EM算法

1 任务描述

一个袋子中三种硬币的混合比例为：s_1, s_2与$1-s_1-s_2$ $(0 <= s_i <= 1)$，三种硬币掷出正面的概率分别为：p, q, r。自己指定系数$s1, s2, p, q, r$，生成N个投掷硬币的结果（由01构成的序列，其中1为正面，0为反面），利用期望最大算法（expectation-maximization algorithm，EM算法）来对参数进行估计并与预先假定的参数进行比较。

2 算法原理

EM算法是一种从不完全数据或有数据丢失的数据集（存在隐含变量）中求解概率模型参数的最大似然估计方法。EM算法由两步组成：E步和M步，是最常用的迭代算法。以任务中的系数为例，算法步骤如下：

设 y_j 是第 j 次实验抛硬币的观测数据，$z_i=(\alpha_i,\beta_i)$ 为第 i 次迭代中的隐变量，其中 α_i 表示用硬币 A抛的概率， β_i 表示用硬币 B抛的概率，模型参数 $\theta=(s_1,s_2,p,q,r)$，第 i 次迭代时参数估计为 $\theta^{(i)}=(s_1^{(i)},s_2^{(i)},p^{(i)},q^{(i)},r^{(i)})$ 。观测数据的似然函数为：

$$P(Y|\theta)=\prod_{j=1}^{n}[s_1p^{y_j}(1-p)^{1-y_j}+s_2q^{y_j}(1-q)^{1-y_j}+(1-s_1-s_2)r^{y_j}(1-r)^{1-y_j}]$$

观测数据 Y 关于当前参数估计 $\theta^{(i)}$ 的对数似然函数为：

$$L(\theta)=\log P(Y|\theta)=\log\left(\sum_Z P(Y|Z,\theta)P(Z|\theta)\right)$$

我们希望迭代参数能使得 $L(\theta)$ 极大化，取两次迭代的差值：

$$\begin{aligned}L(\theta)-L\left(\theta^{(i)}\right)&=\log\left(\sum_Z P\left(Z\mid Y,\theta^{(i)}\right)\frac{P(Y\mid Z,\theta)P(Z\mid\theta)}{P\left(Z\mid Y,\theta^{(i)}\right)}\right)-\log P\left(Y\mid\theta^{(i)}\right)\\&\geqslant\sum_Z P\left(Z\mid Y,\theta^{(i)}\right)\log\frac{P(Y\mid Z,\theta)P(Z\mid\theta)}{P\left(Z\mid Y,\theta^{(i)}\right)}-\log P\left(Y\mid\theta^{(i)}\right)\\&=\sum_Z P\left(Z\mid Y,\theta^{(i)}\right)\log\frac{P(Y\mid Z,\theta)P(Z\mid\theta)}{P\left(Z\mid Y,\theta^{(i)}\right)P\left(Y\mid\theta^{(i)}\right)}\end{aligned}$$

则迭代过程可表示为：

$$\begin{aligned}\theta^{(i+1)}&=\arg\max_\theta\left(L\left(\theta^{(i)}\right)+\sum_Z P\left(Z\mid Y,\theta^{(i)}\right)\log\frac{P(Y\mid Z,\theta)P(Z\mid\theta)}{P\left(Z\mid Y,\theta^{(i)}\right)P\left(Y\mid\theta^{(i)}\right)}\right)\\&=\arg\max_\theta\left(\sum_Z P\left(Z\mid Y,\theta^{(i)}\right)\log(P(Y\mid Z,\theta)P(Z\mid\theta))\right)\\&=\arg\max_\theta\left(\sum_Z P\left(Z\mid Y,\theta^{(i)}\right)\log P(Y,Z\mid\theta)\right)\end{aligned}$$

定义 Q 函数：

$$Q\left(\theta,\theta^{(i)}\right)\hat{=}\sum_Z P\left(Z\mid Y,\theta^{(i)}\right)\log P(Y,Z\mid\theta)$$

则问题转化为 $\arg\max_\theta Q(\theta,\theta_i)$. 代入本问题，得：

$$Q(\theta,\theta_i)=\sum_{j=1}^{n}\{\alpha_j^{(i+1)}[\log s_1+y_j\log p+(1-y_i)\log(1-p)]+\beta_j^{(i+1)}[\log s_2+y_j\log q+(1-y_j)\log(1-q)]+(1-\alpha_j^{(i+1)}-\beta_j^{(i+1)})[\log(1-s_1$$

2.1 E _Step

已知第 i 次迭代得参数估计为 $\theta^{(i)}$ ，在该参数下观测数据 y_j 来自硬币 A 的概率为：

$$\alpha_j^{(i+1)}=\frac{s_1^{(i)}(p^{(i)})^{y_j}(1-p^{(i)})^{1-y_j}}{s_1^{(i)}(p^{(i)})^{y_j}(1-p^{(i)})^{1-y_j}+s_2^{(i)}(q^{(i)})^{y_j}(1-q^{(i)})^{1-y_j}+(1-s_1^{(i)}-s_2^{(i)})(r^{(i)})^{y_j}(1-r^{(i)})^{1-y_j}}$$

