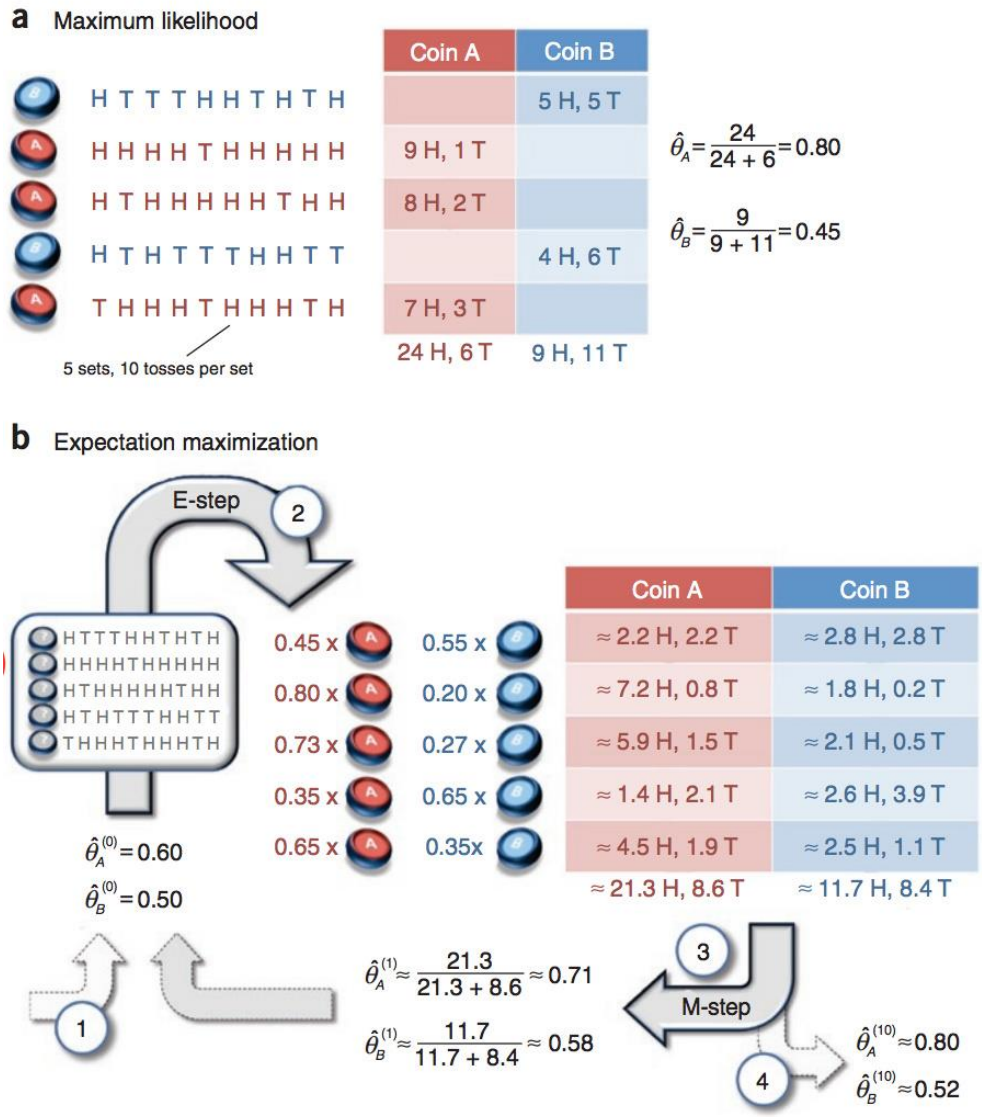
人工智能实验报告

姓名：孟衍璋 学号：16337183

一、 实验内容

假设有两枚硬币 A、B，以相同的概率随机选择一个硬币，进行如下的抛硬币实验：共做 5 次实验，每次实验独立的抛十次，结果如图中 a 所示，例如某次实验产生了 H、T、T、T、H、H、T、H、T、H，H 代表正面朝上。

假设试验数据记录员可能是实习生，业务不一定熟悉，造成如下图的 a 和 b 两种情况：



a 表示实习生记录了详细的试验数据，我们可以观测到试验数据中每次选择的是 A 还

是 B

b 表示实习生忘了记录每次试验选择的是 A 还是 B，我们无法观测实验数据中选择的硬币是哪个

问：在两种情况下分别如何估计两个硬币正面出现的概率？并完成相应的 python 代码实现。

二、 算法实现

首先考虑第一种情况，有两枚硬币 A、B，以相同的概率随机选择一个硬币，进行如下的抛硬币实验：共做 5 次实验，每次实验独立的抛十次。第一种情况实习生记录了详细的实验数据，我们知道每次选择的是 A 还是 B。

