

# 深度学习 训练营

一步一个脚印，掌握深度学习

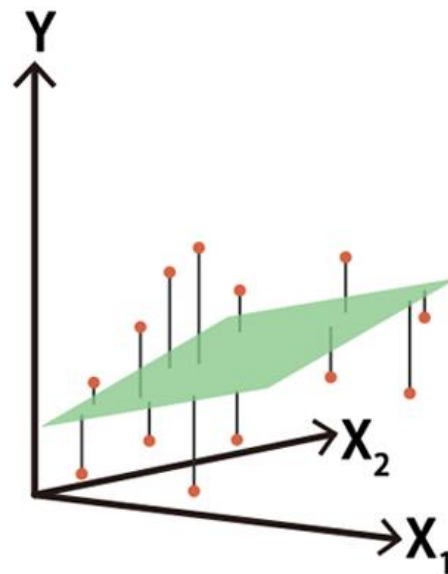


# 第3周，矩阵和多元线性回归

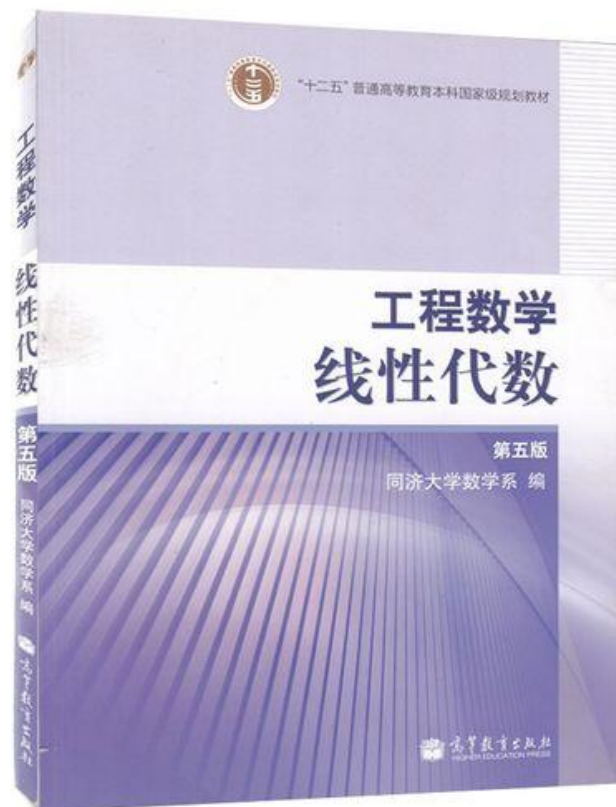
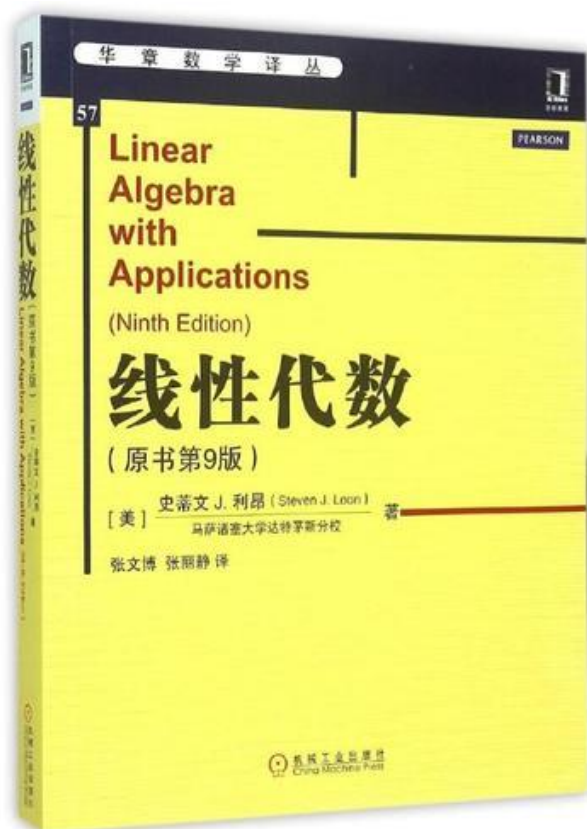


## 第3周，课程核心内容：

- 3.1-多元线性回归的整体概述
- 3.2-梯度下降求解多元线性回归
- 3.3-实验，多元线性回归
- 3.4-特征缩放的算法设计
- 3.5-实验，特征缩放
- 3.6-实验，sklearn的线性回归模型
- 3.7-矩阵的基本概念和运算
- 3.8-单位矩阵与求解逆矩阵
- 3.9-标准方程方法的数学表示
- 3.10-标准方程方法的推导过程
- 3.11-实验，标准方程方法



# 关于线性代数



# 多元线性回归的问题概述

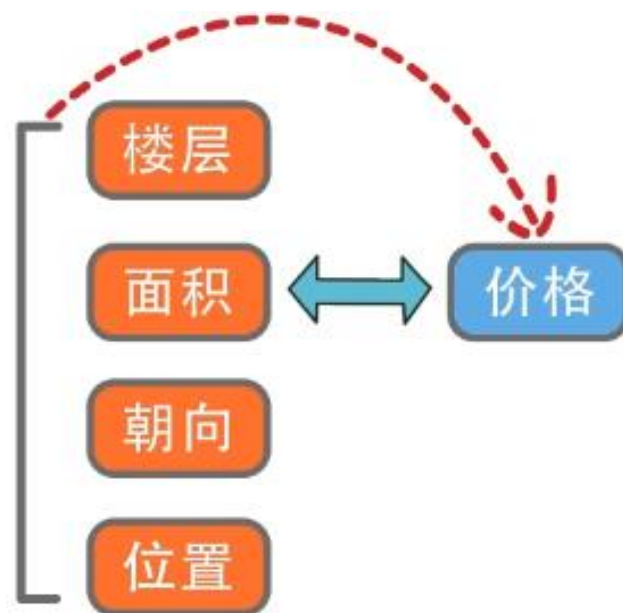
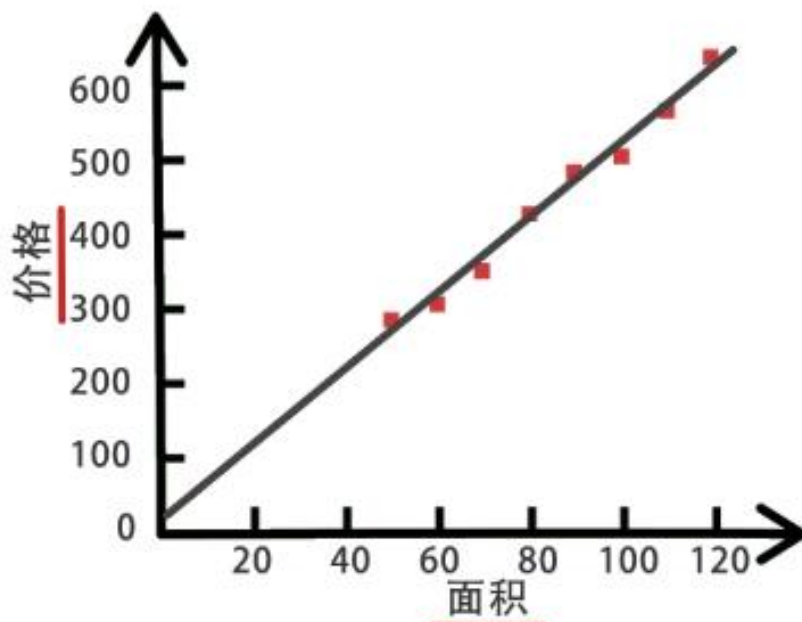
编号	面积( $x_1$ )	卧室( $x_2$ )	朝向( $x_3$ )	楼层( $x_4$ )	房价( $y$ )
1	96.79	2	南北 1	高楼层 2	287w
2	110.39	3	南北 1	低楼层 0	343w
3	70.25	1	北 0	高楼层 2	199w
4	99.96	2	南北 1	中楼层 1	298w
5	118.15	3	南北 1	低楼层 0	340w
6	115.08	3	南北 1	高楼层 2	350w

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

# 课程回顾

在前面的课程中，我们学习了使用一元线性回归解决回归预测问题，即只通过一个自变量的值预测因变量。

在大多情况下，往往需要同时考虑多个特征来预测问题的结果。



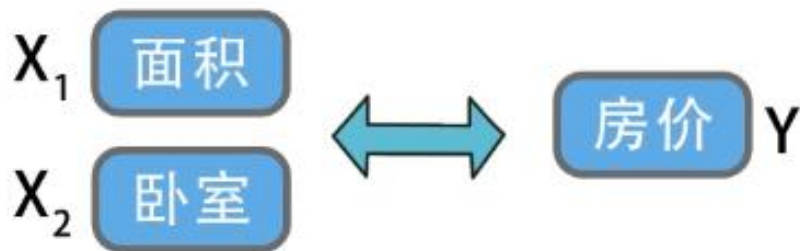


# 多元线性回归

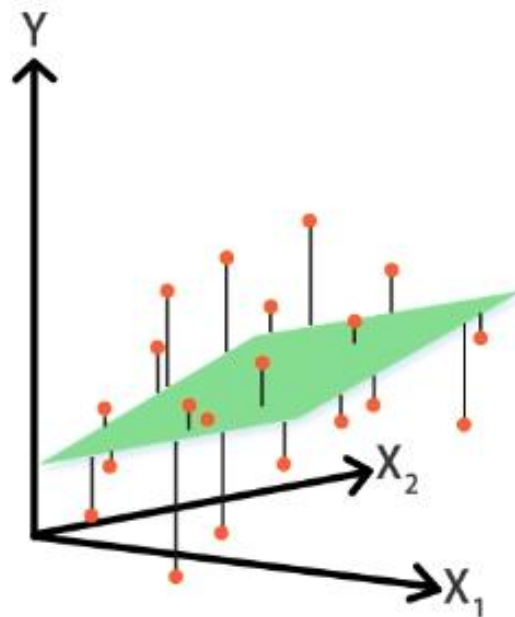


例如，在预测房价时，不仅需要考虑房子的面积，还要关注楼层、小区位置、朝向、卧室数量等关键特征，才能更准确的预测结果。

同时使用多个特征来进行回归预测，在更高维度的空间中，找到一条能够拟合样本特征与预测值关系的超平面，就是多元线性回归问题。



多元线性回归





# 样本和特征的数学表示

设训练集中有 $m$ 个样本，每个样本有 $n$ 个特征。

使用 $x$ 的下角标表示特征的编号， $n$ 个特征分别表示为 $x_1$ 、 $x_2$ 等等到 $x_n$ 。

使用 $x$ 的上角标加括号的形式表示样本的编号，第 $i$ 个样本的特征向量就表示为， $x_{1-i}$ 、 $x_{2-i}$ 等等到 $x_{n-i}$ 。

$m$ 个样本对应的真实值使用 $y$ 表示，其中 $y$ 的上角标加括号表示样本的编号。



第 $i$ 个样本  $(x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)})$

$x_i$  -----> 特征编号

$x^{(i)}$  -----> 样本编号  
 $y^{(i)}$

**$m$ 个样本，每个样本中有 $n$ 个特征：**

$(x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)}, y^{(1)}), (x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)}, y^{(2)}), \dots, (x_1^{(m)}, x_2^{(m)}, \dots, x_n^{(m)}, y^{(m)})$



# 样本和特征的数学表示

例如， $m=6$ ， $n=4$ ，代表训练集中有6个样本，4个特征。4个特征分别是面积、卧室、朝向、楼层。

将其中的字符串换作整数表示，第1个样本表示为(96.79, 2, 1, 2, 287)



编号	面积	卧室个数	朝向	楼层	房价
1	96.79	2	南北 1	高楼层 2	287w
2	110.39	3	南北 1	低楼层 0	343w
3	70.25	1	北 0	高楼层 2	199w
4	99.96	2	南北 1	中楼层 1	298w
5	118.15	3	南北 1	低楼层 0	340w
6	115.08	3	南北 1	高楼层 2	350w

