

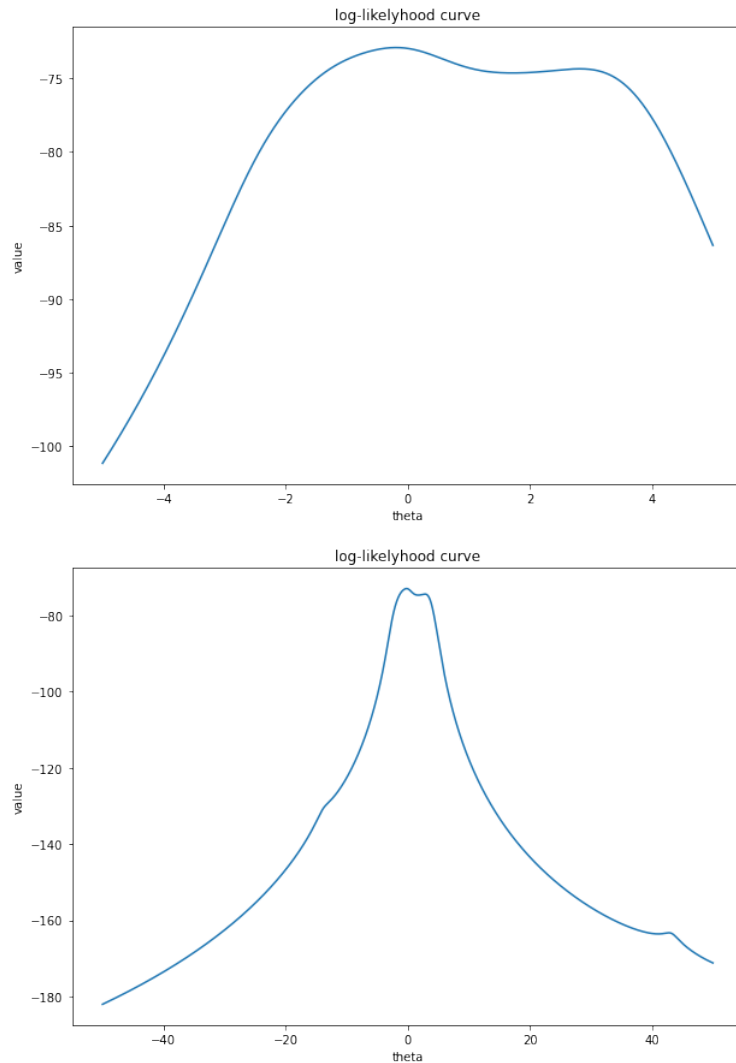
高等统计计算第 1 次作业

自硕 21 崔晏菲 2021210976

注：因为我不懂 R，所以代码都是用 python 写的

2.1

a.



初始值	收敛值
-11	-4.30537E+11
-1	-0.192286613
0	-0.192286613
1.5	1.713586835
4	2.817472166
4.7	-0.192286613
7	41.04084782
8	-6.00407E+11
38	42.79537747

可见，初始值的选取会影响最优值得收敛，这是显然的，因为这个对数似然函数

并不是凸函数。数据点的平均值不是好选择，因为柯西分布的数学期望是发散的。

b.

得到最优解为-0.5，最优值为-73.050692。显然这个方法大部分情况下并不能获得全局最优值，因为这要求函数的导数在区间内是单调的，但是我们的对数似然函数的导数并不是单调的。

c.

```
for alpha in [1, 0.64, 0.25]:
    point3 = log_likely.fixed_point_method(-1, alpha, delta = 1e-10)
    if(point3 is not None):
        print("最优解为%f, 最优值为%f"%(point3, log_likely.value(point2)))
```

```
not lipschitz continous at scale factor = 1.000000
not lipschitz continous at scale factor = 0.640000
not lipschitz continous at scale factor = 0.250000
```

不动点法很难使用，因为很难保证 lipchitz 连续。

d.

```
: log_likely.secant_method(x0=-2, x1=-1, delta = 1e-10)
```

```
: -0.1922866132319395
```

```
: log_likely.secant_method(x0=-3, x1=3, delta = 1e-10)
```

```
: 2.8174721655730948
```

可以看到，算法收敛到不同的极值点，这是很正常的，毕竟这个函数不是凸函数。

e.

