AI principle HW4

張語楹 110511164 蔡雅婷 110511068 李紹穎 109261007

Code: https://github.com/Benchangatrul284/AI_principle/tree/main/HW4

Part 1: Maximizing the reward

$\epsilon - greedy$

$\epsilon - greedy$算法是說有ϵ的機率會隨機選擇一個 arm 做 exploration。並有$1 - \epsilon$的機率選擇目前已知最好的 arm。這方法最容易實作，但缺點是很容易卡在不是最佳的 arm 當中。

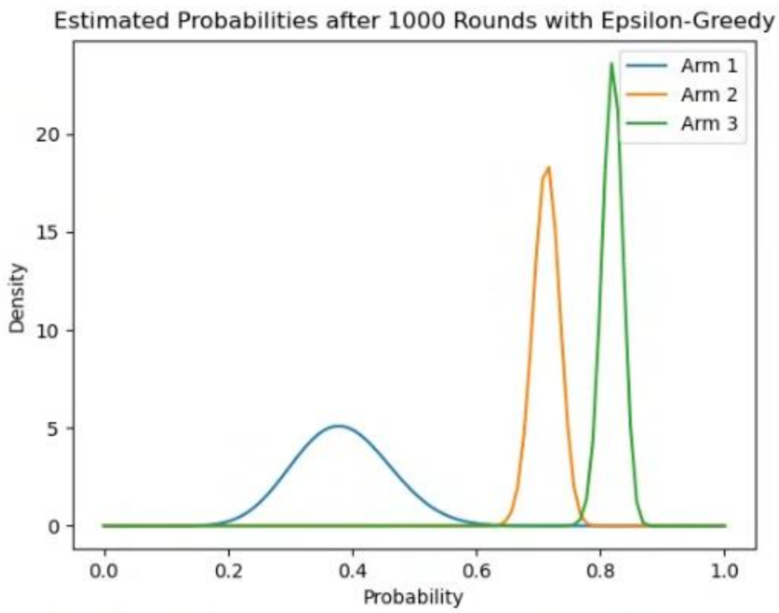
```
for _ in range(n_rounds):
    if np.random.rand() < epsilon:
        # Exploration: choose a random arm
        chosen_arm = np.random.choice(n_arms)
    else:
        # Exploitation: choose the best known arm
        # Arm with the highest success rate so far (wins/trials)
        # Avoid division by zero by adding a small value to trials
        success_rates = wins / (trials + 1e-5)
        chosen_arm = np.argmax(success_rates)

    # Simulate pulling the chosen arm
    reward = np.random.rand() < true_probabilities[chosen_arm]
    rewards.append(reward)
    wins[chosen_arm] += reward
    trials[chosen_arm] += 1
```

Final accumulated reward: 757*20 = 15140

Final accumulated regret: 43*20 = 860

Number of times each arm was pulled: [37 440 523]



UCB algorithm

UCB 會根據下面這個式子來決定下一個要選哪一個 arm。

$$A_{t+1} = argmax[Q_t(a) + c \cdot \sqrt{\frac{\ln t}{N_t(a)}}]$$

C = 0.5，是一個超參數，當 c 越大，UCB 會多多的去做 exploration。但不論 c 多小，因為每個 arm 的$N_t(a)$初始值皆為零，因此此式保證每個 arm 皆會被探索至少一次。且隨著 t 增加，$c \cdot \sqrt{\frac{\ln t}{N_t(a)}}$那項會越來越小。表示傾向 exploitation。

```
# Function to calculate the UCB values
def calculate_ucb(total_plays, wins, trials):
    ucb_values = []
    for arm in range(n_arms):
        if trials[arm] > 0:
            average_reward = wins[arm] / trials[arm]
            delta_i = math.sqrt(c * math.log(total_plays) / trials[arm])
            ucb_values.append(average_reward + delta_i)
        else:
            ucb_values.append(float('inf'))
    return ucb_values
```

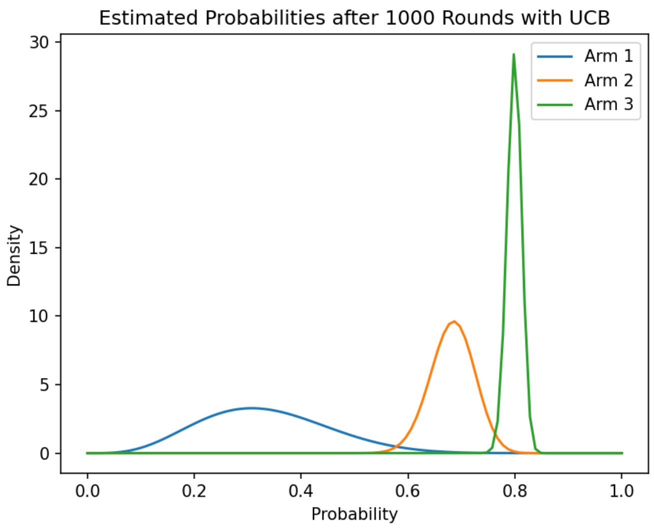
前面的 function 會計算等等要用到的 UCB value。

```
# UCB Algorithm
for round_number in range(1, n_rounds + 1):
    ucb_values = calculate(total_plays, wins, trials)
    chosen_arm = np.argmax(ucb_values)
    chosen_arms.append(chosen_arm)
    reward = np.random.rand() < true_probabilities[chosen_arm]
    rewards.append(reward)
    wins[chosen_arm] += reward
    trials[chosen_arm] += 1
    total_plays += 1
```