$m = 6$

$n = 4$

样本1: 96.79-2-1-2-287





# 将特征向量保存到矩阵

将所有样本的特征向量保存到矩阵中，就得到了一个 $m \times (n+1)$ ，即6乘5的矩阵X，该矩阵被称为特征矩阵。

将样本的标记保存到列向量y中，会得到一个 $m \times 1$ 的列向量，该向量与特征矩阵的每一行都是对应的。

编号	面积( $x_1$ )	卧室( $x_2$ )	朝向( $x_3$ )	楼层( $x_4$ )	房价(y)
1	96.79	2	南北 1	高楼层 2	287w
2	110.39	3	南北 1	低楼层 0	343w
3	70.25	1	北 0	高楼层 2	199w
4	99.96	2	南北 1	中楼层 1	298w
5	118.15	3	南北 1	低楼层 0	340w
6	115.08	3	南北 1	高楼层 2	350w

$$X = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 \end{matrix} \\ \begin{bmatrix} 1 & 96.79 & 2 & 1 & 2 \\ 1 & 110.39 & 3 & 1 & 0 \\ 1 & 70.25 & 1 & 0 & 2 \\ 1 & 99.96 & 2 & 1 & 1 \\ 1 & 118.15 & 3 & 1 & 0 \\ 1 & 115.08 & 3 & 1 & 2 \end{bmatrix} & \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \end{matrix}$$

$m \times n+1$

$$y = \begin{matrix} & 0 \\ \begin{bmatrix} 287 \\ 343 \\ 199 \\ 298 \\ 340 \\ 350 \end{bmatrix} & \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \end{matrix}$$

$m \times 1$



# 特殊参数 $\theta_0$

它表示在训练集的样本中，除了我们最开始预设的 $n$ 个特征， $x_1$ 到 $x_n$ ，还包括了一个额外的特征 $x_0$ ，而 $x_0$ 的值被固定设置为1。因此，特征的编号从0开始，一直到 $n$ ，组成了一个 $n+1$ 维的特征向量。

$$h_{\theta}(x) = \underbrace{\theta_0}_{\theta_0 x_0} + \underbrace{\theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n}_{n \text{ 个特征 } (x_1, x_2, \dots, x_n)}$$

$\downarrow$   
 $x_0 = 1$



# 预测直线的数学表示

设直线 $h(x) = 1 + 2x_1 + 3x_2 + 4x_3 - x_4$ ，该直线是一个可能的假设函数。用它进行多元线性回归时，该方程表示了4个自变量与因变量的关系。

由于 $x_1$ 、 $x_2$ 、 $x_3$ 的系数是正的， $x_4$ 的系数是负的，所以房价随着面积、卧室个数、朝向值的增加而增加，随着楼层的增加而减小。

因为 $x_2$ 的参数3要大于 $x_1$ 的参数2，即代表卧室个数的变化比面积的变化对于房价的影响更大。

编号	自变量( $x_1, x_2, x_3, x_4$ )				因变量 $y$
	面积( $x_1$ )	卧室( $x_2$ )	朝向( $x_3$ )	楼层( $x_4$ )	房价( $y$ )
1	96.79	2	南北 1	高楼层 2	287w
2	110.39	3	南北 1	低楼层 0	343w
3	70.25	1	北 0	高楼层 2	199w
4	99.96	2	南北 1	中楼层 1	298w
5	118.15	3	南北 1	低楼层 0	340w
6	115.08	3	南北 1	高楼层 2	350w

$$h(x) = 1 + \underline{2}x_1 + \underline{3}x_2 + \underline{4}x_3 - x_4$$





# 回归方程的矩阵形式

将方程的全部参数 $\theta$ ，保存到列向量中：

将 $\theta$ 向量的转置乘以某样本特征的列向量 $X$ ，展开后就得到直线回归方程的预测值，这是多元线性回归的矩阵表示形式。

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \quad \underbrace{\begin{bmatrix} \theta_0 & \theta_1 & \theta_2 & \dots & \theta_n \end{bmatrix}}_{\theta^T} * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{\mathbf{X}}$$

↓

$$\theta^T \mathbf{X} = h_{\theta}(x) \\ = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$



# 多元线性回归的代价函数

对于多元线性回归的代价函数 $J(\theta)$ ，和一元线性回归一样，同样是1/2倍的均方误差。

不同之处在于，如果对 $h_{\theta}(x)$ 展开，会得到一个 $n$ 元的回归方程。

最终在多元线性回归问题中，我们需要求出，对于 $m$ 个样本，使得代价函数 $J(\theta)$ 取得最小值的 $\theta_0$ 、 $\theta_1$ 、等等到 $\theta_n$ ，这 $n+1$ 个参数。

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x_i) - y_i)^2$$

$\downarrow$   
 $\min(J(\theta))$

$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$

$n+1$  个参数



# 梯度下降解决多元线性回归

编号	面积( $x_1$ )	卧室( $x_2$ )	朝向( $x_3$ )	楼层( $x_4$ )	房价( $y$ )
1	96.79	2	南北 1	高楼层 2	287w
2	110.39	3	南北 1	低楼层 0	343w
3	70.25	1	北 0	高楼层 2	199w
4	99.96	2	南北 1	中楼层 1	298w
5	118.15	3	南北 1	低楼层 0	340w
6	115.08	3	南北 1	高楼层 2	350w

$$\nabla J(\theta_0, \theta_1, \theta_2, \dots, \theta_n) = \left( \frac{\partial J}{\partial \theta_0}, \frac{\partial J}{\partial \theta_1}, \frac{\partial J}{\partial \theta_2}, \dots, \frac{\partial J}{\partial \theta_n} \right)$$



# 梯度下降解决多元线性回归

为了选出最合适的假设函数 $h_{\theta}(x)$ ，需要求出使全部样本的总误差最小。

即 $J(\theta)$ 取得最小值时，多元线性回归方程中的全部参数， $\theta_0$ 、 $\theta_1$ 、 $\theta_2$ 等等到 $\theta_n$ 的值。

$J(\theta)$ 是一个关于 $\theta_0$ 到 $\theta_n$ 的多元函数，我们无法将它随 $\theta_0$ 到 $\theta_n$ 变化的过程直接使用图像表示出来。

但可以确定的是， $J(\theta)$ 是一个凸函数，通过梯度下降算法可以求出代价函数 $J(\theta)$ 的最小值。

$$\begin{array}{ccc} h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n & \rightarrow & J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \\ \text{min(MSE)} & & \text{min}(J(\theta)) \end{array}$$
  
$$\begin{array}{ccccc} J(\theta_0, \theta_1, \theta_2, \dots, \theta_n) & \rightarrow & \nabla J(\theta_0, \theta_1, \theta_2, \dots, \theta_n) & \rightarrow & \text{min}(J(\theta_0, \theta_1, \theta_2, \dots, \theta_n)) \\ \text{多元函数} & & & & \end{array}$$



# $J(\theta)$ 的梯度向量

对于 $J(\theta)$ 的梯度向量，它等于对 $\theta_0$ 、 $\theta_1$ 等等到 $\theta_n$ ，每个自变量求偏导后，然后一起构成的 $n+1$ 维向量。求出 $J(\theta)$ 的梯度向量后，我们以负梯度的方向来决定每次迭代时 $\theta_0$ 、 $\theta_1$ 等等到 $\theta_n$ ，这些参数的变化方向，从而使得每次迭代后，目标函数 $J(\theta)$ 逐渐减小。

$$\nabla J(\theta_0, \theta_1, \theta_2, \dots, \theta_n) = \left( \frac{\partial J}{\partial \theta_0}, \frac{\partial J}{\partial \theta_1}, \frac{\partial J}{\partial \theta_2}, \dots, \frac{\partial J}{\partial \theta_n} \right)$$



$n+1$ 维向量

$$\min(J(\theta_0, \theta_1, \theta_2, \dots, \theta_n))$$

$$\theta_0 = \theta_0 - \alpha * \frac{\partial J}{\partial \theta_0}$$

$$\theta_1 = \theta_1 - \alpha * \frac{\partial J}{\partial \theta_1}$$

$\vdots$

$$\theta_n = \theta_n - \alpha * \frac{\partial J}{\partial \theta_n}$$



# 梯度下降算法

在每次迭代时过程中，需要同时修改 $\theta_0$ 到 $\theta_n$ 的值，使它们分别以 $\alpha$ 的学习速率，沿着各自偏导数的反方向进行移动。

进而经过number轮迭代后，达到 $J(\theta)$ 取得最小值的位置，此时 $\theta_0$ 、 $\theta_1$ 等等到 $\theta_n$ 的值，即为迭代的参数结果。

对于其中的学习速率 $\alpha$ ，是迭代前选择的一个常数，它的调试方法与一元线性回归中是一样的。

for i in range(0,number): ➡  $\min(J(\theta))$

$$\begin{aligned}\theta_0 &= \theta_0 - \alpha * \frac{\partial J}{\partial \theta_0} \\ \theta_1 &= \theta_1 - \alpha * \frac{\partial J}{\partial \theta_1} \\ &\vdots \\ \theta_n &= \theta_n - \alpha * \frac{\partial J}{\partial \theta_n}\end{aligned}$$

$\alpha$  取值范围：0.0001到0.01



# 偏导数的推导和计算

为了实现多元线性回归的梯度下降算法，同样要求出代价函数J对于所有 $\theta$ 参数的偏导数。

具体为函数J对第j个特征 $x_j$ 的参数 $\theta_j$ ，求偏导的过程。

有了所有 $\theta$ 参数的偏导数计算结果，即可继续设计多元线性回归的实现方法。

对 $x_j$ 的参数 $\theta_j$ 求偏导过程

$$\frac{\partial}{\partial \theta_j} J(\theta) = \left( \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \right)'$$

$$= \frac{1}{2m} \sum_{i=1}^m ((h_{\theta}(x^{(i)}) - y^{(i)})^2)'$$

$$= \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) \underline{(h_{\theta}(x^{(i)}) - y^{(i)})'}$$

$$= \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) (\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \dots + \theta_j \underline{x_j^{(i)}} + \dots + \theta_n x_n^{(i)} - y^{(i)})'$$

未知数

$$= \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$\frac{\partial}{\partial \theta_0} J(\theta)$$

$$\frac{\partial}{\partial \theta_1} J(\theta)$$


$\vdots$

$$\frac{\partial}{\partial \theta_n} J(\theta)$$





# 多元线性回归的代码实现

面积( $x_1$ )	卧室( $x_2$ )	朝向( $x_3$ )	楼层( $x_4$ )	房价( $y$ )
96.79	2	南北 1	高楼层 2	287w
110.39	3	南北 1	低楼层 0	343w
70.25	1	北 0	高楼层 2	199w
99.96	2	南北 1	中楼层 1	298w
118.15	3	南北 1	低楼层 0	340w
115.08	3	南北 1	高楼层 2	350w

 假设函数  $h_{\theta}(x)$

  $J(\theta)$  关于  $\theta_j$  的偏导数

 梯度下降迭代

 代价函数  $J(\theta)$



# 功能函数

为了使用梯度下降算法实现多元线性回归，需要设计相应的计算函数，具体包括：

- 1) 求假设函数值  $h_{\theta}(x)$  的计算
- 2)  $J(\theta)$  关于  $\theta_j$  的偏导数
- 3) 梯度下降迭代
- 4) 代价函数  $J(\theta)$

01 假设函数  $h_{\theta}(x)$

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_{n-1} x_{n-1} + \theta_n x_n$$

02  $J(\theta)$  关于  $\theta_j$  的偏导数

$$\frac{\partial}{\partial \theta_1} J(\theta) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

03 梯度下降迭代

```
def gradient_descent(x,y,n,m,alpha,iterate)
```

04 代价函数  $J(\theta)$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$



# 假设函数值 $h_{\theta}(x)$ 的计算函数hypothesis

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_{n-1} x_{n-1} + \theta_n x_n$$

