4. 多变量线性回归(Linear Regression with Multiple Variables)

4.1 多维特征

符号约定：

$$n：特征的数量$$
$$\theta：参数$$
$$x(i)：第i个训练实例，是特征矩阵中的第i行，是一个向量（vector）$$
$$x_j^{(i)}：特征矩阵中第i行的第j个, 第i个训练实例的第j个特征$$
$$支持多变量的假设函数h：h\theta(x)=\theta 0+\theta 1x1+\theta 2x2+...+\theta nxn$$

$$第2个训练样本：x^{(2)}=\begin{bmatrix}1416\\40\\3\\2\end{bmatrix}$$

$$第二个样本的第二个特征x_2^{(2)}=3第二个样本的第三个特征x_3^{(2)}=2$$
$$假设函数中有n+1个参数和n个变量，为了使得公式能够简化，引入x_0=1，则公式为$$
$$\theta_0x0+\theta 1x1+\theta 2x2+...\theta nxn$$
$$此时模型中的参数是一个n+1维的向量，任何一个训练实例也都是n+1维的向量，$$
$$特征矩阵X的维度是m*(n+1)。所以公式可以简化为：h\theta(x)=\theta^TX，$$
$$其中上标T代表矩阵转置$$

1. **特征向量：**
   - 每个样本有多个特征，以第二个样本为例，第二个样本的特征向量为 $x^{(2)} = [1, x_1^{(2)}, x_2^{(2)}, x_3^{(2)}]$，其中 $x_1^{(2)} = 1, x_2^{(2)} = 3, x_3^{(2)} = 2$。
   - 这里引入了 $x_0 = 1$，作为特征向量中的第一个元素，目的是为了简化模型表示。
2. **参数向量：**
   - 假设函数中有 $n+1$ 个参数，即 $\theta_0, \theta_1, \ldots, \theta_n$。
   - 参数向量为 $\theta = [\theta_0, \theta_1, \ldots, \theta_n]$。
3. **线性模型表示：**
   - 假设函数 $h_\theta(x)$ 通过参数向量和特征向量的内积来表示，即 $h_\theta(x) = \theta^T x$。
   - 这里 θ^T 代表参数向量的转置。
4. **特征矩阵 X：**
   - 假设有 m 个样本，每个样本有 $n+1$ 个特征，将所有样本的特征向量按行排列形成特征矩阵 X，其维度是 $m \times (n+1)$。
   - 具体而言，第 i 行表示第 i 个样本的特征向量。
5. **简化表示：**
   - 为了方便表示，引入 $x_0 = 1$，使得特征向量的第一个元素为1，从而将参数向量和特征向量的内积表示简化为 $\theta^T x$。

综合起来，模型的假设函数可以简化为 $h_\theta(x) = \theta^T x$，其中 θ 是参数向量，x 是特征向量。这种形式更方便用矩阵表示，并在训练和预测中进行操作。

4.2 多变量梯度下降

与单变量线性回归类似，在多变量线性回归中，我们也构建一个代价函数，

则这个代价函数是所有建模误差的平方和，即：

$$J(\theta_0, \theta_1, \ldots \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}),$$

其中$h_\theta(x) = \theta^T X = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$

我们的目标和单变量线性回归问题中一样，是要找出使得代价函数最小的一系列参数。 多变量线性回归的批量梯度下降算法为：

$$
\begin{aligned}
&\text{Repeat } \{ \\
&\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1, \ldots, \theta_n) \\
&\}
\end{aligned}
$$

即：

$$
\begin{aligned}
&\text{Repeat } \{ \\
&\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \\
&\}
\end{aligned}
$$

求导数后得到：

$$
\begin{aligned}
&\text{Repeat } \{ \\
&\quad \theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}) \\
&\quad (\text{ simultaneously update } \theta_j \\
&\quad \text{for } j=0,1,\ldots,n\ ) \\
&\}
\end{aligned}
$$

当$n >= 1$时，$\theta_0 := \theta_0 - a\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$

$\theta_1 := \theta_1 - a\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}$