Final accumulated reward: 779*20 = 15580

Final accumulated regret: 21*20 = 420

Number of times each arm was pulled: [13 124 863]



Thompson sampling

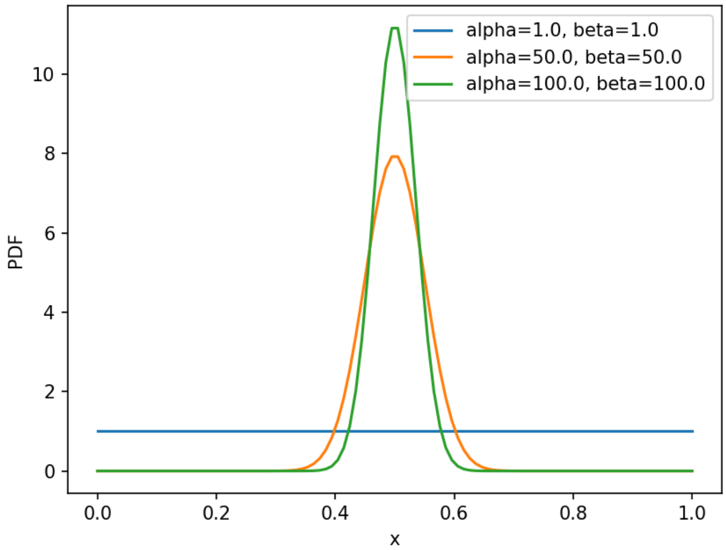
Thompson sampling 看準了 Bernoulli 和 beta 是 conjugate pair 的特質。

Thompson sampling 以 beta distribution 作為每個 arm 的 prior。Beta distribution 有兩個參數，α、β 。每個 arm 的初始值為$\alpha = 1$、$\beta = 1$。當一個 success 後，對於該 arm 的$\alpha += 1$，如果失敗，對於該 arm 的 $\beta += 1$。因此每當 pull 一次後，該 arm 就會有一個新的 distribution。

那如何決定要 pull 哪一個 arm 呢? 當要決定要取哪一個 arm 時，我們會從各個 arm 的 beta distribution 來 sample 出一個值。我們就 pull 最高的那個 arm。

```
# Thompson Sampling Algorithm
for round_number in range(1, n_rounds + 1):
    sampled_probs = [beta.rvs(a=wins[i]+1, b=trials[i]-wins[i]+1) for i
in range(n_arms)]
    chosen_arm = np.argmax(sampled_probs)
    reward = np.random.rand() < true_probabilities[chosen_arm]
    rewards.append(reward)
    chosen_arms.append(chosen_arm)
    wins[chosen_arm] += reward
    trials[chosen_arm] += 1
```

exploration 和 exploitation 的描述藏在哪裡呢? 其實就在 beta distribution 裡面，下面畫出對於不同α、β的 beta distribution:

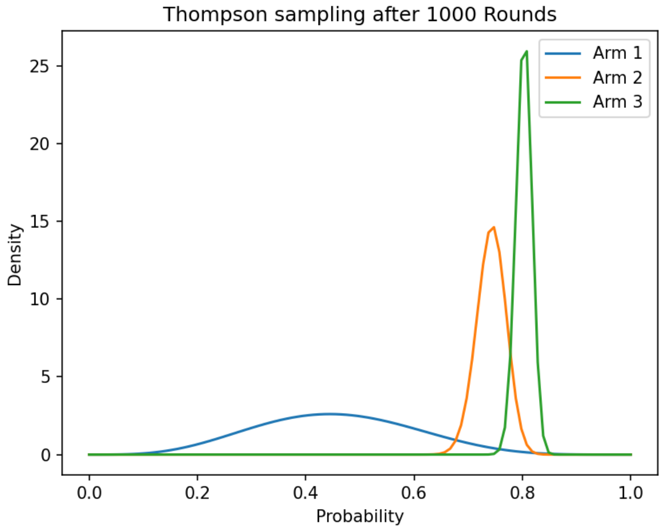


可以發現當α、β越大時，beta distribution 會更集中在$\frac{\alpha}{\alpha+\beta}$，相反的可以注意到當$\alpha = 1$、$\beta = 1$時，beta distribution 會變成 uniform distribution，也符合我們對每個 arm 的初始假設。所以當$\alpha + \beta = t$ 較小時，exploration 較多。