# 设函数值 $h_{\theta}(x)$ 的计算函数

# 函数传入参数theta、样本特征向量x和特征的个数n

```
def hypothesis(theta, x, n):  
    h = 0.0 # 保存预测结果  
    for i in range(0, n + 1):  
        # 将theta-i和xi的乘积累加到h中  
        h += theta[i] * x[i]  
    return h # 返回预测结果h
```



# J(θ)关于theta-j的偏导数计算函数

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

#J(θ) 关于theta-j的偏导数计算函数

#函数传入样本特征矩阵x和标记y，特征的参数列表theta，特征数n

#样本数量m和待求偏导数的参数下标j

```
def gradient_thetaj(x, y, theta, n, m, j):  
    sum = 0.0 #保存sigma求和累加的结果  
    for i in range(0, m): #遍历m个样本  
        h = hypothesis(theta, x[i], n) #求出样本的预测值h  
        #计算预测值与真实值的差，再乘以第i个样本的第j个特征值x[i][j]  
        #将结果累加到sum  
        sum += (h - y[i]) * x[i][j]  
    return sum / m #返回累加结果sum除以样本的个数m
```



# 梯度下降的迭代函数gradient\_descent



#梯度下降的迭代函数

#函数传入样本特征矩阵x和样本标签y，特征数n，样本数量m

#迭代速率alpha和迭代次数iterate

```
def gradient_descent(x, y, n, m, alpha, iterate):  
    theta = [0] * (n + 1) #初始化参数列表theta，长度为n+1  
    for i in range(0, iterate): #梯度下降的迭代循环  
        temp = [0] * (n + 1)  
        #使用变量j，同时对theta0到theta-n这n+1个参数进行更新  
        for j in range(0, n + 1):  
            #通过临时变量列表temp，先保存一次梯度下降后的结果  
            #在迭代的过程中调用theta-j的偏导数计算函数  
            temp[j] = theta[j] - alpha * gradient_thetaj(x, y, theta, n, m, j)  
        #将列表temp赋值给列表theta  
        for j in range(0, n + 1):  
            theta[j] = temp[j]  
    return theta #函数返回参数列表theta
```





# 代价函数 $J(\theta)$ 的计算函数costJ

#实现代价函数 $J(\theta)$ 的计算函数costJ

#函数传入样本的特征矩阵x和标签y，参数列表theta，特征个数n和样本个数m

```
def costJ(x, y, theta, n, m):  
    sum = 0.0 #定义累加结果  
    for i in range(0, m): #遍历每个样本  
        h = hypothesis(theta, x[i], n)  
        #将样本的预测值与真实值差的平方累加到sum中  
        sum += (h - y[i]) * (h - y[i])  
    return sum / (2 * m) #返回sum除以2m
```



# main函数

```
if __name__ == '__main__':  
    m = 6 #6个样本  
    n = 4 #每个样本4个特征  
    alpha = 0.0001 #迭代速率  
    iterate = 1500 #迭代次数  
    #样本特征矩阵x  
    #需要给特征矩阵补一列，即特征x0的值，每个样本的x0都为1  
    x = [[1, 96.79, 2, 1, 2],  
          [1, 110.39, 3, 1, 0],  
          [1, 70.25, 1, 0, 2],  
          [1, 99.96, 2, 1, 1],  
          [1, 118.15, 3, 1, 0],  
          [1, 115.08, 3, 1, 2]]  
    #样本标签向量y  
    y = [287, 343, 199, 298, 340, 350]
```

编号	面积(x <sub>1</sub> )	卧室(x <sub>2</sub> )	朝向(x <sub>3</sub> )	楼层(x <sub>4</sub> )	房价(y)
1	96.79	2	南北 1	高楼层 2	287w
2	110.39	3	南北 1	低楼层 0	343w
3	70.25	1	北 0	高楼层 2	199w
4	99.96	2	南北 1	中楼层 1	298w
5	118.15	3	南北 1	低楼层 0	340w
6	115.08	3	南北 1	高楼层 2	350w

$$X = \begin{bmatrix} 1 & 96.79 & 2 & 1 & 2 \\ 1 & 110.39 & 3 & 1 & 0 \\ 1 & 70.25 & 1 & 0 & 2 \\ 1 & 99.96 & 2 & 1 & 1 \\ 1 & 118.15 & 3 & 1 & 0 \\ 1 & 115.08 & 3 & 1 & 2 \end{bmatrix} \quad y = \begin{bmatrix} 287 \\ 343 \\ 199 \\ 298 \\ 340 \\ 350 \end{bmatrix}$$



# gradient\_descent模型训练

```
theta = gradient_descent(x, y, n, m, alpha, iterate)
costJ = costJ(x, y, theta, n, m)
#打印参数列表与代价值
print("After %d iterate, theta is:"%(iterate))
for i in range(0, len(theta)):
    print("theta[%d] = %.3lf"%(i, theta[i]))
print("Cost J is %lf"%(costJ))
```

#打印6个训练样本的预测值与真实值进行比对

```
print("Check h and y:")
for i in range(0, m):
    h = hypothesis(theta, x[i], n)
    print("i = %d h = %.3f y = %d"%(i, h, y[i]))
```

```
test1 = [1, 112, 3, 1, 0]
```

```
test2 = [1, 110, 3, 1, 1]
```

#打印两个测试样本的结果

```
print("test1 = %.3f"%(hypothesis(theta, test1, n)))
print("test2 = %.3f" % (hypothesis(theta, test2, n)))
```

After 1500 iterate, theta is:

theta[0] = -0.053

theta[1] = 2.974

theta[2] = 0.365

theta[3] = 0.196

theta[4] = -0.199

Cost J is 40.559510

Check h and y:

i = 0 h = 288.292 y = 287

i = 1 h = 329.496 y = 343

i = 2 h = 208.811 y = 199

i = 3 h = 297.918 y = 298

i = 4 h = 352.571 y = 340

i = 5 h = 343.044 y = 350

test1 = 334.283

test2 = 328.137



# 特征缩放的算法设计

房屋面积	缩放后	卧室个数	缩放后
60	-0.446	1	-0.5
200	0.554	4	0.5
80	-0.304	2	-0.167
150	0.196	3	0.167

01

均值归一化

$$nx = \frac{(x - \text{average})}{(\max\_x - \min\_x)}$$

02

均值标准化

$$S = \sqrt{\frac{\sum_{i=1}^m (x_i - \bar{x})^2}{m - 1}}$$

# 特征范围的缩放

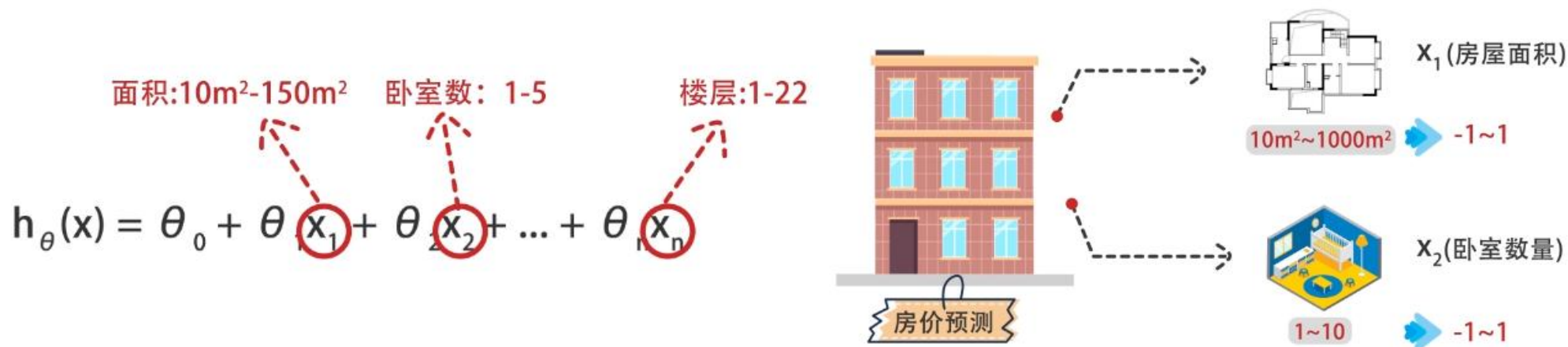
在多元线性回归中，使用多个特征来预测回归问题的结果，这些特征可能具有不同的取值范围。

例如，在房价预测时，包括两个特征 $x_1$ 和 $x_2$ ：

$x_1$ 是房屋的面积，取值范围一般在10平米到1000平米。

$x_2$ ，是卧室的数量，取值范围在1到10。

这两个特征的取值范围相差很大，我们需要将两个特征的取值缩放到相近的范围后，才能保证梯度下降算法更快速的收敛。





# 举例说明

设置参数 $\theta_1$ 为特征 $x_1$ 的系数， $\theta_2$ 是 $x_2$ 的系数。

在空间坐标系中，暂时忽略掉参数 $\theta_0$ ，画出 $J(\theta)$ 随 $\theta_1$ 和 $\theta_2$ 变化的等高图。

如果 $x_1$ 与 $x_2$ 的取值范围相差很大，那么就会得到一个形状非常细长的 $J(\theta)$ 。

基于这样的代价函数运行梯度下降算法，需要走很长的路径，花费很长时间才能收敛到全局最小值。

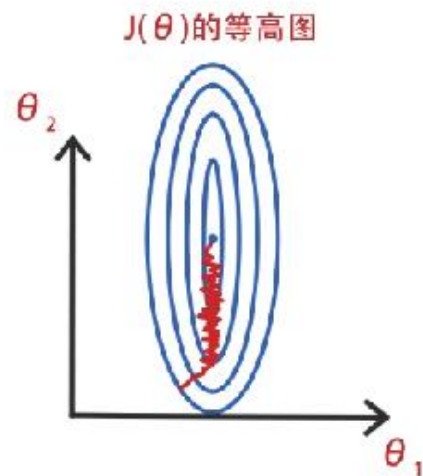
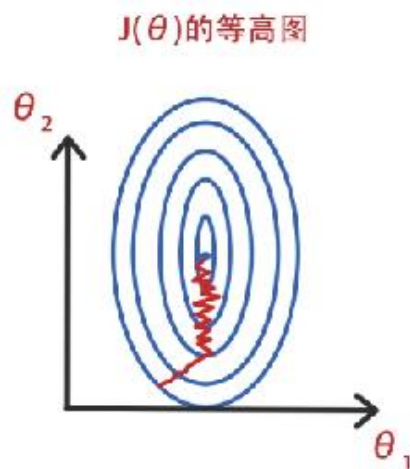
例如， $x_1$ 使用平方厘米来表示，梯度下降就会执行的更加缓慢，需要更长的时间收敛。

$\theta_1$ :  $x_1$ 的系数

$\theta_2$ :  $x_2$ 的系数

$x_1$ : 1000~100000

$x_2$ : 1~10





# 举例说明

如果将 $x_1$ 和 $x_2$ 的取值缩放到相同的范围，就会得到一个形状对称的 $J(\theta)$ 。

对它进行梯度下降，由于到达最小值的路径更短，所以很快就可以收敛到全局最小值。

关于它们的对比过程，会通过实验来验证。

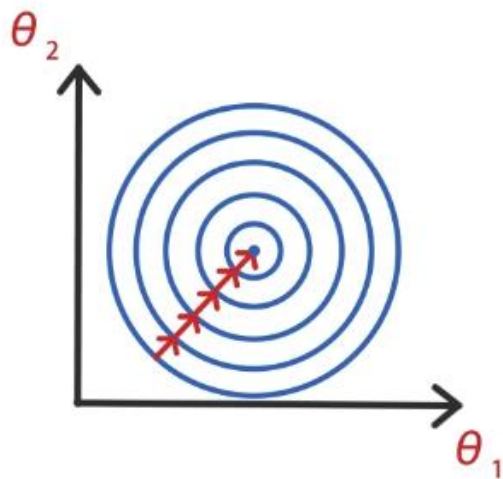
$\theta_1$ :  $x_1$ 的系数

$\theta_2$ :  $x_2$ 的系数

$x_1$ :  $-1 \sim 1$

$x_2$ :  $-1 \sim 1$

$J(\theta)$ 的等高图





# 均值归一化

对于特征缩放，有多种实现的方式。最常见的是均值归一化和均值标准化。

在均值归一化的算法中，所有特征的特征值都约束在-1到1之间。

设 $n_x$ 为某样本特征 $x$ 缩放后的取值。为了计算 $n_x$ ，需要先计算所有样本中特征 $x$ 的平均值 $average$ 、最大值 $max\_x$ 和最小值 $min\_x$ 。

