DL-NLP 第二次作业

一、问题描述

一个袋子中三种硬币的混合比例为:s1, s2 与 1-s1-s2 (0<=si<=1), 三种硬币 掷出正面的概率分别为:p, q, r。

(1)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)

(2)利用 EM 算法来对参数进行估计并与预先假定的参数进行比较。

二、背景介绍

EM算法是一种迭代优化策略，由于它的计算方法中每一次迭代都分两步，其中一个为期望步（E步），另一个为极大步（M步），所以算法被称为EM算法（Expectation Maximization Algorithm）。其基本思想是：首先根据己经给出的观测数据，估计出模型参数的值；然后再依据上一步估计出的参数值估计缺失数据的值，再根据估计出的缺失数据加上之前己经观测到的数据重新再对参数值进行估计，然后反复迭代，直至最后收敛，迭代结束。

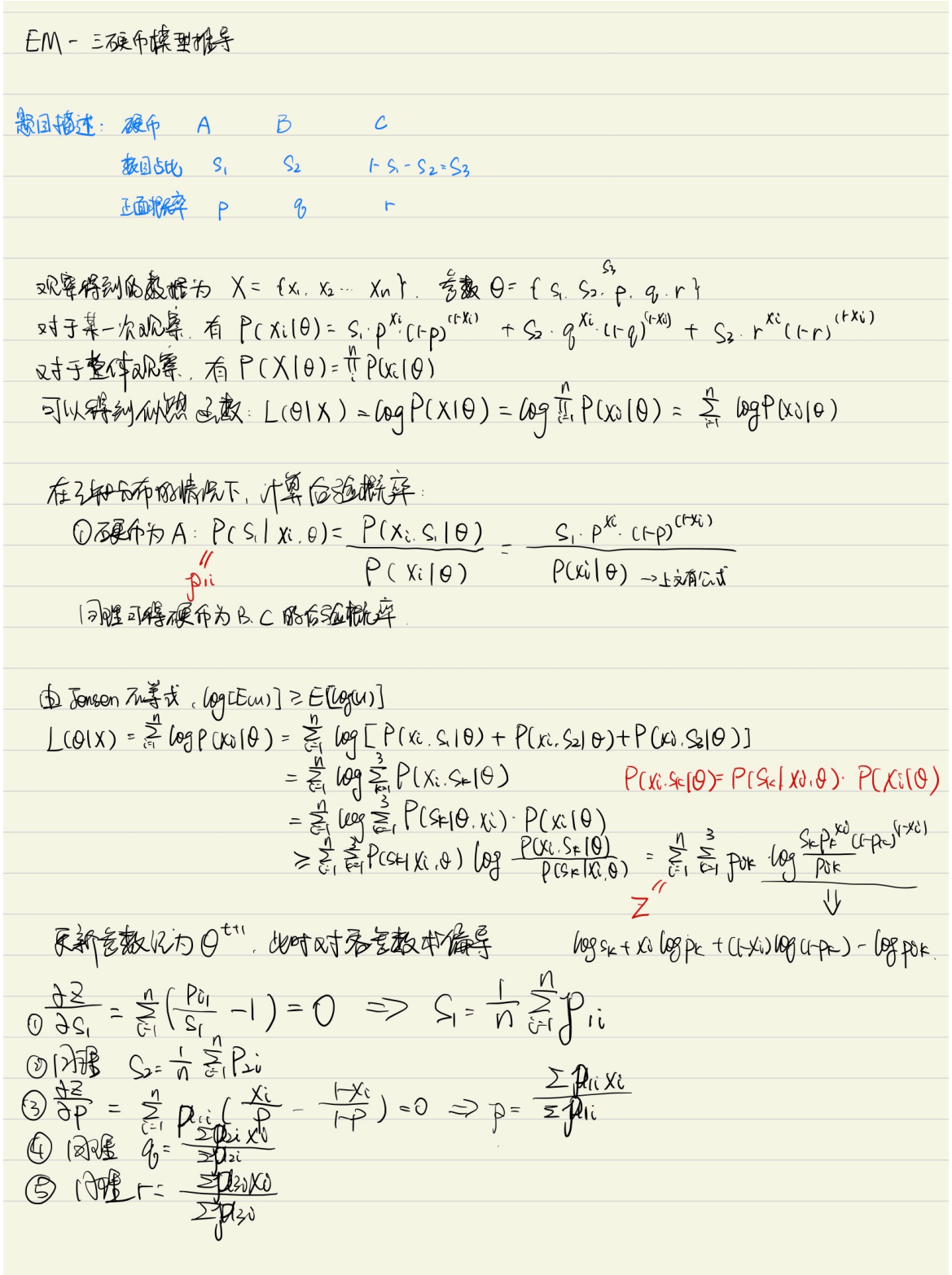
EM算法流程：

Step 1：选择EM算法的初值 （初值对结果由影响，EM算法只能达到局部最优）

Step 2：E步：就是求Q函数，

Step 3：M步：使参数 [公式] 的似然函数增大，达到局部最大值；

三、公式推导



四、代码介绍

- 参数定义

  theta 表示给定的真实参数；theta_init 表示迭代一次后的参数；theta_record 记录了每一次迭代的结果；times 表示一共进行10000次实验；x_seq 记录硬币结果序列；group_size 表示一组实验对同一个硬币抛掷多少次

```
theta = {'s1':0.2, 's2':0.5, 'p': 0.8, 'q': 0.2, 'r': 0.5}
theta_init = {'s1':0.8, 's2':0.1, 'p': 0.7, 'q': 0.3, 'r': 0.4}
theta_record = {'s1':[0.8], 's2':[0.1], 'p': [0.7], 'q': [0.3], 'r':
[0.4]}
times = 10000
x_seq = []
group_size = 20
```

- 模拟生成序列

```
def generate_res():
    for i in range(times):
        xx_seq = []
        rand_value = random.random()
        for k in range(group_size):
            if rand_value < theta['s1']:
                tmp = 1 if random.random() > theta['p'] else 0
            elif rand_value < theta['s1'] + theta['s2']:
                tmp = 1 if random.random() > theta['q'] else 0
            else:
                tmp = 1 if random.random() > theta['r'] else 0
            xx_seq.append(tmp)
        x_seq.append(xx_seq)
```

- 执行EM算法（根据推导公式）

```
def em():
    p1i_record = []
    p2i_record = []
    p3i_record = []
    for xi in x_seq:
        real_xi = sum(xi)
        real_neg_xi = len(xi) - real_xi
        p = theta_init['p']
        q = theta_init['q']
        r = theta_init['r']
        s1 = theta_init['s1']
        s2 = theta_init['s2']
