第一周学习笔记

第一节：引言部分

机器学习按学习方法分类，可以分为监督学习和无监督学习两种。

监督学习这个想法是指，我们将教计算机如何去完成任务，而在无监督学习中，我们打算让它自己进行学习。

无监督学习中没有任何的标签或者是有相同的标签或者就是没标签。所以我们已知数据集，却不知如何处理，也未告知每个数据点是什么。别的都不知道，就是一个数据集。

“监督”一词会不明就里，其实可以把这个词理解为习题的“参考答案”，专业术语叫做“标记”。比如有监督学习就是有参考答案的学习，而无监就是无参考答案。

其中监督学习分为回归问题和分类问题，两者最大的区别在于预测结果是否是连续的。如果预测结果是连续的，那便是回归问题，如果预测结果是离散的，那便是分类问题。

将相似的样本聚合成一类让后进行分析，是聚类问题。

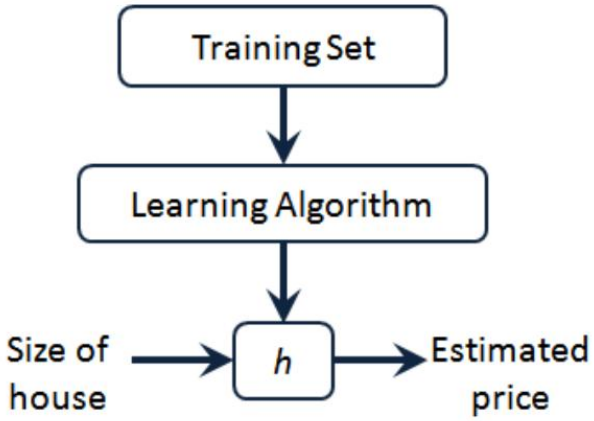
第二节：单变量线性回归

1. 模型的表示

方程表达式为 $h=\theta_0+\theta_1 x$ 。

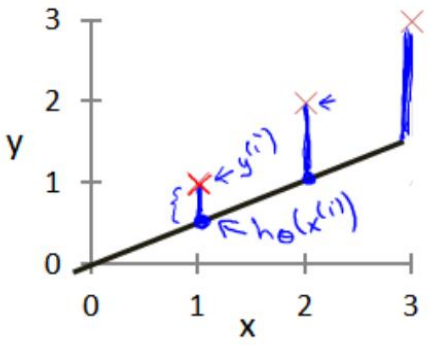
对于一个回归问题的训练集，我们描述为：

m 代表训练集中实例的数量；x 代表特征/输入变量；y 代表目标变量/输出变量；(x,y) 代表训练集中的实例；$\left(x^{(i)},y^{(i)}\right)$ 代表第 i 个观察实例；h 代表学习算法的解决方案或函数也称为假设。



2. 代价函数

为我们的模型选择合适的参数，我们选择的参数决定了我们得到的直线相对于我们的训练集的准确程度，模型所预测的值与训练集中实际值之间的差距就是建模误差。如下图所示：



我们的目标便是选择出可以使得建模误差的平方和能够最小的模型参数。即使得代价函数

$$J(\theta_0,\theta_1)=\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2 \text{ 最小。}$$

其中 $h_\theta\left(x^{(i)}\right)$ 是模型所预测的值， $y^{(i)}$ 是训练集中实际值。

代价函数也被称作平方误差函数，有时也被称为平方误差代价函数。

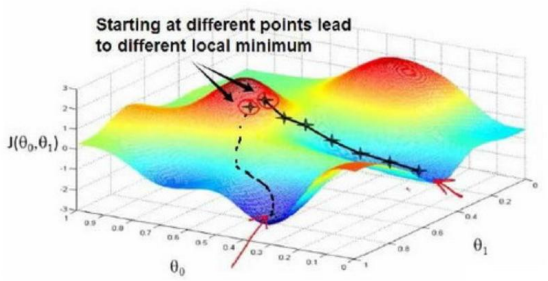
之所以要求出误差的平方和，是因为误差平方代价函数，对于大多数问题，特别是回归问题，都是一个合理的选择。还有其他的代价函数也能很好地发挥作用，但是平方误差代价函数可能是解决回归问题最常用的手段了。

