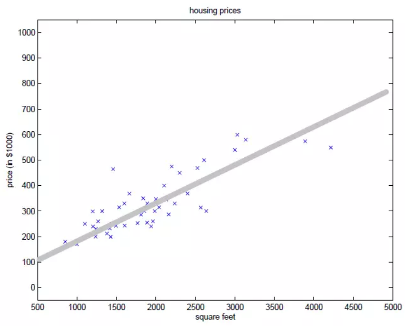
Lecture02

大纲（outline）
-Linear regression
-Batch/stochastic gradient descent
-Normal equation

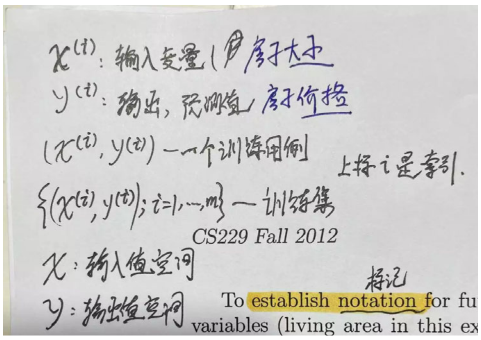
Supervised Learning

房价例子

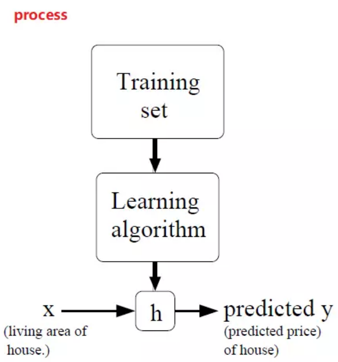
Living area ($feet^2$)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮



Notation



Process



训练集通过 学习算法 可以生成一个 假设H函数，通过这个函数，可以对 输入的房屋大小 预测出 房屋价格。

which we also know the number of bedrooms in each ho

Living area (feet^2) x_1	#bedrooms x_2	Price (1000$s) y
2104 $x_1^{(1)}$	3	400
1600	3	330
2400 $x_1^{(3)}$	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

how to represent the Hypothesis?

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

因为 假设h是由两个参数θ和x共同决定的所以写成左边那种形式，然后写成 y近似为x的线性函数，之后用θ对 x到y线性函数空间进行参数化。

$x_0 = 1$ (this is the **intercept term**), so that

化，规定 $x_0=1$　　截距

$$h(x)=\theta_0 x_0+\theta_1 x_1+\dots = \boxed{h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x} = [\theta_0\ \theta_1\ \theta_2]\cdot\begin{bmatrix}x_0\\x_1\\x_2\end{bmatrix} = \theta_0 x_0+\theta_1 x_1+\theta_2 x_2$$

把θ和x都看成"向量"

where on the right-hand side above we are viewing θ and x both as vectors, and here n is the number of input variables (not counting x_0).

"n"是输入变量的个数

$$\vec{\theta}=\begin{bmatrix}\theta_0\\\theta_1\\\theta_2\end{bmatrix} \quad \vec{x}=\begin{bmatrix}x_0\\x_1\\x_2\end{bmatrix}$$

x_1 → 房子面积

x_2 → 房间数目

x：input features，

　x1：房屋大小

　x2：房间数目

术语：

θ：参数

m：训练实例个数，表格中一行代表一个 training example

x：input/features，

y：output/target variable

(x,y)：training example

($x^{(i)}, y^{(i)}$)：第 i 个训练实例

例：

Living area (feet^2)	Price (1000$s)
2104 $x_1^{(1)}$	400
1600 $x_1^{(2)}$	330

n：features，特征个数，这里的特征个数为2（area、bedrooms）

how choose parameters?