Final accumulated reward: 785*20 = 15580

Final accumulated regret: 15*20 = 300

Number of times each arm was pulled: [9 258 733]



Problem 2: Hyperparameter selection

Upper confident bound:

Confident bound:

根據公式推導:

$$P\left(|S_n-\mu|\geq\sqrt{\frac{c\ln(t^\alpha)}{n}}\right)\leq\frac{1}{t^\alpha}$$

n 是該 arm 被 pull 的次數。這裡我們將α設為 1。

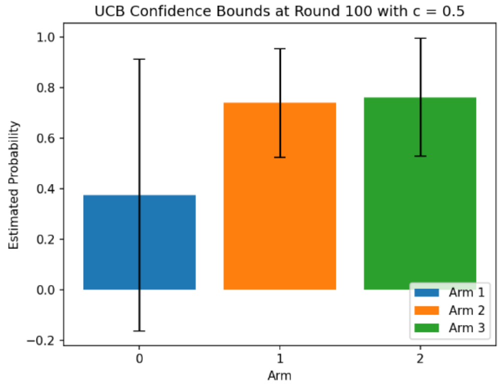
信賴區間即是:

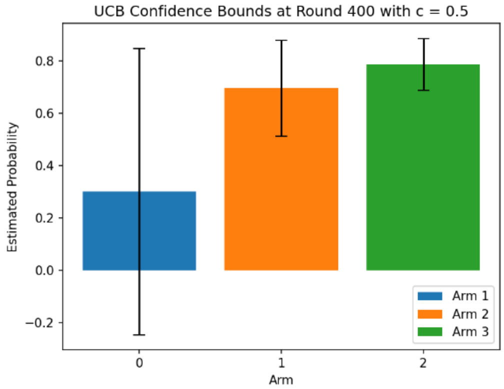
$$confidence\ bound=\pm\sqrt{\frac{c\ln t}{n}}$$

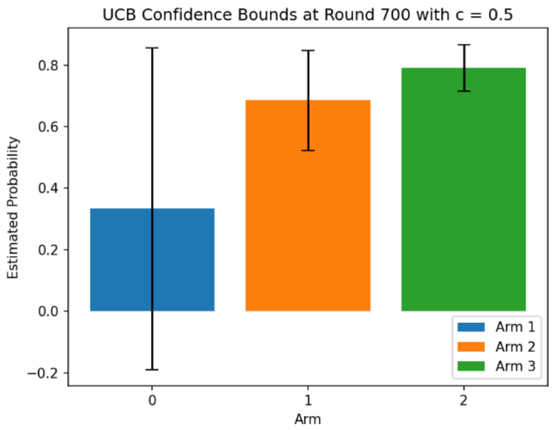
也就是在 T=t 時，有$1-\frac{1}{T}$的機率 sample mean 和實際的 mean 相差不到$\sqrt{\frac{c\ln T}{n}}$。

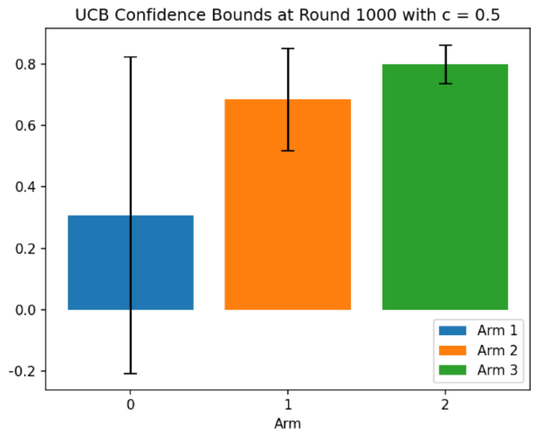
從上面式子可以發現，當 n 增加時，$confidence\ bound$會越來越小。

