

1. 矩阵和向量。矩阵加法、标量乘法、矩阵乘法。

用矩阵运算简化编码，提升运行速度。如：预测函数：

单个预测函数计算：

House sizes:

$$\begin{matrix} \rightarrow 2104 \\ \rightarrow 1416 \\ \rightarrow 1534 \\ \rightarrow 852 \end{matrix}$$

Matrix  $4 \times 2$

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix}$$

Vector  $2 \times 1$

$$\begin{bmatrix} -40 \\ 0.25 \end{bmatrix}$$

Equation:

$$h_{\theta}(x) = -40 + 0.25x$$

Resulting  $4 \times 1$  matrix:

$$\begin{bmatrix} -40 \times 1 + 0.25 \times 2104 \\ -40 \times 1 + 0.25 \times 1416 \\ \dots \\ \dots \end{bmatrix}$$

for  $i = 1:4$ ,  
prediction(i) = ...

prediction = Data Matrix  $\times$  Parameters.

当你有

多个预测函数同时计算：

House sizes:

$$\begin{bmatrix} 2104 \\ 1416 \\ 1534 \\ 852 \end{bmatrix}$$

Matrix  $4 \times 2$

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix}$$

Have 3 competing hypotheses:

- $h_{\theta}(x) = -40 + 0.25x$
- $h_{\theta}(x) = 200 + 0.1x$
- $h_{\theta}(x) = -150 + 0.4x$

Matrix  $4 \times 3$

$$\begin{bmatrix} -40 & 200 & -150 \\ 0.25 & 0.1 & 0.4 \end{bmatrix}$$

Equation:

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \times \begin{bmatrix} -40 & 200 & -150 \\ 0.25 & 0.1 & 0.4 \end{bmatrix} = \begin{bmatrix} 486 & 410 & 692 \\ 314 & 342 & 416 \\ 344 & 353 & 464 \\ 173 & 285 & 191 \end{bmatrix}$$

Prediction

为你实现矩阵乘法

Predictions of 2nd  $h_{\theta}$

Andrew Ng

2. 矩阵相乘的性质：（1）不满足交换率 （2）满足结合率 （3）单位矩阵  $AI=IA=A$  （4）矩阵的逆 （5）矩阵转置

3. 多变量：n表示特征/变量数量。

## Multiple features (variables).

Size (feet <sup>2</sup> ) $x_1$	Number of bedrooms $x_2$	Number of floors $x_3$	Age of home (years) $x_4$	Price (\$1000) $y$
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

Notation:

- $n$  = number of features  $n=4$
- $x^{(i)}$  = input (features) of  $i^{th}$  training example.
- $x_j^{(i)}$  = value of feature  $j$  in  $i^{th}$  training example.

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

$$x_3^{(2)} = 2$$

假设的形式改变：

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

为了记号方便起见，定义  $x_0 = 1$ 。有：

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

For convenience of notation, define  $x_0 = 1$ . ( $x_0^{(i)} = 1$ )

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$h_{\theta}(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$= \theta^T x$$

$\theta^T$  is a  $(n+1) \times 1$  matrix.

4. 多元模型中的梯度下降方法：可以将theta写成向量形式。

Hypothesis:  $h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n$  ↗  $x_0 = 1$

Parameters:  $\theta_0, \theta_1, \dots, \theta_n$  ⓪ n+1-dimensional vector

Cost function:

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

Gradient descent:

Repeat {

$$\rightarrow \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} \underbrace{J(\theta_0, \dots, \theta_n)}_{J(\theta)}$$

}

就是代价函数的对参数  $\theta_j$  的偏导数 every  $j = 0, \dots, n$

对  $\theta_j$  求偏导数，会剩余  $x_j$ ，于是有：

## Gradient Descent

Previously ( $n=1$ ):

Repeat {

$$\rightarrow \theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial \theta_0} J(\theta)}$$

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

(simultaneously update  $\theta_0, \theta_1$ )

}

New algorithm ( $n \geq 1$ ):

Repeat {

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

(simultaneously update  $\theta_j$  for  $j = 0, \dots, n$ )

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

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

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

...