而 $n_x = x - average$ 的差，再除以 $x$ 的取值范围( $max\_x - min\_x$ 的差)。

特征值在  
-1到1之间

1 均值归一化

2 均值标准化

$n_x$  特征 $x$ 缩放后取值

$average\_x$ :  $x$ 的平均值

$max\_x$ : 最大值

$min\_x$ : 最小值

$$n_x = \frac{(x - average)}{(max\_x - min\_x)}$$



# 均值归一化举例

例如，有四个样本，每个样本有房屋面积和卧室个数两个特征，对应x1和x2。

对x1缩放，计算x1的平均值： $(60+200+80+150)/4 = 122.5$ 。

x1的范围是最大值200减最小值60，等于140。

对样本1的x1缩放： $(60-122.5)/140 = -0.446$ 。

对样本2的x1缩放： $(200-122.5)/140 = 0.554$ 。其他样本的x1特征的缩放也按照同样的计算方式计算。

## 1 均值归一化

$x_1$		$x_2$	
房屋面积	缩放后	卧室个数	缩放后
60	-0.446	1	
200	0.554	4	
80	-0.304	2	
150	0.196	3	

$$nx = \frac{(x - \text{average})}{(\max\_x - \min\_x)}$$

$$\begin{aligned} \text{average\_}x_1 &= (60+200+80+150)/4 = 122.5 \end{aligned}$$

$$\begin{aligned} \max\_x_1 - \min\_x_1 &= 200-60 = 140 \end{aligned}$$

$$(\text{样本1})nx_1 = \frac{60 - 122.5}{140} = -0.446$$

$$(\text{样本1})nx_2 = \frac{200 - 122.5}{140} = 0.554$$

$$(\text{样本1})nx_3 = \frac{80 - 122.5}{140} = -0.304$$

$$(\text{样本1})nx_4 = \frac{150 - 122.5}{140} = 0.196$$



# 均值归一化举例

对于x2缩放，先计算出平均值2.5和范围3，然后将每个样本x2的特征值减去x2的平均值2.5再除以3，就会得到x2的缩放结果。

## 1 均值归一化

$x_1$		$x_2$	
房屋面积	缩放后	卧室个数	缩放后
60	-0.446	1	-0.5
200	0.554	4	0.5
80	-0.304	2	-0.167
150	0.196	3	0.167

$$nx = \frac{(x - \text{average})}{(\max\_x - \min\_x)}$$

$$\begin{aligned}\text{average\_}x_2 \\ &= (1+4+2+3)/4 = 2.5\end{aligned}$$

$$\begin{aligned}\max\_x_2 - \min\_x_2 \\ &= 4-1 = 3\end{aligned}$$

$$(\text{样本2})nx_1 = \frac{1 - 2.5}{3} = -0.5$$

$$(\text{样本2})nx_2 = \frac{4 - 2.5}{3} = 0.5$$

$$(\text{样本2})nx_3 = \frac{2 - 2.5}{3} = -0.167$$

$$(\text{样本2})nx_4 = \frac{3 - 2.5}{3} = 0.167$$



## 均值标准化

在均值标准化中，如果对特征x进行特征缩放，同样需要计算所有样本中特征x的平均值average。

不同的是，将最大值减最小值得到的取值范围替换为该特征的标准差S。

设有m个样本，其中待计算标准差的特征为x:

根据标准差S的计算公式，累加每个样本特征x的值减去特征平均值的平方，然后除以m-1，再求平方根。

例如，计算特征x1的标准差S。

## 2 均值标准化

$X_1$		$X_2$	
房屋面积	缩放后	卧室个数	缩放后
60	-0.446	1	-0.5
200	0.554	4	0.5
80	-0.304	2	-0.167
150	0.196	3	0.167

$$\text{average\_x}_1 = (600+200+80+150)/4 = 122.5$$

$\max_x - \min_x$

$S = \sqrt{\frac{\sum_{i=1}^m (x_i - \bar{x})^2}{m - 1}}$

$$S(x_1) = \sqrt{\frac{(60-122.5)^2 + (200-122.5)^2 + (80-122.5)^2 + (150-122.5)^2}{3}}$$

# 特征缩放的代码实现

$x_1$	$x_2$	$y$
房屋面积	卧室个数	房价
9679	2	287w
11039	3	343w
7025	1	199w
9996	2	298w
11815	3	340w
11508	3	350w

$$S = \sqrt{\frac{\sum_{i=1}^m (x_i - \bar{x})^2}{m - 1}}$$



# 均值标准化函数

#计算特征x的标准差，函数传入某一个特征x的全部取值

#样本个数m和特征x的平均取值average

```
def standard_deviation(x, m, average):  
    sum = 0.0 #保存累加和的结果  
    for i in range(0, m): #循环m个样本  
        #将每个样本的特征x的取值与平均值差的平方累加到sum  
        sum += (x[i] - average) * (x[i] - average)  
    sum = sum / (m - 1) #除以m-1  
    return math.sqrt(sum) #求出算数平方根返回
```

$x_1$	$x_2$	y
房屋面积	卧室个数	房价
9679	2	287w
11039	3	343w
7025	1	199w
9996	2	298w
11815	3	340w
11508	3	350w

$m = 6$        $\text{average\_}x_2$   
 $= (2+3+1+2+3+3)/6$   
 $= 2.33$

$$S(x_2) = \sqrt{\frac{(2-2.33)^2 + (3-2.33)^2 + (1-2.33)^2 + (2-2.33)^2 + (3-2.33)^2 + (3-2.33)^2}{6-1}} = 0.82$$



# 特征缩放函数

#将传入的特征值矩阵x进行特征缩放，其中包括n个特征和m个样本  
#函数返回特征值矩阵中每个特征的平均值与标准差

```
def feature_normalize(x, n, m):  
    average = [0] * (n + 1) #保存每个特征的平均值  
    s = [0] * (n + 1) #保存每个特征的标准差  
    #使用i遍历特征x1到xn  
    for i in range(1, n + 1):  
        sum = 0.0 #设置sum累加特征值的和  
        temp = list() #temp保存当前正在处理的特征xi的全部取值  
        for j in range(0, m):  
            sum += x[j][i]  
            temp.append(x[j][i])  
        #计算特征xi的平均值和标准差  
        average[i] = sum / m  
        s[i] = standard_deviation(temp, m, average[i])  
    #遍历每个样本  
    for i in range(0, m):  
        x[i][0] = 1.0 #将样本的特征x0设置为1.0  
        for j in range(1, n + 1): #对特征x1到xn进行特征缩放  
            #缩放后的值为当前值减平均值再除以标准差  
            x[i][j] = (x[i][j] - average[j]) / s[j]  
    return average, s #返回平均值列表和标准差列表
```

$$X = \begin{matrix} & \begin{matrix} x_1 & x_2 \end{matrix} \\ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} & \begin{bmatrix} 9679 & 2 \\ 11039 & 3 \\ 7025 & 1 \\ 9996 & 2 \\ 11815 & 3 \\ 11508 & 3 \end{bmatrix} \end{matrix}$$

sum = [61062, 14]

temp = [9679, 11039, ..., 11508]  
[2, 3, 1, 2, 3, 3]

average = [10177, 2.33]

s = [1755.82, 0.82]

以此类推

$$x_1 = \frac{9679 - 10177}{1755.82} = -0.28$$

$$x_2 = \frac{2 - 2.33}{0.82} = -0.41$$





# 设置初始化的数据

设置6个样本，每个样本2个特征，迭代速率alpha和迭代次数iterate。

定义样本的特征矩阵x和样本标签y。

两个特征的特征值取值范围差别要设置的大一些，比如特征x1的取值在1w左右，x2的取值在1左右。

```
if __name__ == '__main__':
    m = 6 # 6个样本
    n = 2 # 每个样本2个特征
    alpha = 0.01 # 更大迭代速率
    #alpha = 0.0001 # 迭代速率
    #alpha = 0.00000001 # 非常小的迭代速率才可能正常迭代
    iterate = 1500 # 迭代次数iterate
    #样本的特征矩阵x和样本标签y
    #其中两个特征的特征值取值范围差别要设置的大一些
    x = [[1, 9679, 2],
          [1, 11039, 3],
          [1, 7025, 1],
          [1, 9996, 2],
          [1, 11815, 3],
          [1, 11508, 3]]
    y = [287, 343, 199, 298, 340, 350]
    #完成样本的特征值缩放
    average, s = feature_normalize(x, n, m)
    #运行梯度下降算法对参数进行迭代
    theta = gradient_descent(x, y, n, m, alpha, iterate)
    costJ = costJ(x, y, theta, n, m)
    for i in range(0, n + 1):
        print("theta[%d] = %lf"%(i, theta[i]))
    print("costJ = %lf"%(costJ))
    test1 = [1, 11200, 3]
    test2 = [1, 11000, 3]
    #对两个测试样本进行预测
    print(normalize_hypothesis(theta, test1, n, average, s))
    print(normalize_hypothesis(theta, test2, n, average, s))
```

$x_1$	$x_2$	$y$
房屋面积	卧室个数	房价
9679	2	287w
11039	3	343w
7025	1	199w
9996	2	298w
11815	3	340w
11508	3	350w

$$X = \begin{bmatrix} 1 & 9679 & 2 \\ 1 & 11039 & 3 \\ 1 & 7025 & 1 \\ 1 & 9996 & 2 \\ 1 & 11815 & 3 \\ 1 & 11508 & 3 \end{bmatrix} \quad y = \begin{bmatrix} 287 \\ 343 \\ 199 \\ 298 \\ 340 \\ 350 \end{bmatrix}$$





# 运行梯度下降算法

运行梯度下降算法对参数进行迭代，会发现结果直接超出了数据显示的范围。这是由于迭代速率alpha过大引起的问题。

因为特征取值范围的原因，引起了其中一个特征对于迭代速率更加敏感，而另外一个特征不敏感的情况。

除非我们将alpha调整到非常小的值，比如0.0000001，才能正常的进行迭代，但这样又会引起迭代很难收敛到函数最小值的问题。

```
if __name__ == '__main__':  
    m = 6 # 6个样本  
    n = 2 # 每个样本2个特征  
    alpha = 0.01 # 更大迭代速率  
    #alpha = 0.0001 # 迭代速率  
    #alpha = 0.00000001 # 非常小的迭代速率才可能正常迭代  
    iterate = 1500 # 迭代次数iterate  
    #样本的特征矩阵x和样本标签y  
    #其中两个特征的特征值取值范围差别要设置的大一些  
    x = [[1, 9679, 2],  
         [1, 11039, 3],  
         [1, 7025, 1],  
         [1, 9996, 2],  
         [1, 11815, 3],  
         [1, 11508, 3]]  
    y = [287, 343, 199, 298, 340, 350]  
    #完成样本的特征值缩放  
    average, s = feature_normalize(x, n, m)  
    #运行梯度下降算法对参数进行迭代  
    theta = gradient_descent(x, y, n, m, alpha, iterate)  
    costJ = costJ(x, y, theta, n, m)  
    for i in range(0, n + 1):  
        print("theta[%d] = %lf"%(i, theta[i]))  
    print("costJ = %lf"%(costJ))  
    test1 = [1, 11200, 3]  
    test2 = [1, 11000, 3]  
    #对两个测试样本进行预测  
    print(normalize_hypothesis(theta, test1, n, average, s))  
    print(normalize_hypothesis(theta, test2, n, average, s))
```

$x_1$	$x_2$	$y$
房屋面积	卧室个数	房价
9679	2	287w
11039	3	343w
7025	1	199w
9996	2	298w
11815	3	340w
11508	3	350w