讓 c=0.5，來看 confidence interval 怎麼變化。



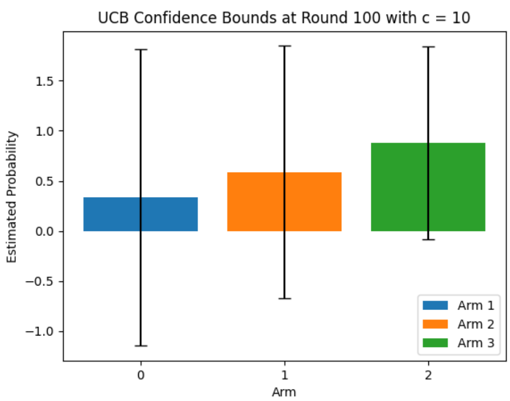


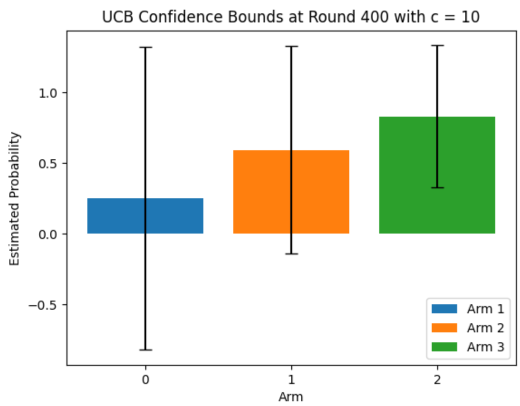


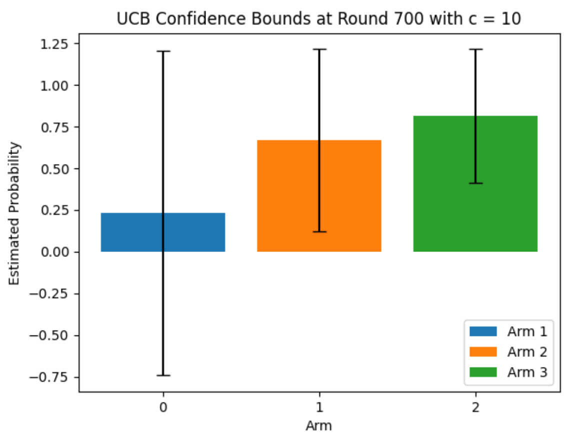


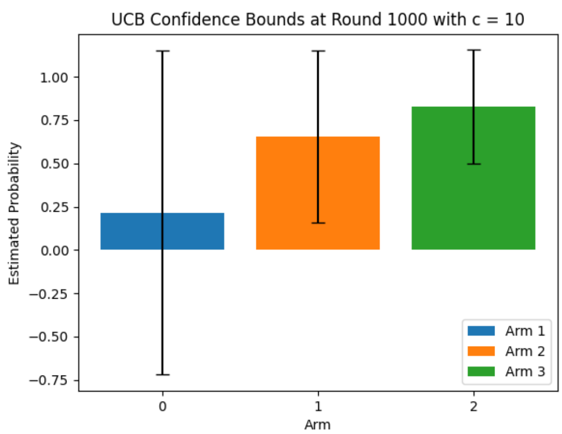
可以發現隨著時間 t 增加，arm 3 的 confidence interval 會下降。這是因為 arm 3 被 pull 的次數比其他兩個 arm 還要多。因此對 arm3 比較了解。

接下來如果我們改動 c 會造成甚麼影響呢?