n=1的情况相同

5. 特征缩放 (Features Scaling)：确保不同特征在相似的范围之内，从而加快代价函数收敛速度：

## Feature Scaling

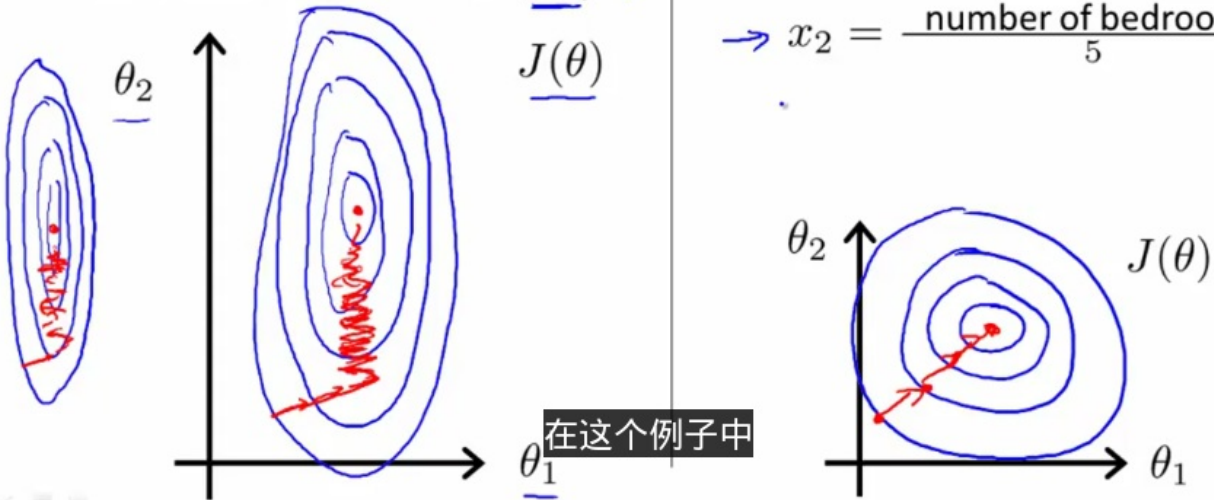
Idea: Make sure features are on a similar scale.

E.g.  $x_1 = \text{size (0-2000 feet}^2\text{)}$  ←

$x_2 = \text{number of bedrooms (1-5)}$  ←

$$\rightarrow x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

$$\rightarrow x_2 = \frac{\text{number of bedrooms}}{5}$$



Andrew Ng

通常情况下，将特征都缩放在大约 $[-1, 1]$ 范围之内。这里不管太大或者太小都不合适，绝对值在接近1的适当范围是可以接受的。

## Feature Scaling

Get every feature into approximately a  $-1 \leq x_i \leq 1$  range.

$$x_0 = 1$$

$$0 \leq x_1 \leq 3 \quad \checkmark$$

$$-2 \leq x_2 \leq 0.5 \quad \checkmark$$

$$-100 \leq x_3 \leq 100 \quad \times$$

$$-0.0001 \leq x_4 \leq 0.0001 \quad \checkmark$$

$$-3 \text{ to } 3 \quad \checkmark$$

$$-\frac{1}{3} \text{ to } \frac{1}{3} \quad \checkmark$$

只要它们足够接近的话 梯度下降法就会正常地工作

Andrew Ng

均值归一化 (Mean normalization) : 用  $x_i - \mu_i$  来代替  $x_i$  从而保证特征均值大约为0. ( $\mu_i$  是数据集  $i$  的均值, 注意不要运用到  $x_0=1$  上)。



## Mean normalization

Replace  $x_i$  with  $x_i - \mu_i$  to make features have approximately zero mean (Do not apply to  $x_0 = 1$ ).

E.g.  $x_1 = \frac{\text{size} - 1000}{2000}$       Average size = 1000

$$x_2 = \frac{\# \text{bedrooms} - 2}{5}$$

1-5 bedrooms

$$-0.5 \leq x_1 \leq 0.5, -0.5 \leq x_2 \leq 0.5$$

$$x_1 \leftarrow \frac{x_1 - \mu_1}{S_1} \quad \left| \quad x_2 \leftarrow \frac{x_2 - \mu_2}{S_2}$$

← avg value of  $x_1$  in training set  
← range (max-min) (or standard deviation)