选择参数，让房子预估值(hθ(x)) 离 真实价格(y)很近

定义一个函数 J(θ) 表示这个接近程度

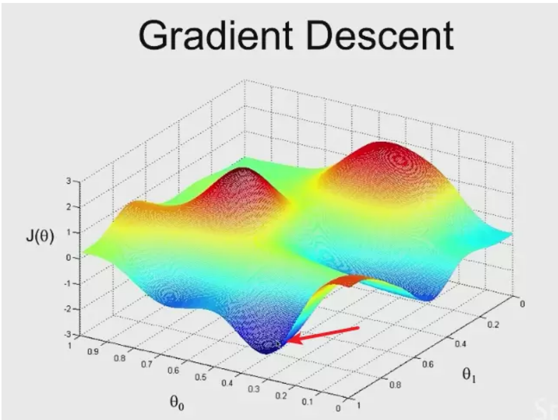
$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$

我们要做的事：找到参数θ使得 J(θ)最小，从而 房子预估值 与 真实值 就越接近

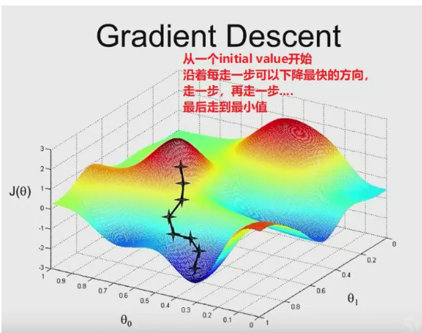
how to implement this algorithm?

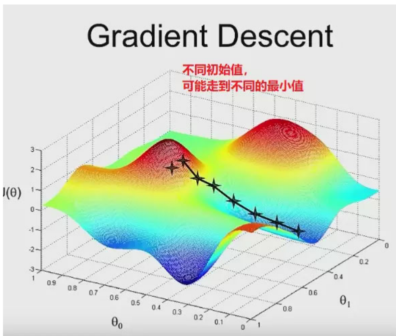
Gradient descent（梯度下降）

start with some initial θ,
keep changing θ to reduce J(θ)
解释：从一个初始值开始，然后重复修改θ值，以使减小 J(θ)，最终收敛到最小值
梯度可视化：



在这个例子中，θ有两个参数，就是找到 适合的 θ0和 θ1 使得 J(θ)最小，（这里箭头指向的这个值就是了）





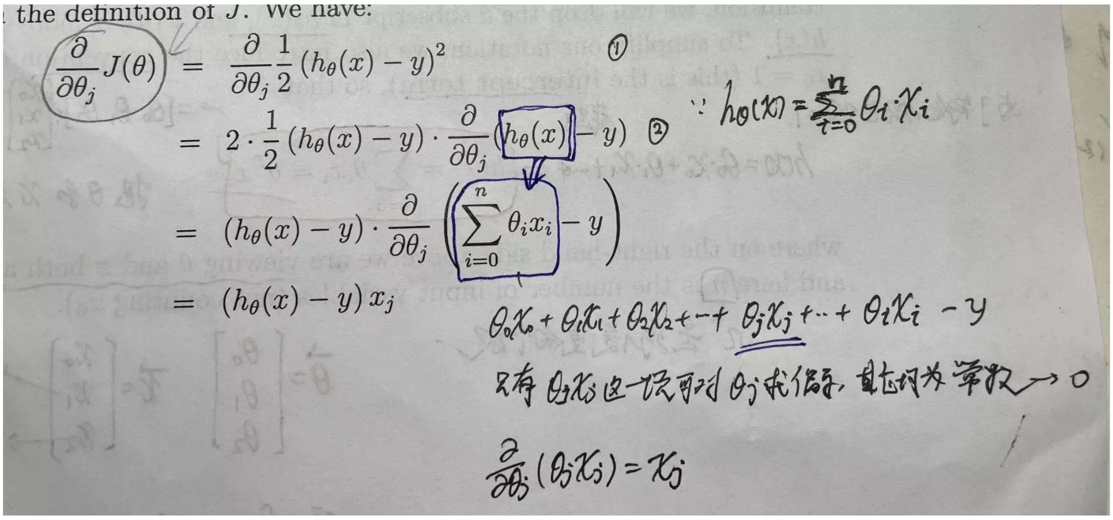
转一圈找到走一步就可以下降最大的方向

formalize the gradient descent algorithm

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

这里的 := 为赋值运算符(assign)，= 是判断运算符
每一步梯度下降，都要选取一个 **θj**，这里 **j** 取值为 0,1,2...n，（n为features个数）
α 为 learning rate，
后面是 J(θ) 对 θj 求偏导数

