



9.5 实例：解析法实现多元线性回归

9.5 实例：解析法多元线性回归

商品房销售记录

序号	面积 (平方米)	房间数	销售价格 (万元)	序号	面积 (平方米)	房间数	销售价格 (万元)
1	137.97	3	145.00	9	106.69	2	62.00
2	104.50	2	110.00	10	138.05	3	133.00
3	100.00	2	93.00	11	53.75	1	51.00
4	124.32	3	116.00	12	46.91	1	45.00
5	79.20	1	65.32	13	68.00	1	78.50
6	99.00	2	104.00	14	63.02	1	69.65
7	124.00	3	118.00	15	81.26	2	75.69
8	114.00	2	91.00	16	86.21	2	95.30

x_1

x_2

y

x_1

x_2

y

- 加载样本数据
- 数据处理
- 学习模型：计算 W
 $W = (X^T X)^{-1} X^T Y$
- 预测房价



■ 加载样本数据

```
In [1]: import numpy as np
```

```
In [2]: x1=np.array([137.97, 104.50, 100.00, 124.32, 79.20, 99.00, 124.00, 114.00,  
                    106.69, 138.05, 53.75, 46.91, 68.00, 63.02, 81.26, 86.21])  
x2=np.array([3, 2, 2, 3, 1, 2, 3, 2, 2, 3, 1, 1, 1, 1, 2, 2])  
y=np.array([145.00, 110.00, 93.00, 116.00, 65.32, 104.00, 118.00, 91.00,  
            62.00, 133.00, 51.00, 45.00, 78.50, 69.65, 75.69, 95.30])
```

```
In [3]: x1.shape, x2.shape, y.shape
```

```
Out[3]: ((16,), (16,), (16,))
```



$$Y = X W$$

$n \times (m+1)$

$(m+1) \times 1$

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & \dots & x_1^m \\ 1 & x_2^1 & \dots & x_2^m \\ \dots & \dots & \dots & \dots \\ 1 & x_n^1 & \dots & x_n^m \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_m \end{bmatrix}$$



```
In [4]: x0=np.ones(len(x1))
```

```
In [5]: X=np.stack((x0,x1,x2), axis = 1) (16, )
X (16, 3)
```

```
Out[5]: array([[ 1. , 137.97,  3. ],
 [ 1. , 104.5 ,  2. ],
 [ 1. , 100. ,  2. ],
 [ 1. , 124.32,  3. ],
 [ 1. ,  79.2 ,  1. ],
 [ 1. ,  99. ,  2. ],
 [ 1. , 124. ,  3. ],
 [ 1. , 114. ,  2. ],
 [ 1. , 106.69,  2. ],
 [ 1. , 138.05,  3. ],
 [ 1. ,  53.75,  1. ],
 [ 1. ,  46.91,  1. ],
 [ 1. ,  68. ,  1. ],
 [ 1. ,  63.02,  1. ],
 [ 1. ,  81.26,  2. ],
 [ 1. ,  86.21,  2. ]])
```

9.5 实例：解析法多元线性回归

■ 数据处理

$$Y = X W$$

$n \times (m+1)$ $(m+1) \times 1$

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & \dots & x_1^m \\ 1 & x_2^1 & \dots & x_2^m \\ \dots & \dots & \dots & \dots \\ 1 & x_n^1 & \dots & x_n^m \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ \dots \\ w_m \end{bmatrix}$$

16×1

```
In [6]: Y = np.array(y).reshape(-1, 1)
```

```
Out[6]: array([[145. ],  
               [110. ],  
               [ 93. ],  
               [116. ],  
               [ 65.32],  
               [104. ],  
               [118. ],  
               [ 91. ],  
               [ 62. ],  
               [133. ],  
               [ 51. ],  
               [ 45. ],  
               [ 78.5 ],  
               [ 69.65],  
               [ 75.69],  
               [ 95.3 ]])
```



■ 求解模型参数

$$W = (X^T X)^{-1} X^T Y$$

功 能	函 数
矩阵相乘	np.matmul()
矩阵转置	np.transpose()
矩阵求逆	np.linalg.inv()

```
In [7]: Xt=np.transpose(X)           #计算X'  
XtX_1 = np.linalg.inv(np.matmul(Xt,X)) #计算(X'X)-1  
XtX_1_Xt= np.matmul(XtX_1,Xt)         #计算(X'X)-1X'  
W= np.matmul(XtX_1_Xt,Y)              #W=((X'X)-1)X'Y
```

```
In [8]: W
```

```
Out[8]: array([[11.96729093],  
               [ 0.53488599],  
               [14.33150378]])
```

```
In [9]: W=W.reshape(-1)  
W
```

```
Out[9]: array([11.96729093,  0.53488599, 14.33150378])
```

```
In [10]: print("多元线性回归方程：")  
print("Y=",W[1],"*x1+",W[2],"*x2+",W[0])
```

多元线性回归方程：

Y= 0.5348859949724747 *x1+ 14.331503777673149 *x2+ 11.967290930535732



■ 预测房价

```
In [11]: print("请输入房屋面积和房间数，预测房屋销售价格：")
          x1_test=float(input("商品房面积："))
          x2_test=int(input("房间数："))

          y_pred = W[1]*x1_test+W[2]*x2_test+W[0]
          print("预测价格：",round(y_pred,2),"万元")
```

请输入房屋面积和房间数，预测房屋销售价格：
商品房面积:140
房间数: 3
预测价格: 129.85 万元

请输入房屋面积和房间数，预测房屋销售价格：
商品房面积:140
房间数: 4
预测价格: 144.18 万元



■ NumPy数组运算函数

功 能	函 数
数组堆叠	<code>np.stack()</code>
改变数组形状	<code>np.reshape()</code>
矩阵相乘	<code>np.matmul()</code>
矩阵转置	<code>np.transpose()</code>
矩阵求逆	<code>np.linalg.inv()</code>