来自硬币 B 的概率为：

$$\beta_j^{(i+1)}=\frac{s_2^{(i)}(q^{(i)})^{y_j}(1-q^{(i)})^{1-y_j}}{s_1^{(i)}(p^{(i)})^{y_j}(1-p^{(i)})^{1-y_j}+s_2^{(i)}(q^{(i)})^{y_j}(1-q^{(i)})^{1-y_j}+(1-s_1^{(i)}-s_2^{(i)})(r^{(i)})^{y_j}(1-r^{(i)})^{1-y_j}}$$

2.2 M _Step

要极大化 $Q(\theta,\theta_i)$，需对参数求偏导。对 s_1, s_2 ：

$$\frac{\partial Q}{\partial s_1}=\sum_{j=1}^{n}[\frac{\alpha_j^{(i+1)}}{s_1}-\frac{1-\alpha_j^{(i+1)}-\beta_j^{(i+1)}}{1-s_1-s_2}]=0$$

$$\frac{\partial Q}{\partial s_2}=\sum_{j=1}^{n}[\frac{\beta_j^{(i+1)}}{s_2}-\frac{1-\alpha_j^{(i+1)}-\beta_j^{(i+1)}}{1-s_1-s_2}]=0$$

解得$s_1=\frac{1}{n}\sum_{j=1}^{n}\alpha_j^{(i+1)}, s_2=\frac{1}{n}\sum_{j=1}^{n}\beta_j^{(i+1)}$

再对 p, q, r 求偏导，由：

$$\frac{\partial Q}{\partial p} = \sum_{j=1}^{n} \alpha_j^{(i+1)} [\frac{y_j}{p} - \frac{1-y_j}{1-p}] = 0$$

得 $p^{(i+1)} = \frac{\sum_{j=1}^{n} \alpha_j^{(i+1)} y_j}{\sum_{j=1}^{n} \alpha_j^{(i+1)}}$. 同理有 $q^{(i+1)} = \frac{\sum_{j=1}^{n} \beta_j^{(i+1)} y_j}{\sum_{j=1}^{n} \beta_j^{(i+1)}}$, $r^{(i+1)} = \frac{\sum_{j=1}^{n} (1-\alpha_j^{(i+1)}-\beta_j^{(i+1)}) y_j}{\sum_{j=1}^{n} (1-\alpha_j^{(i+1)}-\beta_j^{(i+1)})}$

再由迭代得参数，重复进行E步骤和M步骤，直到达到最大迭代次数或参数收敛（即 $\left\| \theta^{(i+1)} - \theta^{(i)} \right\| < \varepsilon$）.

3 任务过程

首先自己设定好真实的参数值，这里分别为s_1, s_2, p, q, r，根据参数生成投掷n次硬币的观测结果

```
def init_data(s1, s2, p, q, r, n):
    data = []
    for i in range(n):
        coin = random()
        if 0 <= coin < s1:
            side = np.random.binomial(1, p)
        elif s1 <= coin < s1 + s2:
            side = np.random.binomial(1, q)
        else:
            side = np.random.binomial(1, r)
        data.append(side)
    if isdebug:
        print(data)
    return data
```

再将初始的参数估计、迭代的终止条件和之前生成的观测数据灌入，开始不断迭代E步骤和M步骤：

```
def EM(theta, y, iter_num, epsilon):
    s1 = theta[0]
    s2 = theta[1]
    p  = theta[2]
    q  = theta[3]
    r  = theta[4]
    n  = len(y)
    i = 0
    threshold = 99999
    while(i < iter_num and threshold >= epsilon):
        # E_Step
        a = np.random.rand(n)
        b = np.random.rand(n)
        for j in range(n):
            a[j] = (s1*pow(p,y[j])*pow(1-p,1-y[j]))/(s1*pow(p,y[j])*pow(1-p,1-y[j])+s2*pow(q,y[j])*pow(1-q,1-
y[j])+(1-s1-s2)*pow(r,y[j])*pow(1-r,1-y[j]))
            b[j] = (s2*pow(q,y[j])*pow(1-q,1-y[j]))/(s1*pow(p,y[j])*pow(1-p,1-y[j])+s2*pow(q,y[j])*pow(1-q,1-
y[j])+(1-s1-s2)*pow(r,y[j])*pow(1-r,1-y[j]))
        # M_Step
        s1_next = 1/n * sum(a)
        s2_next = 1/n * sum(b)
        p_next = sum([a[j]*y[j] for j in range(n)]) / sum(a)
        q_next = sum([b[j]*y[j] for j in range(n)]) / sum(b)
        r_next = sum([(1-a[j]-b[j])*y[j] for j in range(n)]) / sum([(1-a[j]-b[j]) for j in range(n)])
        # Threshold
        threshold = np.linalg.norm(np.array([s1-s1_next,s2-s2_next,p-p_next,q-q_next,r-r_next]), ord = 2)
        s1 = s1_next
        s2 = s2_next
        p  = p_next
        q  = q_next
        r  = r_next
        i += 1
        print(i, [s1, s2, p, q, r])
    return s1, s2, p, q, r
```

