EE551000 System Theory

Homework 1: Multi-Armed Bandit

謝昉澂 109061589

Implementation

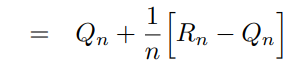
1. In 𝜀-Greedy, how do you select action if the probabilities are equal?

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。 {\displaystyle [a,b]}

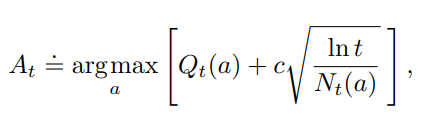
1. In UCB, how do you select action when time steps < num of bandits?

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。

1. Briefly describe your implementation
2. 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)。

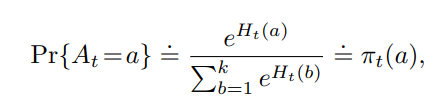
1. UCB裡面我更改了act函數還有update函數，act函數會根據以下演算法選擇動作



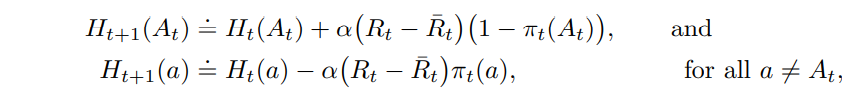


若有多個maximizing的動作，則從中以等機率選擇，update則跟EpislonGreedy的update方法一樣。

1. 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:

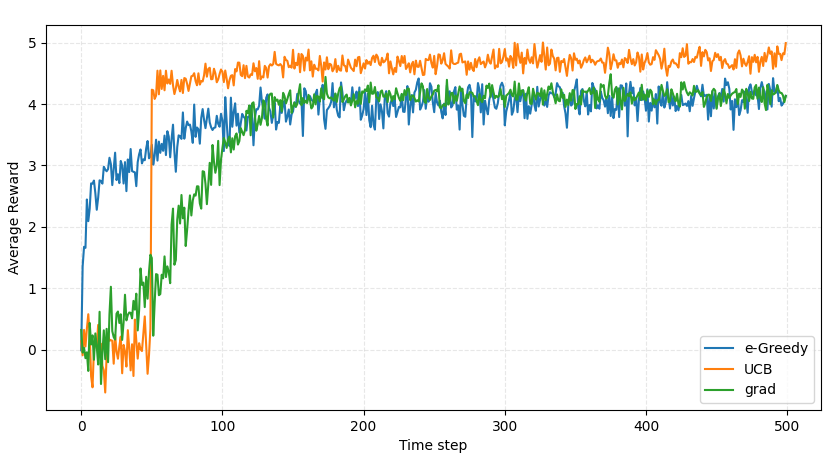


Update函數則是實作以下演算法

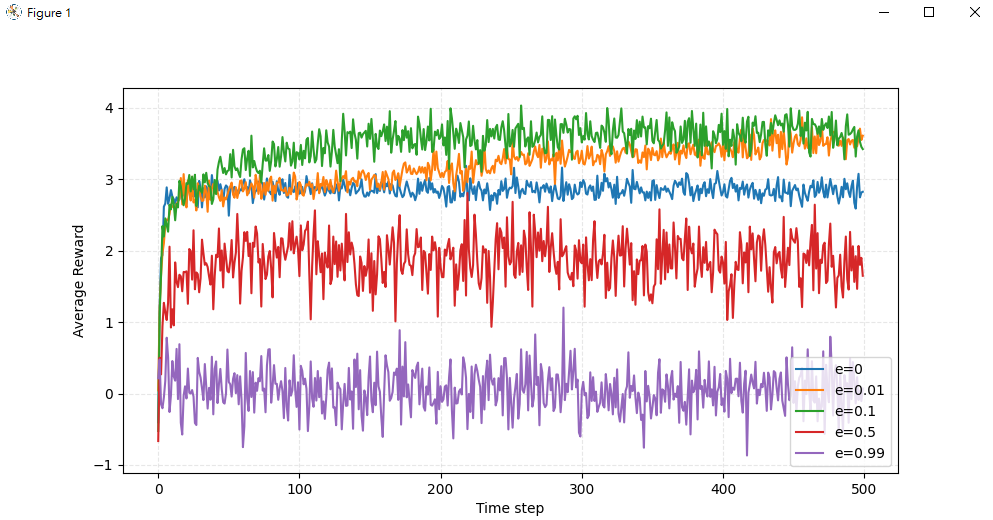


Experiments and Analysis

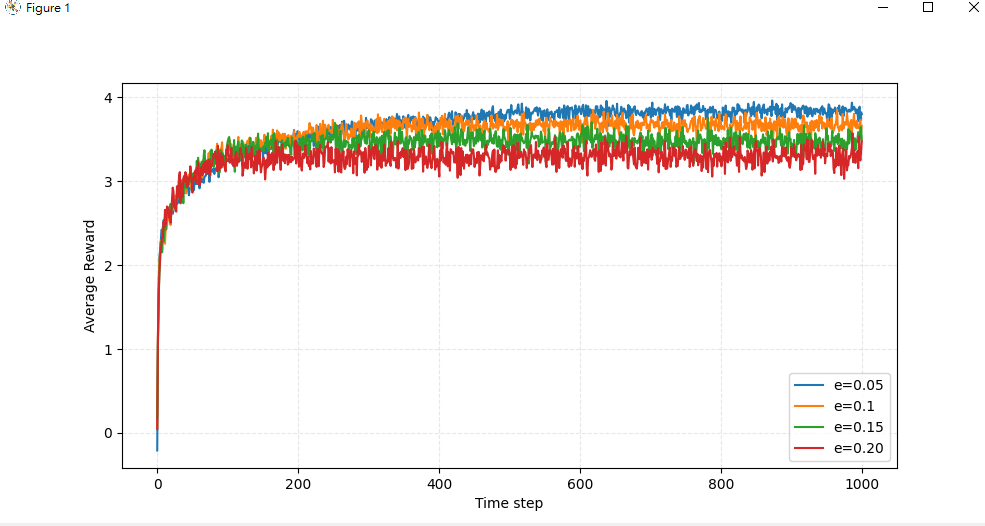
1.Plot the average reward curves of different methods into a figure.



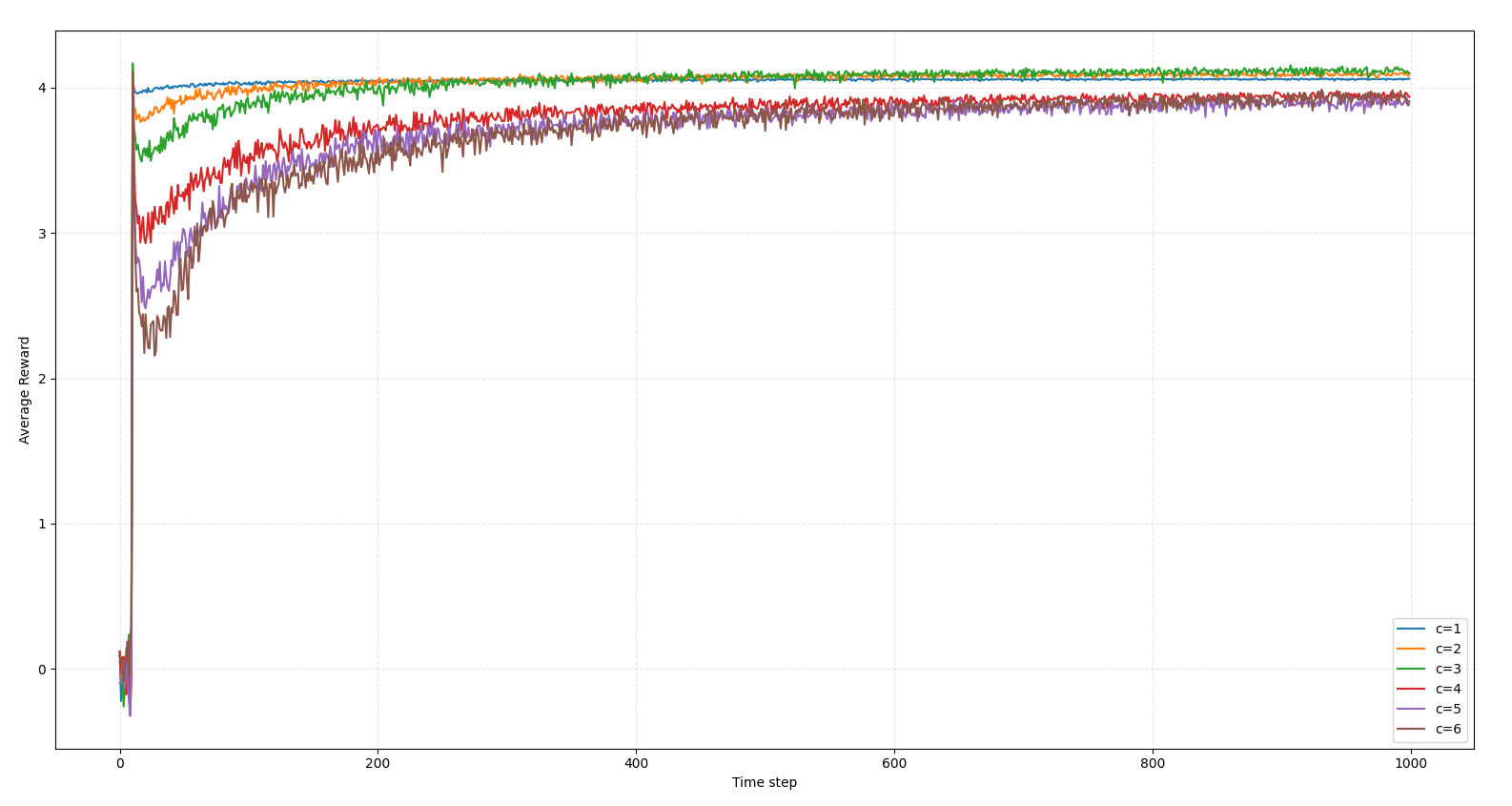
2.Vary 𝜀 value with 0, 0.01, 0.1, 0.5 and 0.99. What happens? Why? Please plot it.