求偏导的过程


the definition of J. We have:

$$\frac{\partial}{\partial\theta_j}J(\theta) = \frac{\partial}{\partial\theta_j}\frac{1}{2}(h_\theta(x)-y)^2 \quad ①$$

$$= 2\cdot\frac{1}{2}(h_\theta(x)-y)\cdot\frac{\partial}{\partial\theta_j}(\boxed{h_\theta(x)}-y) \quad ② \qquad \because h_\theta(x)=\sum_{i=0}^{n}\theta_i x_i$$

$$= (h_\theta(x)-y)\cdot\frac{\partial}{\partial\theta_j}\left(\boxed{\sum_{i=0}^{n}\theta_i x_i}-y\right)$$

$$= (h_\theta(x)-y)\,x_j$$

$\theta_0x_0+\theta_1x_1+\theta_2x_2+\cdots+\underline{\theta_jx_j}+\cdots+\theta_ix_i-y$

对θ_jx_j这一项求对θ_j求偏导，其他为常数→0

$\frac{\partial}{\partial\theta_j}(\theta_jx_j)=x_j$

综上，表达式如下

1个training example $\theta_j := \theta_j - \alpha(h_\theta(x)-y)x_j$

m个 training examples $\theta_j := \theta_j - \alpha\sum_{i=0}^{m}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)}$

$x^{(i)}$ 是第 i 个 训练实例的 input features

$y^{(i)}$ 是第 i 个 训练实例的 target label

Batch Gradient Descent

batch的含义：

例如房价中有49个training example，则就可以把这49个例子看做成一个 batch of data

Repeat until convergence{

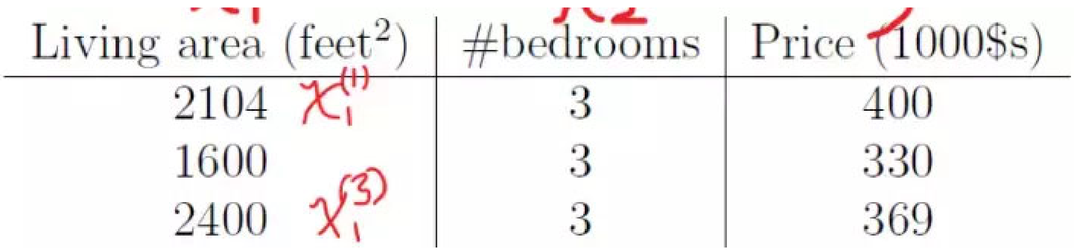
$$\theta_j := \theta_j - \alpha\sum_{i=0}^{m}(h_\theta(x^{(i)})-y^{(i)})x_j^{(i)} \quad \text{(j=0,1,2...n \quad n是features)}$$

}

n：features，n=2，(房子大小、房间个数)

m：number of training examples，训练实例个数

用房价例子举例，n=2（area、bedrooms），m=3, 3个训练用例



Living area (feet^2) x_1	#bedrooms x_2	Price (1000$s) y
2104 $x_1^{(1)}$	3	400
1600	3	330
2400 $x_1^{(3)}$	3	369

对于参数 θ0：

Repeat until convergence{

$$\theta_0 := \theta_0 - \alpha\sum_{i=0}^{m}(h_\theta(x^{(i)})-y^{(i)})x_0^{(i)}$$

$$:= \theta_0 - \alpha((h_\theta(x^{(1)})-y^{(1)})x_0^{(1)} + (h_\theta + (x^{(2)})-y^{(2)})x_0^{(2)} + (h_\theta(x^{(3)})-y^{(3)})x_0^{(3)})$$

}

对于参数 θ1：

Repeat until convergence{

$$\theta_1 := \theta_1 - \alpha \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_1^{(i)}$$

}

对于参数 θ2:

