参数估计

哈爾濱工業大學

随机过程实验

实验题目 参数估计

学　　号 1190200122

姓　　名 袁野

指导教师 范晓鹏

日　　期 2021.9.25

一、 实验目的

掌握参数估计的方法，通过编写代码模拟随机生成分布的过程和方式，理解分布的内涵

二、 实验内容

1、 随机数和随机序列的生成

1.1 生成随机序列 X，其中每个 Xi 服从[-1,1]的均匀分布。

1.2 生成随机序列 Y，每个 Yi 服从[-1,1]的均匀分布。

1.3 利用随机序列{(Xi,Yi)}计算圆周率（蒙特卡罗投点法）。

2、 随机分布的计算机模拟

2.1 利用 normrnd 生成均值为 10，方差为 5 的正态分布。

2.2 利用 1 生成的样本估计分布的均值和方差，并画出均值和方差随样本数增加而变化的图。

2.3 敌坦克分队对我方阵地实施突袭，其到达规律服从泊松分布，平均每分钟到达 4 辆.（1）模拟敌坦克在 3 分钟内到达目标区的数量，以及在第 1 、 2 、 3 分钟内各到达几辆坦克.（2）模拟在 3 分钟内每辆敌坦克的到达时刻.（1.用 poissrnd(4)进行模拟。2.用 exprnd（1/4）模拟）。

3、 EM 算法

下表中有 30 条水果数据，包括密度和含糖率，根据其特征将水果分类，求其高斯混合模型（类别数 k=3）。

编号	密度	含糖率	编号	密度	含糖率	编号	密度	含糖率
1	0.697	0.460	11	0.245	0.057	21	0.748	0.232
2	0.774	0.376	12	0.343	0.099	22	0.714	0.346
3	0.634	0.264	13	0.639	0.161	23	0.483	0.312
4	0.608	0.318	14	0.657	0.198	24	0.478	0.437
5	0.556	0.215	15	0.360	0.370	25	0.525	0.369
6	0.403	0.237	16	0.593	0.042	26	0.751	0.489
7	0.481	0.149	17	0.719	0.103	27	0.532	0.472
8	0.437	0.211	18	0.359	0.188	28	0.473	0.376
9	0.666	0.091	19	0.339	0.241	29	0.725	0.445
10	0.243	0.267	20	0.282	0.257	30	0.446	0.459

三、 实验过程

1、 随机数和随机序列的生成

i. 使用 numpy 中的 random 函数生成长度为10^7的服从[0,1]均匀分布的序列，然后将其均匀扩展到$[-1,1]$上从而构成序列 X 和 Y。

ii. 蒙特卡洛投点法计算圆周率：正方形内部有一个相切的圆，它们的面积之比是$\frac{\pi r^2}{(2r)^2}=\frac{\pi}{4}$。我们随机生成的序列 X 和 Y 可以看作是对应$10^7$个点的坐标，由于 X 和 Y 是服从[0,1]均匀分布，因此我们可以认为这些点均匀撒在边长为 2 的正方形里，我们可以通过计算距离正方形中心的距离小于 1 的点的个数来计算有多少个点在这个正方形的内切圆内。由二者点数之比我们就能大概知道面积之比，再代入公式计算即可得到π。

以上代码为 1.1.py

2、 随机分布的计算机模拟

i. 使用 numpy 中的 normal 函数生成长度递增的 10000 个均值为 10，方差为 5 正态分布序列 X，再使用 numpy_array 中的 mean()和 var()函数分别计算不同长度的正态分布序列的均值和方差。

ii. 记录下来之后用 matplotlib.pyplot 函数画出均值和方差随样本数量增加而变化的折线图。
以上代码为 1.2.1_2.py

iii. 使用 numpy 中的 poisson 函数生成一个均值为 4，长度为 3 的泊松分布序列，对应地这三个数字即为模拟的三分钟内敌方坦克数量的结果，分别输出后并输出和即可。由泊松分布与指数分布的关系可知，使用 numpy 中的 exponential 生成指数分布序列，每次生成一个数字之后将其加在上一个坦克达到时间上作为当前坦克的到达时间，直到当前坦克到达时间为 3 分钟以上为止。
以上代码为 1.2.3.py

3、 EM 算法

首先我们从文件中读入数据，然后如下初始化所有的参数：

$$\alpha_1=\alpha_2=\alpha_3=1/3;\quad \mu_1=x_6,\ \mu_2=x_{22},\ \mu_3=x_{27};$$

$$\Sigma_1=\Sigma_2=\Sigma_3=\begin{pmatrix}0.1 & 0.0\\ 0.0 & 0.1\end{pmatrix}.$$

然后我们写出二维正态分布的矩阵计算函数：

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})=\frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \qquad (2.43)$$

where $\boldsymbol{\mu}$ is a D-dimensional mean vector, $\boldsymbol{\Sigma}$ is a $D\times D$ covariance matrix, and $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$.