$x_1$ : 取值范围1w左右

$x_2$ : 取值范围1左右

~~alpha = 0.001~~ ➡ alpha = 0.00000001





# 预测样本数据的特征缩放

由于对特征进行了特征缩放，所以未来在使用线性回归进行预测时，同样需要对预测样本的特征值进行特征缩放。然后再代入到 $h_{\theta}(x)$ 中，才能得到正确的预测结果。

1	9679	2
1	11039	3
1	7025	1
1	9996	2
1	11815	3
1	11508	3

特征缩放前



1	-0.28	-0.41
1	-0.49	0.82
1	-1.8	-1.62
1	-0.1	-0.41
1	0.93	0.82
1	0.76	0.82

特征缩放后

$$h_{\theta}(x) = \theta_0 + \theta_1 * 9679 + \theta_2 * 2$$

$$h_{\theta}(x) = \theta_0 + \theta_1 * (-0.28) + \theta_2 * (-0.41)$$





# 带特征缩放的预测函数

#对样本进行特征缩放并预测样本的特征值

#函数传入参数列表theta、特征向量x、特征个数n、每个特征的平均值与标准差

```
def normalize_hypothesis(theta, x, n, average, s):  
    nx = [0] * (n + 1) #保存特征缩放后的特征值列表  
    nx[0] = 1.0  
    for i in range(1, n + 1): #将x1到xn进行缩放  
        nx[i] = (x[i] - average[i]) / s[i]  
    h = 0.0 # 保存预测结果  
    for i in range(0, n + 1):  
        # 将theta-i和xi的乘积累加到h中  
        h += theta[i] * nx[i]  
    return h # 返回预测结果h
```

$$\begin{matrix} \begin{bmatrix} 1 & 9679 & 2 \\ 1 & 11039 & 3 \\ 1 & 7025 & 1 \\ 1 & 9996 & 2 \\ 1 & 11815 & 3 \\ 1 & 11508 & 3 \end{bmatrix} & \xrightarrow{\text{blue arrow}} & \begin{bmatrix} 1 & -0.28 & -0.41 \\ 1 & -0.49 & 0.82 \\ 1 & -1.8 & -1.62 \\ 1 & -0.1 & -0.41 \\ 1 & 0.93 & 0.82 \\ 1 & 0.76 & 0.82 \end{bmatrix} \\ \mathbf{x} & & \mathbf{nx} \end{matrix}$$

average = [10177, 2.33]

s = [1755.82, 0.82]

$$h_{\theta}(x) = \theta_0 + \theta_1 * (-0.28) + \theta_2 * (-0.41)$$

$$h_{\theta}(x) = \theta_0 + \theta_1 * (-0.49) + \theta_2 * (0.82)$$

⋮

$$h_{\theta}(x) = \theta_0 + \theta_1 * (0.76) + \theta_2 * (0.82)$$

# main函数



```
if __name__ == '__main__':  
    m = 6 # 6个样本  
    n = 2 # 每个样本2个特征  
    alpha = 0.01 # 更大迭代速率  
    #alpha = 0.0001 # 迭代速率  
    #alpha = 0.00000001 # 非常小的迭代速率才可能正常迭代  
    iterate = 1500 # 迭代次数iterate  
    #样本的特征矩阵x和样本标签y  
    #其中两个特征的特征值取值范围差别要设置的大一些  
    x = [[1, 9679, 2],  
         [1, 11039, 3],  
         [1, 7025, 1],  
         [1, 9996, 2],  
         [1, 11815, 3],  
         [1, 11508, 3]]  
    y = [287, 343, 199, 298, 340, 350]  
    #完成样本的特征值缩放  
    average, s = feature_normalize(x, n, m)  
    #运行梯度下降算法对参数进行迭代  
    theta = gradient_descent(x, y, n, m, alpha, iterate)  
    costJ = costJ(x, y, theta, n, m)  
    for i in range(0, n + 1):  
        print("theta[%d] = %lf"%(i, theta[i]))  
    print("costJ = %lf"%(costJ))  
    test1 = [1, 11200, 3]  
    test2 = [1, 11000, 3]  
    #对两个测试样本进行预测  
    print(normalize_hypothesis(theta, test1, n, average, s))  
    print(normalize_hypothesis(theta, test2, n, average, s))
```

```
theta[0] = 302.833247  
theta[1] = 30.449821  
theta[2] = 26.402722  
costJ = 26.298273  
342.132649041  
338.664091919
```





# 标准方程方法的数学表示

$$X = \begin{bmatrix} x_0 & x_1 & x_2 & \dots & x_n \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ \dots \\ n \end{matrix} * \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{bmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ \dots \\ n \end{matrix}$$

$$h_{\theta}(x) = X * \theta = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$





# 知识回顾

在前面的课程中，我们学习了基于迭代的梯度下降算法来解决线性回归问题，通过一轮轮的迭代，求出代价函数  $J(\theta)$  的全局最小值。

求多元函数  $J(\theta)$  取得最小值时，参数  $\theta_0$  到  $\theta_n$  的值。可以分别求出  $J(\theta)$  关于  $\theta_0$ 、 $\theta_1$  等等到  $\theta_n$  的偏导数，然后令这  $n+1$  个算式等于 0。

联立后，会得到关于  $\theta_0$  到  $\theta_n$ ，一共  $n+1$  个未知数的线性方程。求出这个线性方程的解，就解决了线性回归问题。

$$\begin{array}{c} \nabla J(\theta_0, \theta_1, \theta_2, \dots, \theta_n) \\ \downarrow \\ \min(J(\theta_0, \theta_1, \theta_2, \dots, \theta_n)) \end{array} \quad \begin{array}{c} n+1 \uparrow \\ \left\{ \begin{array}{l} \frac{\partial J}{\partial \theta_0} = 0 \\ \frac{\partial J}{\partial \theta_1} = 0 \\ \vdots \\ \frac{\partial J}{\partial \theta_n} = 0 \end{array} \right. \end{array} \Rightarrow \left\{ \begin{array}{l} \theta_0 = ? \\ \theta_1 = ? \\ \vdots \\ \theta_n = ? \end{array} \right.$$



# 标准方程方法

基于线性代数与微积分的知识，可以直接求出线性回归矩阵方程的解。

如果用矩阵表示方程的解，参数矩阵 $\theta$ 可以表示为关于特征向量矩阵 $X$ 与标记矩阵 $y$ 的相关形式，根据该公式可以直接求出 $\theta_0$ 到 $\theta_n$ 。

该方法也被称为正规方程法、最小二乘法。

矩阵方程公式：

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

线性代数与  
微积分的知识

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

参数  $\theta$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

特征向量  $x$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

样本标签  $y$





# 特征与参数的矩阵表示

样本包含 $n$ 个特征，设样本向量的一般形式 $X$ ，是一个 $1 \times (n+1)$ 的行向量。

将方程的全部参数 $\theta$ ，保存到 $(n+1) \times 1$ 的列向量中。

预测值 $h_{\theta}(x) = X * \theta$ 。

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

$$h_{\theta}(x) = X * \theta = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$

$$X = \begin{bmatrix} 0 & 1 & 2 & \dots & n \\ x_0 & x_1 & x_2 & \dots & x_n \end{bmatrix} * \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$



# 预测值和真实值的矩阵表示

设训练集中包含 $m$ 条数据， $X$ 为一个 $m \times (n+1)$ 的矩阵。

保存 $m$ 条样本数据的矩阵 $X$ 乘以参数矩阵 $\theta$ ，会得到一个 $m \times 1$ 的矩阵，在这个矩阵中，保存了 $m$ 个样本的预测值。

将 $m$ 个样本的标记值，保存到 $m \times 1$ 的列向量 $y$ 中。

$$\begin{aligned} X &= \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \quad m \times (n+1) \quad * \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix} \\ &= \begin{bmatrix} \theta_0 + \theta_1 x_{11} + \theta_2 x_{12} + \dots + \theta_n x_{1n} \\ \theta_0 + \theta_1 x_{21} + \theta_2 x_{22} + \dots + \theta_n x_{2n} \\ \dots \\ \theta_0 + \theta_1 x_{m1} + \theta_2 x_{m2} + \dots + \theta_n x_{mn} \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_m \end{bmatrix} \quad m \times 1 \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix} \quad m \times 1 \end{aligned}$$





# 代价函数的矩阵表示

基于特征向量矩阵 $X$ 、参数矩阵 $\theta$ 、样本标记矩阵 $y$ ，表示 $J(\theta)$ 。

将 $y - X * \theta$ 展开，得到一个 $m$ 乘 $1$ 的列向量，保存了 $m$ 个样本的真实值与预测值的差。

将该矩阵转置，得到 $1 * m$ 的矩阵，转置后的矩阵乘以自身，就是矩阵中每个元素的平方和。该结果除以 $2m$ ，得到 $J(\theta)$ 。

$J(\theta)$ 的矩阵形式

$$J(\theta) = \frac{1}{2m} (y - X * \theta)^T (y - X * \theta)$$

$J(\theta)$ 矩阵表示方式推导过程

$$y - X * \theta = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix} - \begin{bmatrix} h_1 \\ h_2 \\ \dots \\ h_m \end{bmatrix} = \begin{bmatrix} y_1 - h_1 \\ y_2 - h_2 \\ \dots \\ y_m - h_m \end{bmatrix}$$

$$\begin{aligned} & (y - X * \theta)^T (y - X * \theta) \\ &= \begin{bmatrix} y_1 - h_1 & y_2 - h_2 & \dots & y_m - h_m \end{bmatrix} * \begin{bmatrix} y_1 - h_1 \\ y_2 - h_2 \\ \dots \\ y_m - h_m \end{bmatrix} \\ &= (y_1 - h_1)^2 + (y_2 - h_2)^2 + \dots + (y_m - h_m)^2 \\ &= \sum_{i=1}^m (h_i - y_i)^2 \\ &= \sum_{i=1}^m (h_{\theta}(x_i) - y_i)^2 \end{aligned}$$

第 $i$ 个样本



# 总结

完成 $J(\theta)$ 的矩阵表示，回到最初的问题，即求 $n+1$ 元的线性方程的解。

该过程可以基于 $J(\theta)$ 的矩阵形式进行计算。想一想如何进行推导求解？

$$\min(J(\theta_0, \theta_1, \theta_2, \dots, \theta_n))$$

$$n+1 \uparrow \begin{cases} \frac{\partial J}{\partial \theta_0} = 0 \\ \frac{\partial J}{\partial \theta_1} = 0 \\ \vdots \\ \frac{\partial J}{\partial \theta_n} = 0 \end{cases} \Rightarrow \begin{cases} \theta_0 = ? \\ \theta_1 = ? \\ \vdots \\ \theta_n = ? \end{cases}$$


$J(\theta)$ 的矩阵形式

$$J(\theta) = \frac{1}{2m} (y - X * \theta)^T (y - X * \theta)$$




# 标准方程解的计算与推导

$$J(\theta) = \frac{1}{2m} (y - X * \theta)^T (y - X * \theta)$$



$$\min(J(\theta))$$



$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix}$$

公式1:  $\frac{dAX}{dX} = A^T$

公式2:  $\frac{dX^T A}{dX} = A$

公式3:  $\frac{dX^T A X}{dX} = (A + A^T)X$





# 问题归纳

在代价函数 $J(\theta)$ 的矩阵形式中， $X$ 是一个 $m \times (n+1)$ 的矩阵，代表 $m$ 个样本的特征向量。

$y$ 是 $m \times 1$ 的列向量，代表样本的标记向量。 $\theta$ 是 $(n+1) \times 1$ 的列向量，代表了 $n+1$ 个特征的特征系数。

我们需要求出使得 $J(\theta)$ 取得最小值时，列向量 $\theta$ 中各项元素的值。

$$J(\theta) = \frac{1}{2m} (y - X * \theta)^T (y - X * \theta)$$



$\min(J(\theta))$

$$X = \begin{bmatrix} 0 & 1 & 2 & \dots & n \\ 1 & x_{11} & x_{12} & \dots & x_{1n} \\ 1 & x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{m1} & x_{m2} & \dots & x_{mn} \end{bmatrix} \begin{matrix} 0 \\ 1 \\ \dots \\ m-1 \end{matrix}$$