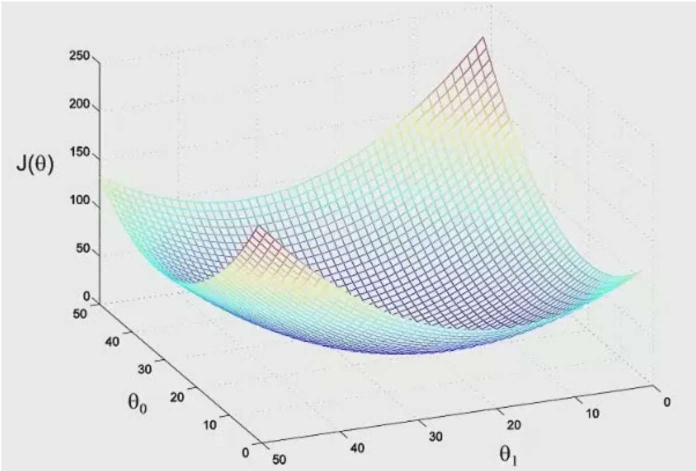
Repeat until convergence{

$$\theta_2 := \theta_2 - \alpha \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_2^{(i)}$$

}

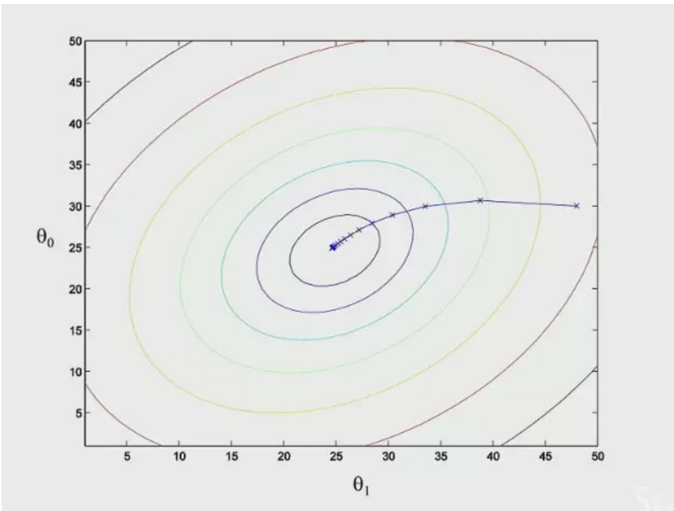
如果把 θj 定义为 若干平方项求和，那么它就是一个二次函数，

那么 θj 就是下面这个形式:



θj 没有局部最优，局部最优就是全局最优

看这个函数的另一种方法是看 轮廓线，对上面这个大碗进行水平分层，



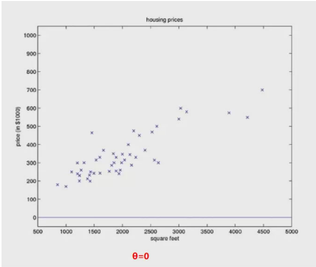
the choice of the learning rate α

设置的很大，跨的步很大，可能会错过最小的地方;

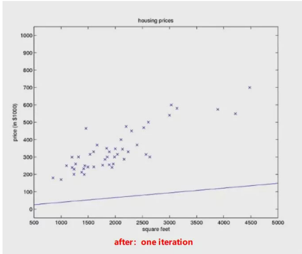
设置的很小，需要迭代很多次，算法会很慢。

设置经验:

随机尝试几个值，看如果哪个值可以最有效的推动 θj 的下降，如果你看到 θj在增加而不是减少，就说明你设置的α太大了。Ng的做法是在指数尺度上面尝试。

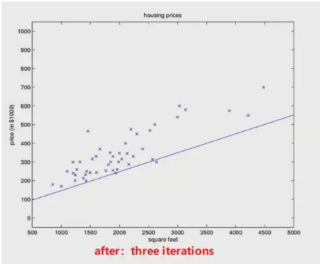


θ=0

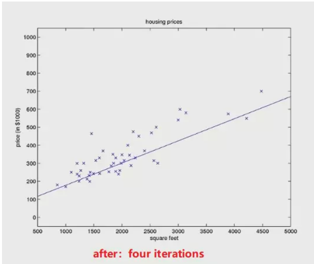


after: one iteration

after: two iterations



after: three iterations



after: four iterations

BGD的缺点:
当数据集很大的时候，更新一次 θ，就要对整个数据集进行求和一次

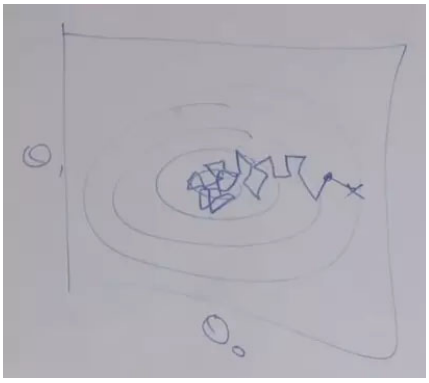
Repeat until convergence{

$$\theta_j := \theta_j - \alpha \boxed{\sum_{i=0}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}}$$ (j=0,1,2...n　n是features)