3. 梯度下降

梯度下降是一个用来求函数最小值的算法，我们将使用梯度下降算法来求出代价函数$J(\theta_0,\theta_1)$的最小值。

梯度下降背后的思想是：开始时我们随机选择一个参数的组合$(\theta_0,\theta_1,\cdots\cdots,\theta_n)$，计算代价函数，然后我们寻找下一个能让代价函数值下降最多的参数组合。

我们持续这么做直到到到一个局部最小值，因为我们并没有尝试完所有的参数组合，所以不能确定我们得到的局部最小值是否便是全局最小值，选择不同的初始参数组合，可能会找到不同的局部最小值。



想象一下我们正站立在山的这一点上，站立在这座红色山上，在梯度下降算法中，我们要做的就是旋转 360 度，看看我们的周围，并问自己要在某个方向上，用小碎步尽快下山。

这些小碎步需要朝什么方向？

如果我们站在山坡上的这一点，我们看一下周围，会发现最佳的下山方向，你再看看周围，然后再一次想想，我应该从什么方向迈着小碎步下山？然后我们按照自己的判断又迈出一步，重复上面的步骤，从这个新的点，你环顾四周，并决定从什么方向将会最快下山，然后又迈进了一小步，并依此类推，直到你接近局部最低点的位置。

批量梯度下降算法的公式为：

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0,\theta_1)$$

$$(j=0,1)$$

其中α是学习率，它决定了我们沿着能让代价函数下降程度最大的方向向下迈出的步子有多大，在批量梯度下降中，我们每一次都同时让所有的参数减去学习速率乘以代价函数的导数。

$$temp0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$temp1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 = temp0$$
$$\theta_1 = temp1$$

梯度下降中，我们要更新θ_0和θ_1，当$j=0$和$j=1$时，会产生更新，所以将更新$J(\theta_0)$和$J(\theta_1)$。实现梯度下降算法的微妙之处是，在这个表达式中，如果你要更新这个等式，你需要同时更新θ_0和θ_1，我的意思是在这个等式中，我们要这样更新：$\theta_0 := \theta_0$，并更新$\theta_1 := \theta_1$。

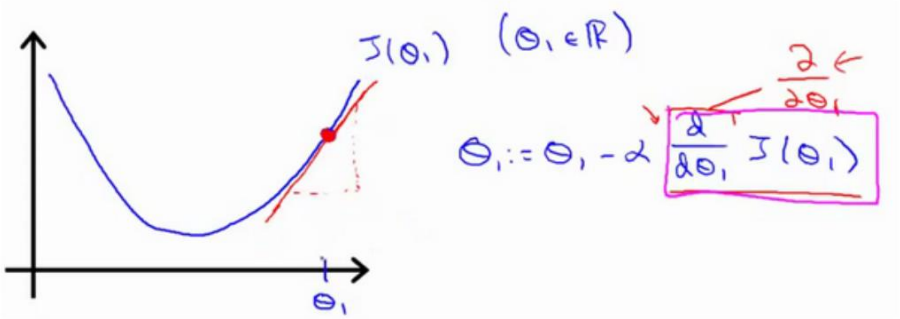
实现方法是：你应该计算公式右边的部分，通过那一部分计算出θ_0和θ_1的值，然后同时更新θ_0和θ_1。

4. 梯度下降的直观理解

梯度下降算法如下：

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

描述：对θ赋值，使得按梯度下降最快方向进行，一直迭代下去，最终得到局部最小值。其中α是学习率，它决定了我们沿着能让代价函数下降程度最大的方向向下迈出的步子有多大。



