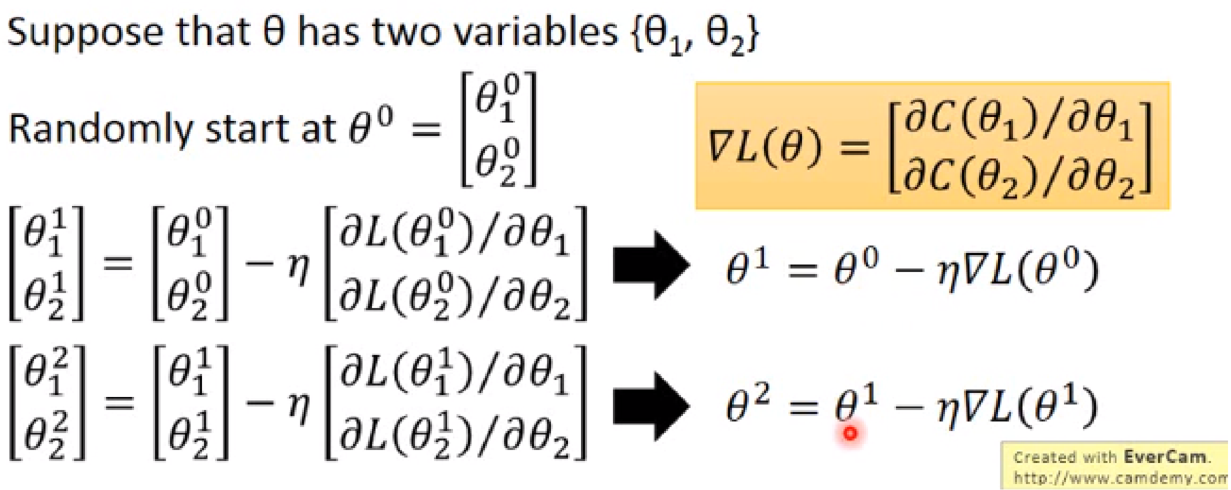
梯度下降

我们在确定函数之后，需要找出使损失函数值最小的参数值。即：

$$\theta^* = argmax_\theta L(\theta)$$

其中，L——损失函数，θ——为参数

假设只有两个参数，则迭代方法如下：



从而一次类推。

如何调整学习率（Learning Rate）

最流行最简单的方法是：学习率自动随着每一次参数的迭代，缩小一点点。

- 在一开始，我们是远离目标的，所以我们使用较高的学习率，使得加快收敛速度。
- 在经过多次迭代之后，我们将离目的地越来越近，所以这个时候减小学习率。

自适应梯度下降

通过使用自适应梯度算法可以实现自动变化学习率：

将每个参数的学习率除以之前算出来的微分值的均方根。

- 梯度下降

$$W^{(t+1)} \leftarrow W^t - \eta^t g^t$$

- 自适应梯度下降

$$W^{(t+1)} \leftarrow W^t - \frac{\eta^t}{\sigma^t} g^t$$

$$\text{其中} \eta^t = \frac{\eta}{\sqrt{t+1}}$$

$$g^t = \frac{\partial C(\theta^t)}{\partial w}$$

$$\sigma^t \text{：代表的是过去所有已经计算出来的微分值的均方根}$$

$$\text{再通过简化公式之后可以得到：} W^{(t+1)} \leftarrow W^t - \frac{\eta}{\sum_{i=0}^{t} (g^i)^2} g^t$$