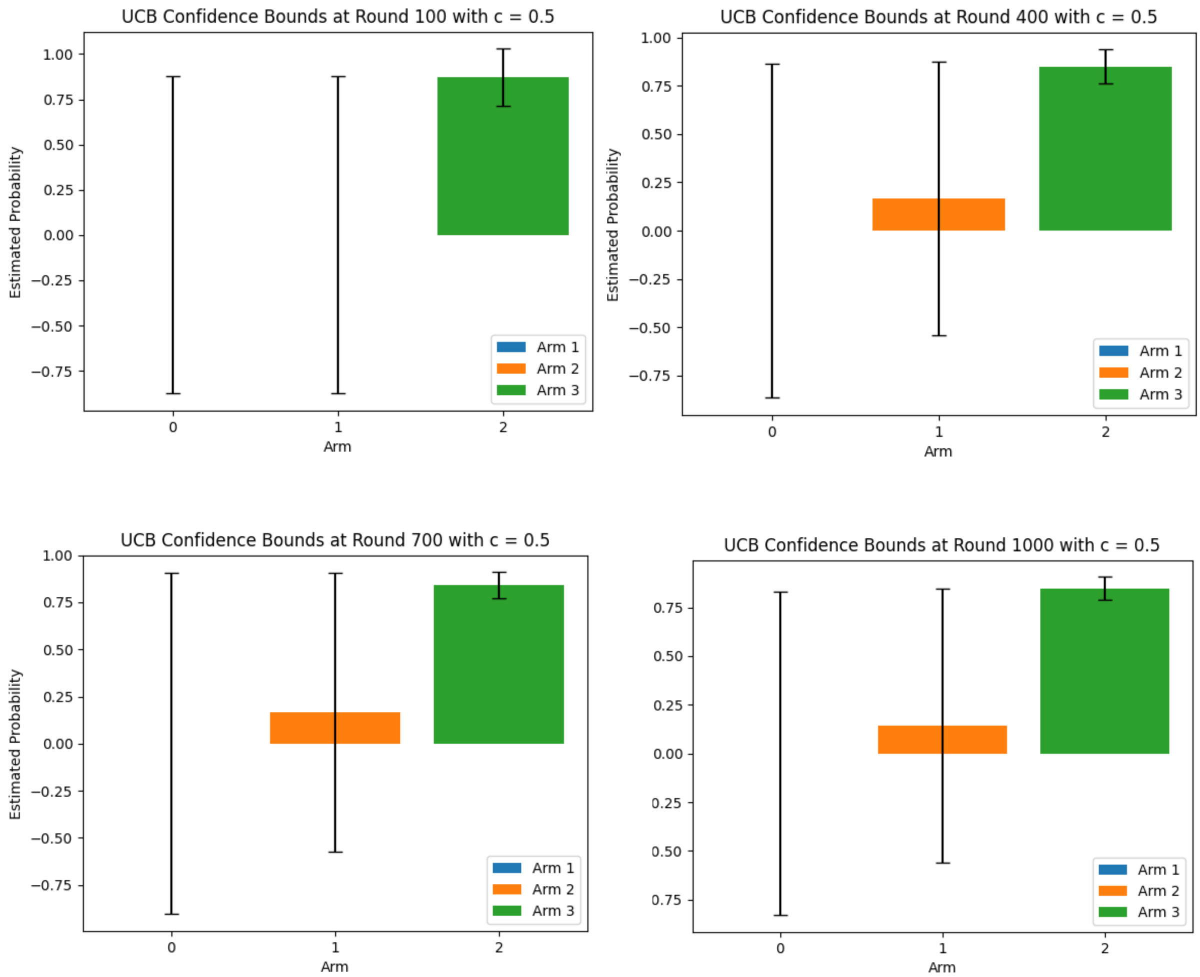




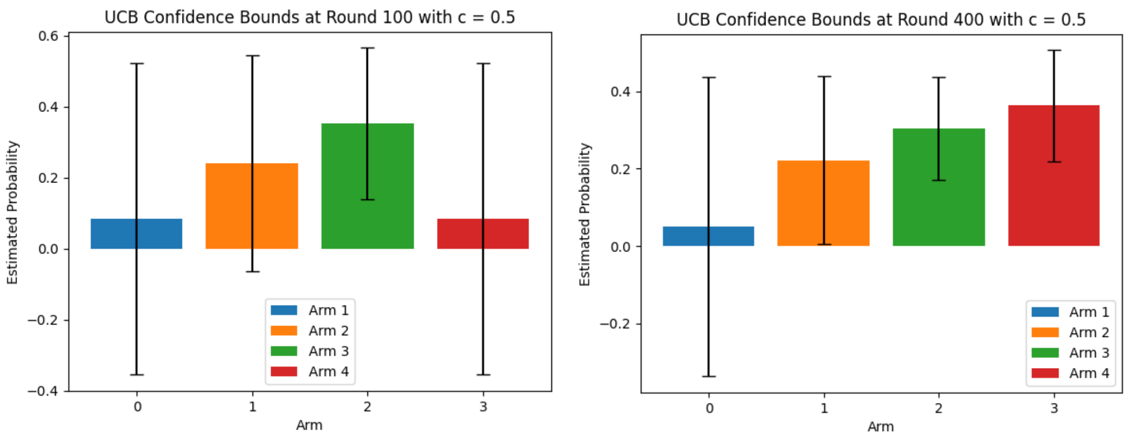
可以發現 arm 3 的 confidence interval 會下降但較 c=0.5 還大，這是因為

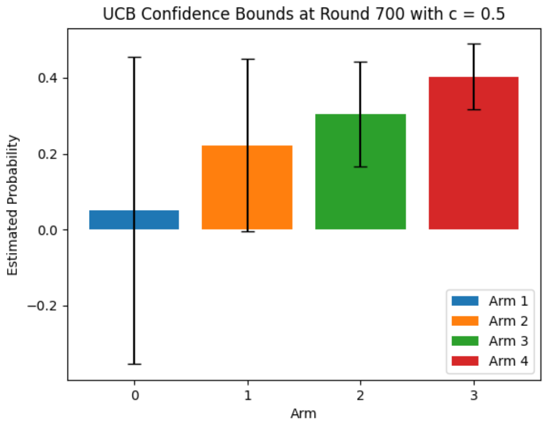
$$confidence\ bound = \pm\sqrt{\frac{c \ln t}{n}}$$