μ_i为一个 2*1 的矩阵

紧接着我们根据以下公式去更新各数据：

for $j = 1,2,\dots,m$ m表示样本的总个数

利用下面这个式子计算x_j由各混合成分生成的后验概率γ_{ji}.

$$\gamma_{ji} = \frac{\alpha_i \cdot p(x_j|\mu_i, \Sigma_i)}{\sum_{l=1}^{k} \alpha_l \cdot p(x_j|\mu_l, \Sigma_l)}$$

$for\ i = 1,2,\dots,k$

下面的三个式子更新均值向量，协方差矩阵，新混合系数，直至收敛.

$$\mu_i' = \frac{\sum_{j=1}^{m} \gamma_{ji} x_j}{\sum_{j=1}^{m} \gamma_{ji}}$$

$$\Sigma_i' = \frac{\sum_{j=1}^{m} \gamma_{ji} (x_j - \mu_i')((x_j - \mu_i')^T}{\sum_{j=1}^{m} \gamma_{ji}}$$

$$\alpha_i' = \frac{\sum_{j=1}^{m} \gamma_{ji}}{m}$$

这样的方式我们重复迭代 50 次，得到最终的后验概率。对于最终的所有点，我们观察其属于哪一类的概率最大，那么它就属于哪类，将最终结果用 plt.scatter 绘制出来。

以上代码为 1.3.py

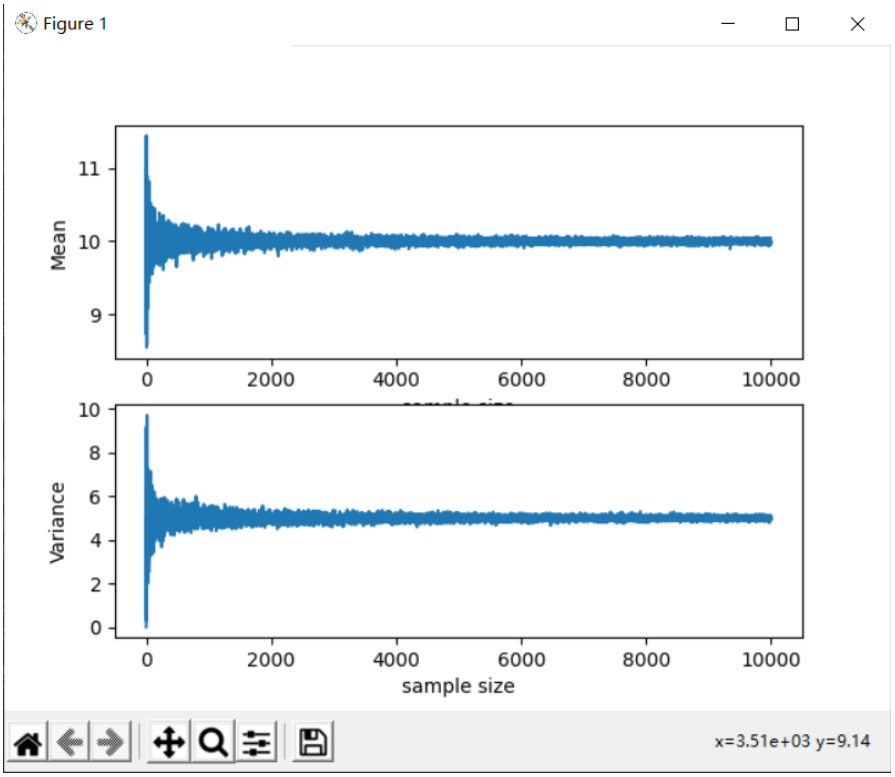
四、 实验结果

1.1.py

```
点的个数为: 10000000
前十个数字为:
[-0.62910692 -0.37604548 -0.36270584 -0.24288635 -0.17371158  0.05606735
 -0.93267126  0.07619044 -0.26603542  0.67736977]
[ 0.31738795 -0.52656137  0.36097039 -0.70104848 -0.76044239 -0.44130499
 -0.94663733  0.10176335 -0.47383892  0.9917283 ]
π= 3.1407796

Process finished with exit code 0
```