Andrew Ng

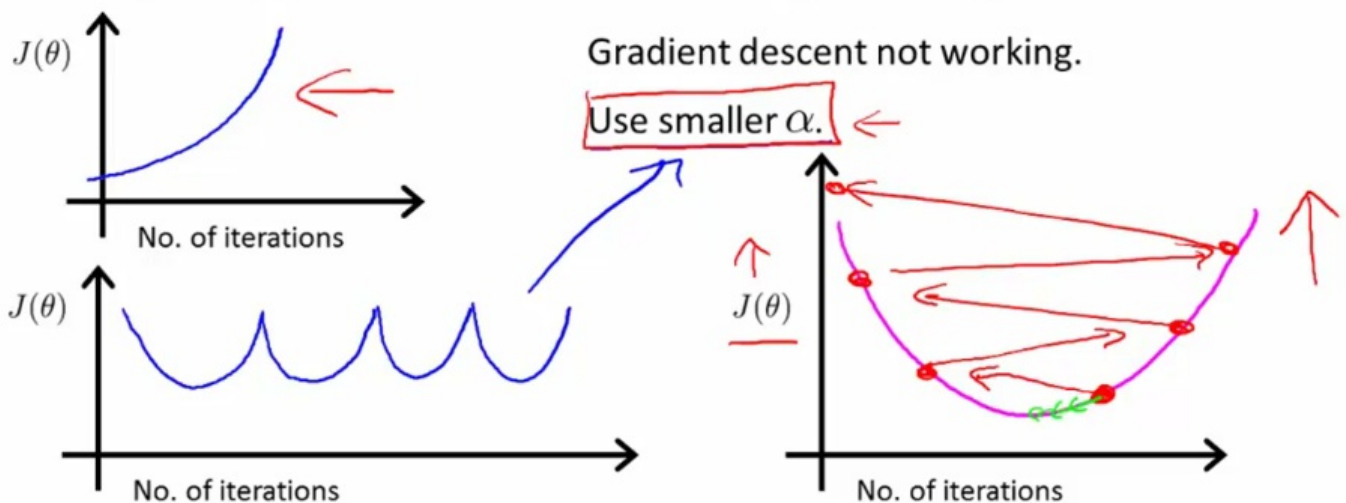
其中S不必是标准差，范围即可（即最大-最小）。

### 6. 学习速率。

首先要确保梯度下降算法正确工作，也就是说在每一次迭代中代价函数J都必须下降，在达到某一步代价函数不再变化，已经接近收敛。

如果出现问题，需要调整学习速率：

## Making sure gradient descent is working correctly.



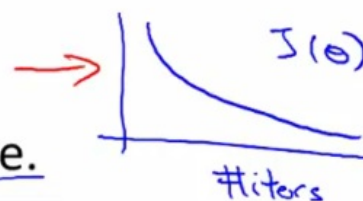
- For sufficiently small  $\alpha$ ,  $J(\theta)$  should decrease on every iteration. ←
- But if  $\alpha$  is too small, gradient descent may take a long time to converge.

总结：

选择学习速率，... 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, ....

## Summary:

- If  $\alpha$  is too small: slow convergence.
- If  $\alpha$  is too large:  $J(\theta)$  may not decrease on every iteration; may not converge. (Slow converge also possible.)



To choose  $\alpha$ , try

..., 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, ...

而当我做了以上工作时

Andrew Ng

## 7. 特征和多项式回归。

选择合适的特征，比如：有两个特征，长度和宽度，有时候我们可以选择他们的乘积，即面积。有时候定义新的特征可能会得到更好的模型。

## Housing prices prediction

$$h_{\theta}(x) = \theta_0 + \theta_1 \times \underbrace{\text{frontage}}_{x_1} + \theta_2 \times \underbrace{\text{depth}}_{x_2}$$