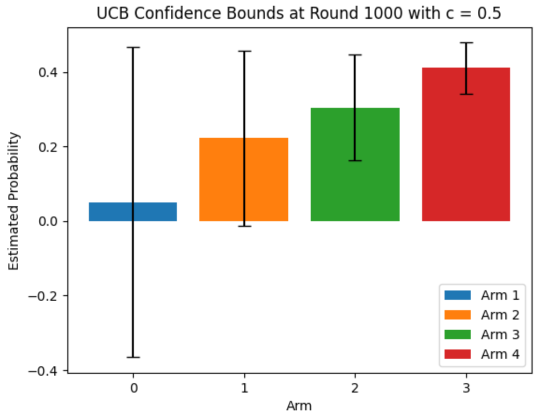
較大的 c 會給 UCB 比較大的 exploration 空間。
我們將 wining probability 改成: `[0.05, 0.1, 0.85]`
來看一下對於 confidence interval 如何改變。



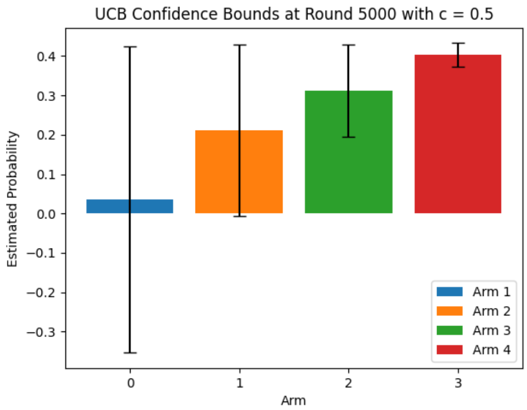
可以看到 arm 3 的 confidence interval 又縮得更小，因為 UCB 對 arm3 做更多的 exploitation。
接下來，我們來實驗看看更多的 arm 的情況:
`n_arms = 4 true_probabilities = [0.1, 0.2, 0.3, 0.4]`

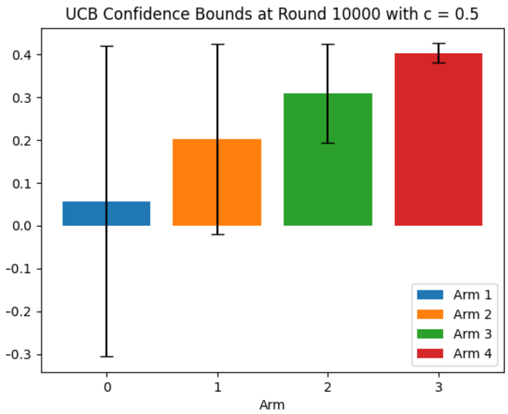






可以發現更多的 arm 結果也相同。
如果將 rounds 增加至 10000。

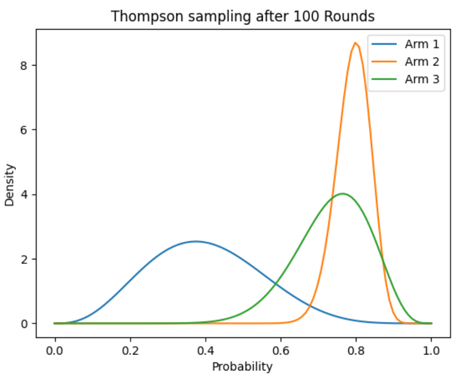


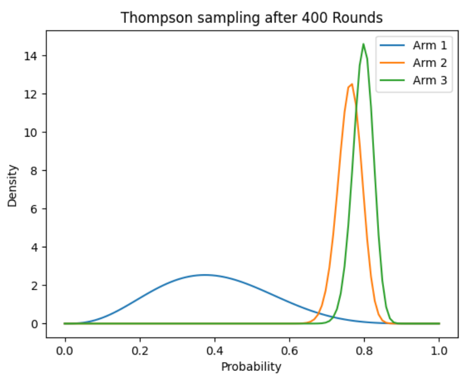


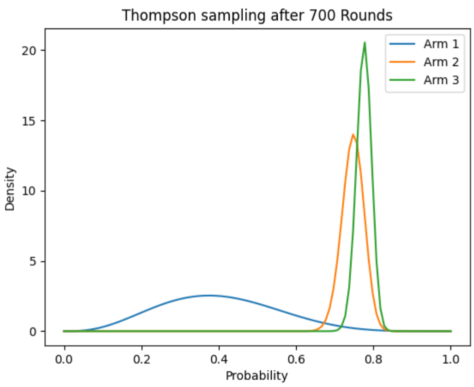
arm 4 的 confidence bound 進一步縮小，且 UCB 趨向收斂。

