

### (1) 演算法公式 (LaTeX)

```
\textbf{UCB1 Algorithm}

\begin{align*}
& \& \text{Initialize: } Q(a) = 0, \quad N(a) = 0 \quad \forall a \in \{1, \dots, k\} \\
& \& \text{For each time step } t = 1 \quad \text{to } T: \\
& \quad \& \text{If } N(a) = 0: \quad \text{select } a \quad \text{immediately} \\
& \quad \& \text{Else: } UCB(a) = Q(a) + c \cdot \sqrt{\frac{\ln t}{N(a)}} \\
& \quad \& \text{Choose } a_t = \arg\max_a UCB(a) \\
& \quad \& \text{Receive reward } r_t \sim \mathcal{N}(\mu_{a_t}, 1) \\
& \quad \& N(a_t) \leftarrow N(a_t) + 1 \\
& \quad \& Q(a_t) \leftarrow Q(a_t) + \frac{1}{N(a_t)} (r_t - Q(a_t))
\end{align*}
```

### (2) ChatGPT 提示語

請實作 UCB1 多臂拉霸演算法，使用置信區間公式評估每個動作的不確定性，並觀察其如何在探索與利用之間取得平衡。

### (3) Python 程式碼

```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(42)
k = 10
steps = 10000
c = 2
true_rewards = np.random.normal(0, 1, k)

Q = np.zeros(k)
N = np.zeros(k)
rewards = np.zeros(steps)

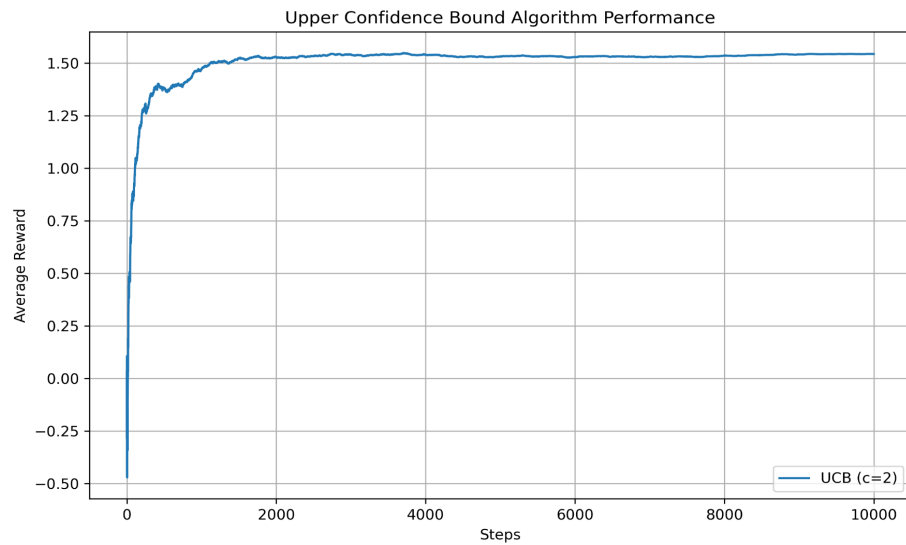
for t in range(steps):
    if 0 in N:
        a = np.argmin(N)
    else:
        ucb = Q + c * np.sqrt(np.log(t + 1) / N)
        a = np.argmax(ucb)

    reward = np.random.normal(true_rewards[a], 1)
    N[a] += 1
    Q[a] += (reward - Q[a]) / N[a]
    rewards[t] = reward

cumulative_average = np.cumsum(rewards) / (np.arange(steps) + 1)

plt.figure(figsize=(10, 6))
plt.plot(cumulative_average, label=f"UCB (c={c})")
plt.xlabel("Steps")
plt.ylabel("Average Reward")
plt.title("Upper Confidence Bound Algorithm Performance")
plt.legend()
plt.grid(True)
plt.savefig("ucb_result.png", dpi=300)
plt.show()
```

### (3-1) 圖表



#### (4) 結果分析

時間複雜度：  $O(T * k)$

空間複雜度：  $O(k + T)$

說明：

- 具有理論保證，能有效控制探索幅度
- 適合動態平衡場景，但對參數  $c$  敏感
- 初期會平均探索每個 arm，後期快速聚焦於最佳臂