对于这个问题，求导的目的，基本上可以说取这个红点的切线，就是这样一条红色的直线，刚好与函数相切于这一点，让我们看看这条红色直线的斜率，就是这条刚好与函数曲线相切的这条直线，这条直线的斜率正好是这个三角形的高度除以这个水平长度，现在，这条线有一个正斜率，也就是说它有正导数，因此，我得到的新的θ_1，θ_1更新后等于θ_1减去一个正数乘以α。

这就是梯度下降法的更新规则：$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$。

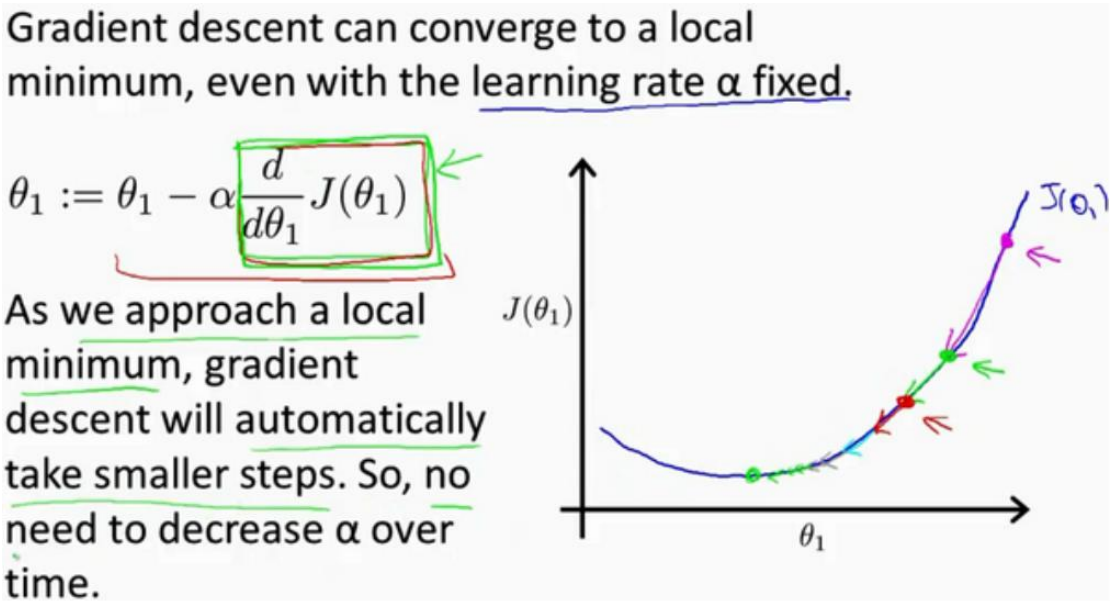
α的不同情况：

如果α太小了，即学习速率太小，这样就需要很多步才能到达最低点，所以如果太小的话，可能会很慢，因为它会一点点挪动，它会需要很多步才能到达全局最低点。

如果α太大，那么梯度下降法可能会越过最低点，甚至可能无法收敛，下一次迭代又移动了一大步，越过一次，又越过一次，一次次越过最低点，直到你发现实际上离最低点越来越远，所以，如果太大，它会导致无法收敛，甚至发散。

假设你将θ_1初始化在局部最低点，它已经在一个局部的最优处或局部最低点。结果是局部最优点的导数将等于零，因为它是那条切线的斜率。这意味着你已经在局部最优点，它使得θ_1不再改变，也就是新的等于原来的θ_1，因此，如果你的参数已经处于局部最低点，那么梯度下降法更新其实什么都没做，它不会改变参