1. 2. 1_2. py



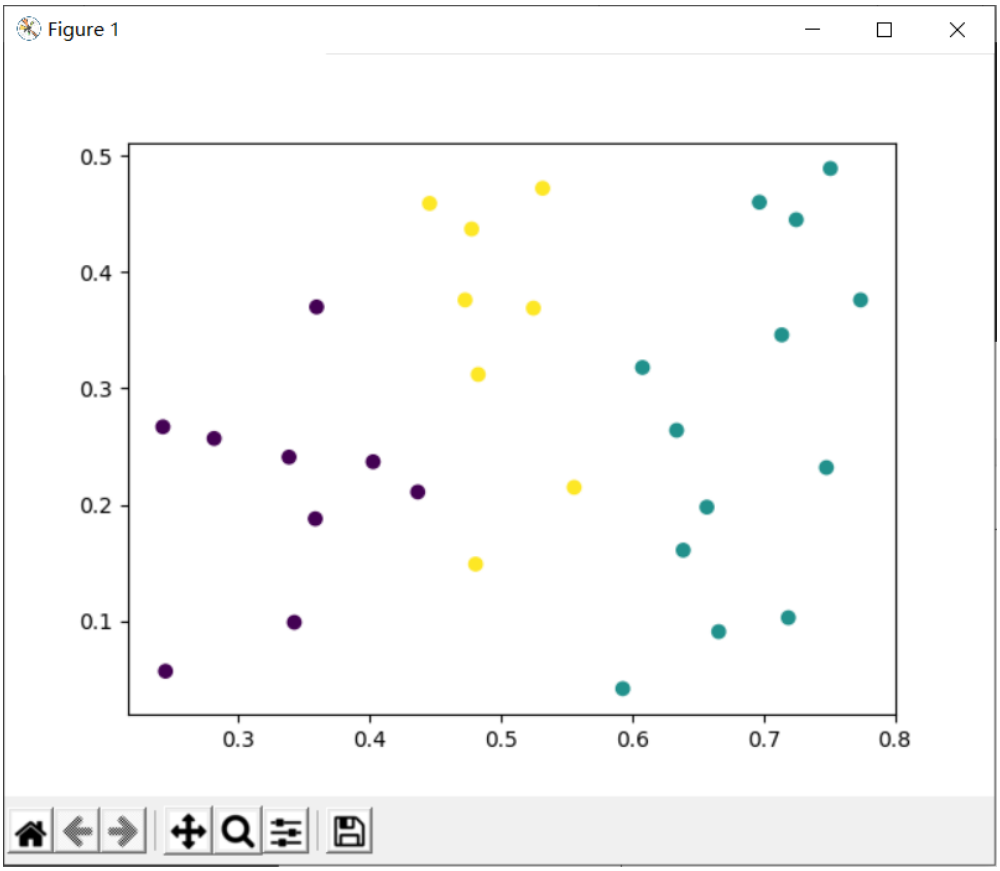
1. 2. 3. py

```
D:\Codefield\PY\计算建模\Lab1\venv\Scripts\python.exe D:/Codefield/PY/计算建模/Lab1/1.2.3.py
All: 11
the first minute: 2
the second minute: 4
the third minute: 5

All:  8
0.445140857579863
0.6469612938905352
1.3008783806980064
1.4957907210081696
1.4970692211784389
1.8644055459629638
2.8817743019817907
2.905886150672866

Process finished with exit code 0
```

1. 3. py



五、 心得体会

这次实验让我对不同分布的序列有了更深的理解，学习了如何去模拟简单的随机分布，如何去进行参数估计，学习了 EM 算法的过程。我个人的 python 基础几乎为 0，因此这次实验也带我学习了 python 语言的特点以及 python 语言中一些包的应用，比如 numpy、matplotlib.pyplot 等，而通过 python 完成随机过程实验帮助我实现了二者的有机结合，整体来说本次实验是我受益匪浅。