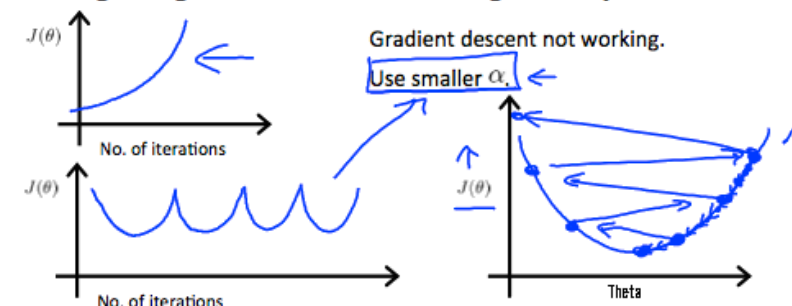


Making sure gradient descent is working correctly.



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

mean normalization:
(x - mu) / sigma or
range(x), 除数可能是
标准差也可能是
range差值

feature scaling: x /
(max(x) - min(x))

$$h_{\theta}(x) = [\theta_0 \quad \theta_1 \quad \dots \quad \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} = \theta^T x$$

注意特征标准化

不是直线，而是曲线

特征转换, x1,
x2=x1^2, x3 =
sqrt(x1), x4 = x1^3,
x5 = x1 * x2

确保损失函数一直
变小, alpha太小收敛
较慢; 太大可能不
收敛

加快theta梯度下降
的速度

方法

多种特征

多项式回归

学习步长

特征标准化

多种特征

多变量线性回归

矩阵计算参数

多变量的梯度下降

矩阵不可逆

太多特征(m <= n) ->
正则化

重复的特征

参数求解公式

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

当n>10K时, 梯度下
降也能够工作的很
好

与梯度下降区别

O(kn^2)与O(n^3)

需要许多迭代步骤

需要步长alpha

Gradient Descent

Previously (n=1):

Repeat {

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

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

}

New algorithm (n ≥ 1):

Repeat {

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

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

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

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

...}