形状, 圆圈

描述已自动生成

**卡通人物

中度可信度描述已自动生成**

**深度学习与自然语言处理第二次作业**

EM算法计算硬币分布

|  |  |
| --- | --- |
| 院（系）名称 | 自动化科学与电气工程学院 |
| 专业名称 | 电子信息 |
| 学生姓名 | 彭程 |
| 学号 | ZY2103807 |
| 指导老师 | 秦曾昌 |

2022年 4月

# 实验背景

## 问题描述

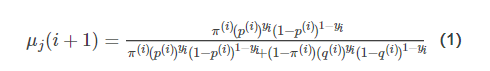
一个袋子中三种硬币的混合比例为：s1, s2与1-s1-s2 (0<=si<=1), 三种硬币掷出正面的概率分别为：p, q, r。指定系数s1, s2, p, q, r，生成N个投掷硬币的结果（由01构成的序列，其中1为正面，0为反面），利用EM算法来对参数进行估计并与预先假定的参数进行比较。

## 实验原理

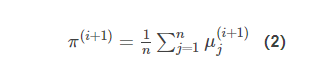
**1.2.1 EM算法简介**

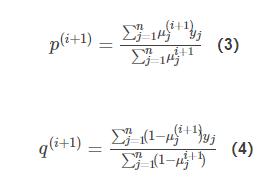
EM算法是一种迭代优化策略，用于含有隐变量（Hidden Variable）的概率模型参数的极大似然估计。EM算法的每次迭代由两步组成：E步，求期望（expectation）；M步，求极大（maximum），所以算法被称为EM算法（Expectation-Maximization Algorithm）。EM算法受到缺失思想影响，最初是为了解决数据缺失情况下的参数估计问题，其算法基础和收敛有效性等问题在Dempster、Laird和Rubin三人于1977年所做的文章《Maximum likelihood from incomplete data via the EM algorithm》中给出了详细的阐述。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。

**1.2.2 算法原理**

E步：计算在模型参数下观测数据yj来自掷硬币B的概率

M步：计算模型参数的新估计值：





# 实验方法

## 算法流程

输入：观察变量数据*x* = (*x1,x2...xn*), 隐变量z，联合分布概率P(*x ,z | θ*)，条件分布概率P(*z |x, θ*)，迭代次数iterNum；

输出：模型参数*θ*；

1. 选定模型参数的初始值*θ0*；
2. *j*= 1,2,3,…, iterNum,利用EM算法进行计算；

E步：计算联合分布的条件概率期望：

M步：求使极大化的，确定第*j*+1次迭代的参数估计 *j+1*

1. 重复迭代步骤直至算法收敛或到达迭代次数后退出。

## 公式推导

设初始参数构成为,根据参数生成一个长度为N的投掷序列，投掷结果可以用一个长度为N的01序列*x* = (*x1,x2...xn*)，*xi* = 0 或1。则各个种类的硬币掷出的条件分布概率为：

联合三类硬币出现的概率，可以得到投掷硬币结果的联合分布概率：

由此，可以估算出三种硬币的概率为：

进行加权处理可得：

那么，每类硬币是正面的概率分布为：

# 实验结果与分析

## 生成随机投掷序列

投掷数据为1-N的01序列，按照给定参数值生成序列：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 参数 | s1 | s2 | p | q | r | N |
| 数值 | 0.3 | 0.2 | 0.8 | 0.6 | 0.7 | 10000 |

表 1. 序列参数表



图1. 投掷结果

## 实验结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 参数 | s1 | s2 | p | q | r | iterNum |
| 真实值 | 0.3 | 0.2 | 0.8 | 0.6 | 0.7 | 1000 |
| 初始值 | 0.4 | 0.2 | 0.4 | 0.5 | 0.6 | 1000 |
| 预测值 | 0.3674 | 0.2 | 0.6131 | 0.7039 | 0.7810 | 2 |