4 任务结果

设定真实的参数值为$s_1 = 0.2, s_2 = 0.4, p = 0.3, q = 0.4, r = 0.7$，投掷次数为$n = 50$;

设定初始的参数分别为$s_1 = 0.3, s_2 = 0.3, p = 0.2, q = 0.5, r = 0.6$，与真实参数相差较小，多次运行该算法，在不同的观测数据下，发现获得的结果有时与真实值较为接近，有时较远，但基本上还是比较接近的。

```
观测数据:
[1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1,
0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0]
1 [0.2666666666666665, 0.3066666666666669, 0.28000000000000014, 0.6086956521739124, 0.6999999999999998]
2 [0.26666666666666633, 0.30666666666666703, 0.2800000000000003, 0.6086956521739122, 0.7000000000000001]
3 [0.2666666666666663, 0.30666666666666703, 0.28000000000000036, 0.6086956521739121, 0.7000000000000001]
4 [0.2666666666666663, 0.3066666666666671, 0.28000000000000036, 0.6086956521739121, 0.7]
5 [0.2666666666666663, 0.3066666666666671, 0.28000000000000036, 0.6086956521739121, 0.7]
```

```
观测数据:
[1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0]
1 [0.2666666666666665, 0.3066666666666669, 0.2800000000000002, 0.6086956521739125, 0.6999999999999996]
```

```
2 [0.2666666666666663, 0.3066666666666671, 0.2800000000000004, 0.6086956521739122, 0.7000000000000005]
3 [0.26666666666666616, 0.30666666666666714, 0.28000000000000047, 0.6086956521739119, 0.7]
4 [0.26666666666666605, 0.3066666666666673, 0.2800000000000006, 0.6086956521739116, 0.6999999999999998]
5 [0.26666666666666605, 0.30666666666666736, 0.28000000000000064, 0.6086956521739116, 0.7]
6 [0.26666666666666605, 0.3066666666666677, 0.28000000000000064, 0.608695652173911, 0.7000000000000005]
7 [0.26666666666666605, 0.3066666666666679, 0.28000000000000064, 0.6086956521739105, 0.7000000000000012]
8 [0.266666666666666, 0.3066666666666679, 0.2800000000000007, 0.6086956521739103, 0.7000000000000007]
9 [0.2666666666666657, 0.3066666666666681, 0.28000000000000097, 0.6086956521739102, 0.7000000000000007]
10 [0.26666666666666555, 0.3066666666666681, 0.28000000000000114, 0.6086956521739101, 0.7000000000000005]
11 [0.2666666666666654, 0.3066666666666683, 0.2800000000000013, 0.6086956521739096, 0.7000000000000004]
12 [0.2666666666666653, 0.3066666666666686, 0.2800000000000014, 0.6086956521739091, 0.7000000000000004]
13 [0.2666666666666653, 0.3066666666666687, 0.28000000000000147, 0.608695652173909, 0.7000000000000005]
14 [0.2666666666666652, 0.3066666666666688, 0.280000000000015, 0.6086956521739095, 0.7000000000000012]
15 [0.26666666666666516, 0.30666666666666886, 0.28000000000000147, 0.6086956521739094, 0.7000000000000014]
16 [0.26666666666666516, 0.30666666666666886, 0.2800000000000014, 0.6086956521739094, 0.7000000000000014]
17 [0.26666666666666516, 0.30666666666666886, 0.2800000000000014, 0.6086956521739094, 0.7000000000000014]
```

改变真实参数值，设定真实的参数值为$s_1 = 0.1, s_2 = 0.6, p = 0.8, q = 0.2, r = 0.3$，使得真实参数和初始参数相差较远，运行该算法，发现结果与真实参数相差较远。

```
观测数据：
[1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0]
1 [0.3454545454545452, 0.29090909090909106, 0.11578947368421061, 0.3437499999999998, 0.43999999999999967]
2 [0.34545454545454496, 0.2909090909090914, 0.11578947368421068, 0.34374999999999944, 0.43999999999999967]
3 [0.3454545454545449, 0.2909090909090915, 0.1157894736842107, 0.34374999999999956, 0.4400000000000001]
4 [0.3454545454545449, 0.2909090909090915, 0.11578947368421069, 0.34374999999999956, 0.4400000000000001]
5 [0.3454545454545449, 0.2909090909090915, 0.11578947368421068, 0.34374999999999956, 0.4400000000000001]
6 [0.3454545454545449, 0.2909090909090915, 0.11578947368421068, 0.34374999999999956, 0.4400000000000001]
```