$\theta_2 := \theta_2 - a\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}$

我们开始随机选择一系列的参数值，计算所有的预测结果后，再给所有的参数一个新的值，如此循环直到收敛。

代码示例：

计算代价函数 $J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$ 其中：$h_\theta(x) = \theta^T X = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$

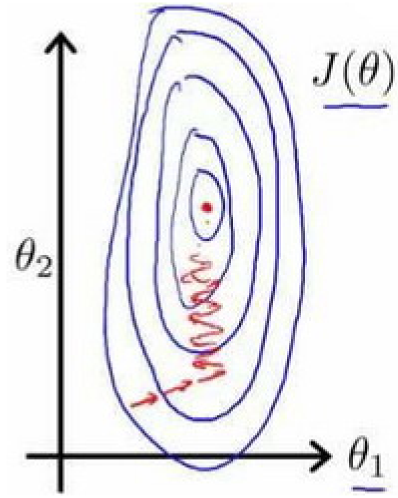
Python 代码：

```python
def computeCost(X, y, theta):
    inner = np.power(((X * theta.T) - y), 2)
    return np.sum(inner) / (2 * len(X))
```

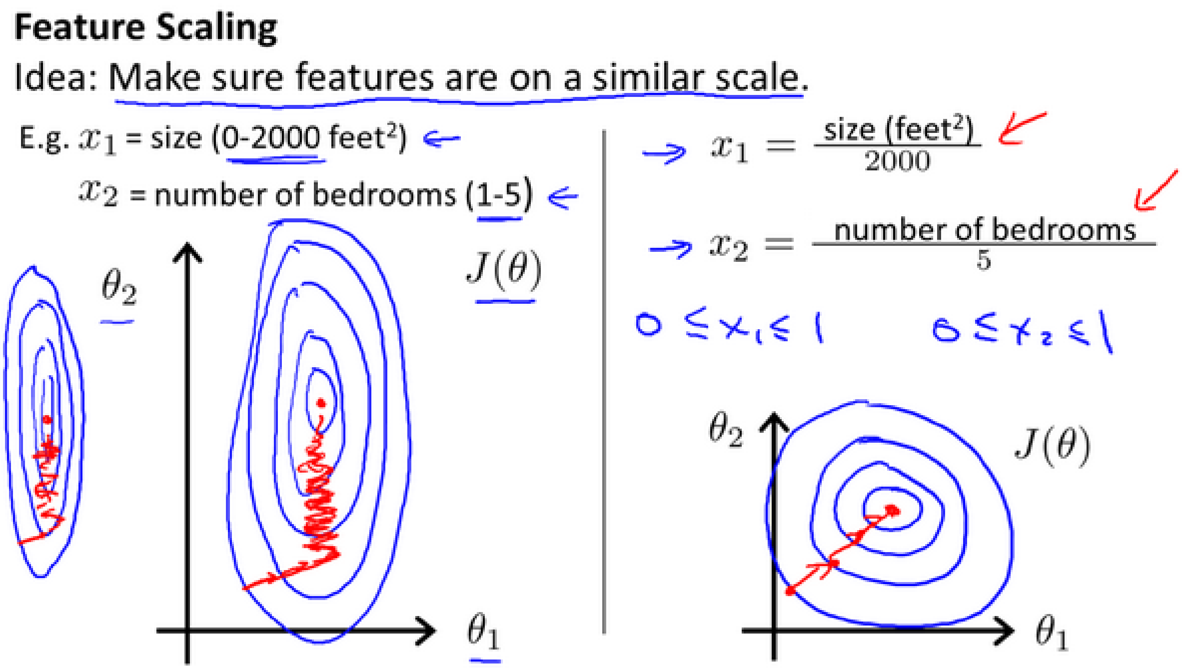
4.3 梯度下降实践1-特征缩放（归一化）

在我们面对多维特征问题的时候，我们要保证这些特征都具有相近的尺度，这将帮助梯度下降算法更快地收敛。

以房价问题为例，假设我们使用两个特征，房屋的尺寸和房间的数量，尺寸的值为 0-2000平方英尺，而房间数量的值则是0-5，以两个参数分别为横纵坐标，绘制代价函数的等高线图能，看出图像会显得很扁，梯度下降算法需要非常多次的迭代才能收敛。



