

六, qr&MM

(一)分位数回归

1.1定义分位数回归

(二) MM算法

2.1MM算法的主要思想

(三) MM算法计算中位数回归

## 六, qr&MM

---

### (一)分位数回归

Date: / /

分位数回归: ~~Autogressive MM no AR~~ ~~autoregressive~~ ~~sliding~~

一个连续变量  $y$ , 其总体第  $T$  分位数是  $y(T)$  的定义是:  $y$  小于等于  $y(T)$  的概率是  $T$ . 即.

$$T = P(y \leq y(T)) = F(y(T))$$

离差绝对值 LAD.

定理: 连续变量用  $y$  表示, 其概率密度函数用  $f(y)$  表示, 累计概率密度函数用  $F(y)$  表示.  $y$  的中位数用  $y_{(0.5)}$  表示. 则  $y$  与任一值  $\alpha$  的离差绝对值的期望  $E(|y - \alpha|)$  以  $\alpha = y_{(0.5)}$  最小.

证明:

$$\begin{aligned} E(|y - \alpha|) &= - \int_{-\infty}^{\alpha} (y - \alpha) f(y) dy + \int_{\alpha}^{\infty} (y - \alpha) f(y) dy \\ \frac{\partial E(|y - \alpha|)}{\partial \alpha} &= 2F(\alpha) - 1 = 0 \\ \Rightarrow F(\alpha) &= 0.5 \text{ 即 } \alpha = y_{(0.5)} \end{aligned}$$

分位数回归:

若用  $\hat{y}(T)_t$  表示  $y_t$  的分位数回归估计量, 则 对于以检查函数  $W_T$  为权数,  $y_t$  对于任意  $\alpha$  值的加权离差绝对值和

$\sum W_T |y_t - \alpha|$  在  $\alpha = \hat{y}(T)_t$  取得最小值:

$$\sum W_T |y_t - \alpha| = - \sum_{i:y_i < \alpha}^{T-1} (1-T) |y_i - \alpha| + \sum_{t:y_t > \alpha}^T T |y_t - \alpha|$$

Date: / /

设函数

$$\rho_\tau(u) = u(\tau - I(u < 0))$$

那么  $\sum_{t=1}^T |y_t - \alpha|$  也可以表示为

$$\sum \rho_\tau(|y_t - \alpha|)$$

我们希望  $E \rho_\tau(y - \alpha)$  尽可能小

$$E \rho_\tau(y - \alpha) = (\tau - 1) \int_{-\infty}^{\alpha} (y - \alpha) dy + \tau \int_{\alpha}^{+\infty} (y - \alpha) dy$$

同样有:  $F(\alpha) = \tau$

$$\text{即 } \alpha = \hat{Y}(1)\tau$$

## 1.1 定义分位数回归

$$Q_y(\tau|x) = \alpha_0 + \alpha_1 * x_1 + \alpha_2 * x_2 + \dots + \alpha_n * x_n$$

分位数岭回归的目标函数为:

$$\min_{\beta} \frac{1}{n} \sum_{i=1}^n \rho_\tau(y_i - x_i^T \beta) + \lambda_n \|\beta\|_2^2$$

中位数回归的目标函数为:

$$\min_{\beta} \sum_{i=1}^n |y_i - x_i^\beta|$$

不能直接求导数计算, 因此要引入MM算法进行计算。

## (二) MM算法

### 2.1 MM算法的主要思想

找到一个辅助函数  $Q(\theta|\theta^t)$ , 去逼近似然函数  $l(\theta|Y_{obs})$ 。步骤如下:

令:  $Q(\theta|\theta^t) \leq l(\theta|Y_{obs})$ , for any  $\theta \in \Theta$

仅当  $\theta = \theta^{(t)}$  时, 有  $Q(\theta^{(t)}|\theta^t) = l(\theta|Y_{obs})$

MM算法不直接最大化  $l(\theta|Y_{obs})$ , 而是最大化  $Q(\theta|\theta^t)$ :

在这个过程中，我们最大化Q函数的同时，也在最大化似然函数：

$$l(\theta^{(t+1)} | Y_{obs}) \geq Q(\theta^{(t+1)} | \theta) \geq Q(\theta^{(t)} | \theta) = l(\theta^{(t)} | Y_{obs})$$

### (三) MM算法计算中位数回归

1. 令  $Q(\beta | \beta^{(l)}) = \sum_{i=1}^n \frac{(y_i - X_i^T \beta)^2}{2|y_i - X_i^T \beta^{(l)}|} + \frac{1}{2} |y_i - X_i^T \beta^{(l)}|$
2. 初始的 $\beta$ 值为OLS估计
3. 给定 $\beta^{(l)}$ ，可以计算  $w_i = |y_i - X_i^T \beta^{(l)}|$
4.  $W_{(l)} = diag(w_i^{(l)})^{-1}$ , 对Q函数求导即可得之。
5. 重复以上步骤

```
import numpy as np
import numpy.linalg as la
import matplotlib.pyplot as plt
n=100

#data generation
x1=np.random.randn(n,1)
x2=np.random.randn(n,1)
X=np.hstack((x1,x2))
error = np.random.randn(n,1)*0.2
y = 3*x1+5*x2+error

#中位数回归
def reg_llw(x,y,tol):
    beta1 = la.inv(X.T.dot(x)).dot(x.T).dot(y)#默认的β1
    diff = 1
    iter = 0
    path = []
    while np.abs(diff)>tol:#当目标函数差异小于容忍度时停止迭代
        iter = iter + 1
        L1sum0 = np.sum(np.abs(y-x.dot(beta1)))#目标函数
        w = np.abs(y - x.dot(beta1))
        W = np.diag(w.ravel()**(-1))
        beta1 = la.inv(X.T.dot(W).dot(x)).dot(x.T).dot(W).dot(y)#计算替代函数的最小值。
        L1sum1 = np.sum(np.abs(y - x.dot(beta1)))
        diff = L1sum1 - L1sum0#计算两次迭代的差异，差异小于容忍度时停止迭代
        path.append(L1sum0)#查看目标函数是否在减小
    return beta1,iter,path
(beta1,iter,path) = reg_llw(x,y,0.00005)
print(beta1)
```