}　**当数据集很大，即m很大的时候，更新一次 θj，就要求和一次m个数据！**

Stochastic Gradient Descent（随机梯度下降）

Repeat {
　　for i=1 to m, {
　　　　$\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$　(for every j)
　　}
}



在某个地方初始化参数之后，看第一个房子，修改参数提高预测当前房子价格的准确性，只对当前这个房子的数据进行了拟合，虽然改善了参数，但是并没有朝最直接的方向走下去。每次改参数都是为了更好的拟合当前那个房子，若干次之后，它在最小值附近震荡，但不会收敛于最小值。
数据集很大时，通常使用SGD（随机梯度下降算法）

Q：可以从SGD切换到BGD吗？
A：可以，但是当数据集很大时，BGD太慢了，而且SGD生成的参数性能也不错。

数据集很大时，用SGD；
数据集很小时，用BGD，可以得到全局最优，而不用振荡。

如果使用的算法是线性回归，有一种方法可以直接解出参数theta的最佳值，直接一步跳到全局最优，而不需要使用迭代算法。

2 The Normal equations

Matrix derivatives（矩阵求导）

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \cdots & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}$$

例：

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

$$f(A) = \frac{3}{2}A_{11} + 5A_{12}^2 + A_{21}A_{22}$$

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \frac{\partial f}{\partial A_{12}} \\ \frac{\partial f}{\partial A_{21}} & \frac{\partial f}{\partial A_{22}} \end{bmatrix}$$

$$\begin{cases} \frac{\partial f}{\partial A_{11}} = \frac{\partial}{\partial A_{11}}\left(\frac{3}{2}A_{11} + 5A_{12}^2 + A_{21}A_{22}\right) = \frac{3}{2} \\ \frac{\partial f(A)}{\partial A_{12}} = \frac{\partial}{\partial A_{12}}\left(\frac{3}{2}A_{11} + 5A_{12}^2 + A_{21}A_{22}\right) = 10A_{12} \\ \frac{\partial f(A)}{\partial A_{21}} = \frac{\partial}{\partial A_{21}}\left(\frac{3}{2}A_{11} + 5A_{12}^2 + A_{21}A_{22}\right) = A_{22} \\ \frac{\partial f(A)}{\partial A_{22}} = \frac{\partial}{\partial A_{22}}\left(\frac{3}{2}A_{11} + 5A_{12}^2 + A_{21}A_{22}\right) = A_{21} \end{cases}$$

$$= \begin{bmatrix} \frac{3}{2} & 10A_{12} \\ A_{22} & A_{21} \end{bmatrix}$$

trace operator（矩阵的迹）

$$trA = \sum_{i=1}^{n} A_{ii}$$

运算法则

$trA = trA^T$

$trAB = trBA$

$trABC = trCAB = trBCA$

$\nabla_A trAB = B^T$

$$A=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \qquad B=\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix}$$

$$AB=\begin{bmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} & a_{11}b_{13}+a_{12}b_{23} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} & a_{21}b_{13}+a_{22}b_{23} \\ a_{31}b_{11}+a_{32}b_{21} & a_{31}b_{12}+a_{32}b_{22} & a_{31}b_{13}+a_{32}b_{23} \end{bmatrix}$$

$$trAB = a_{11}b_{11}+a_{12}b_{21} + a_{21}b_{12}+a_{22}b_{22} + a_{31}b_{13}+a_{32}b_{23}$$

$$\frac{\partial trAB}{\partial a_{11}} = b_{11} \qquad \frac{\partial trAB}{\partial a_{21}} = b_{12} \qquad \frac{\partial trAB}{\partial a_{31}} = b_{13}$$