数的值。这也解释了为什么即使学习速率保持不变时，梯度下降也可以收敛到局部最低点。



我想找到它的最小值，首先初始化我的梯度下降算法，在那个品红色的点初始化，如果我更新一步梯度下降，也许它会带我到这个点，因为这个点的导数是相当陡的。现在，在这个绿色的点，如果我再更新一步，你会发现我的导数，也即斜率，是没那么陡的。随着我接近最低点，我的导数越来越接近零，所以，梯度下降一步后，新的导数会变小一点点。然后我想再梯度下降一步，在这个绿点，我自然会用一个稍微跟刚才在那个品红点时比，再小一点的一步，到了新的红色点，更接近全局最低点了，因此这点的导数会比在绿点时更小。所以，我再进行一步梯度下降时，我的导数项是更小的，θ_1更新的幅度就会更小。所以随着梯度下降法的运行，你移动的幅度会自动变得越来越小，直到最终移动幅度非常小，你会发现，已经收敛到局部极小值。

在梯度下降法中，当我们接近局部最低点时，梯度下降法会自动采取更小的幅度，这是因为当我们接近局部最低点时，很显然在局部最低时导数等于零，所以当我们接近局部最低时，导数值会自动变得越来越小，所以梯度下降将自动采取较小的幅度，这就是梯度下降的做法。所以实际上没有必要再另外减小α。

5. 梯度下降的线性回归

梯度下降算法：

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$
$$(j = 0,1)$$

线性回归算法：

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

对我们之前的线性回归问题运用梯度下降法，关键在于求出代价函数的导数，即：

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$
$$j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$
$$j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)}) \bullet x^{(i)})$$

则算法改写成：

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x^{(i)}) - y^{(i)}) \bullet x^{(i)})$$

批量梯度下降法这个名字说明了我们需要考虑所有这一批训练样本，而事实上，有时也有其他类型的梯度下降法，不是这种"批量"型的，不考虑整个的训练集，而是每次只关注训练集中的一些小的子集。

第三节：线性代数回顾

此章节未作笔记

第四节：多变量线性回归

1. 多维特征

增添更多特征后，我们引入一系列新的注释：

n 代表特征的数量；$x^{(i)}$ 代表第 i 个训练实例，是特征矩阵中的第 i 行，是一个向量；$x_j^{(i)}$ 代表特征矩阵中第 i 行的第 j 个特征，也就是第 i 个训练实例的第 j 个特征。

支持多变量的假设 h 表示为：$h_\theta(x)=\theta_0+\theta_1x_1+\theta_2x_2+\cdots+\theta_nx_n$

这个公式中有 $n+1$ 个参数和 n 个变量，为了使得公式能够简化一些，引入 $x_0=1$，则公式转化为：$h_\theta(x)=\theta_0x_0+\theta_1x_1+\theta_2x_2+\cdots+\theta_nx_n$

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 $m*(n+1)$。 因此公式可以简化为：$h_\theta(x)=\theta^TX$。

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：

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

其中：$h_\theta(x)=\theta^TX=\theta_0+\theta_1x_1+\theta_2x_2+\cdots+\theta_nx_n$

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：

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

求导后得到：$\theta_j:=\theta_j-\alpha\frac{1}{m}\sum_{i=1}^{m}((h_\theta(x^{(i)})-y^{(i)})\cdot x_j^{(i)})$， $j=0,1,\cdots,n$

$$n \geq 1, \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

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

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

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。

3. 梯度下降法实践

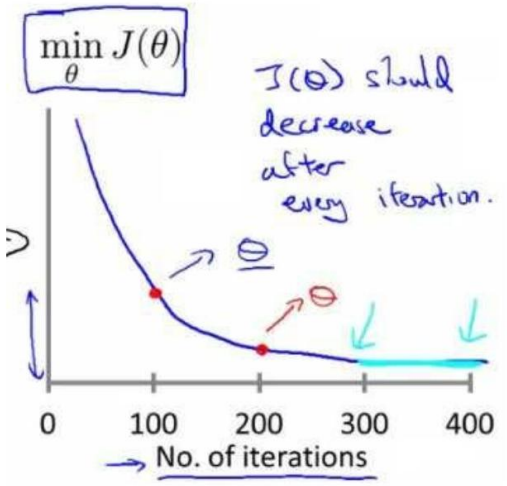
- **特征放缩**

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。

最简单的方法是令：$x_n = \frac{x_n - \mu_n}{s_n}$，其中 μ_n 是均值，s_n 是标准差。

- **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，如果学习率过小，则达到收敛所需的迭代次数会非常高；如果学习率过大，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。