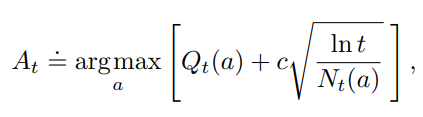
從上圖可以發現，𝜀如果太大，動作大部分都是隨機選擇，因此沒有辦法得到好的reward，而當𝜀夠小的時候，可以根據Q值去選擇比較好的動作，然而當𝜀=0，沒有辦法探索是否有更好的動作可以選擇，當𝜀稍微大於0的時候，𝜀越大探索速度越快，但是最後收斂的結果也會因𝜀越大越容易隨機選擇所以reward越少，如下圖。



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。



1. 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之間的均勻分佈，在bandit較多的時候比較可能出現接近5的reward。

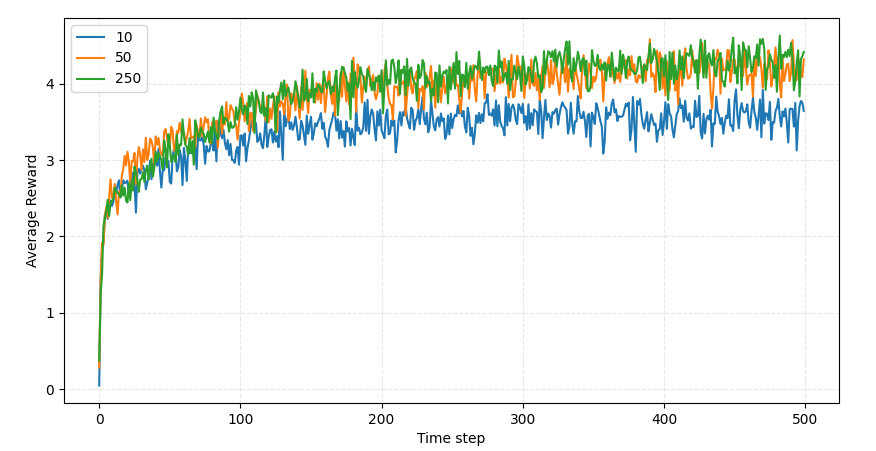
增加bandit對運算時間沒什麼影響，如下實驗結果。

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之間的均勻分佈，在bandit較多的時候比較可能出現接近5的reward。

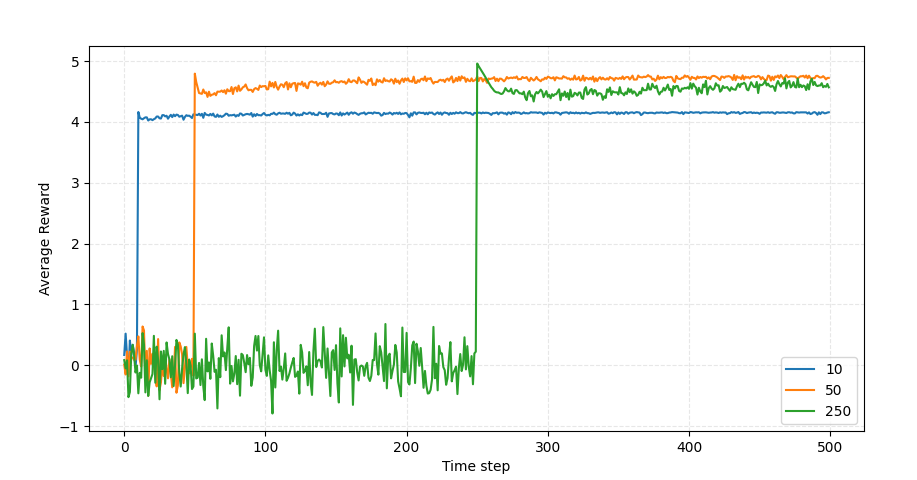
倍數增加bandit數量，運算時間大概也是倍數增加，如下實驗結果。

UCB

10 bandits 2.85s

50 bandits 12.5s

1. 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之間的均勻分佈，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

1. bandits 4.4s
2. bandits 16.3.s