特征向量

$$y = \begin{bmatrix} 0 \\ y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix} \begin{matrix} 0 \\ 1 \\ \dots \\ m-1 \end{matrix}$$

样本标记

$$\theta = \begin{bmatrix} 0 \\ \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix} \begin{matrix} 0 \\ 1 \\ \dots \\ n-1 \end{matrix}$$

特征系数





# 问题归纳

将 $X$ 、 $y$ 、 $\theta$ 看做是一个整体，进行矩阵的加、减、乘和求导运算。

即求出代价函数 $J(\theta)$ 对于矩阵 $\theta$ 的偏导数，令该偏导数为0，从而求矩阵 $\theta$ 的值。

在这个方程中，矩阵 $\theta$ 是未知数，矩阵 $X$ 和 $y$ 是已知的。为了解出该矩阵方程，需要对矩阵运算进行化简，并对矩阵求导。

$$J(\theta) = \frac{1}{2m} (y - X * \theta)^T (y - X * \theta)$$

$$\frac{\partial}{\partial \theta} J(\theta) = 0 \Rightarrow \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix}$$

- 未知数： $\theta$
- 已知数： $X, y$

$$\frac{\partial (y - X * \theta)^T (y - X * \theta)}{\partial \theta} = 0$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix}$$

化简

求导



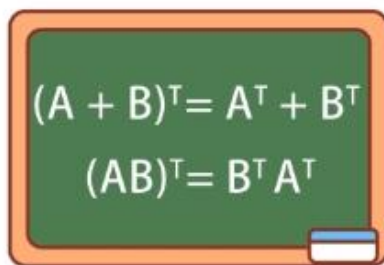


# 问题化简

根据矩阵转置的公式进行化简：

- 1) 将转置运算到括号内，得到 $y^T - \theta^T X^T$ ，然后再乘以 $y - X\theta$ 。
- 2) 将该算式展开，会得到一个关于 $\theta$ 矩阵的多项式，一共有4项，然后分别对这四项求关于 $\theta$ 矩阵的偏导数。

01  
化简



矩阵转置公式

$$\begin{aligned} & \frac{\partial (y - X\theta)^T (y - X\theta)}{\partial \theta} \\ &= \frac{\partial (y^T - \theta^T X^T)(y - X\theta)}{\partial \theta} \\ &= \frac{\partial (y^T(y - X\theta) - \theta^T X^T(y - X\theta))}{\partial \theta} \\ &= \frac{\partial (y^T y - y^T X \theta - \theta^T X^T y + \theta^T X^T X \theta)}{\partial \theta} \\ &= \frac{\partial y^T y}{\partial \theta} - \frac{\partial y^T X \theta}{\partial \theta} - \frac{\partial \theta^T X^T y}{\partial \theta} + \frac{\partial \theta^T X^T X \theta}{\partial \theta} \end{aligned}$$



# 矩阵求导

在公式中，矩阵X是未知量，矩阵A是常数矩阵，即求关于X的偏导数。

注意，在使用公式时，对应各项的顺序是不能变化的，每一项都要对应使用。

另外，我们不必深究矩阵求导的公式是如何推导的，只需要知道如何使用。



● 未知量：X    ● 常数：A

公式1:  $\frac{d\mathbf{A}\mathbf{X}}{d\mathbf{X}} = \mathbf{A}^T$

公式2:  $\frac{d\mathbf{X}^T\mathbf{A}}{d\mathbf{X}} = \mathbf{A}$

公式3:  $\frac{d\mathbf{X}^T\mathbf{A}\mathbf{X}}{d\mathbf{X}} = (\mathbf{A} + \mathbf{A}^T)\mathbf{X}$

$\frac{\partial \mathbf{y}^T \mathbf{X} \theta}{\partial \theta}$

$\frac{\partial \theta^T \mathbf{X}^T \mathbf{y}}{\partial \theta}$

$\frac{\partial \theta^T \mathbf{X}^T \mathbf{X} \theta}{\partial \theta}$

# 公式1



常数矩阵A乘未知矩阵X，对它的求导结果是矩阵A的转置。

公式1:  $\frac{dAX}{dX} = A^T$

公式2:  $\frac{dX^T A}{dX} = A$

公式3:  $\frac{dX^T A X}{dX} = (A + A^T)X$

$$(A * X)' = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$





# 公式2



X转置乘常数矩阵A，对它的求导结果是矩阵A。

公式1:  $\frac{dAX}{dX} = A^T$

公式2:  $\frac{dX^T A}{dX} = A$

公式3:  $\frac{dX^T A X}{dX} = (A + A^T)X$

$$(X^T A)' = \begin{bmatrix} x_0 & x_1 & x_2 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$



# 公式3



对矩阵X转置、常数矩阵A、矩阵X，这三项乘积求导，结果是矩阵A加矩阵A的转置的和，再乘X。

公式1:  $\frac{dAX}{dX} = A^T$

公式2:  $\frac{dX^T A}{dX} = A$

公式3:  $\frac{dX^T A X}{dX} = (A + A^T)X$

$$(X^T A X)' = \begin{bmatrix} x_0 & x_1 & x_2 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

$$= \left( \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix} \right) * \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$



# 根据公式进行化简

第1项是对常数矩阵求导，结果是0。

第2、3、4项，需要将未知量矩阵 $\theta$ ，看做是公式中的矩阵 $X$ ，已知 $X$ 和 $y$ 整体看做公式中的矩阵 $A$ ：

第2项，根据公式1， $y$ 的转置乘 $X$ 看做是矩阵 $A$ ， $\theta$ 是 $X$ 。

第3项，根据公式2，将 $X$ 的转置乘 $y$ 看做 $A$ 。

第4项，根据公式3，将 $X$ 的转置乘 $X$ 看做 $A$ 。

$$\frac{\partial y^T y}{\partial \theta} - \frac{\partial y^T X \theta}{\partial \theta} - \frac{\partial \theta^T X^T y}{\partial \theta} + \frac{\partial \theta^T X^T X \theta}{\partial \theta} = 0 - X^T y - X^T y + 2X^T X \theta$$

$$\text{公式1: } \frac{dAX}{dX} = A^T$$

$$\frac{\overset{A}{\partial y^T X} \theta}{\partial \theta} = (y^T X)^T = X^T y$$

$$\text{公式2: } \frac{dX^T A}{dX} = A$$

$$\frac{\partial \theta^T \overset{A}{X^T y}}{\partial \theta} = X^T y$$

$$\text{公式3: } \frac{dX^T A X}{dX} = (A + A^T) X$$

$$\begin{aligned} \frac{\partial \theta^T \overset{A}{X^T X} \theta}{\partial \theta} &= (X^T X + (X^T X)^T) \theta \\ &= 2X^T X \theta \end{aligned}$$



# 解矩阵方程

经过矩阵求导后，令四项结果相加后为0，继续整理得到方程。

解关于 $\theta$ 的矩阵方程，方程的两边同时乘以 $X$ 转置乘 $X$ 的逆矩阵，从解出未知矩阵 $\theta$ 。

$$\frac{\partial y^T y}{\partial \theta} - \frac{\partial y^T X \theta}{\partial \theta} - \frac{\partial \theta^T X^T y}{\partial \theta} + \frac{\partial \theta^T X^T X \theta}{\partial \theta} = \underline{0 - X^T y - X^T y + 2X^T X \theta}$$

$$0 - X^T y - X^T y + 2X^T X \theta = 0$$

$$-2X^T y + 2X^T X \theta = 0$$

移项： $X^T X \theta = X^T y$

化简： $\underline{(X^T X)^{-1} X^T X} \theta = \underline{(X^T X)^{-1} X^T y}$

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



# 总结



在代价函数 $J(\theta)$ 取得最小值时，参数矩阵 $\theta$ 的值，它是一个关于特征向量矩阵 $X$ 和标签矩阵 $y$ 的算式。该方法即为标准方程方法。

$$0 - X^T y - X^T y + 2X^T X \theta = 0$$

$$-2X^T y + 2X^T X \theta = 0$$

移项： $X^T X \theta = X^T y$

化简： $(X^T X)^{-1} X^T X \theta = (X^T X)^{-1} X^T y$

$$\theta = \underline{(X^T X)^{-1} X^T y}$$

$$J(\theta) = \frac{1}{2m} (y - X * \theta)^T (y - X * \theta)$$



$\min(J(\theta))$

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix}$



# 标准方程算法的设计与实现


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

$$X = \begin{matrix} & \begin{matrix} 0 & 1 & 2 & 3 & 4 \end{matrix} \\ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} & \begin{bmatrix} 96.79 \\ 110.39 \\ 70.25 \\ 99.96 \\ 118.15 \\ 115.08 \end{bmatrix} & \begin{bmatrix} 2 \\ 3 \\ 1 \\ 2 \\ 3 \\ 3 \end{bmatrix} & \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} & \begin{bmatrix} 2 \\ 0 \\ 2 \\ 1 \\ 0 \\ 2 \end{bmatrix} \end{matrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$
$$y = \begin{matrix} & 0 \\ \begin{bmatrix} 287 \\ 343 \\ 199 \\ 298 \\ 340 \\ 350 \end{bmatrix} & \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{matrix} \end{matrix}$$



# 本节内容

基于标准方程的解设计并实现标准方程算法，解决多元线性回归问题。



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



$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$$



# 使用举例



设训练集中有6个样本，每个样本4个特征。构造出特征向量矩阵X和样本标签矩阵y。

根据公式，计算出矩阵X的转置，将它和矩阵X相乘后求逆，然后计算乘以X转置和y的结果。

编号	面积( $x_1$ )	卧室( $x_2$ )	朝向( $x_3$ )	楼层( $x_4$ )	房价(y)
1	96.79	2	南北 1	高楼层 2	287w
2	110.39	3	南北 1	低楼层 0	343w
3	70.25	1	北 0	高楼层 2	199w
4	99.96	2	南北 1	中楼层 1	298w
5	118.15	3	南北 1	低楼层 0	340w
6	115.08	3	南北 1	高楼层 2	350w

$$\theta = \underline{(X^T X)^{-1} X^T y}$$

$$X = \begin{bmatrix} 1 & 96.79 & 2 & 1 & 2 \\ 1 & 110.39 & 3 & 1 & 0 \\ 1 & 70.25 & 1 & 0 & 2 \\ 1 & 99.96 & 2 & 1 & 1 \\ 1 & 118.15 & 3 & 1 & 0 \\ 1 & 115.08 & 3 & 1 & 2 \end{bmatrix}$$

$$y = \begin{bmatrix} 287 \\ 343 \\ 199 \\ 298 \\ 340 \\ 350 \end{bmatrix}$$





# 使用举例

例如， $X^T$ 是5\*6的矩阵， $X$ 是6\*5的矩阵，它们相乘会得到一个5乘5的方阵。

对于方阵求逆矩阵，是不确定是否能求出结果的，是否有解，需要要看矩阵的行列式值是否不为0。

如果行列式的值不为0，才可以继续按照公式，计算出矩阵 $\theta$ 。

$$\theta = \underline{(X^T X)^{-1} X^T y}$$

$$X^T = \begin{bmatrix} 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\ 96.79 & 110.39 & 70.25 & 99.96 & 118.15 & 115.08 \\ 2.00 & 3.00 & 1.00 & 2.00 & 3.00 & 3.00 \\ 1.00 & 1.00 & 0.00 & 1.00 & 1.00 & 1.00 \\ 2.00 & 0.00 & 2.00 & 1.00 & 0.00 & 2.00 \end{bmatrix}$$