$$\frac{\partial trAB}{\partial a_{12}} = b_{21}, \qquad \frac{\partial trAB}{\partial a_{22}} = b_{22} \qquad \frac{\partial trAB}{\partial a_{32}} = b_{23}$$

$$\Rightarrow \nabla_A trAB = \begin{bmatrix} \frac{\partial trAB}{\partial a_{11}} & \frac{\partial trAB}{\partial a_{12}} \\ \frac{\partial trAB}{\partial a_{21}} & \frac{\partial trAB}{\partial a_{22}} \\ \frac{\partial trAB}{\partial a_{31}} & \frac{\partial trAB}{\partial a_{32}} \end{bmatrix} = \begin{bmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \\ b_{13} & b_{23} \end{bmatrix} = B^T$$

$$\nabla_A trAA^TC = CA + C^TA$$

矩阵向量的形式重写 J(θ)

m：训练实例的个数

design X

x：input values

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — (x^{(m)})^T — \end{bmatrix}.$$

y：target values

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}.$$

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

由

$$h_\theta(x^{(i)}) = (x^{(i)})^T \cdot \theta$$

$$X\theta = \begin{bmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \cdot \theta = \begin{bmatrix} (x^{(1)})^T \cdot \theta \\ (x^{(2)})^T \cdot \theta \\ \vdots \\ (x^{(m)})^T \cdot \theta \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) \\ h_\theta(x^{(2)}) \\ \vdots \\ h_\theta(x^{(m)}) \end{bmatrix}$$

$$X\theta - \vec{y} = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}$$

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

由 $a^T a = a_1^2 + a_2^2 + \cdots + a_n^2 = \sum_{i=1}^{n} a_i^2$

有 $= \frac{1}{2} (X\theta - \vec{y})^T \cdot (X\theta - \vec{y})$

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \frac{1}{2} (X\theta - \vec{y})^T \cdot (X\theta - \vec{y}) \\
&= \frac{1}{2} \cdot \nabla_\theta (X\theta - \vec{y})^T \cdot (X\theta - \vec{y}) \\
&\quad \text{拆开} \\
&= \frac{1}{2} \cdot \nabla_\theta \left[(X\theta)^T \cdot X\theta - (X\theta)^T \cdot \vec{y} - \vec{y}^T \cdot X\theta + \vec{y}^T \vec{y} \right] \\
&= \frac{1}{2} \cdot \nabla_\theta \left[\theta^T X^T \cdot X\theta - \theta^T X^T \cdot \vec{y} - \vec{y}^T \cdot X \cdot \theta + \vec{y}^T \vec{y} \right] \\
&\quad \mathrm{tr}\, a = a \\
&= \frac{1}{2} \nabla_\theta \mathrm{tr} \left(\theta^T X^T X\theta - \theta^T X^T \cdot \vec{y} - \vec{y}^T \cdot X \cdot \theta + \vec{y}^T \vec{y} \right) \\
&\quad \text{tr代回去} \quad \mathrm{tr}(A+B) = \mathrm{tr} A + \mathrm{tr} B \\
&= \frac{1}{2} \nabla_\theta \left[\mathrm{tr}\, \theta^T X^T X\theta - \underbrace{\mathrm{tr}\, \theta^T X^T \vec{y} - \mathrm{tr}\, \vec{y} X\theta}_{\mathrm{tr} A^T = \mathrm{tr} A} + \mathrm{tr}\, \vec{y}^T \vec{y} \right] \\
&= \frac{1}{2} \nabla_\theta \left[\mathrm{tr}\, \theta^T X^T X\theta - \underline{2\, \mathrm{tr}\, \vec{y}^T X\theta} + \vec{y}^T \vec{y} \right] \\
&= \frac{1}{2} \nabla_\theta \mathrm{tr}\, \theta^T X^T X\theta - \nabla_\theta \mathrm{tr}\, \vec{y}^T X\theta + 0 \\
&= \frac{1}{2} (X^T X\theta + X^T X\theta) - X^T \vec{y} \\
&= X^T X\theta - X^T \vec{y}
\end{aligned}$$

为了求最小值，让导数等于0，

即： $X^T X\theta = X^T \vec{y}$

则求得： $\theta = (X^T X)^{-1} X^T \vec{y}$