梯度下降最小化函数 J

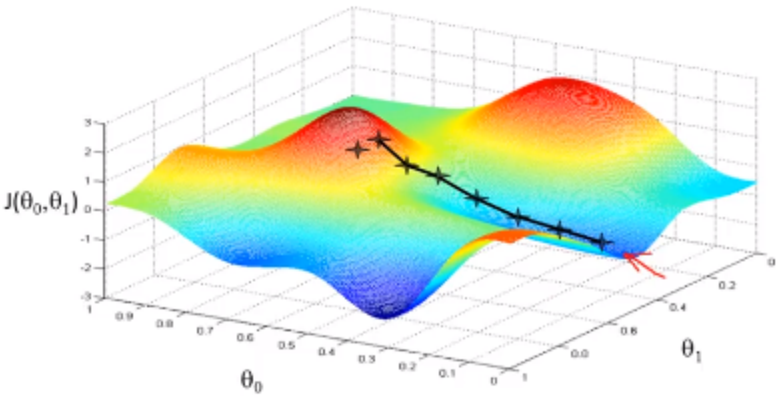
Have some function $J(\theta_0, \theta_1)$ J·

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

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

使用一个函数 J(θ0, θ1)，这是一个线性回归的代价函数 也许是一些其他函数要使其最小化 我们需要用一个算法来最小化函数 J(θ0, θ1) 就像刚才说的，事实证明 梯度下降算法可应用于多种多样的函数求解 所以想象一下如果你有一个函数 J(θ0, θ1, θ2, ...,θn) 你希望可以通过最小化 θ0 到 θn 来最小化此代价函数 J(θ0 到 θn)用 n 个 θ 是为了证明梯度下降算法可以解决更一般的问题但为了简洁起见 为了简化符号在接下来的视频中 我只用两个参数 θ0,和 θ1，寻找最近的下山方向



寻找局部最优解

梯度下降算法定义：

Gradient descent algorithm

Assignment $a := b$, $a := a+1$

Truth assertion $a = b$, $a = a+1$

θ_0, θ_1

repeat until convergence {

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

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

$\theta_0 := \text{temp0}$

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

$\theta_1 := \text{temp1}$

注： :=是复制运算符，a := a+1 将 a 的值加上 1，这里的 α 被称为学习速率，在梯度下降中它控制了我们下山买多大的步子，α 值很大—下山迈的步子会很大，

onvergence {

$\boxed{\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)}$ 是微分项，持续更新。梯度下降算法的核心是重复直到收敛

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

（后面会说积分公式的推导）

梯度下降过程更新：

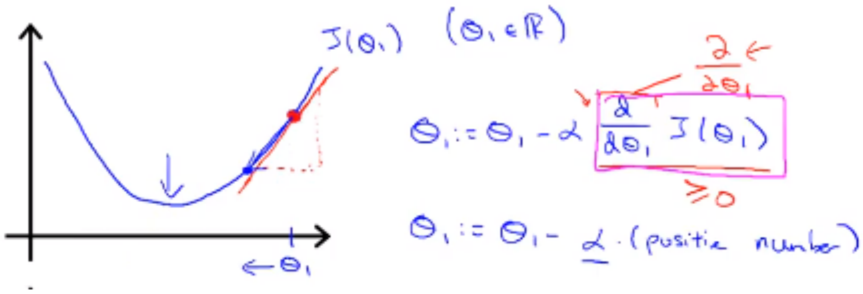
repeat until convergence {

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1) \qquad \text{(simultaneously update } j=0 \text{ and } j=1\text{)}$$

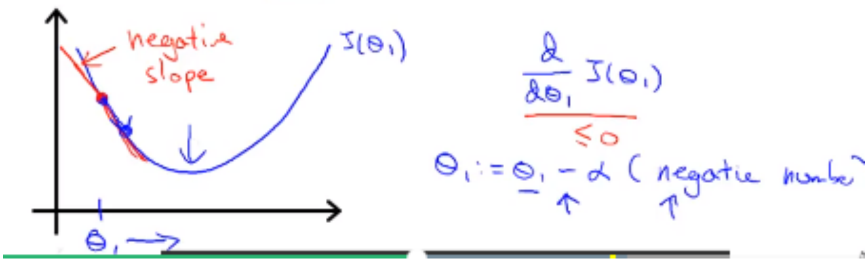
}

梯度下降算法：

α 代表学习率，控制以多大的幅度更新这个参数 θj.第二部分是导数项，$\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)$。



