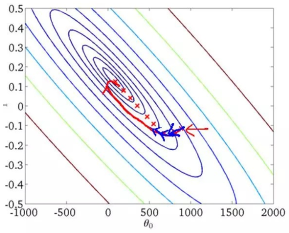
17.4.12（梯度下降，二元线性回归）

1. 梯度下降（沿着梯度向量相反的方向）
    a. 梯度下降是一种最优化算法，用来最小化损失函数
    b. 梯度下降每一步下降的步长我们称之为学习率
2. 梯度下降的三种训练形式：BGD、SGD以及MBGD
    a. 批量梯度下降法BGD
        i. 批量梯度下降法（Batch Gradient Descent，简称BGD）是梯度下降法最原始的形式，它的具体思路是在更新每一参数时都使用所有的样本来进行更新。
        ii. 优点：全局最优解；易于并行实现；

缺点：当样本数目很多时，训练过程会很慢。



b.批量梯度下降法SGD

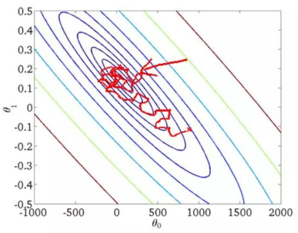
i:由于批量梯度下降法在更新每一个参数时，都需要所有的训练样本，所以训练过程会随着样本数量的加大而变得异常的缓慢。

ii:SGD伴随的一个问题是噪音较BGD要多，使得SGD并不是每次迭代都向着整体最优化方向。

iii:随机梯度下降是通过每个样本来迭代更新一次.如果样本量很大的情况（例如几十万），那么可能只用其中几万条或者几千条的样本，就已经将theta迭代到最优解.