Area

$$x = \text{frontage} * \text{depth}$$

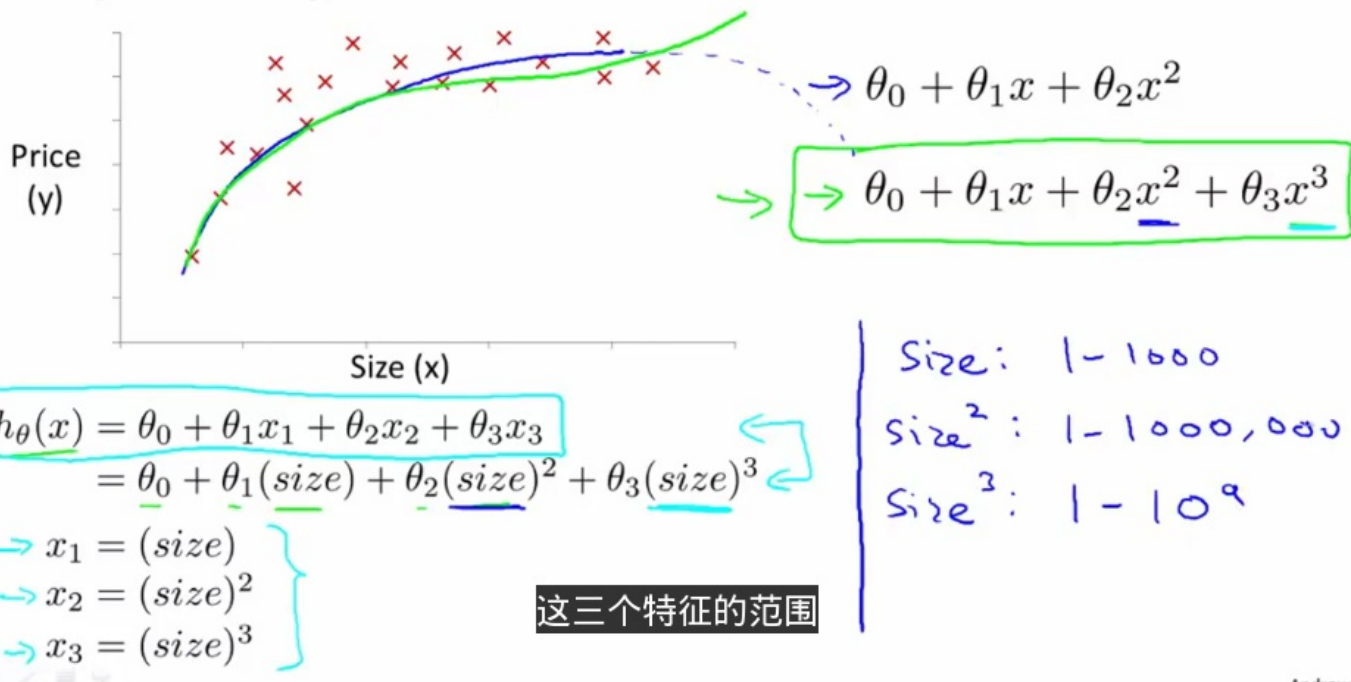
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

land area



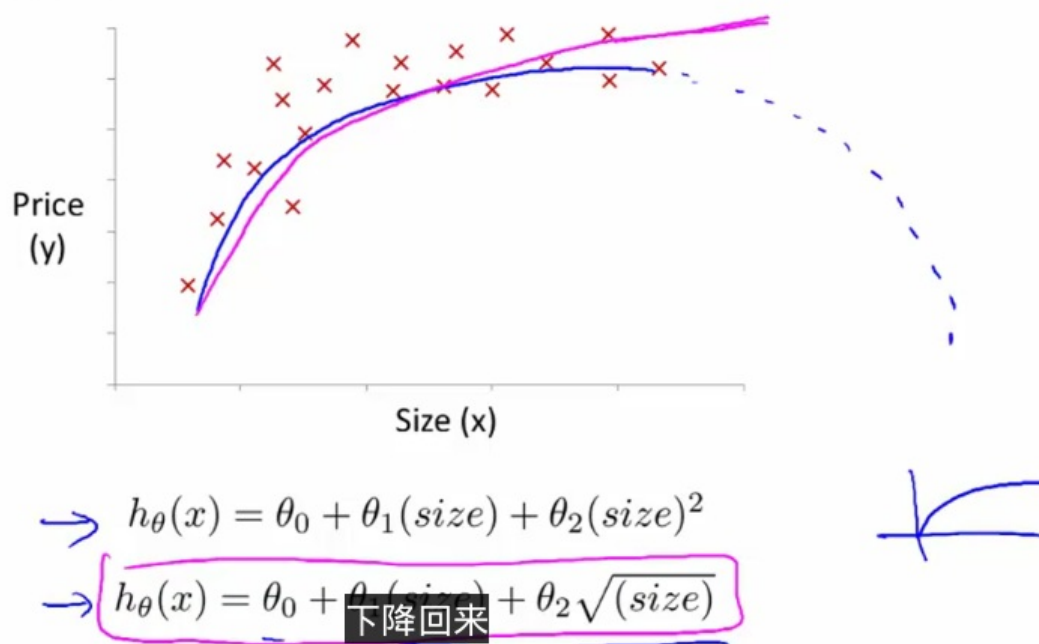
多项式回归：(Polynomial Regression) 数据关系不是线性。但是可以通过线性回归的简单变换拟合，变换 $x$ 。但是要注意特征缩放，缩放到相似的范围。