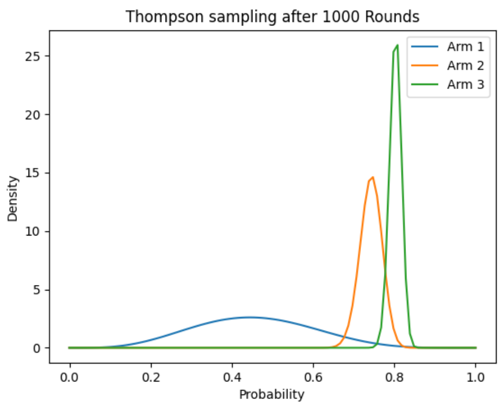
Thompson sampling algorithm

我們先不動 wining probability。印出各種 arm 在不同 iteration 的 beta distribution。



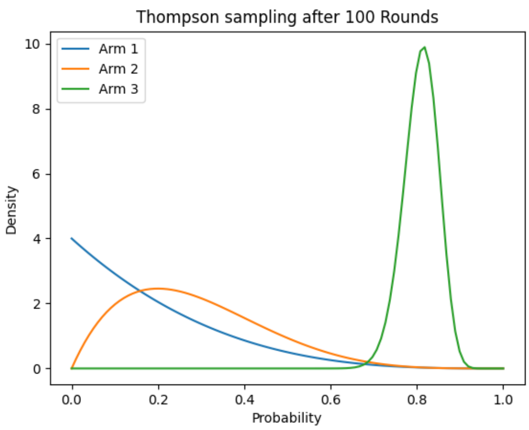


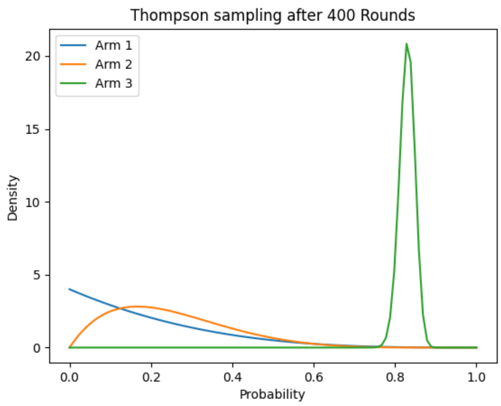


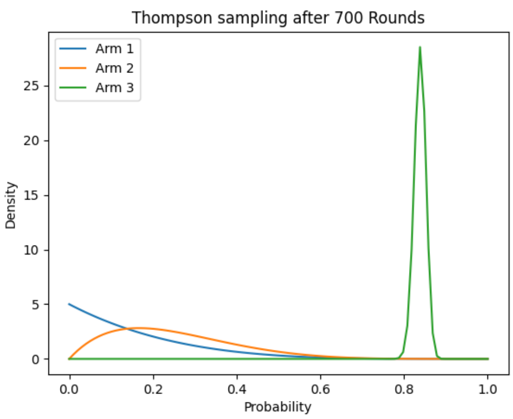


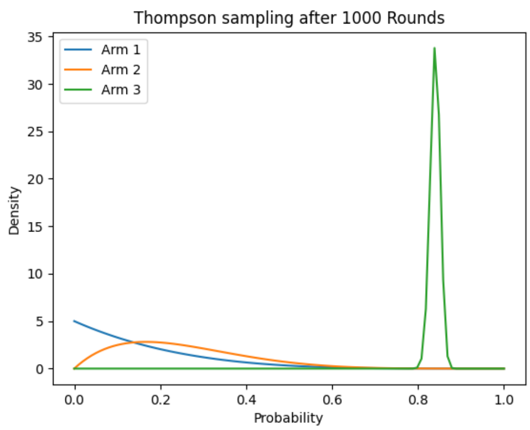
可以發現隨著 rounds 增加，arm3 的 peak 越明顯，且其值約在 0.8 左右。而對於 arm 1，peak 不明顯較接近 uniform，因為對於 arm1 的 exploration 較少。

