17.4.13（多元梯度，特征缩放，学习率调整）

1. 多元梯度，特征缩放，学习率调整

2. 多元线性回归的假设函数：h(X)=theta'X

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

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

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n \quad (x_0 = 1)$$

$$= \theta^T x$$

θ^T: $(n+1) \times 1$ matrix

Multivariate linear regression.

2.多元线性回归的梯度下降：Gradient Descent

Gradient Descent

Previously (n=1):

Repeat {

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

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

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

(simultaneously update θ_0, θ_1)

}

New algorithm $(n \geq 1)$:

Repeat {

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

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

(simultaneously update θ_j for $j = 0, \ldots, n$)

}

$x_0^{(i)} = 1$

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

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

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

...

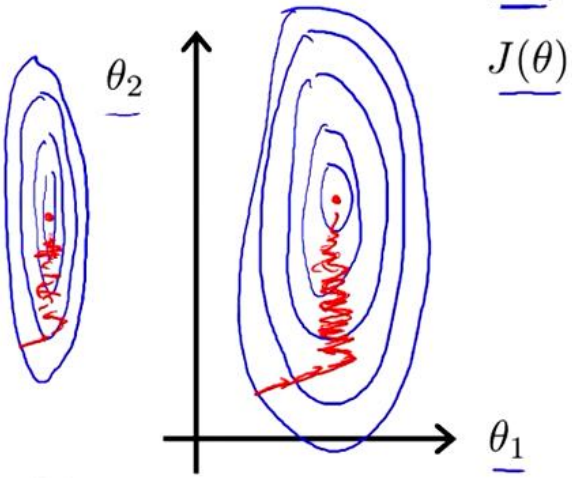
3.X的Feature Scaling（特征缩放，消耗掉不同特征的范围，最后得到的两个feature都在类似于-1~1这样的相近的范围之间，这样可以让你的梯度下降算法更快的收敛）：

Feature Scaling

Idea: Make sure features are on a similar scale.

E.g. x_1 = size (0-2000 feet^2)

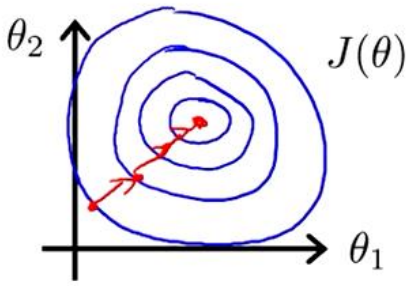
x_2 = number of bedrooms (1-5)



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

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

$0 \le x_1 \le 1$ $0 \le x_2 \le 1$



Feature Normalization（特征标准化）数学公式里（：=）是赋值，（=）是相等

求出新的特征值替换原来的，其中mu是所有特征的平均值，S是最大值减去最小值（也可以是标准差，sigma）的一个范围，这样求出新的特征值就大概在一个很小的范围内了，也就能让你的梯度下降算法更快的收敛，就是特征缩放的方法。

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. $x_1 = \frac{size - 1000}{2000}$ Average size = 100

$x_2 = \frac{\#bedrooms - 2}{5}$ 1-5 bedrooms

$-0.5 \le x_1 \le 0.5$ $-0.5 \le x_2 \le 0.5$

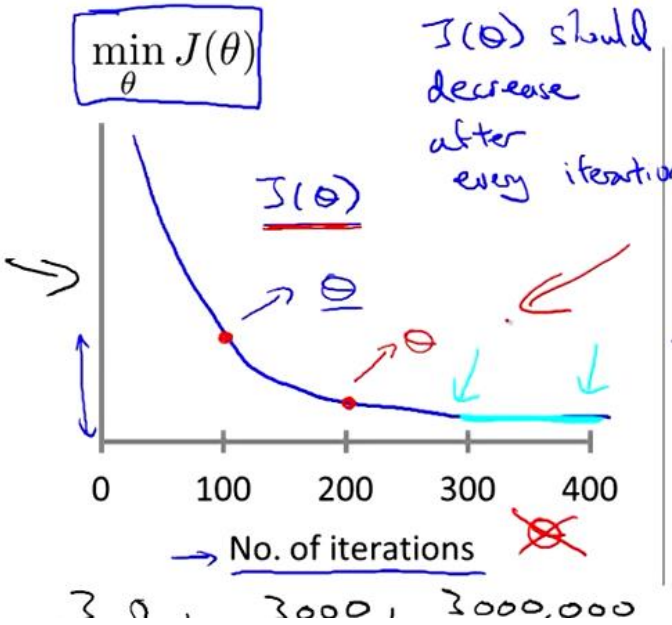
$x_1 \leftarrow \frac{x_1 - \mu_1}{s_1}$ (μ_1: avg value of x_1 in training set; s_1: range (max - min) (or standard deviation))