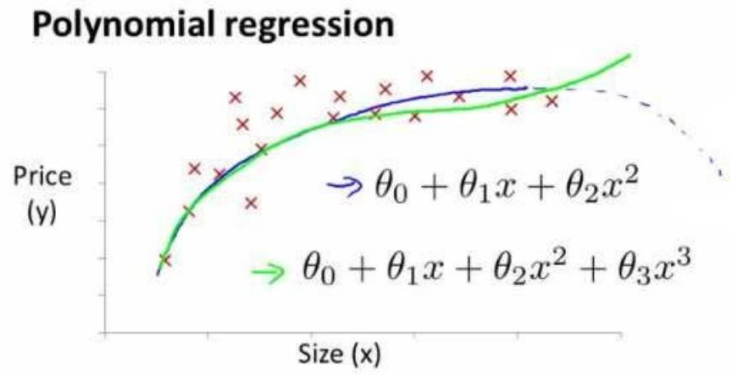
通常可以考虑尝试些学习率：$\alpha = 0.01, 0.03, 0.1, 0.3, 1, 3, 10$

4. 特征和多项式回归

如房价预测问题：

$$h_\theta(x) = \theta_0 + \theta_1 \times frontage + \theta_2 \times depth$$
$$x_1 = frontage, x_2 = depth, x = depth * frontage = area$$
$$h_\theta(x) = \theta_0 + \theta_1 x$$

线性回归并不适用于所有数据，有时我们需要曲线来适应我们的数据，比如一个二次方模型：$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2$ 或者三次方模型：$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3$。



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：$x_2 = x_2^2, x_3 = x_3^3$，从而将模型转化为线性回归模型。

根据函数图形特性，我们还可以使：$h_\theta(x)=\theta_0+\theta_1(size)+\theta_2(size)^2$ 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。

5. 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。

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 $\theta=(X^TX)^{-1}X^Ty$ 。设矩阵 $A=X^TX$，则：$(X^TX)^{-1}=A^{-1}$ 。

运用正规方程方法求解参数。

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α	不需要
需要多次迭代	一次运算得出
当特征数量n大时也能较好适用	需要计算$\left(X^TX\right)^{-1}$ 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$O\left(n^3\right)$，通常来说当n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。

增加内容：

$\theta=(X^TX)^{-1}X^Ty$ 的推导内容：

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

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（m 为样本个数，n 为特征个数），θ 为 n 行一列，y 为 m 行一列的矩阵。对 $J(\theta)$ 进行如下变换：

$$
\begin{aligned}
J(\theta)&=\frac{1}{2}(X\theta-y)^T(X\theta-y)\\
&=\frac{1}{2}(\theta^TX^T-y^T)(X\theta-y)\\
&=\frac{1}{2}(\theta^TX^TX\theta-\theta^TX^Ty-y^TX\theta-y^Ty)
\end{aligned}
$$

接下来对 $J(\theta)$ 偏导，需要用到以下几个矩阵的求导法则：

$$
\begin{aligned}
\frac{dAB}{dB}&=A^T\\
\frac{dX^TAX}{dX}&=2AX
\end{aligned}
$$

所以有：

$$
\begin{aligned}
\frac{\partial J(\theta)}{\partial\theta}&=\frac{1}{2}(2X^TX\theta-X^Ty-(y^TX)^T-0)\\
&=\frac{1}{2}(2X^TX\theta-X^Ty-X^Ty-0)\\
&=X^TX\theta-X^Ty
\end{aligned}
$$

令$\frac{\partial J(\theta)}{\partial \theta}=0$，则有$\theta=(X^TX)^{-1}X^Ty$