$$X = \begin{bmatrix} 1.00 & 96.79 & 2.00 & 1.00 & 2.00 \\ 1.00 & 110.39 & 3.00 & 1.00 & 0.00 \\ 1.00 & 70.25 & 1.00 & 0.00 & 2.00 \\ 1.00 & 99.96 & 2.00 & 1.00 & 1.00 \\ 1.00 & 118.15 & 3.00 & 1.00 & 0.00 \\ 1.00 & 115.08 & 3.00 & 1.00 & 2.00 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 6.00 & 610.62 & 14.00 & 5.00 & 7.00 \\ 610.62 & 63684.15 & 1494.61 & 540.37 & 664.20 \\ 14.00 & 1494.61 & 36.00 & 13.00 & 14.00 \\ 5.00 & 540.37 & 13.00 & 5.00 & 5.00 \\ 7.00 & 664.20 & 14.00 & 5.00 & 13.00 \end{bmatrix}$$

$$\det(X^T X) = 0 \quad \times$$

$$\underline{\det(X^T X) \neq 0} \quad \checkmark$$

方阵



# python的实现方法

- 1)直接调用numpy的库函数计算numpy数组。
- 2)将numpy数组转为矩阵，然后再进行计算。



调用库函数：

```
numpy.transpose  
numpy.dot  
numpy.linalg.det
```



转矩阵后计算：

```
numpy.mat  
 $XTX.I * X.T * y$ 
```

# 方法1



#函数传入特征向量矩阵和标签列向量y

```
def normal_equation_array(X, y):  
    XT = numpy.transpose(X) #调用transpose计算矩阵X的转置  
    XTX = numpy.dot(XT, X) #调用dot计算X转置乘X  
    det = numpy.linalg.det(XTX) #计算结果矩阵的行列式的值  
    #在行列式值不为0的情况下,才可以进行求逆矩阵的运算  
    if det == 0.0:  
        return "error"  
    XTX_inv = numpy.linalg.inv(XTX) #计算XTX的逆矩阵  
    XTy = numpy.dot(XT, y) #计算X转置和y相乘  
    theta = numpy.dot(XTX_inv, XTy) #计算theta矩阵  
    return theta
```

$$X = \begin{bmatrix} 1.00 & 96.79 & 2.00 & 1.00 & 2.00 \\ 1.00 & 110.39 & 3.00 & 1.00 & 0.00 \\ 1.00 & 70.25 & 1.00 & 0.00 & 2.00 \\ 1.00 & 99.96 & 2.00 & 1.00 & 1.00 \\ 1.00 & 118.15 & 3.00 & 1.00 & 0.00 \\ 1.00 & 115.08 & 3.00 & 1.00 & 2.00 \end{bmatrix} \quad y = \begin{bmatrix} 287.00 \\ 343.00 \\ 199.00 \\ 298.00 \\ 340.00 \\ 350.00 \end{bmatrix}$$

$$X^T = \begin{bmatrix} 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\ 96.79 & 110.39 & 70.25 & 99.96 & 118.15 & 115.08 \\ 2.00 & 3.00 & 1.00 & 2.00 & 3.00 & 3.00 \\ 1.00 & 1.00 & 0.00 & 1.00 & 1.00 & 1.00 \\ 2.00 & 0.00 & 2.00 & 1.00 & 0.00 & 2.00 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 6.00 & 610.62 & 14.00 & 5.00 & 7.00 \\ 610.62 & 63684.15 & 1494.61 & 540.37 & 664.20 \\ 14.00 & 1494.61 & 36.00 & 13.00 & 14.00 \\ 5.00 & 540.37 & 13.00 & 5.00 & 5.00 \\ 7.00 & 664.20 & 14.00 & 5.00 & 13.00 \end{bmatrix}$$

$$(X^T X)^{-1} = \begin{bmatrix} 88.17 & -1.54 & 23.06 & 19.09 & -1.14 \\ -1.54 & 0.03 & -0.45 & -0.34 & 0.00 \\ 23.06 & -0.45 & 8.29 & 4.01 & 0.19 \\ 19.09 & -0.34 & 4.01 & 7.43 & -0.16 \\ -1.14 & 0.00 & 0.19 & -0.16 & 0.32 \end{bmatrix}$$

# 方法1图解



$$X = \begin{bmatrix} 1.00 & 96.79 & 2.00 & 1.00 & 2.00 \\ 1.00 & 110.39 & 3.00 & 1.00 & 0.00 \\ 1.00 & 70.25 & 1.00 & 0.00 & 2.00 \\ 1.00 & 99.96 & 2.00 & 1.00 & 1.00 \\ 1.00 & 118.15 & 3.00 & 1.00 & 0.00 \\ 1.00 & 115.08 & 3.00 & 1.00 & 2.00 \end{bmatrix} \quad y = \begin{bmatrix} 287.00 \\ 343.00 \\ 199.00 \\ 298.00 \\ 340.00 \\ 350.00 \end{bmatrix}$$

$$X^T = \begin{bmatrix} 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\ 96.79 & 110.39 & 70.25 & 99.96 & 118.15 & 115.08 \\ 2.00 & 3.00 & 1.00 & 2.00 & 3.00 & 3.00 \\ 1.00 & 1.00 & 0.00 & 1.00 & 1.00 & 1.00 \\ 2.00 & 0.00 & 2.00 & 1.00 & 0.00 & 2.00 \end{bmatrix}$$

$$X^T X = \begin{bmatrix} 6.00 & 610.62 & 14.00 & 5.00 & 7.00 \\ 610.62 & 63684.15 & 1494.61 & 540.37 & 664.20 \\ 14.00 & 1494.61 & 36.00 & 13.00 & 14.00 \\ 5.00 & 540.37 & 13.00 & 5.00 & 5.00 \\ 7.00 & 664.20 & 14.00 & 5.00 & 13.00 \end{bmatrix}$$

$$(X^T X)^{-1} = \begin{bmatrix} 88.17 & -1.54 & 23.06 & 19.09 & -1.14 \\ -1.54 & 0.03 & -0.45 & -0.34 & 0.00 \\ 23.06 & -0.45 & 8.29 & 4.01 & 0.19 \\ 19.09 & -0.34 & 4.01 & 7.43 & -0.16 \\ -1.14 & 0.00 & 0.19 & -0.16 & 0.32 \end{bmatrix}$$

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

$$= \begin{bmatrix} 88.17 & -1.54 & 23.06 & 19.09 & -1.14 \\ -1.54 & 0.03 & -0.45 & -0.34 & 0.00 \\ 23.06 & -0.45 & 8.29 & 4.01 & 0.19 \\ 19.09 & -0.34 & 4.01 & 7.43 & -0.16 \\ -1.14 & 0.00 & 0.19 & -0.16 & 0.32 \end{bmatrix} \begin{bmatrix} 1.00 & 1.00 & 1.00 & 1.00 & 1.00 & 1.00 \\ 96.79 & 110.39 & 70.25 & 99.96 & 118.15 & 115.08 \\ 2.00 & 3.00 & 1.00 & 2.00 & 3.00 & 3.00 \\ 1.00 & 1.00 & 0.00 & 1.00 & 1.00 & 1.00 \\ 2.00 & 0.00 & 2.00 & 1.00 & 0.00 & 2.00 \end{bmatrix} \begin{bmatrix} 287.00 \\ 343.00 \\ 199.00 \\ 298.00 \\ 340.00 \\ 350.00 \end{bmatrix}$$

$$= \begin{bmatrix} 124.56 \\ 0.32 \\ 48.26 \\ 37.22 \\ 1.89 \end{bmatrix}$$





## 方法2

#直接使用numpy中的矩阵类，使用起来更简单方便

```
def normal_equation_matrix(X, y):  
    #将数组x和y转换为矩阵  
    X = numpy.mat(X)  
    y = numpy.mat(y)  
    #矩阵X，可以直接调用X点T来得到X的转置  
    XTX = X.T * X #矩阵的相乘可以直接使用乘法符号*号  
    det = numpy.linalg.det(XTX)  
    if det == 0.0:  
        return "error"  
    #X点I得到X的逆矩阵，根据公式，计算出theta矩阵  
    theta = XTX.I * X.T * y  
    return theta
```

$$X = \begin{bmatrix} 1 & 96.79 & 2 & 1 & 2 \\ 1 & 110.39 & 3 & 1 & 0 \\ 1 & 70.25 & 1 & 0 & 2 \\ 1 & 99.96 & 2 & 1 & 1 \\ 1 & 118.15 & 3 & 1 & 0 \\ 1 & 115.08 & 3 & 1 & 2 \end{bmatrix} \quad y = \begin{bmatrix} 287 \\ 343 \\ 199 \\ 298 \\ 340 \\ 350 \end{bmatrix}$$

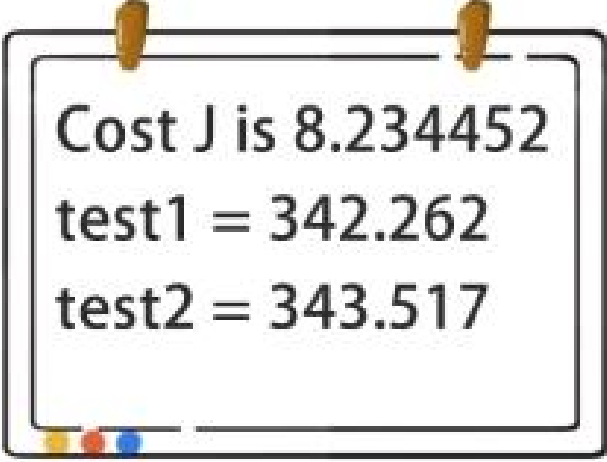
$$\theta = (X^T X)^{-1} X^T y = \begin{bmatrix} 124.56 \\ 0.32 \\ 48.26 \\ 37.22 \\ 1.89 \end{bmatrix}$$



# main函数

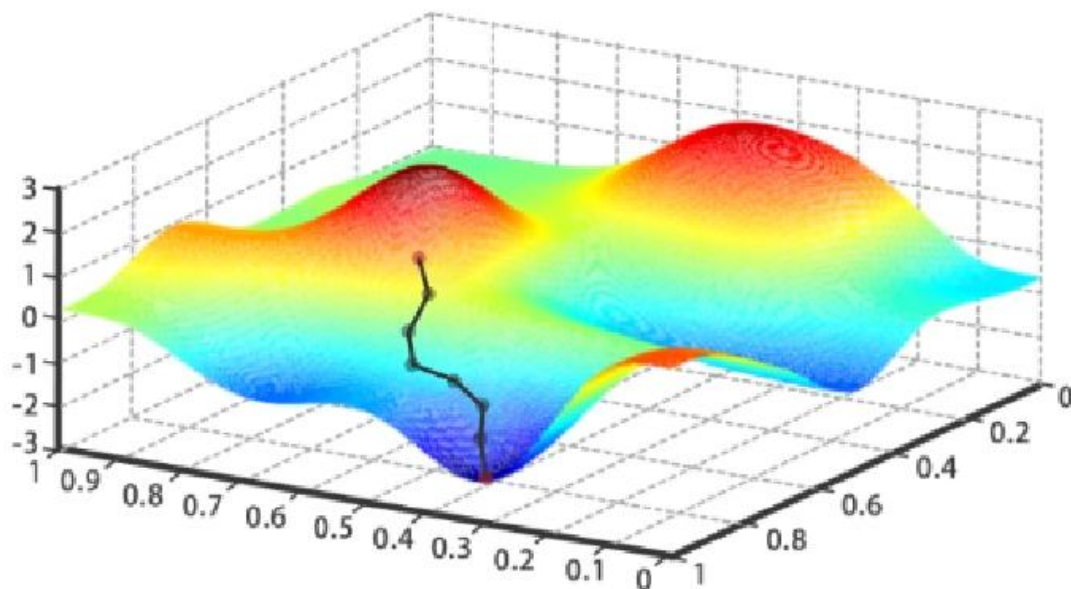


```
if __name__ == '__main__':  
    #设置特征向量矩阵X  
    X = numpy.array([[1, 96.79, 2, 1, 2],  
                     [1, 110.39, 3, 1, 0],  
                     [1, 70.25, 1, 0, 2],  
                     [1, 99.96, 2, 1, 1],  
                     [1, 118.15, 3, 1, 0],  
                     [1, 115.08, 3, 1, 2]])  
  
    #设置标签列向量y  
    y = numpy.array([[287],  
                     [343],  
                     [199],  
                     [298],  
                     [340],  
                     [350]])  
  
    #测试两个函数，求出theta的值  
    theta = normal_equation_array(X, y)  
    print(theta)  
    theta = normal_equation_matrix(X, y)  
    print(theta)  
  
    #计算代价J  
    costJ = (y - X * theta).T * (y - X * theta) / (2 * len(y))  
    print("Cost J is %lf" % (costJ))  
    # 预测两个样本结果  
    test1 = [1, 112, 3, 1, 0]  
    test2 = [1, 110, 3, 1, 1]  
    # 打印两个测试样本的结果  
    print("test1 = %.3f" % (test1 * theta))  
    print("test2 = %.3f" % (test2 * theta))
```