$x_2 \leftarrow \frac{x_2 - \mu_1}{s_2}$

Andrew Ng

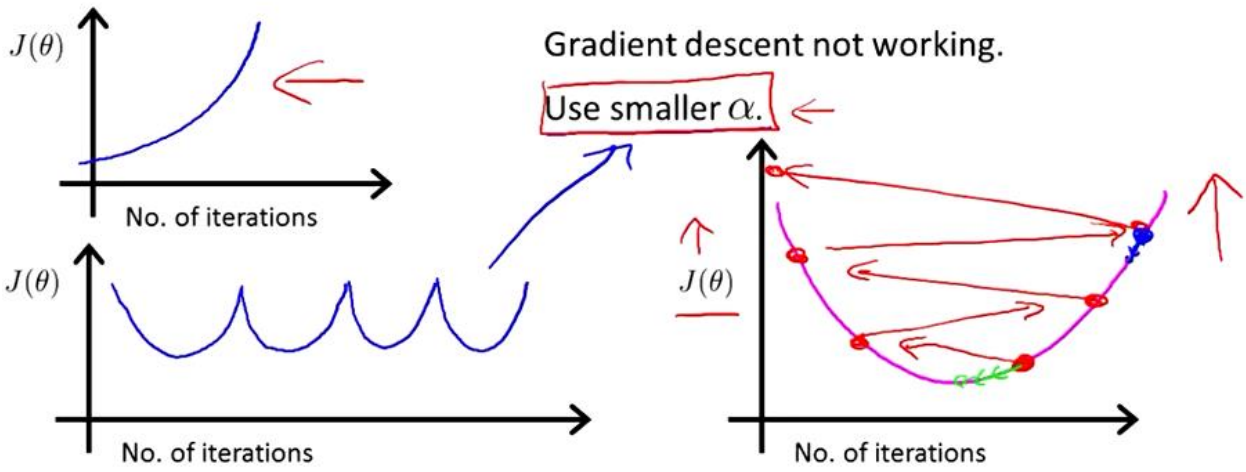
"Debugging" 调试看你的梯度下降算法有没有正常工作。

Making sure gradient descent is working correctly.



这里的X轴是指梯度下降算法迭代的次数，Y轴就是计算的cost function。我们可以通过这条曲线进行判断你的梯度下降算法的工作效率。

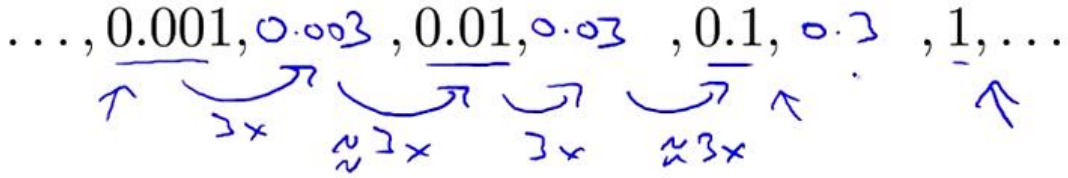
Making sure gradient descent is working correctly.



- For sufficiently small α, $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

上图中左边两张图是因为学习率过大，随着迭代次数，cost function并没有持续减小。可以试着减小学习率，来调整你的梯度下降算法。

To choose α, try



所以选择学习率可以在上面这些数里to try.

MATLAB中让一个小矩阵变成重复的大矩阵：

```
>>B=repmat( [1 2;3 4],2,3)
B =

1     2     1     2     1     2

3     4     3     4     3     4

1     2     1     2     1     2

3     4     3     4     3     4
```

如果X是一个矩阵，则其均值是一个向量组。mean(X,1)为列向量的均值，mean(X,2)为行向量的均值。mean(X(:,1)) 求X矩阵中第一列的平均值

std，均方差，std(X,0,1)求列向量方差，std(X,0,2)求行向量方差。std(X(:,1)) 求X矩阵中第一列的标准差

size(A,n)

如果在size函数的输入参数中再添加一项n，并用1或2为n赋值，则 size将返回矩阵的行数或列数。其中r=size(A,1)该语句返回的是矩阵A的行数， c=size(A,2) 该语句返回的是矩阵A的列数。