表 2. EM算法运行结果

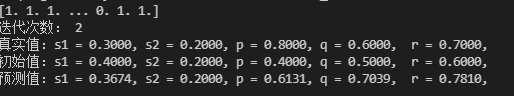


图2. EM算法运行结果

## 结果分析

通过以上实验可以看出EM算法能够快速收敛，但精度较差，预测结果依赖于初始值，最终可能收敛于局部最优解

**附录：**

import math

import numpy as np

def GetData(s1,s2,p,q,r,N):

    #随机生成硬币序列 0代表反面，1代表正面

    CoinSeq = np.zeros(N)

    CoinChance = [p,q,r]

    for i in range(N):

        #抽取硬币

        rand1 = np.random.random()

        if rand1 < s1:

            coin = 0

        elif rand1 <s1+s2:

            coin = 1

        else:

            coin = 2

        #掷硬币

        rand2 = np.random.random()

        if rand2 < CoinChance[coin]:

            CoinSeq[i] = 1

    return CoinSeq

#隐参数计算

def EStep(s1, s2, p, q, r, coinseq):

    Z1, Z2, Z3 = [], [], []

    for x in coinseq:

        Miu1 = s1\*pow(p, x)\*pow(1-p,1-x)

        Miu2 = s2\*pow(q, x)\*pow(1-q,1-x)

        Miu3 = (1 - s1 -s2)\*pow(r, x)\*pow(1-r, 1-x)

        sum = Miu1 + Miu2 + Miu3

        Miu1 = Miu1/sum

        Miu2 = Miu2/sum

        Miu3 = Miu3/sum

        Z1.append(Miu1)

        Z2.append(Miu2)

        Z3.append(Miu3)

    return [Z1, Z2, Z3]

def MStep(z, coinseq):

    s,p = [],[]

    for i in range(len(z)):

        sum1 = 0.0

        if i < 2:

            s.append(sum(z[i])/len(z[i]))

        for j,x in enumerate(coinseq):

            sum1 += z[i][j]\*x

        p.append(sum1/sum(z[i]))

    return s,p

def EM(s, p, coinseq, iterNum, epsilon):

    s\_pre, p\_pre = s, p

    for i in range(iterNum):

        z = EStep(s\_pre[0],s\_pre[1],p\_pre[0],p\_pre[1],p\_pre[2],coinseq)

        s\_temp , p\_temp = MStep(z,coinseq)

        sum1, sum2 = 0.0, 0.0

        for j in range(len(s\_temp)):

            sum1 += abs(s\_temp[j] -  s\_pre[j])

        for k in range(len(p\_temp)):

            sum2 += abs(p\_temp[k] -  p\_pre[k])

        if (sum1 + sum2 < epsilon):

            print("迭代次数：", i+1)

            break

        s\_pre, p\_pre = s\_temp, p\_temp

    return  s\_pre, p\_pre

if \_\_name\_\_ == '\_\_main\_\_':

    #参数预设

    s1, s2, p, q, r, N = 0.3, 0.2, 0.8, 0.6, 0.7, 10000

    coinseq = GetData(s1,s2,p,q,r,N)

    print(coinseq)

    start\_s = [0.4,0.2]

    start\_p = [0.4, 0.5, 0.6]

    s\_p,p\_p = EM(start\_s, start\_p, coinseq, 10000, 0.0000001)

    print("真实值：s1 = {:.4f}, s2 = {:.4f}, p = {:.4f}, q = {:.4f},  r = {:.4f},".format(s1, s2, p, q, r))

    print("初始值：s1 = {:.4f}, s2 = {:.4f}, p = {:.4f}, q = {:.4f},  r = {:.4f},".format(start\_s[0],start\_s[1], start\_p[0], start\_p[1], start\_p[2]))

    print("预测值：s1 = {:.4f}, s2 = {:.4f}, p = {:.4f}, q = {:.4f},  r = {:.4f},".format(s\_p[0],s\_p[1], p\_p[0], p\_p[1], p\_p[2]))