用一个稍微简单的例子 比如我们想最小化的那个函数只有一个参数的情形 所以假如我们有一个代价函数 J 只有一个参数 θ1，那么可以画出一维的曲线看起来很简单。为什么梯度下降法会在这个函数上起作用？所以假如这是我的函数关于 θ1 的函数 J θ1 是一个实数现在已经对这个点上用于梯度下降法的 θ1 进行了初始化，想象一下在我的函数图像上从那个点出发 那么梯度下降要做的事情是不断更新 θ1 等于 θ1 减 α 倍的 d/dθ1J(θ1)这个项 对吧这是一个偏导数 这是一个导数这取决于函数 J 的参数数量 我们要计算这个导数，对于这个问题 求导的目的 基本上可以说取这一点的切线 就是这样一条红色的直线刚好与函数相切于这一点 让我们看看这条红色直线的斜率其实这就是导数 也就是说 直线的斜率 也就是这条刚好与函数曲线相切的这条直线 这条直线的斜率正好是这个高度除以这个水平长度 现在 这条线有一个正斜率 也就是说它有正导数 因此 我得到的新的 θ，θ1 更新后等于 θ1 减去一个正数乘以 α.α 也就是学习速率也是一个正数 所以我要使 θ1 减去一个东西所以相当于我将 θ1 向左移 使 θ1 变小了 我们可以看到这么做是对的 因为实际上我往这个方向移动确实让我更接近那边的最低点 所以梯度下降到目前为止似乎是在做正确的事

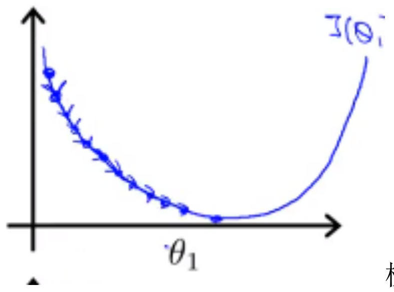


用同样的函数同样画出函数 θ1 图像，此时斜率为负 θ1 更新之后会增加，从这个图看出，增加 θ1 之后，让我更接近最小值，这就是导数项的意义。

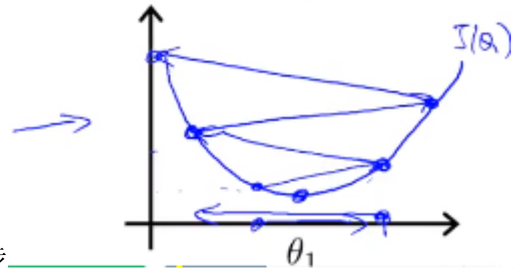
学习率 α，越大下一次迭代移动步数越大，但是

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

从公式可以知道，当达到最优解的时候导数为 0，θ1：=θ1+α*0，θ1 不会再发生改变。α 特别小的时候

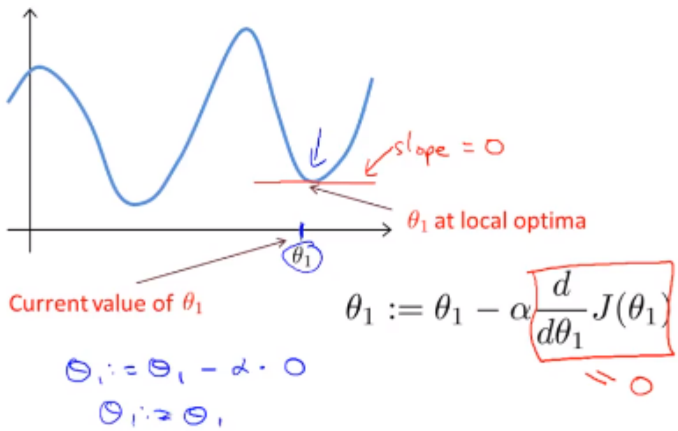


梯度下降很慢，α 特别大的时候，会迈出一大步



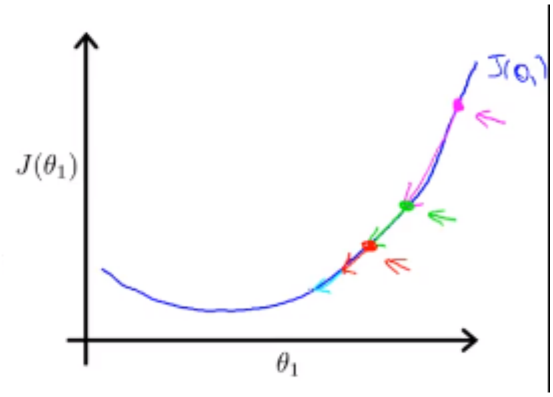
，实际上 α 值越大会导致无法收敛，甚至发散。

当它到了局部最优解的时候，导数为 0，θ1：=θ1+α*0，θ1 不会发生变化，梯度下降到达局部最优点。



梯度下降的过程详细介绍：

$$\theta_1 := \theta_1 - \alpha \boxed{\frac{d}{d\theta_1} J(\theta_1)}$$



从最上面红点开始往下降低，导数比较大，相当陡，更新以不梯度下降之后到了绿点，会发现倒数也就是斜率没那么陡了，随着接近最低点，倒数趋向于 0，知道最终移动幅度特别的小，所以梯度下降会采取更小的幅度不需要减小 α，最终到达局部最低点，最小化任何代价函数。

梯度下降应用-具体拟合直线

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

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

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

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

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

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

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

偏导项可以简化成

Gradient descent algorithm

$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

repeat until convergence {

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

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

}

$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

update θ_0 and θ_1 simultaneously