        s3 = 1 - theta_init['s1'] - theta_init['s2']
        pA = s1 * pow(p, real_xi) * pow(1-p, real_neg_xi)
        pB = s2 * pow(q, real_xi) * pow(1-q, real_neg_xi)
        pC = s3 * pow(r, real_xi) * pow(1-r, real_neg_xi)
        p1i = pA / (pA + pB + pC)
        p2i = pB / (pA + pB + pC)
        p3i = pC / (pA + pB + pC)
        p1i_record.append(p1i)
        p2i_record.append(p2i)
        p3i_record.append(p3i)
```

```
    theta_init['s1'] = sum(p1i_record)/len(x_seq)
    theta_init['s2'] = sum(p2i_record)/len(x_seq)
    theta_init['p'] = sum(p1i_record[i]*sum(x_seq[i])/len(x_seq[i]) for i
in range(len(x_seq))) / sum(p1i_record)
    theta_init['q'] = sum(p2i_record[i]*sum(x_seq[i])/len(x_seq[i]) for i
in range(len(x_seq))) / sum(p2i_record)
    theta_init['r'] = sum(p3i_record[i]*sum(x_seq[i])/len(x_seq[i]) for i
in range(len(x_seq))) / sum(p3i_record)
    record()
```

- 记录参数的变化

```
def record():
    theta_record['s1'].append(theta_init['s1'])
    theta_record['s2'].append(theta_init['s2'])
    theta_record['p'].append(theta_init['p'])
    theta_record['q'].append(theta_init['q'])
    theta_record['r'].append(theta_init['r'])
```

五、实验结果

在实验过程中，有些给定参数

- times = 10000 表示一共执行10000次实验，即抛掷10000次硬币
- group_size = ? 表示一组实验的大小。本组实验抛掷同一枚硬币。
- epoch = 20 表示迭代20次
-

参数	s1	s2	p	q	r
真实值	0.2	0.5	0.8	0.2	0.5

-

参数	s1	s2	p	q	r
起始值	0.8	0.1	0.4	0.5	0.7

group_size	s1	s2	p	q	r	s3
1	0.7796	0.1166	0.8147	0.5568	0.6533	0.1038
5	0.4003	0.4329	0.7956	0.3111	0.7535	0.1668
20	0.2059	0.5028	0.798	0.2047	0.5073	0.2913
50	0.2073	0.4981	0.8004	0.2011	0.4993	0.2946
100	0.1973	0.5012	0.7996	0.2008	0.499	0.3015
500	0.2066	0.493	0.8002	0.1998	0.4997	0.3004

- 根据实验结论可以得到，group_size这一参数会对em算法产生较大影响，因为在分组实验的时候，更容易判断出该硬币属于哪种类型，因为做了多次重复实验，可以根据实验结果初步进行估计预测。
- 若不进行分组实验，则会导致有较大误差，极大似然估计容易困在局部最优值的情况
- 当 group_size 大于等于20时，根据结果可以发现能够很好的预测出真实参数了