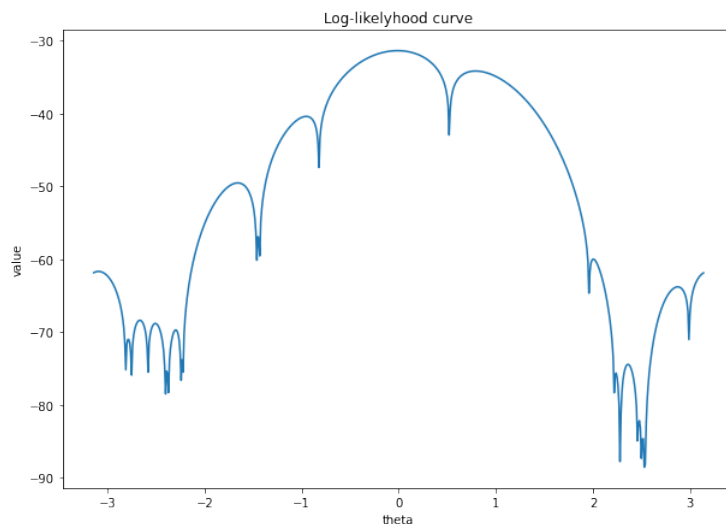
速度由大到小：二分法、牛顿法、弦截法、不动点法

稳定性由大到小：牛顿法、弦截法、二分法、不动点法

结论会在样本数据量改变时改变。

2.2

a.



b.

$$\int_0^{2\pi} xf(x)dx = \frac{1}{2\pi} \int_0^{2\pi} x(1 - \cos(x - \theta))dx$$

$$= \pi + \sin \theta$$

故

$$\pi + \sin \theta = \frac{1}{n} \sum_{i=1}^n x_n$$

解得

$$\theta \approx 0.05844060614042408$$

c.

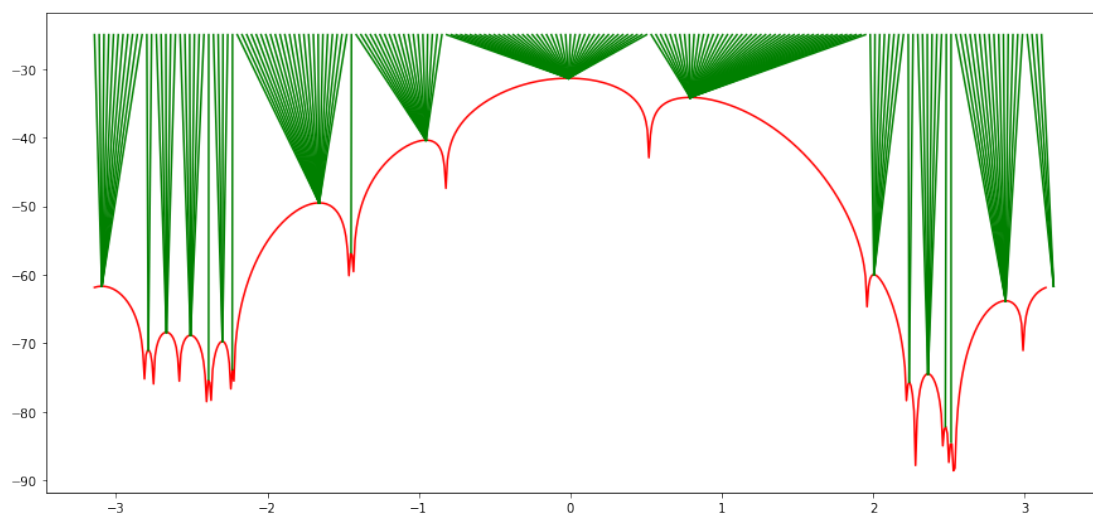
用上一问初始点，算出 $MLE = -0.011972002283305973$

用-2.7，算出 $MLE = -2.666699926100948$

用 2.7，算出 $MLE = 2.8730945142450826$

可见，牛顿法会快速收敛到距离最近的极值点。

d.



可见，牛顿法会快速收敛到距离最近的极值点。

e.

