线性回归的数学推导

数学表达

回归方程

$$y = w_0 + x_1w_1 + x_2w_2 + x_3w_3 + x_4w_4 + x_5w_5 + \cdots + w_nx_n \\ = XW^T$$

其中,

$$\mathrm{X} = [1, \mathrm{x}_1, \cdots, \mathrm{x}_\mathrm{n}]$$

$$\mathrm{W} = [\mathrm{w}_0, \mathrm{w}_1, \cdots, \mathrm{w}_\mathrm{n}]$$

损失函数

$$L(W) = \sum (y - \hat{y})^2 \\ = (Y - XW^T)^T(Y - XW^T)$$

参数估计

这一部分与上文参考自不同文章，函数表达略有不同，但并不影响理解。

注意到我们现在需要最小化的函数是

> $Q(\beta_0, \beta_1, \ldots, \beta_p) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_{i1} - \ldots - \beta_p x_{ip})^2$

（这里为了方便，我没有给每一个 β_i 加上帽子，实际要写的时候应该写成 $\hat{\beta}_i$ ）

令其对β_i的导数为0，求解方程组得到向量β，即上文的W

那么同样的，对每一个需要估计的参数求偏导，我们可以得到一系列的方程组如下

> $$\sum (y_i - \beta_0 - \beta_1 x_{i1} - \ldots - \beta_p x_{ip}) = 0 \\ \sum (y_i - \beta_0 - \beta_1 x_{i1} - \ldots - \beta_p x_{ip}) x_{i1} = 0 \\ \ldots \\ \sum (y_i - \beta_0 - \beta_1 x_{i1} - \ldots - \beta_p x_{ip}) x_{ip} = 0$$

首先我们来看第一个式子，求和号需要求和的这一个元素是什么？如果我们设 $\hat{y}_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}$ ，那么相当于每一个元素的残差求和为0。也就是说，$\sum e_i = 0$ 。关键来了，我们把它写成矩阵形式，也就是 $e^T \mathbf{1} = 0$ ，这里 $e = (e_1, e_2, \ldots, e_n)^T$ 。你没有忘记啥是 $\mathbf{1}$ 吧？

上文的黑体1表示全1列向量。

那么同样的，我们看看第二个式子，我需要说的是它是针对下标 i 来做的求和，所以如果我们同样的可以得到 $\sum e_i x_{i1} = 0$ ，也就是 $e^T X_1 = 0$ ，其中 $X_1 = (x_{11}, x_{21}, \ldots, x_{n1})^T$ 。

以此类推，我们事实上可以得到一系列内积为0的式子。这样的话，我们事实上可以把这些关于 X_i 的矩阵拼在一起，也就是说 $e^T \begin{bmatrix} \mathbf{1} & X_1 & \cdots & X_n \end{bmatrix} = 0$ 。这样的话我们就可以得到我们最终的结论

> $e^T X = 0$

这里 X 就是我们上面的设计矩阵。这就很方便了，因为 $e^T = y - X\hat{\beta}$ ，所以我们代进去做一些化简

> $(y - X\hat{\beta})^T X = 0$
> $y^T X = \hat{\beta}^T X^T X$
> $X^T y = X^T X \hat{\beta}$
> $\hat{\beta} = (X^T X)^{-1} X^T y$

这里要注意矩阵 X 是列满秩的（多元回归基本假定第一条），所以 $X^T X$ 是可逆矩阵，因此运算是合法的。通过这一套计算，我们也最终得到了我们想要的结果。

梯度计算

$$令\mathrm{H} = \mathrm{Y} - \mathrm{X}\mathrm{W}^{\mathrm{T}}$$

$$L = H^T H$$

$$\begin{aligned}
dL &= (dH)^T H + H^T dH \\
tr(dL) &= tr((dH)^T H) + tr(H^T dH) \\
tr(dL) &= tr((H^T dH)^T) + tr(H^T dH) \\
tr(dL) &= tr(H^T dH) + tr(H^T dH) \\
dL &= tr(2H^T dH) \\
\frac{\partial L}{\partial H} &= 2H
\end{aligned}$$

$$dH = -X(dW)^T$$

$$\begin{aligned} dL &= -tr(2H^T X(dW)^T) \\ &= -tr((dW)^T 2H^T X) \\ &= -tr((2H^T X)^T dW) \end{aligned}$$

$$\frac{\partial L}{\partial W} = -2(Y - XW^T)^T X$$

可以看到，参数估计部分对β求导的结果与梯度计算的部分差了个系数“-2”，因为其只需计算=0的结果，系数可以不管。

验证代码

```
import numpy as np
import matplotlib.pyplot as plt

# 支持中文
plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False  # 用来正常显示负号
# 测试数据
x1 = np.arange(0,10,0.2)
x2 = np.arange(-10,0,0.2)
y = 10*x1 + 3*x2 + 5 + np.random.random((len(x1)))*20

# 初始化参数
W = np.mat(np.random.random(3)*2-1)
X = np.mat(np.column_stack((np.ones(len(x1)),x1,x2))) #按列组合
Y = np.mat(y).T

# 梯度下降
lr = 0.0001
th = 1000000
while(True):
    # 更新梯度
    w = -2 * (Y - X @ W.T).T @ X  # @代表矩阵相乘
    W = W - lr * w
    # 绘图
    Y_Test = X @ W.T
    e = (Y - Y_Test).T @ (Y - Y_Test)
    x = np.arange(len(Y)) + 1
    plt.clf()
    plt.scatter(x,list(Y), s=10)
    plt.plot(x, Y, 'b')
    plt.plot(x, Y_Test, 'r')
```

```
    plt.pause(0.01)
    if (th - e) < 0.01:
        plt.show()  # 保持绘图窗口，否则程序直接退出，窗口会关闭
        break
    if e < th:
        th = e

```

参考链接

回归分析|笔记整理（6）——多元线性回归（上）

多元线性回归之矩阵求导推导与python实现

3分钟带你掌握标量对矩阵求导方法