## Polynomial regression



最后，注意模型的选择，以便对数据更好的拟合。（为防止二次函数下降，不用三次函数，用根号）

## Choice of features



但是，这么多模型要怎么选择呢？

9. 正规方程（Normal Equation）：对于某些情况求解theta更快。不是用迭代，而是用解析方法直接求解出theta的值。

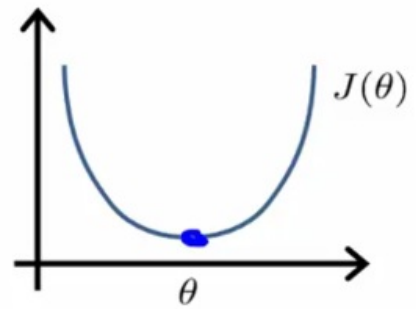
代价函数对每个theta求偏导数，令偏导数为0，然后解出所有的theta值，便得到最小代价函数的theta。

Intuition: If 1D ( $\theta \in \mathbb{R}$ )

$\rightarrow J(\theta) = a\theta^2 + b\theta + c$

$\frac{\partial}{\partial \theta} J(\theta) = \dots \stackrel{\text{set}}{=} 0$

Solve for  $\theta$



$\theta \in \mathbb{R}^{n+1} \quad J(\theta_0, \theta_1, \dots, \theta_m) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$

$\frac{\partial}{\partial \theta_j} J(\theta) = \dots \stackrel{\text{set}}{=} 0 \quad (\text{for every } j)$

Solve for  $\theta_0, \theta_1, \dots, \theta_n$  这样就能得到能够最小化代价函数J的θ值

向量方法：

Examples:  $m = 4$ .

	Size (feet <sup>2</sup> )	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

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

$m \times (n+1)$

$$\underline{y} = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$m$ -dimensional vector

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

X和y的构建如下：



$m$  examples  $(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})$  ;  $n$  features.

$$\underline{x^{(i)}} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1} \quad \bigg| \quad X = \begin{bmatrix} \text{---} (x^{(1)})^T \text{---} \\ \text{---} (x^{(2)})^T \text{---} \\ \vdots \\ \text{---} (x^{(m)})^T \text{---} \end{bmatrix}$$

(design matrix)

E.g. If  $\underline{x^{(i)}} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$   $\rightarrow$   $X = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_1^{(2)} \\ \vdots & \vdots \\ 1 & x_1^{(m)} \end{bmatrix}$   $y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$

$m \times (n+1)$   $m \times 2$   $m \times 1$

我们就可以通过计算  $X^T X$  乘以  $X^T y$  来得到  $\theta$

Andrew Ng

从而得到了最小二乘矩阵形式：也即正规方程：

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

$(X^T X)^{-1}$  is inverse of matrix  $X^T X$ .

Set  $A = X^T X$

$$(X^T X)^{-1} = A^{-1}$$

Octave: `pinv(X' * X) * X' * y`

$$\text{pinv}(X^T * X) * X^T * y$$

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

但如果你使用梯度下降法

$X'$   $X^T$

~~Feature Scaling~~

$0 \leq x_1 \leq 1$

$0 \leq x_2 \leq 1000$

$0 \leq x_3 \leq 10^{-5}$  ✓

同时注意，正规方程方法不需要进行特征缩放。

10. 梯度下降和特征方程方法的比较：

$m$  training examples,  $n$  features.

### Gradient Descent

- • Need to choose  $\alpha$ .
- • Needs many iterations.
- Works well even when  $n$  is large.

↗  
 $n = 10^6$

### Normal Equation

- • No need to choose  $\alpha$ .
- • Don't need to iterate.
- Need to compute
- •  $(X^T X)^{-1}$   $\frac{n \times n}{O(n^3)}$
- Slow if  $n$  is very large.

$n = 100$   
 $n = 1000$

在一万左右 我会开始考虑换成梯度下降法

Andrew Ng

正规方程公式： $\theta = \text{inv}(X^T X) * X^T y$ , 即最小二乘法。