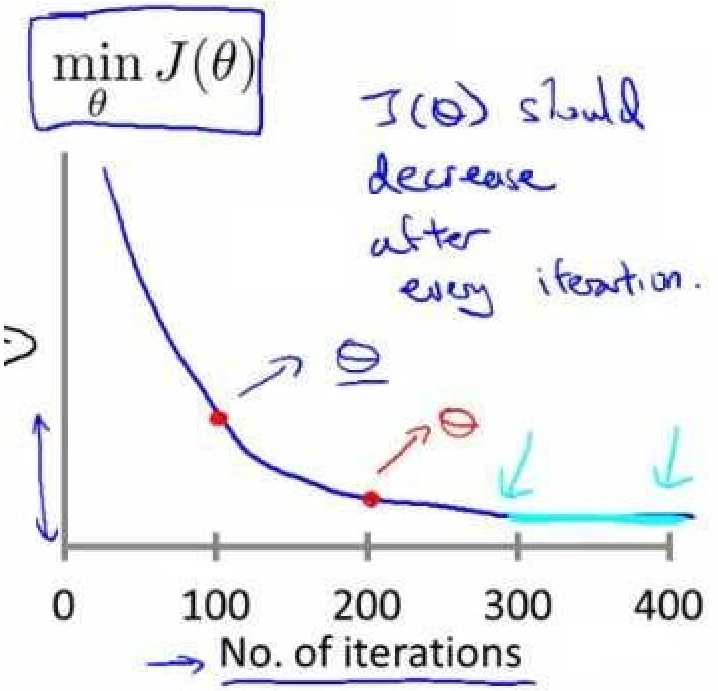
解决的方法是尝试将所有特征的尺度都尽量缩放到-1到1之间。如图：



最简单的方法是令$x_n = \frac{x_n - u_n}{sn}$其中$u_n$是平均值，$s_n$是标准差

4.4 梯度下降法实践2-学习率

梯度下降算法收敛所需要的迭代次数根据模型的不同而不同，我们不能提前预知，我们可以绘制迭代次数和代价函数的图表来观测算法在何时趋于收敛。



也有一些自动测试是否收敛的方法，例如将代价函数的变化值与某个阀值（例如0.001）进行比较，但通常看上面这样的图表更好。

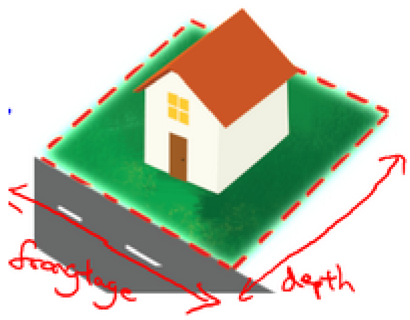
梯度下降算法的每次迭代受到学习率的影响，如果学习率a过小，则达到收敛所需的迭代次数会非常高；如果学习率a过大，每次迭代可能不会减小代价函数，可能会越过局部最小值导致无法收敛。

