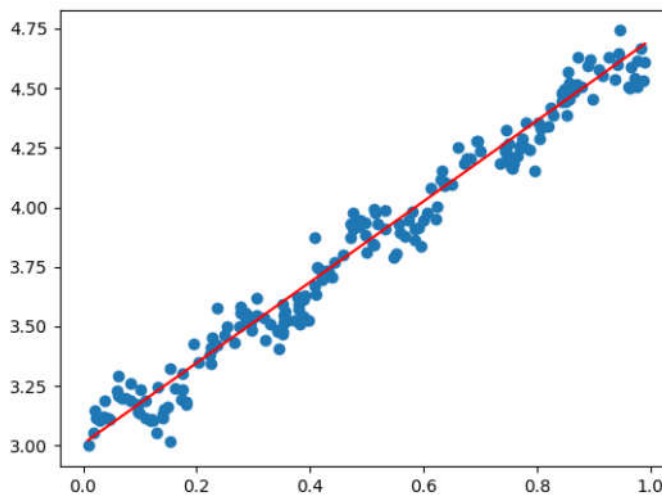
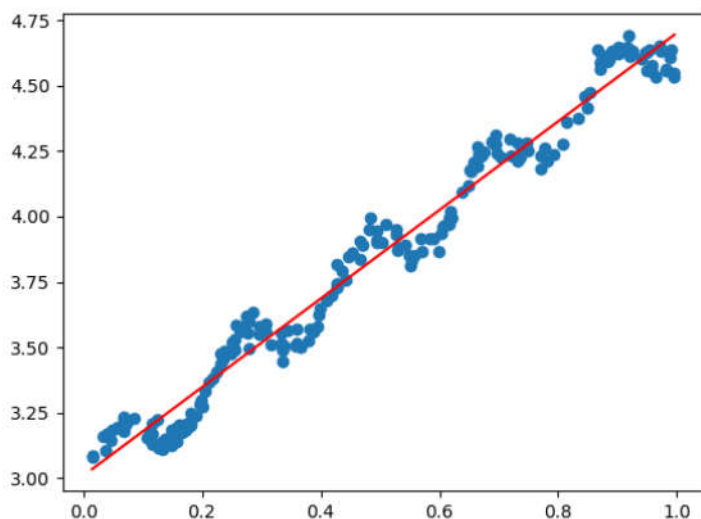


Python编程与人工智能实践

算法篇：线性回归-Linear Regression



于泓
鲁东大学
信息与电气工程学院
2021.9.24

回归 (Regress) 问题

- 回归问题也叫预测问题，即用一组测量数据来对目标的**数值**进行预测

例如：已知

x_1	x_2	x_3	x_4	y
0.455	0.365	0.095	0.514	15
0.35	0.265	0.09	0.2255	7
0.53	0.42	0.135	0.677	9
0.44	0.365	0.125	0.516	?

即构建一个函数 $y_i = f(\mathbf{x}_i; \mathbf{w})$ 根据输入的**特征向量** \mathbf{x} 来预测输出**数值** y

线性回归 (Linear Regress)

设已知输入特征矢量 \mathbf{x} 的维度为 D , $\mathbf{x}=\{x_1, x_2, \dots, x_D\}$

共有 N 个训练样本 $\mathbf{X}=\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$

以及其对应的真实输出 $\mathbf{y}=\{y_1, y_2, \dots, y_N\}$

利用线性回归预测的输出为 $\hat{\mathbf{y}} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_N\}$ 其中 $\mathbf{w} = \{w_1, w_2, \dots, w_D\}^T$ 所求的参数

$$\hat{y}_1 = w_1 x_{11} + w_2 x_{12} + \dots + w_D x_{1D}$$

$$\hat{y}_2 = w_1 x_{21} + w_2 x_{22} + \dots + w_D x_{2D}$$

....

$$\hat{y}_N = w_1 x_{N1} + w_2 x_{N2} + \dots + w_D x_{ND}$$

矩阵形式

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$$

Diagram illustrating the matrix form of the linear regression equation $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$:

- $\hat{\mathbf{y}}$ is a column vector of size $(N, 1)$.
- \mathbf{X} is a matrix of size (N, D) .
- \mathbf{w} is a column vector of size $(D, 1)$.

损失函数

$$\begin{aligned} L_{\text{MSE}} &= \sum_{i=1}^N (y_i - \hat{y}_i)^2 = (\mathbf{y} - \hat{\mathbf{y}})^T (\mathbf{y} - \hat{\mathbf{y}}) \\ &= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) \\ &= \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \mathbf{X}\mathbf{w} - \mathbf{w}^T \mathbf{X}^T \mathbf{y} + \mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} \end{aligned}$$

$$\frac{\partial L_{\text{MSE}}}{\partial \mathbf{w}} = 2\mathbf{X}^T \mathbf{X} \mathbf{w} - 2\mathbf{X}^T \mathbf{y} = 0$$