2.4

解：概率密度函数为

$$f(x) = \frac{1}{\Gamma(2)} x e^{-x} = x e^{-x}$$

令 $f'(x) = (1+x)e^{-x} = 0$, 得 $x = 1$ 是极值点, 此时 $f(x) = \frac{1}{e}$ 。设 95% 概率密度区

间里的最小概率密度为 M , 则有 $0 \leq M \leq \frac{1}{e}$ 。

设 $f(x) = x e^{-x} - M = 0$ 的解为 x_1, x_2 , 则有

$$\int_{x_1}^{x_2} x e^{-x} dx = 0.95$$

$$e^{-x_1}(1+x_1) - e^{-x_2}(1+x_2) = 0.95$$

得

$$\begin{cases} e^{-x_1} - e^{-x_2} = 0.95 \\ x_1 e^{-x_1} = x_2 e^{-x_2} \end{cases}$$

使用数值算法解得

$$x_1 = 0.042364, x_2 = 4.76524323$$

$[0.042364, 4.76524323]$ 就是 $\text{Gamma}(2,1)$ 的最窄 95% 概率区间。

2.5

解：

- a. 假设我们已经知道了 N_1, \dots, N_i, N_n ($i = 1, \dots, n$), 对于 N_i 的概率密度函数为

$$f(N_i) = \frac{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^{N_i}}{N_i!} e^{-(\alpha_1 b_{i1} + \alpha_2 b_{i2})}$$

故, 对数似然函数为

$$\begin{aligned} l(\alpha_1, \alpha_2 | b_1, b_2, N) &= \ln L(\alpha_1, \alpha_2 | b_1, b_2, N) \\ &= -\alpha_1 \sum_{i=1}^n b_{i1} - \alpha_2 \sum_{i=1}^n b_{i2} - \sum_{i=1}^n \ln(N_i!) \\ &\quad + \sum_{i=1}^n N_i \cdot \ln(\alpha_1 b_{i1} + \alpha_2 b_{i2}) \end{aligned}$$

求导, 得

$$\nabla l(\alpha_1, \alpha_2) = \begin{pmatrix} -\sum_{i=1}^n b_{i1} + \sum_{i=1}^n \frac{N_i b_{i1}}{\alpha_1 b_{i1} + \alpha_2 b_{i2}} \\ -\sum_{i=1}^n b_{i2} + \sum_{i=1}^n \frac{N_i b_{i2}}{\alpha_1 b_{i1} + \alpha_2 b_{i2}} \end{pmatrix}$$

Hessian 矩阵为

$$H(\alpha_1, \alpha_2) = - \sum_{i=1}^n \begin{pmatrix} \frac{N_i b_{i1}^2}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} & \frac{N_i b_{i1} b_{i2}}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} \\ \frac{N_i b_{i1} b_{i2}}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} & \frac{N_i b_{i2}^2}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} \end{pmatrix}$$

$$= - \sum_{i=1}^n \frac{N_i}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} \begin{pmatrix} b_{i1}^2 & b_{i1} b_{i2} \\ b_{i1} b_{i2} & b_{i2}^2 \end{pmatrix}$$

因此，得到牛顿迭代公式为

$$\alpha^{(t+1)} = \alpha^{(t)} - H^{-1}(\alpha^{(t)}) \cdot \nabla l(\alpha^{(t)})$$

b. Fisher 信息量矩阵为

$$I(\alpha_1, \alpha_2) = -E(H(\alpha_1, \alpha_2))$$

$$= \sum_{i=1}^n \int_0^{+\infty} \frac{N_i}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^2} \begin{pmatrix} b_{i1}^2 & b_{i1} b_{i2} \\ b_{i1} b_{i2} & b_{i2}^2 \end{pmatrix} \frac{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^{N_i}}{N_i!} e^{-(\alpha_1 b_{i1} + \alpha_2 b_{i2})} dN_i$$

$$= \sum_{i=1}^n \int_0^{+\infty} \frac{1}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})} \begin{pmatrix} b_{i1}^2 & b_{i1} b_{i2} \\ b_{i1} b_{i2} & b_{i2}^2 \end{pmatrix} \frac{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^{N_i-1}}{(N_i-1)!} e^{-(\alpha_1 b_{i1} + \alpha_2 b_{i2})} dN_i$$

$$= \sum_{i=1}^n \frac{1}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})} \begin{pmatrix} b_{i1}^2 & b_{i1} b_{i2} \\ b_{i1} b_{i2} & b_{i2}^2 \end{pmatrix} \int_0^{+\infty} \frac{(\alpha_1 b_{i1} + \alpha_2 b_{i2})^{N_i-1}}{(N_i-1)!} e^{-(\alpha_1 b_{i1} + \alpha_2 b_{i2})} dN_i$$

$$= \sum_{i=1}^n \frac{1}{(\alpha_1 b_{i1} + \alpha_2 b_{i2})} \begin{pmatrix} b_{i1}^2 & b_{i1} b_{i2} \\ b_{i1} b_{i2} & b_{i2}^2 \end{pmatrix}$$