Stochastic Gradient Descent（随机梯度下降）

$$\text{已知损失函数为} L = \sum_n (y^n - (b + \sum w_i x_i^n))^2$$

$$\text{则参数的梯度下降为} \theta^i = \theta^{(i-1)} - \eta \nabla L(\theta^{i-1})$$

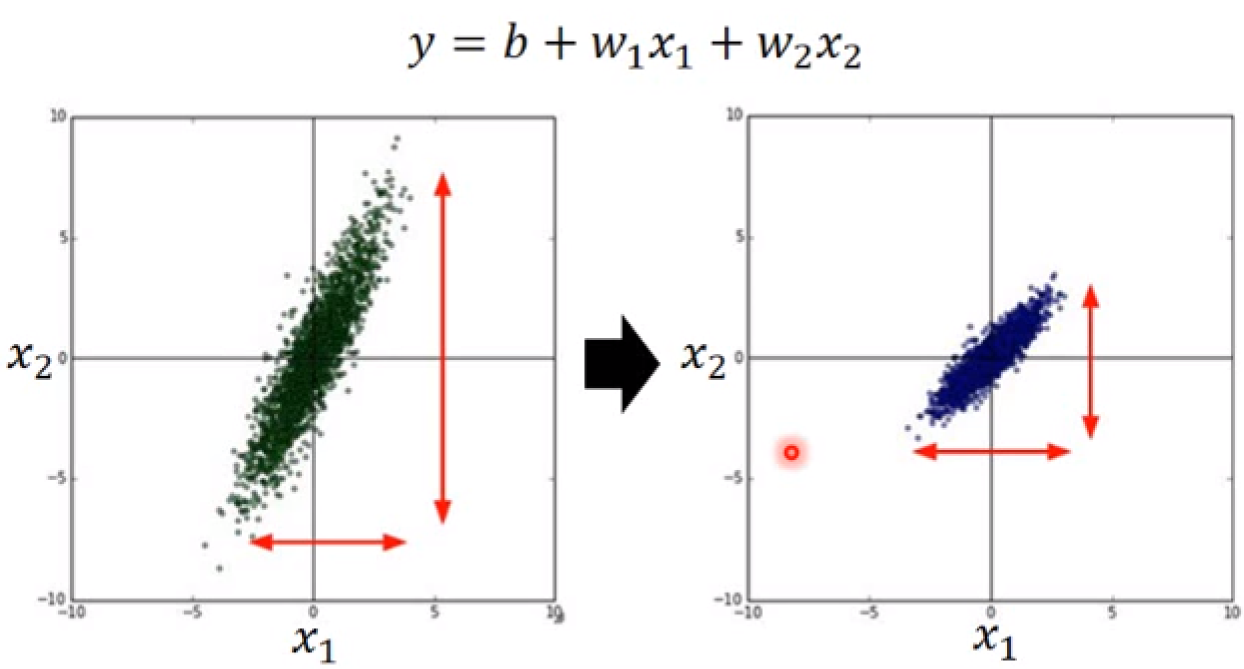
$$\text{首先选择一个样本} x^n,$$

$$\text{只针对样本} x^n \text{进行梯度下降的计算}$$

$$\text{则可以得到} L^n = \sum_n (y^n - (b + \sum w_i x_i^n))^2 \text{（上面是将所有的} x \text{样本进行计算，而这个公式计算的是只有} x^n \text{样本）}$$

$$\text{而参数的梯度下降为} \theta^i = \theta^{(i-1)} - \eta \nabla L^n(\theta^{i-1})$$

Feature Scaling（特征缩放）



如图我们可以看到

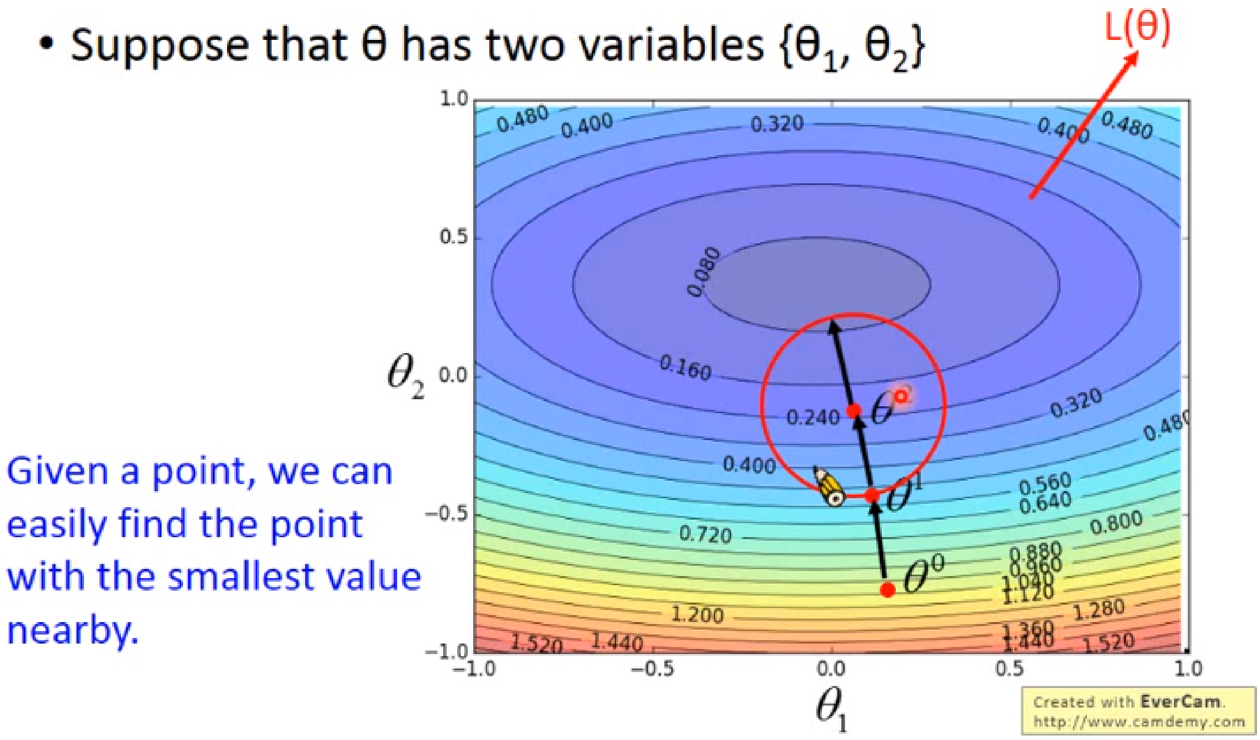
$$x^2 \text{上的样本分布得比较散，因此我们可以使用某些特定的方法来对参数进行缩放。}$$