接下來，我們更改 wining probability = `[0.05,0.1,0.85]`。





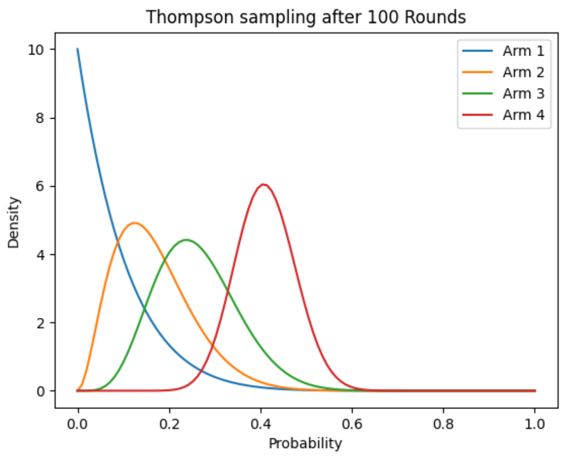


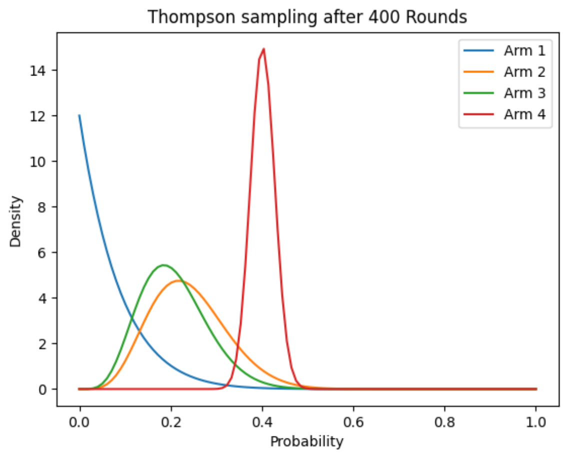


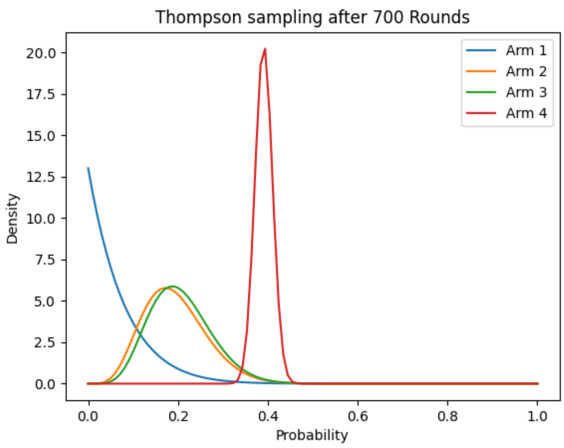
arm3 的 peak 約在 0.85 左右。而對於 arm 1，Thompson sampling 也成功預測它的 probability 大概為 0。

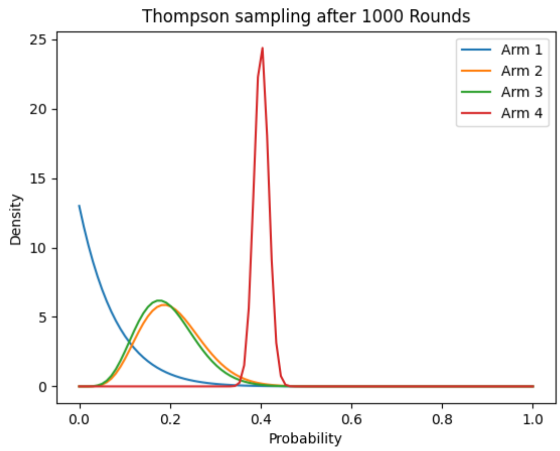
接下來，我們來實驗看看更多的 arm 的情況:

```
n_arms = 4 true_probabilities = [0.1, 0.2, 0.3, 0.4]
```



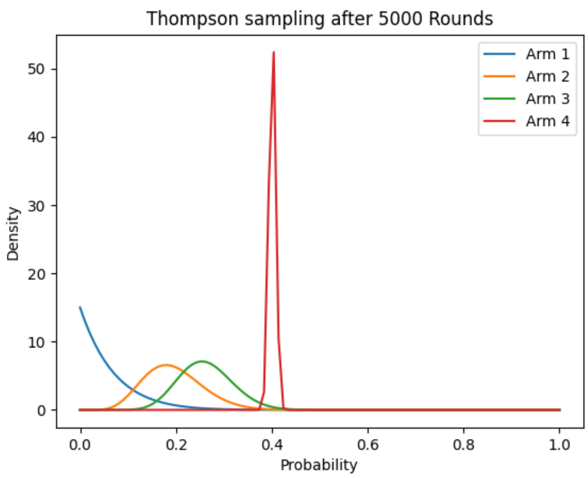


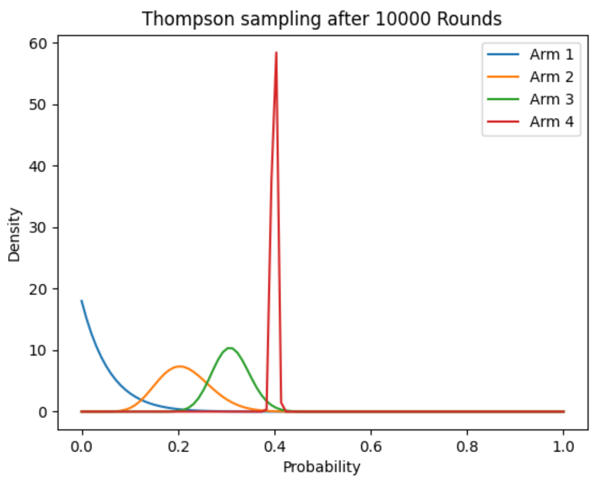




可以發現更多的 arm 結果也相同。

如果將 rounds 增加到 10000。





arm 4 的 beta distribution 的 peak 更明顯。