$$\longrightarrow \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

参数的求解方法

代码实现及测试：

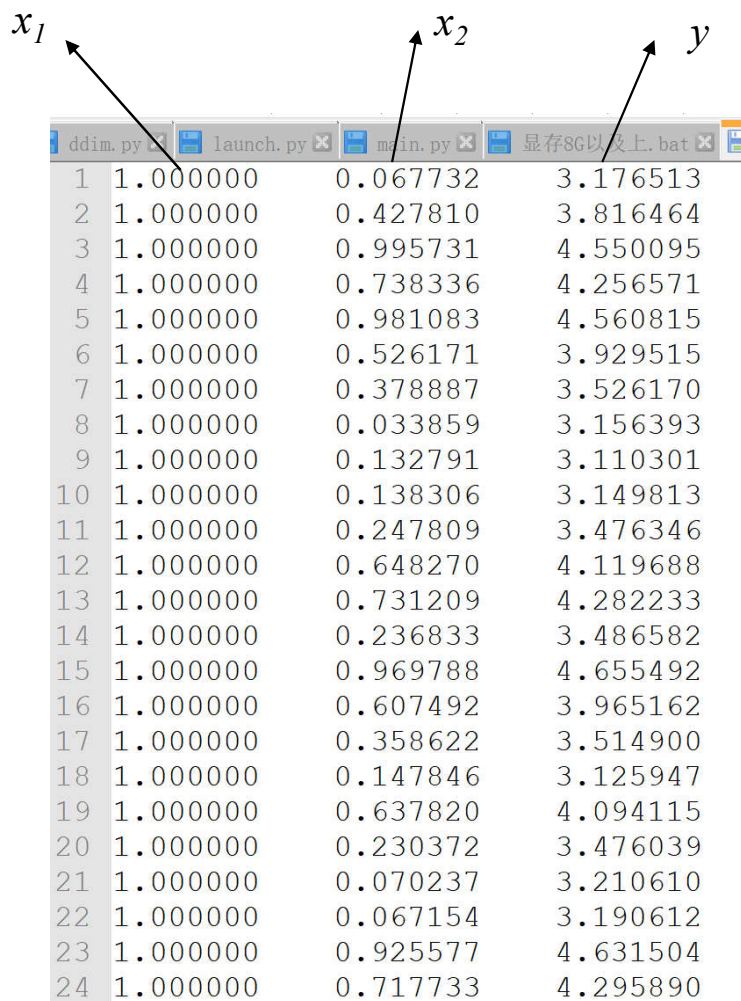
数据加载：

```
def load_DataSet(file_data,col_X=(0,1),col_Y=(2),add_bias=False):
    data_X = np.loadtxt(file_data,dtype=float,delimiter="\t",usecols=col_X)
    data_Y= np.loadtxt(file_data,dtype=float,delimiter="\t",usecols=col_Y)

    # 如果需要偏置, 则为data_X 增加一个数值是1的维度
    if add_bias:
        N,D = np.shape(data_X)
        add_dim = np.ones((N,1))
        data_X = np.concatenate((data_X,add_dim),axis=-1)

    # 对data_Y 进行维度修正 使其为 (N,1)
    if len(np.shape(data_Y))==1:
        data_Y = np.expand_dims(data_Y,axis=-1)
```

ex0.txt



1	1.000000	0.067732	3.176513
2	1.000000	0.427810	3.816464
3	1.000000	0.995731	4.550095
4	1.000000	0.738336	4.256571
5	1.000000	0.981083	4.560815
6	1.000000	0.526171	3.929515
7	1.000000	0.378887	3.526170
8	1.000000	0.033859	3.156393
9	1.000000	0.132791	3.110301
10	1.000000	0.138306	3.149813
11	1.000000	0.247809	3.476346
12	1.000000	0.648270	4.119688
13	1.000000	0.731209	4.282233
14	1.000000	0.236833	3.486582
15	1.000000	0.969788	4.655492
16	1.000000	0.607492	3.965162
17	1.000000	0.358622	3.514900
18	1.000000	0.147846	3.125947
19	1.000000	0.637820	4.094115
20	1.000000	0.230372	3.476039
21	1.000000	0.070237	3.210610
22	1.000000	0.067154	3.190612
23	1.000000	0.925577	4.631504
24	1.000000	0.717733	4.295890

转成mat格式，方便矩阵运算

```
def linear_Regress(X,Y):  
    # # mat()函数将xArr, yArr转换为矩阵 mat().T 代表的是对矩阵进行转置操作  
    xMat = mat(X)  
    yMat = mat(Y)  
    # 矩阵乘法的条件是左矩阵的列数等于右矩阵的行数  
    xTx = xMat.T * xMat  
    # 因为要用到xTx的逆矩阵，所以事先需要确定计算得到的xTx是否可逆，条件是矩阵的行列式不为0  
    # linalg.det() 函数是用来求得矩阵的行列式的，如果矩阵的行列式为0，则这个矩阵是不可逆的，就无法进行接下来的运算  
    if np.linalg.det(xTx) == 0.0:  
        print("This matrix is singular, cannot do inverse")  
        return  
    # 最小二乘法  
    ws = xTx.I * xMat.T * yMat  
    return np.array(ws)
```

$$\mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

绘图与测试:

```
if __name__ == "__main__":  
    # 数据加载  
    X_train,Y_train = load_DataSet('ex0.txt')  
  
    # 参数获取  
    ws = linear_Regress(X_train,Y_train)  
    print(ws)  
  
    # X_test,Y_test = X_train,Y_train  
    X_test,Y_test = load_DataSet('ex1.txt')  
  
    # 回归预测  
    Y_hat = np.dot(X_test,ws)  
  
    # 绘图  
    fig = plt.figure() # 创建绘图对象  
    ax = fig.add_subplot(1,1,1)  
    ax.scatter(X_test[:,1],Y_test)  
    # 数据排序  
    index=np.argsort(X_test[:,1])  
  
    X_copy= X_test[index,:]  
    Y_hat = Y_hat[index,:]  
  
    ax.plot(X_copy[:,1],Y_hat,color=(1,0,0))  
  
    plt.show()
```