通常可以考虑尝试些学习率：

$$a = 0.01, 0.03, 0.1, 1, 3, 10$$

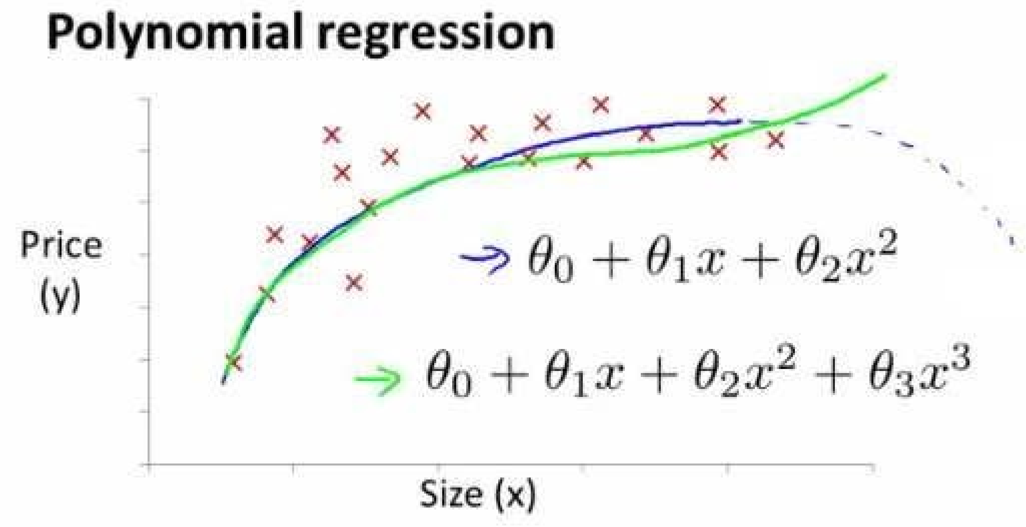
4.5 特征和多项式回归

如房价预测问题，



$$h_\theta(x) = \theta_0 + \theta_1 * frontage + \theta_2 * depth$$
$$x_1 = frontage(\text{临界宽度}) x_2 = depth(\text{纵向深度})$$
$$x = frontage * depth = area(\text{面积})$$
$$\text{则：} h_\theta(x) = \theta_0 + \theta_1 x, \text{线性回归并不适合与所有数据，}$$
$$\text{有时我们需要曲线来适应我们的数据，比如一个二次方程模型：}$$
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 \text{或者三次模型：}$$
$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_2 x_3^3$$

这里为了让函数曲线拥有更好的拟合程度，添加$\theta_2 x_2^2$,$\theta_3 x_3^3$ ，特征变量增加函数的适用性（一次方程，二次方程函数)



通常我们需要先观察数据然后再决定准备尝试怎样的模型，另外，我们可以令：$x_2=x_2^2$，$x_3=x_3^3$，从而将模型转换为线性回归模型。根据函数图形特性，我们还可以使：

$$h\theta(x)=\theta_0+\theta_1(size)+\theta_2(size)^2.\text{或者}$$

$$h\theta(x)=\theta_0+\theta_1(size)+\theta_2\sqrt{size}$$

注：如果我们采用多项式回归模型，在运行梯度下降算法前，特征缩放非常有必要。

4.6 正规方程

到目前为止，我们都在使用梯度下降算法，但是对于某些线性回归问题，正规方程方法是更好的解决方案。如：

正规方程是通过求解下面的方程来找出使得代价函数最小的参数的：$\frac{\partial}{\partial\theta_j}J(\theta_j)=0$。

假设我们的训练集特征矩阵为X（包含了$x_0=1$）并且我们的训练集结果为向量y，则利用正规方程解出向量$\theta=(X^TX)^{-1}X^Ty$。上标T代表矩阵转置，上标-1代表矩阵的逆。设矩阵$A=X^TX$，则：$(X^TX)^{-1}=A^{-1}$以下表示数据为：

解释：

$(X^TX)^{-1}$：矩阵平方

X^T：转置矩阵：矩阵-45°的镜像

转置矩阵的特点（图解）：

（1）转置矩阵的行数是原矩阵的列数，转置矩阵的列数是原矩阵的行数；

（2）转置矩阵下标（i，j）的元素对应于原矩阵下标（j，i）的元素。

向量y：$n*1$的矩阵：这里是数据集所有的结果标签组成

转置矩阵

$$A=\begin{pmatrix}1 & 2 & 2\\ 4 & 5 & 8\end{pmatrix},\qquad A^T=\begin{pmatrix}1 & 4\\ 2 & 5\\ 2 & 8\end{pmatrix};$$

$$B=\begin{pmatrix}18 & 6\end{pmatrix},\qquad B^T=\begin{pmatrix}18\\ 6\end{pmatrix}.$$

例:

Examples: $m = 4.$

	Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X=\begin{bmatrix}1 & 2104 & 5 & 1 & 45\\ 1 & 1416 & 3 & 2 & 40\\ 1 & 1534 & 3 & 2 & 30\\ 1 & 852 & 2 & 1 & 36\end{bmatrix}$$

即:

X(0)	X(1)	X(2)	X(3)	X(4)	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

运用正规方程方法求解参数:

$$\left(\begin{bmatrix}1 & 1 & 1 & 1\\ 2104 & 1416 & 1534 & 852\\ 5 & 3 & 3 & 2\\ 1 & 2 & 2 & 1\\ 45 & 40 & 30 & 36\end{bmatrix} \times \begin{bmatrix}1 & 2104 & 5 & 1 & 45\\ 1 & 1416 & 3 & 2 & 40\\ 1 & 1534 & 3 & 2 & 30\\ 1 & 852 & 2 & 1 & 36\end{bmatrix}\right)^{-1} \times \begin{bmatrix}1 & 1 & 1 & 1\\ 2104 & 1416 & 1534 & 852\\ 5 & 3 & 3 & 2\\ 1 & 2 & 2 & 1\\ 45 & 40 & 30 & 36\end{bmatrix} \times \begin{bmatrix}460\\ 232\\ 315\\ 178\end{bmatrix}$$

注：对于那些不可逆的矩阵（通常是因为特征之间不独立，如同时包含英尺为单位的尺寸和米为单位的尺寸两个特征，也有可能是特征数量大于训练集的数量），正规方程方法是不能用的。

梯度下降与正规方程的比较:

梯度下降	正规方程
需要选择学习率	不需要
需要多次迭代	一次运算得出

梯度下降	正规方程
当特征数量大时也能较好适用	需要计算 如果特征数量n较大则运算代价大，因为矩阵逆的计算时间复杂度为，通常来说当小于10000 时还是可以接受的
适用于各种类型的模型	只适用于线性模型，不适合逻辑回归模型等其他模型

总结一下，只要特征变量的数目并不大，标准方程是一个很好的计算参数的替代方法。具体地说，只要特征变量数量小于一万，我通常使用标准方程法，而不使用梯度下降法。

随着我们要讲的学习算法越来越复杂，例如，当我们讲到分类算法，像逻辑回归算法，我们会看到，实际上对于那些算法，并不能使用标准方程法。对于那些更复杂的学习算法，我们将不得不仍然使用梯度下降法。因此，梯度下降法是一个非常有用的算法，可以用在有大量特征变量的线性回归问题。或者我们以后在课程中，会讲到的一些其他的算法，因为标准方程法不适合或者不能用在它们上。但对于这个特定的线性回归模型，标准方程法是一个比梯度下降法更快的替代算法。所以，根据具体的问题，以及你的特征变量的数量，这两种算法都是值得学习的。

正规方程的**python**实现：

```
import numpy as np

 def normalEqn(X, y):

   theta = np.linalg.inv(X.T@X)@X.T@y #X.T@X等价于X.T.dot(X)

   return theta
```

4.7 正规方程及不可逆性

有些同学曾经问过我，当计算 θ= `inv(X'X ) X'y` ，那对于矩阵$X'X$的结果是不可逆的情况咋办呢? 如果你懂一点线性代数的知识，你或许会知道，有些矩阵可逆，而有些矩阵不可逆。我们称那些不可逆矩阵为奇异或退化矩阵。 问题的重点在于$X'X$的不可逆的问题很少发生，在**Octave**里，如果你用它来实现θ的计算，你将会得到一个正常的解。在**Octave**里，有两个函数可以求解矩阵的逆，一个被称为 `pinv()`，另一个是 `inv()`，这两者之间的差异是些许计算过程上的，一个是所谓的伪逆，另一个被称为逆。使用 `pinv()` 函数可以展现数学上的过程，这将计算出θ的值，即便矩阵$X'X$是不可逆的。