优点：训练速度快；

缺点：准确度下降，并不是全局最优；不易于并行实现。

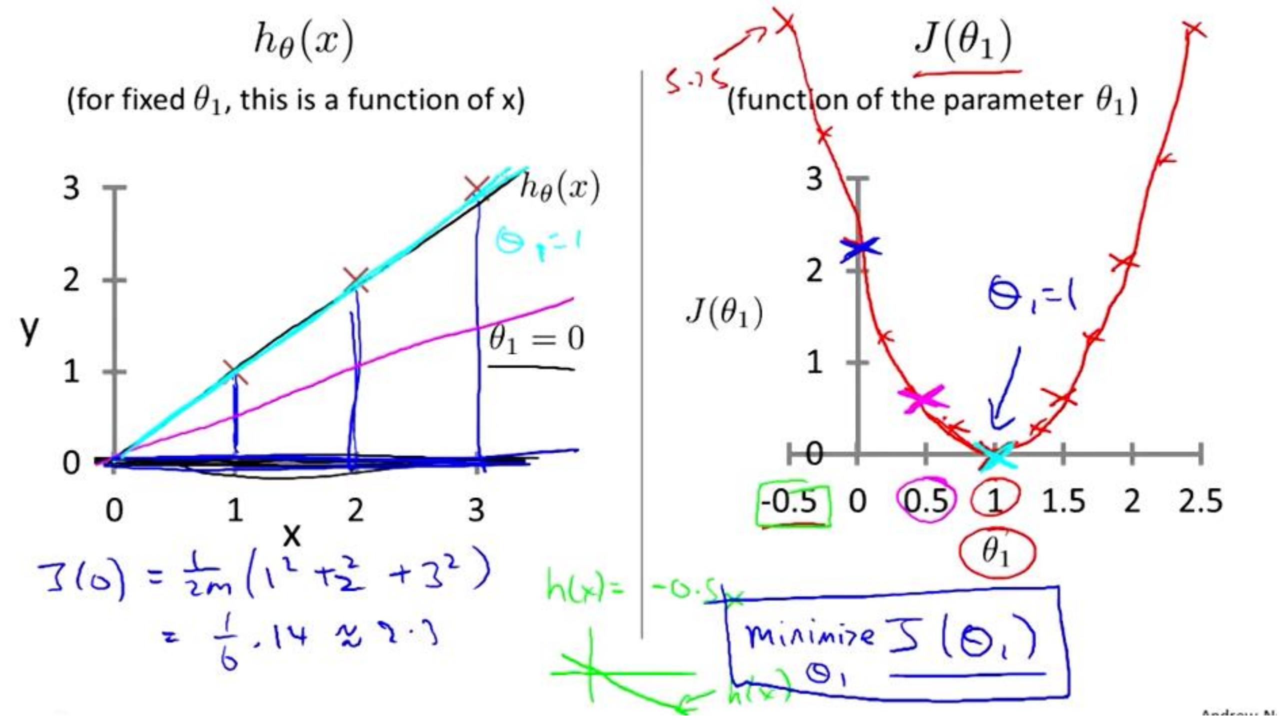


c.小批量梯度下降法MBGD

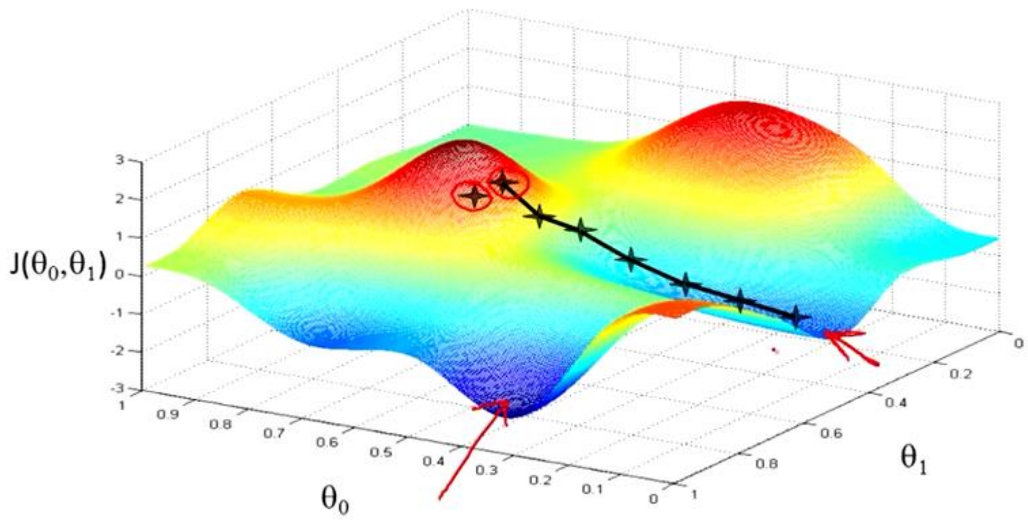
i:算法的训练过程比较快，而且也要保证最终参数训练的准确率，而这正是小批量梯度下降法（Mini-batch Gradient Descent，简称MBGD）的初衷

3. 求二元线性回归的代价函数：J（theta）=sum(X*theta-y)/2m

为何是除以2m，第一反应不应该除以m么？在吴恩达机器学习视频公开课上讲解是为了其他数学计算的方便。其实这里无论除以2m还是m，代价函数最优化的结果θ 都是相同的。



4. 求二元线性回归的梯度下降算法：theta=theta-alpha/m[sum(X*theta-y).sum((X*theta-y).*X(:,2))]
5. 梯度下降是求函数最小值的算法。



Gradient descent algorithm

Assignment

$a := b$

$a := a+1$

Truth assertion

$a = b$

$a = a+1$ ✗

θ_0, θ_1

repeat until convergence {

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

}

learning rate

Simultaneously update θ_0 and θ_1

Correct: Simultaneous update

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

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

$\theta_0 := \text{temp0}$

$\theta_1 := \text{temp1}$

Incorrect:

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

$\theta_0 := \text{temp0}$

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

$\theta_1 := \text{temp1}$

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

$$= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2$$

θ_0 $j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$

θ_1 $j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$

Gradient descent algorithm

repeat until convergence {

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

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

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

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

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$