深度学习与自然语言处理
第二次大作业

EM 算法

院（系）名称	自动化科学与电气工程学院
学 生 学 号	ZY2103803
学 生 姓 名	李鑫磊

2022 年 04 月

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。

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 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、具体算法实现

EM 算法是一种迭代算法，1977 年由 Dempster 等人总结提出，用于含有隐变量的概率模型参数的最大似然估计。

三种硬币的混合比例是 $s_1 : s_2 : 1-s_1-s_2$，三种硬币掷出正面的概率分别为：π, p, q，隐变量 z 表示硬币属于第几种，对于单次抛硬币的结果，可用以下模型来表示：

$$\begin{aligned} p(y_i \mid \theta) &= \sum_z p(y_i, z \mid \theta) \\ &= \sum_z p(z \mid \theta) p(y_j \mid z, \theta) \\ &= s_1 \pi^{y_j} (1-\pi)^{1-y_j} + s_2 p^{y_j} (1-p)^{1-y_j} + (1-s_1-s_2) q^{y_j} (1-q)^{1-y_j} \end{aligned}$$

观测数据的似然函数可表示为：

$$p(y \mid \theta) = \prod_{j=1}^{n} [s_1 \pi^{y_j} (1-\pi)^{1-y_j} + s_2 p^{y_j} (1-p)^{1-y_j} + (1-s_1-s_2) q^{y_j} (1-q)^{1-y_j}]$$

假设 y_i 来自硬币 1 的概率为 u_{1j}，来自硬币 2 的概率为 u_{2j}，完全数据的似然函数可表示为：

$$p(y \mid \theta) = \prod_{j=1}^{n} \{[s_1 \pi^{y_j} (1-\pi)^{1-y_j}]^{u_{1j}} + [s_2 p^{y_j} (1-p)^{1-y_j}]^{u_{2j}} + [(1-s_1-s_2) q^{y_j} (1-q)^{1-y_j}]^{1-u_{1j}-u_{2j}}\}$$

相应的对数似然函数为：

$$\log p(y, u \mid \theta) = \sum_{j=1}^{n} u_{1j} [\log s_1 + y_j \log \pi + (1-y_j) \log(1-\pi)]$$

$$+ u_{2j} [\log s_2 + y_j \log p + (1-y_j) \log(1-p)]$$

$$+(1-u_{1j}-u_{2j})[\log(1-s_1-s_2)+y_j\log q+(1-y_j)\log(1-q)]$$

E-step：

EM 算法是在不断迭代中完成的，对于第 i+1 次迭代，其中的隐变量为：

$$u_{1j}^{(i+1)}=\frac{s_1\pi^{y_j}(1-\pi)^{1-y_j}}{s_1\pi^{y_j}(1-\pi)^{1-y_j}+s_2p^{y_j}(1-p)^{1-y_j}+(1-s_1-s_2)q^{y_j}(1-q)^{1-y_j}}$$

$$u_{2j}^{(i+1)}=\frac{s_2p^{y_j}(1-p)^{1-y_j}}{s_1\pi^{y_j}(1-\pi)^{1-y_j}+s_2p^{y_j}(1-p)^{1-y_j}+(1-s_1-s_2)q^{y_j}(1-q)^{1-y_j}}$$

$$u_{3j}^{(i+1)}=1-u_{1j}^{(i+1)}-u_{2j}^{(i+1)}$$

我们可以直接求取 Q 函数：

$$Q(\theta,\theta_i)=\sum_z p(z\mid y,\theta_i)\log p(y,z\mid\theta)=E_z[\log(y,z\mid\theta,\theta^{(i)})]$$

将上述隐变量代入 Q 函数可得：

$$\begin{aligned}Q(\theta,\theta_i)=&\sum_{j=1}^{n}u_{1j}^{(i+1)}[\log s_1+y_j\log\pi+(1-y_j)\log(1-\pi)]\\&+u_{2j}^{(i+1)}[\log s_2+y_j\log p+(1-y_j)\log(1-p)]\\&+(1-u_{1j}^{(i+1)}-u_{2j}^{(i+1)})[\log(1-s_1-s_2)+y_j\log q+(1-y_j)\log(1-q)]\end{aligned}$$

M-step：

得到 Q 函数后，对参数进行极大化：

$$\theta^{(i+1)}=\arg\max_{\theta}Q(\theta,\theta^{i})$$

使用 Q 函数分别对不同参数求导并令其为 0，可得：

$$s_1^{(i+1)}=\frac{\sum_{j=1}^{n}u_{1j}^{(i+1)}}{n}$$

$$s_2^{(i+1)}=\frac{\sum_{j=1}^{n}u_{2j}^{(i+1)}}{n}$$

$$\pi^{(i+1)}=\frac{\sum_{j=1}^{n}u_{1j}^{(i+1)}y_j}{\sum_{j=1}^{n}u_{1j}^{(i+1)}}$$

$$p^{(i+1)}=\frac{\sum_{j=1}^{n}u_{2j}^{(i+1)}y_j}{\sum_{j=1}^{n}u_{2j}^{(i+1)}}$$

$$q^{(i+1)}=\frac{\sum_{j=1}^{n}(1-u_{1j}^{(i+1)}-u_{2j}^{(i+1)})y_j}{\sum_{j=1}^{n}(1-u_{1j}^{(i+1)}-u_{2j}^{(i+1)})}$$

三、运行结果

随机生成 1000 个 0 和 1 的序列数据，然后设置好五个参数的初始值，令其迭代 100 次之后，得到的结果如下表所示：

表 1 自行生成数据参数估计结果

参数	初始值	迭代 100 次后的值
s_1	0.2	0.1806
s_2	0.4	0.3936
π	0.8	0.6568
p	0.7	0.5274
q	0.6	0.4177

由表中我们可以得出，三种硬币的数量占比并没有发生太大变化，迭代后的结果与初始值相近，而三种硬币抛出正面的概率，随着迭代的次数会渐渐趋向于 0.5，可能是数据是随机生成的原因。

四、个人总结和体会

通过本次大作业的算法设计，对于 EM 算法的理解更加透彻，也理解了初始值对于 EM 算法所起到的关键性作用。同时提高了 Python 语言的编程能力，理解了参数的选择

对于算法效果的重要性。

参考：

https://blog.csdn.net/weiwei19890308/article/details/82943969

https://github.com/Asbee1/DL_NLP_homework2