在 `pinv()` 和 `inv()` 之间，又有哪些具体区别呢？

其中 `inv()` 引入了先进的数值计算的概念。例如，在预测住房价格时，如果x_1是以英尺为尺寸规格计算的房子，x_2是以平方米为尺寸规格计算的房子，同时，你也知道1米等于3.28英尺 (四舍五入到两位小数)，这样，你的这两个特征值将始终满足约束：$x_1 = x_2 * (3.28)^2$。 实际上，你可以用这样的一个线性方程，来展示那两个相关联的特征值，矩阵$X'X$将是不可逆的。

第二个原因是，在你想用大量的特征值，尝试实践你的学习算法的时候，可能会导致矩阵$X'X$的结果是不可逆的。 具体地说，在m小于或等于n的时候，例如，有m等于10个的训练样本也有n等于100的特征数量。要找到适合的$(n+1)$ 维参数矢量θ，这将会变成一个101维的矢量，尝试从10个训练样本中找到满足101个参数的值，这工作可能会让你花上一阵子时间，但这并不总是一个好主意。因为，正如我们所看到你只有10个样本，以适应这100或101个参数，数据还是有些少。

稍后我们将看到，如何使用小数据样本以得到这100或101个参数，通常，我们会使用一种叫做正则化的线性代数方法，通过删除某些特征或者是使用某些技术，来解决当m比n小的时候的问题。即使你有一个相对较小的训练集，也可使用很多的特征来找到很多合适的参数。 总之当你发现的矩阵$X'X$的结果是奇异矩阵，或者找到的其它矩阵是不可逆的，我会建议你这么做。

首先，看特征值里是否有一些多余的特征，像这些x_1和x_2是线性相关的，互为线性函数。同时，当有一些多余的特征时，可以删除这两个重复特征里的其中一个，无须两个特征同时保留，将解决不可逆性的问题。因此，首先应该通过观察所有特征检查是否有多余的特征，如果有多余的就删除掉，直到他们不再是多余的为止，如果特征数量实在太多，我会删除些 用较少的特征来反映尽可能多内容，否则我会考虑使用正规化方法。 如果矩阵$X'X$是不可逆的，（通常来说，不会出现这种情况），如果在**Octave**里，可以用伪逆函数 `pinv()` 来实现。这种使用不同的线性代数库的方法被称为伪逆。即使$X'X$的结果是不可逆的，但算法执行的流程是正确的。总之，出现不可逆矩阵的情况极少发生，所以在大多数实现线性回归中，出现不可逆的问题不应该过多的关注X^TX是不可逆的。

增加内容:

$\theta=\left(X^TX\right)^{-1}X^Ty$ 的推导过程:

$J\left(\theta\right)=\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2$ 其中: $h_\theta\left(x\right)=\theta^TX=\theta_0x_0+\theta_1x_1+\theta_2x_2+\ldots+\theta_nx_n$

将向量表达形式转为矩阵表达形式，则有$J(\theta)=\frac{1}{2}(X\theta-y)^2$，其中$X$为$m$行$n$列的矩阵（$m$为样本个数，$n$为特征个数），$\theta$为$n$行1列的矩阵，$y$为$m$行1列的矩阵，对$J(\theta)$进行如下变换

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

$=\frac{1}{2}\left(\theta^TX^T-y^T\right)\left(X\theta-y\right)$

$=\frac{1}{2}\left(\theta^TX^TX\theta-\theta^TX^Ty-y^TX\theta-y^Ty\right)$

接下来对$J(\theta)$偏导，需要用到以下几个矩阵的求导法则:

$\frac{dAB}{dB}=A^T$

$\frac{dX^TAX}{dX}=2AX$

所以有:

$\frac{\partial J(\theta)}{\partial\theta}=\frac{1}{2}\left(2X^TX\theta-X^Ty-(y^TX)^T-0\right)$

$=\frac{1}{2}\left(2X^TX\theta-X^Ty-X^Ty-0\right)$

$=X^TX\theta-X^Ty$

令$\frac{\partial J(\theta)}{\partial\theta}=0$,

则有$\theta=\left(X^TX\right)^{-1}X^Ty$