

1. 使用梯度下降法寻找最小化代价函数的参数。

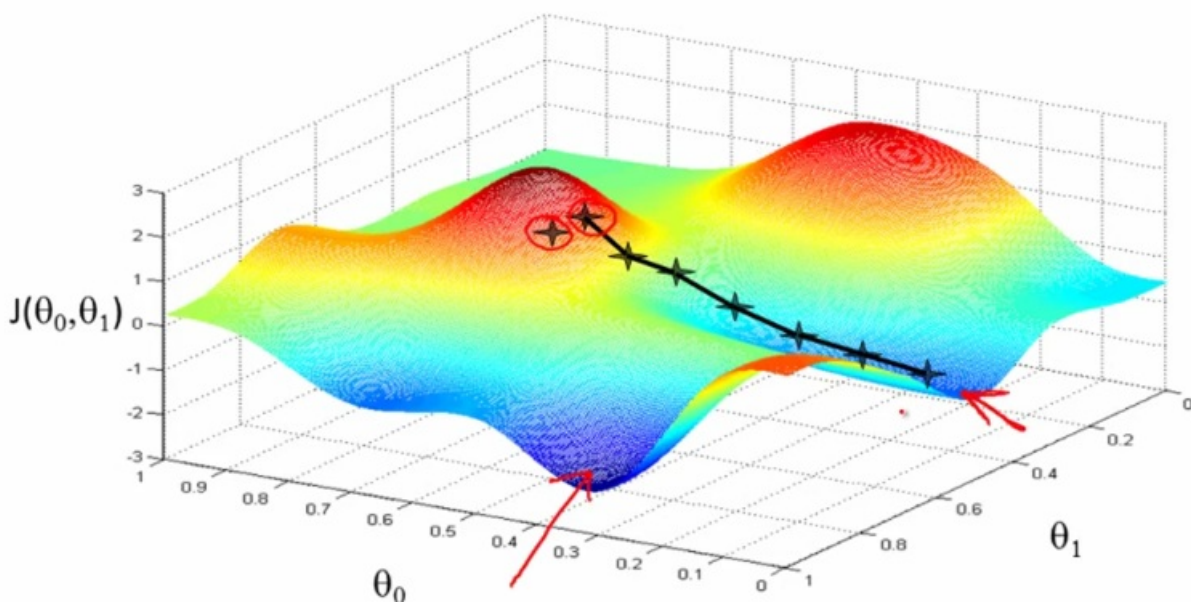
Have some function $J(\theta_0, \theta_1)$ $J(\theta_0, \theta_1, \theta_2, \dots, \theta_n)$

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

Outline:

- Start with some θ_0, θ_1 (say $\theta_0 = 0, \theta_1 = 0$)
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$
until we hopefully end up at a minimum

梯度下降算法的思想是：下山每一步都选择坡度最大的方向，特点是：最终结果取决于初始值的选择，有可能会陷入局部最优。



算法为：

Gradient descent algorithm

$$\begin{aligned} &\rightarrow \tilde{a_i} = b \\ &\quad \uparrow \\ &\quad a_i = a + 1 \end{aligned}$$

repeat until convergence {
 $\rightarrow \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ (for $j = 0$ and $j = 1$)
}
learning rate

迭代直到收敛。其中alpha是学习速率，其值越大，更新地越快。

算法需要同步更新theta0和theta1:

Correct: Simultaneous update

$\rightarrow \text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
 $\rightarrow \text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
 $\rightarrow \theta_0 := \text{temp0}$
 $\rightarrow \theta_1 := \text{temp1}$

Incorrect:

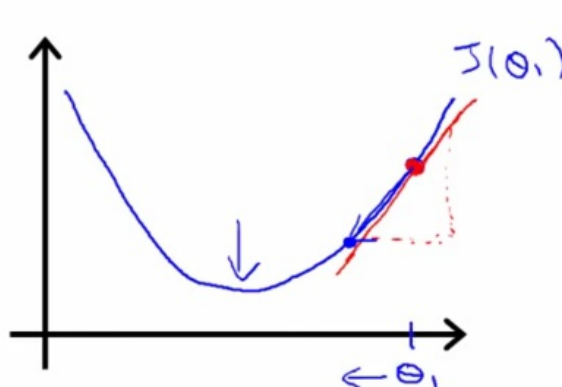
$\rightarrow \text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
 $\rightarrow \theta_0 := \text{temp0}$
 $\rightarrow \text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
 $\rightarrow \theta_1 := \text{temp1}$

我们之后会讲到

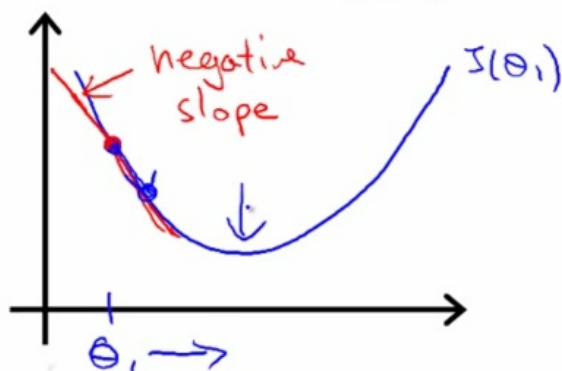
Andrew Ng

2. 理解梯度算法中的偏导数，首先考虑简单情况，即 $\theta_0 = 0$:

很显然，沿着导数的方向，函数下降地最快，并且在最小值左边，由于导数为负，更新 θ_1 会增加 θ_1 的值，同理在右边会减小 θ_1 的值，这样便使得 θ_1 不断靠近最小值点：



$J(\theta_1) \quad (\theta_1 \in \mathbb{R})$
 $\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$
 $\theta_1 := \theta_1 - \alpha \cdot (\text{positive number})$



$\frac{\partial}{\partial \theta_1} J(\theta_1)$
 $\theta_1 := \theta_1 - \alpha (\text{negative number})$

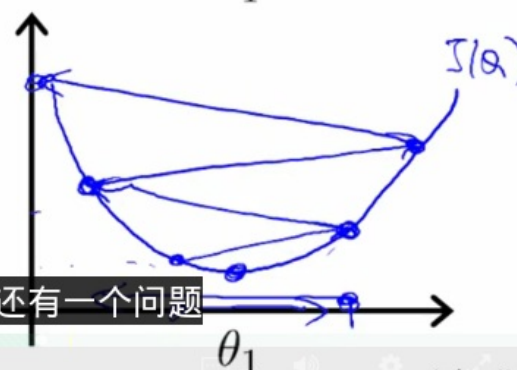
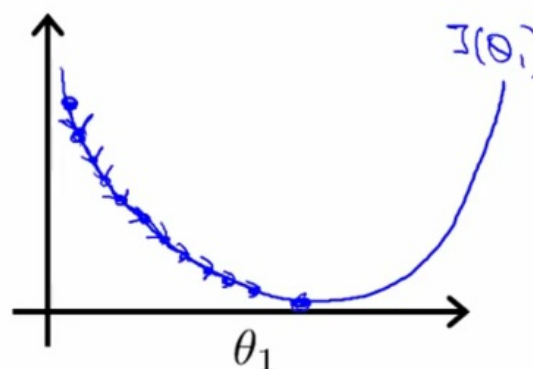
Andrew Ng

学习速率过小，收敛速度慢；过大，可能造成不收敛：

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

If α is too small, gradient descent can be slow.

If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



它会导致无法收敛 甚至发散 现在 我还有一个问题

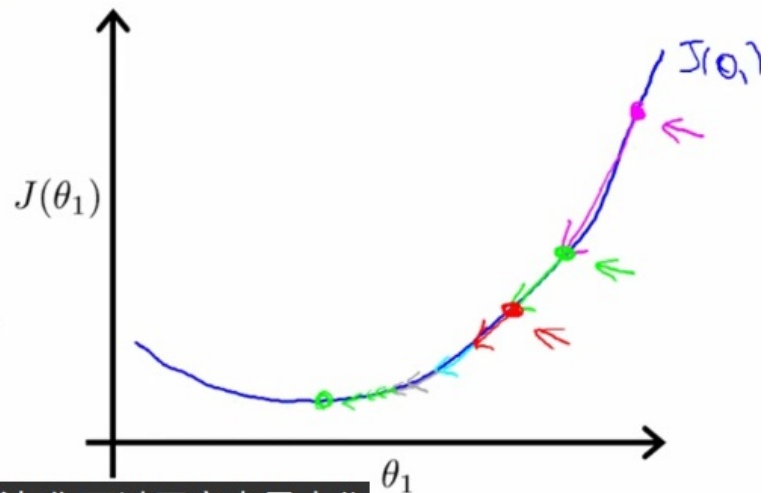
如果已经到达一个局部最低点，因为导数为0，梯度下降法不会更新参数。如果靠近局部最低点（因为学习速率的选择无法到达局部最低点，将会在局部最小值左右摇摆）

另外，靠近最小值时，导数会变小，因此更新速度会下降，所以不用更改学习速率，梯度下降算法在快要收敛时会自动下降更新速度。

Gradient descent can converge to a local minimum, even with the learning rate α fixed.

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

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time



这就是梯度下降算法 你可以用它来最小化

3. 将梯度下降算法应用到线性回归中。

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

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

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

$$\text{Min}_{\theta_0, \theta_1} J(\theta_0, \theta_1)$$

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

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

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

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

梯度下降算法：

Gradient descent algorithm

repeat until convergence {

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

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

}

update
 θ_0 and θ_1
simultaneously

$$\frac{2}{2\theta_j} J(\theta_0, \theta_1)$$

执行梯度下降时

批量梯度下降：每一步用到所有的训练集：

“Batch” Gradient Descent

“Batch”: Each step of gradient descent uses all the training examples.

$$\rightarrow \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$

线性代数中，可以不用梯度下降，使用名为正规方程（normal equations）的数值解法。