常见方法为:

$$\text{假设有}R\text{个样本，而}m_i\text{为所有样本第}i\text{个元素的平均值，而}\sigma_i\text{表示的是所有样本第}i\text{个元素的标准偏差}$$

$$\text{则有}x_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

梯度下降的理论



假设给定一个点，如何在一个特定范围内找到可以让损失函数变得更小的参数呢?

泰勒展开式

$$\text{若函数}h(x)\text{在包含}x_0\text{的某个闭区间}[a,b]\text{上具有}n\text{阶导数(无限次可微)}$$

$$\text{则有：}h(x)=\sum_{k=0}^{\infty}\frac{h^{(k)}(x_0)}{k!}(x-x_0)^k\text{(自己把它展开)}$$

$$\text{而当}x\text{非常接近}x_0\text{的时候，}h(x)\approx h(x_0)+h^{'}(x_0)(x-x_0)$$

多维泰勒展开式

Multivariable Taylor Series

$$h(x,y) = h(x_0,y_0) + \frac{\partial h(x_0,y_0)}{\partial x}(x-x_0) + \frac{\partial h(x_0,y_0)}{\partial y}(y-y_0)$$

+ something related to $(x-x_0)^2$ and $(y-y_0)^2$ + ……

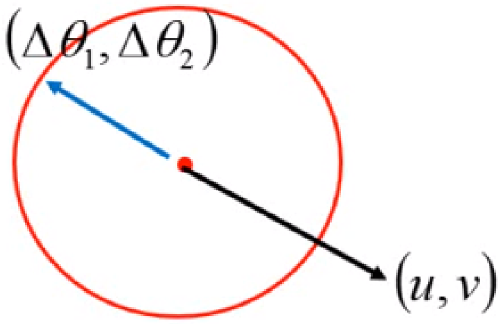
When x and y is close to x_0 and y_0

$$h(x,y) \approx h(x_0,y_0) + \frac{\partial h(x_0,y_0)}{\partial x}(x-x_0) + \frac{\partial h(x_0,y_0)}{\partial y}(y-y_0)$$

Gradient descent – two variables

Red Circle: (If the radius is small)

$$\mathrm{L}(\theta) \approx \cancel{s} + u(\underbrace{\theta_1 - a}_{\Delta\theta_1}) + v(\underbrace{\theta_2 - b}_{\Delta\theta_2})$$



Find θ_1 and θ_2 in the red circle ***minimizing*** $\mathrm{L}(\theta)$

$$(\underbrace{\theta_1 - a}_{\Delta\theta_1})^2 + (\underbrace{\theta_2 - b}_{\Delta\theta_2})^2 \le d^2$$

To minimize $\mathrm{L}(\theta)$

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \end{bmatrix} = -\eta \begin{bmatrix} u \\ v \end{bmatrix}$$

Find θ_1 and θ_2 yielding the smallest value of $L(\theta)$ in the circle

$$\begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix} = \begin{bmatrix}a \\ b\end{bmatrix} - \eta \begin{bmatrix}u \\ v\end{bmatrix} = \begin{bmatrix}a \\ b\end{bmatrix} - \eta \begin{bmatrix}\dfrac{\partial C(a,b)}{\partial \theta_1} \\ \dfrac{\partial C(a,b)}{\partial \theta_2}\end{bmatrix}$$

This is gradient descent.

Not satisfied if the red circle (learning rate) is not small enough