出現接近 5 的 reward。
增加 bandit 對運算時間沒什麼影響，如下實驗結果。

ε-greedy

10 bandits 1.26s

50 bandits 1.26s

250 bandits 1.27s



b)UCB

如下圖，在 UCB 的情況下增加 bandit，導致前面的步數都在探索沒有試過的動作，此外因為每個 bandit 的獎勵是在-5 和 5 之間的高斯分佈，在 bandit 較多的時候比較可能出現接近 5 的 reward。

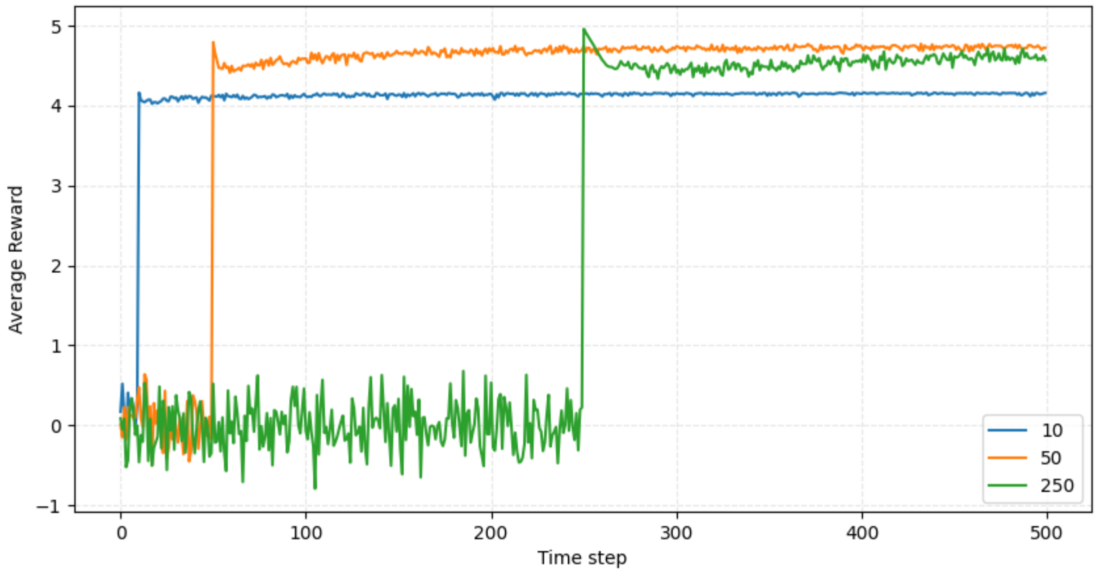
倍數增加 bandit 數量，運算時間大概也是倍數增加，如下實驗結果。

UCB

10 bandits 2.85s

50 bandits 12.5s

250 bandits 69.9s



c) gradient bandit

如下圖，在 gradient bandit 的情況下增加 bandit，會導致探索時間很長，而ε-greedy 較沒有這種現象，因為ε-greedy 探索動作的機率是定值，而 gradient 的方法一開始大家選擇動作的機率都相等，因此 bandit 越多每個動作被選的機率越低，之後才會慢慢增加 reward 較高的動作被選擇的機率，此外因為每個 bandit 的獎勵是在-5 和 5 之間的高斯分佈，在 bandit 較多的時候比較可能出現接近 5 的 reward。

倍數增加 bandit 數量，運算時間超過倍數增加，如下實驗結果。

gradient bandit

10 bandits 2.1s

50 bandits 4.4s

250 bandits 16.3.s