故 Fisher Scoring 迭代公式为

$$\alpha^{(t+1)} = \alpha^{(t)} + I^{-1}(\alpha^{(t)}) \cdot \nabla l(\alpha^{(t)})$$

c. 牛顿法和 Fisher Scoring 方法得到的结果一样， $\alpha_1 = 1.097153, \alpha_2 = 0.937555$ ，但是牛顿法的迭代次数明显少于 Fisher Scoring 方法。这两个方法的实现难度差不多。

当设置的误差 $\epsilon = 1 \times 10^{-10}$ 时，结果如下：

牛顿法搜索得到的最优点为：(1.097153, 0.937555)

迭代次数为6，迭代过程为：

(1.000000, 1.000000)

(1.090831, 0.942676)

(1.097128, 0.937575)

(1.097153, 0.937555)

(1.097153, 0.937555)

(1.097153, 0.937555)

Fisher Scoring方法搜索得到的最优点为: (1.097153, 0.937555)

迭代次数为21, 迭代过程为:

```
(1.000000, 1.000000)
(1.117785, 0.906284)
(1.090702, 0.947331)
(1.099195, 0.934459)
(1.096508, 0.938531)
(1.097356, 0.937246)
(1.097088, 0.937652)
(1.097173, 0.937524)
(1.097146, 0.937564)
(1.097155, 0.937552)
(1.097152, 0.937556)
(1.097153, 0.937554)
(1.097152, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
```

d. 得到在 MLE 处的 Fisher 信息量为

```
np.sqrt(logPoisson.fisherI(max_point_Newton))
✓ 0.0s
array([[4.05118009, 3.06428095],
       [3.06428095, 2.80713521]])
```

故 $std(\alpha_1) = 4.051180009$, $std(\alpha_2) = 2.80713521$

e. Sdada

Steepest Ascent方法搜索得到的最优点为: (1.097153, 0.937555)

迭代次数为65, 迭代过程为:

```
(1.000000, 1.000000)
(1.069551, 1.024199)
(1.049867, 0.996378)
(1.082775, 0.990198)
(1.068151, 0.971739)
(1.090170, 0.969765)
(1.079268, 0.957362)
(1.086722, 0.957386)
(1.085973, 0.948943)
(1.091176, 0.949461)
(1.090110, 0.944052)
(1.093764, 0.944723)
(1.092678, 0.941222)
(1.095259, 0.941884)
(1.094770, 0.940739)
(1.096113, 0.940178)
(1.095704, 0.939422)
(1.096597, 0.939151)
(1.096268, 0.938647)
(1.096868, 0.938531)
(1.096609, 0.938191)
(1.097017, 0.938155)
(1.096817, 0.937924)
...
(1.097153, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
```

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

f.

Quasi-Newton方法使用StepHalvingBacktracking搜索得到的最优点为: (1.097153, 0.937555)

迭代次数为10, 迭代过程为:

```
(1.000000, 1.000000)
(1.069551, 1.024199)
(1.116833, 0.923460)
(1.098062, 0.937130)
(1.097143, 0.937555)
(1.097152, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
```

Quasi-Newton方法不使用StepHalvingBacktracking搜索得到的最优解为: (1.097153, 0.937555)
 迭代次数为11, 迭代过程为:

```
(1.000000, 1.000000)
(2.112823, 1.387177)
(1.218099, 0.819302)
(0.894957, 1.183313)
(1.109377, 0.917175)
(1.098494, 0.941024)
(1.097771, 0.937359)
(1.097155, 0.937550)
(1.097153, 0.937555)
(1.097153, 0.937555)
(1.097153, 0.937555)
```

g.