这样的话就可以直接计算 A 与 B 硬币正面出现的频率，并将频率当作概率处理,相应代码实现如下：

```
observations_head_A = [9,8,7]  # 观察A硬币扔到正面的次数
observations_tail_A = [1,2,3]  # 观察A硬币扔到反面的次数
observations_head_B = [5,4]  # 观察B硬币扔到正面的次数
observations_tail_B = [5,6]  # 观察B硬币扔到反面的次数

count = 0
for i in observations_head_A:
    count += i
probability_A = count / (10 * len(observations_head_A))
count = 0
for i in observations_head_B:
    count += i
probability_B = count / (10 * len(observations_head_B))

print("The probability that coin A will be thrown to the front:", probability_A)
print("The probability that coin B will be thrown to the front:", probability_B)
```

运行结果得到硬币 A 与 B 正面出现的概率：

```
The probability that coin A will be thrown to the front: 0.8
The probability that coin B will be thrown to the front: 0.45
[Finished in 0.1s]
```

然后考虑第二种情况，这时实习生忘记记录每次实验选择了 A 还是 B，只是记录下了 5 次实验的数据，这个时候就没有办法直接估计硬币 A 与 B 扔到正面的概率。

我们先初始化硬币 A 与 B 扔正面的概率，这里先假设为 0.6 与 0.4。然后对其求期望，比如第一次投掷，为 A 的概率为$p_A = (0.6)^5(0.4)^5$，为 B 的概率为：$p_B = (0.5)^5(0.5)^5$，进行归一化后得到$\mathrm{p(A)} = 0.45$，$\mathrm{p(B)} = 0.55$。然后再分别乘以各次实验中正

面与反面出现的概率，多次迭代之后，直到结果收敛变可以得到答案。在代码中判断两次结果的差值是否小于10^{-6}，如果小于的话，就认为已经收敛。

代码实现如下：

```python
def em_single(probability_a, probability_b, observations_head, observations_tail):
    new_observations_head_a = 0.0
    new_observations_tail_a = 0.0
    new_observations_head_b = 0.0
    new_observations_tail_b = 0.0
    for i in range(len(observations_head)):
        pa = pow(probability_a, observations_head[i]) * pow(1-probability_a, observations_tail[i])  # 为A的概率
        pb = pow(probability_b, observations_head[i]) * pow(1-probability_b, observations_tail[i])  # 为B的概率
        normalized_pa = pa / (pa + pb)  # 归一化后的P(A)
        normalized_pb = pb / (pa + pb)  # 归一化后的P(B)

        new_observations_head_a += normalized_pa * observations_head[i]
        new_observations_tail_a += normalized_pa * observations_tail[i]
        new_observations_head_b += normalized_pb * observations_head[i]
        new_observations_tail_b += normalized_pb * observations_tail[i]

    new_probaliblity_a = new_observations_head_a / (new_observations_head_a + new_observations_tail_a)
    new_probaliblity_b = new_observations_head_b / (new_observations_head_b + new_observations_tail_b)

    return new_probaliblity_a, new_probaliblity_b

def em(origin_a, origin_b, observations_head, observations_tail):
    count = 0
    new_probaliblity_a = origin_a
    new_probaliblity_b = origin_b
    while(1):
        probability_a = new_probaliblity_a
        probability_b = new_probaliblity_b
        new_probaliblity_a, new_probaliblity_b = em_single(new_probaliblity_a, new_probaliblity_b, observations_head, observations_tail)
        delta_a = new_probaliblity_a - probability_a
        delta_b = new_probaliblity_b - probability_b
        if delta_a < 1e-6 and delta_b < 1e-6:
            print("Number of iterations:", count)
            print("The probability that coin A will be thrown to the front:", new_probaliblity_a)
            print("The probability that coin B will be thrown to the front:", new_probaliblity_b)
            return new_probaliblity_a, new_probaliblity_b
        else:
            count += 1
            pass

observations_head = [5,9,8,4,7]  # 观察扔到正面的次数
observations_tail = [5,1,2,6,3]  # 观察扔到反面的次数
origin_a = 0.6  # 初始化硬币A扔正面的概率
origin_b = 0.4 # 初始化硬币B扔正面的概率
em(origin_a, origin_b, observations_head, observations_tail)
```

运行结果得到硬币 A 与 B 正面出现的概率：

```
Number of iterations: 13
The probability that coin A will be thrown to the front: 0.7967887863838076
The probability that coin B will be thrown to the front: 0.51958314107701
[Finished in 0.1s]
```