Cost J is 8.234452  
test1 = 342.262  
test2 = 343.517

# 梯度下降与标准方程算法总结



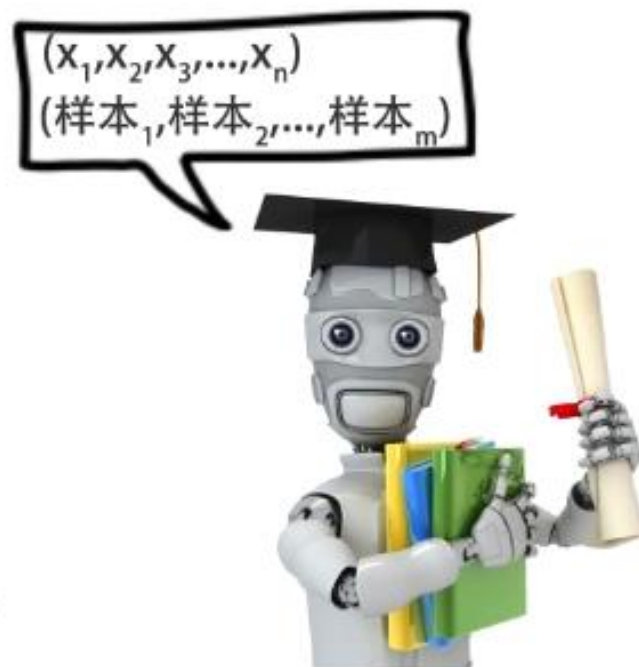
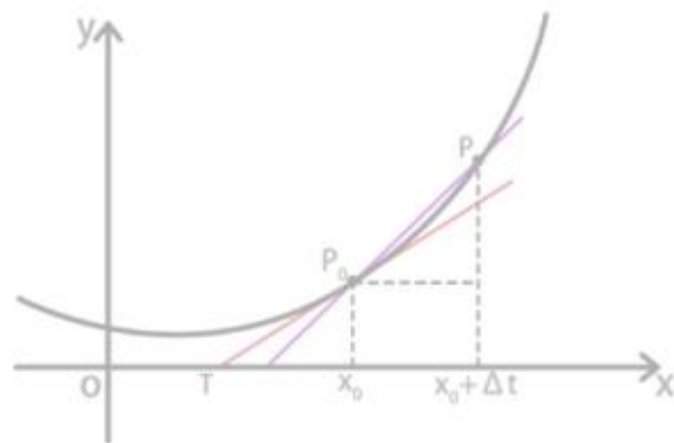

$$\theta = (X^T X)^{-1} X^T y$$

# 多元线性回归的矩阵不可逆情况



思考：多元线性回归中，什么时候会出现矩阵不可逆的情况呢？

这里不去以数学的角度进行推导和证明，只从机器学习中特征向量或样本的角度来解释说明。

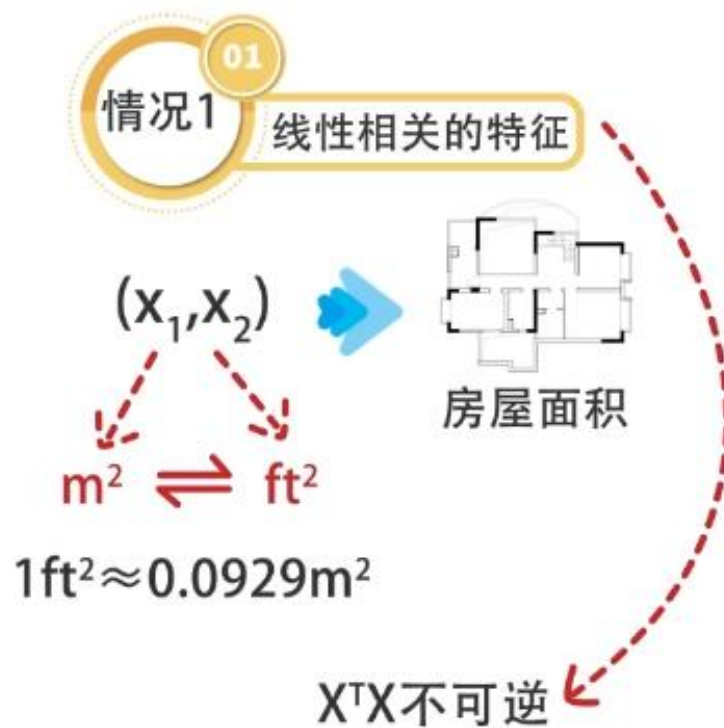


# 情况1

在特征向量中，包括了线性相关的特征。

例如，特征 $x_1$ 和 $x_2$ 都代表房子的面积， $x_1$ 的单位是平方米， $x_2$ 的单位是平方英尺。

由于平方米和平方英尺满足线性换算关系，这就导致这两个特征线性相关，进而导致特征向量矩阵 $X$ 转置乘 $X$ 的结果不可逆。





## 情况2

样本数量 $m$ 小于或等于特征数量 $n$ ，即训练数据过少，或者描述样本的特征太多了。

例如，训练集有10个样本，每个样本都提取了100个特征，这时会出现矩阵不可逆的情况。



(样本<sub>1</sub>, 样本<sub>2</sub>, ..., 样本 <sub>$m$</sub> )  $m=10$

( $x_1, x_2, x_3, \dots, x_n$ )  $n=100$

样本数 $m \leq$  特征数 $n$  ➡  $X^T X$ 不可逆





# 解决的方法

当上面这两种情况发生时，无法直接使用标准方程方法求解线性回归。

解决上述问题的方法很简单，最核心的寻找关键特征，将多余的无效特征删去。

很多时候，对特征描述或提取方式的优化是提升机器学习模型训练效果的关键方法。



✗ 
$$J(\theta) = \frac{1}{2m} (y - X^* \theta)^T (y - X^* \theta)$$

~~( $x_1, x_2, x_3$ ,  $x_4$ ,  $x_5, x_6, \dots, x_n$ )~~

特征提取

$x_1$ : 房屋面积     $x_2$ : 房间数量

特征描述



# 梯度下降与标准方程的对比

从代码实现来看，标准方程方法更加简单，可以认为只需要一步就求出了最优解。

梯度下降方法需要选择合适的迭代速率，并且进行很多轮次的迭代。

## 标准方程法

```
#将数组X和y转换为矩阵
X = numpy.mat(X)
y = numpy.mat(y)
#矩阵X，可以直接调用X点T来得到X的转置
#矩阵的相乘可以直接使用乘法符号*号
XTX = X.T * X
det = numpy.linalg.det(XTX)
if det == 0.0:
    return "error"
#X点I得到X的逆矩阵，
#根据公式，计算出theta矩阵
theta = XTX.I * X.T * y
return theta
```

## 梯度下降法

```
#梯度下降的迭代函数
#函数传入样本特征矩阵x和样本标签y，
#特征数n，样本数量m
#迭代速率alpha和迭代次数iterate
def gradient_descent(x, y, n, m, alpha, iterate):
    #初始化参数列表theta，长度为n+1
    theta = [0] * (n + 1)
    for i in range(0, iterate): #梯度下降的迭代循环
        temp = [0] * (n + 1)
        #使用变量j，同时对theta0到theta-n
        #这n+1个参数进行更新
        for j in range(0, n + 1):
            #通过临时变量列表temp，
            #先保存一次梯度下降后的结果
```



# 梯度下降方法的特点

在实际的应用中，对于机器学习模型训练，更常使用的还是梯度下降算法。

该方法的容错性强，加上很多种优化方式。例如正则化，可以使该方法解决特征过多或无效特征的问题。

而且还能轻松处理较大规模数量的样本。它的时间复杂度和迭代次数 $t$ 、样本数 $m$ 、特征数 $n$ 呈正相关。

方法	优点	缺点	时间复杂度
梯度下降法 (常用)	容错性强 多种优化方式 轻松处理大规模样本	需要合适的学习速率 需要多次迭代 只能得到最优解的近似值	
标准方程法	不需要学习速率 无需迭代 直接计算最优解	样本数量较多时间复杂度较高 $X^T X$ 逆矩阵不存在则无法使用	

正则化

$(x_1, x_2, x_3, \dots, x_n)$

$n=100$

$x_1$ : 面积  $x_2$ : 楼层  ~~$x_3$ : 单元楼号~~



# 标准方程方法的特点

标准方程方法内部包括了矩阵相乘、矩阵求逆等运算，在样本数量或者样本特征较多时，时间复杂度是比较高的。  
该方法需要判断逆矩阵的存在，实际的使用场景并不是很广泛。正则化同样可以优化标准方程方法。

方法	优点	缺点	时间复杂度
梯度下降法 (常用)	<u>容错性强</u> <u>多种优化方式</u> <u>轻松处理大规模样本</u>	需要合适的学习速率 需要多次迭代 只能得到最优解的近似值	$O(tmn)$
标准方程法 (不广泛使用)	不需要学习速率 无需迭代 直接计算最优解	<u>样本数量较多时间复杂度较高</u> <u><math>X^T X</math>逆矩阵不存在则无法使用</u>	$O(n^2 * m + n^3)$

○ 矩阵相乘： $X * Y$

○ 矩阵求逆： $X^{-1}$

○ 矩阵转置： $X^T$

样本<sub>1</sub>, 样本<sub>2</sub>, 样本<sub>3</sub>, ..., 样本<sub>m</sub>  $m=10000$

$(x_1, x_2, x_3, \dots, x_n)$   $n=5000$





# 预习指导问题

同学们在视频学习前，请尝试回答如下问题，并在视频学习后，将答案进行归纳与整理：

1. 在实现多元线性回归时，与实现一元线性回归有什么不同？
2. 如果线性回归中的多个特征的取值范围差别很大时，会出现什么问题？
3. 特征缩放可以解决什么问题，模型训练前一定要进行特征缩放吗？
4. 使用特征缩放后进行模型训练，在预测时也需要对样本进行特征缩放处理吗？
5. 如果两个或多个特征之间存在高度的相关性，会出现什么问题？如何避免这种问题的发生？
6. 如何计算一个矩阵的逆矩阵，为什么会出现矩阵不可逆的情况？
7. 标准方程方法的解的表达式是怎样的？它是如何推导出来的？
8. 对比梯度下降法和标准方程法，各自的优缺点是什么？
9. 在实现标准方程方法时，需要注意什么问题？
10. 当我们有一个大规模的数据集时，你会选择使用标准方程方法还是梯度下降法进行求解？为什么？





# 课后编程练习



同学们在视频学习后，请尝试如下编程练习，并在直播课中，跟着老师一起完成全部编程作业。

- 1.请使用python，实现梯度下降算法解决多元线性回归的代码。
- 2.请使用python，实现带有特征缩放的多元线性回归代码。
- 3.请收集kaggle平台上的数据，训练并调试多元线性回归模型。
- 4.请使用sklearn中的线性回归接口训练模型，并与手动实现的效果做对比。
- 5.请使用python，实现多元线性回归的标准方程方法。
- 6.请基于pytorch，实现多元线性回归的训练。





最后将所有问题，都直接与小黑黑老师讨论清楚吧！

微信号:xhh890921

