<u>**Optimization**</u>

Gradient Descent

$$\theta = \arg\min L(\theta)$$

$$L \rightarrow Loss\ Function$$

$$\theta \rightarrow Parameters$$

Gradient Descent 是用于找到一个 Function 的最小值。首先，随机选取一组起始的点$\begin{bmatrix}\theta_1^0 \\ \theta_2^0\end{bmatrix}$。$\theta$的上标代表第几组参数，下标代表同一组的第几个 Component。

$$\begin{bmatrix}\theta_1^1 \\ \theta_2^1\end{bmatrix} = \begin{bmatrix}\theta_1^0 \\ \theta_2^0\end{bmatrix} - \eta \begin{bmatrix}\dfrac{\partial L(\theta_1^0)}{\partial \theta_1} \\ \dfrac{\partial L(\theta_2^0)}{\partial \theta_2}\end{bmatrix} \rightarrow \begin{bmatrix}\theta_1^2 \\ \theta_2^2\end{bmatrix} = \begin{bmatrix}\theta_1^1 \\ \theta_2^1\end{bmatrix} - \eta \begin{bmatrix}\dfrac{\partial L(\theta_1^1)}{\partial \theta_1} \\ \dfrac{\partial L(\theta_2^1)}{\partial \theta_2}\end{bmatrix}$$

$$\eta \rightarrow Learning\ Rate$$

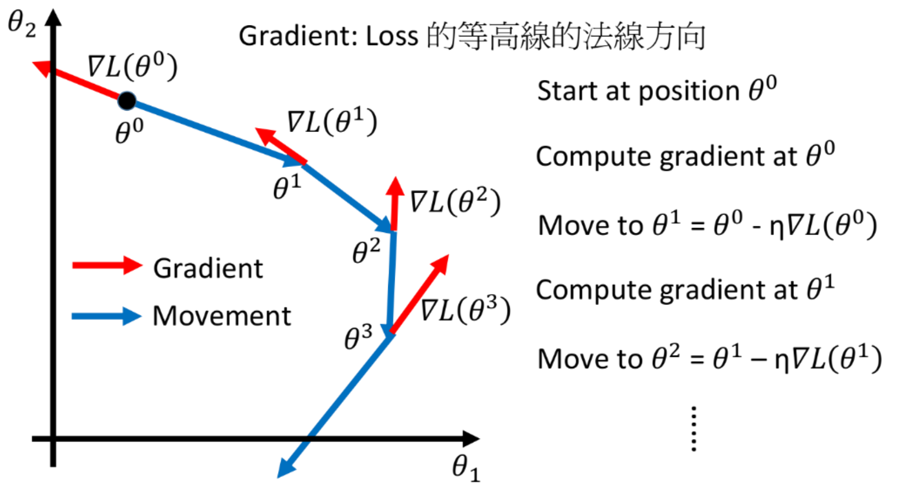
接下来开始 Update θ来找到最小值。计算θ的偏微分 (Partial Derivative)，然后乘上 Negative Learning Rate 再加上原来的θ。反复这个计算直到找到最小值。

$$\text{Gradient} = \nabla L(\theta) = \begin{bmatrix}\dfrac{\partial L(\theta_1)}{\partial \theta_1} \\ \dfrac{\partial L(\theta_2)}{\partial \theta_2}\end{bmatrix}$$

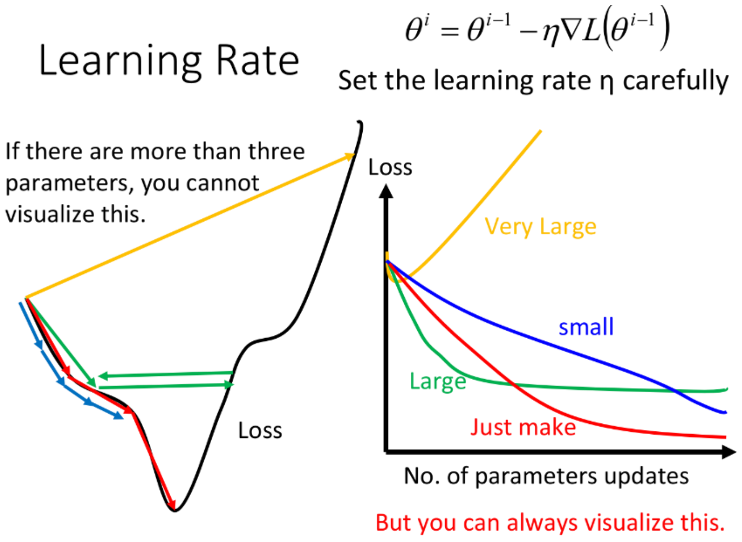
$$\theta^1 = \theta^0 - \eta \nabla L(\theta^0)$$

$$\theta^2 = \theta^1 - \eta \nabla L(\theta^1)$$

计算出的 Gradient 是红色的箭头，如果要找到最小值需要反方向。Learning Rate 决定步数。



Learning Rate 有选好，可以正常的找到最低点 (Minimum Point)，Learning Rate 太小也是可以找到最低点 (Minimum Point)可是要花费很长的时间，Learning Rate 大会找不到最低点(Minimum Point) 一直在几个区域循环，Learning Rate 太大会让 Loss 无限大 。可以通过 Plot Loss 来知道模型训练的情况。



蓝色→ Learning Rate 太小导致下降速度太慢。红色→ Learning Rate 选的刚刚好。 灰色→ Learning Rate 大了导致到一个点就没办法下降。 黄色→ Learning Rate 太大导致 Loss 越来越大。

SGD (Stochastic Gradient Descent) 1847

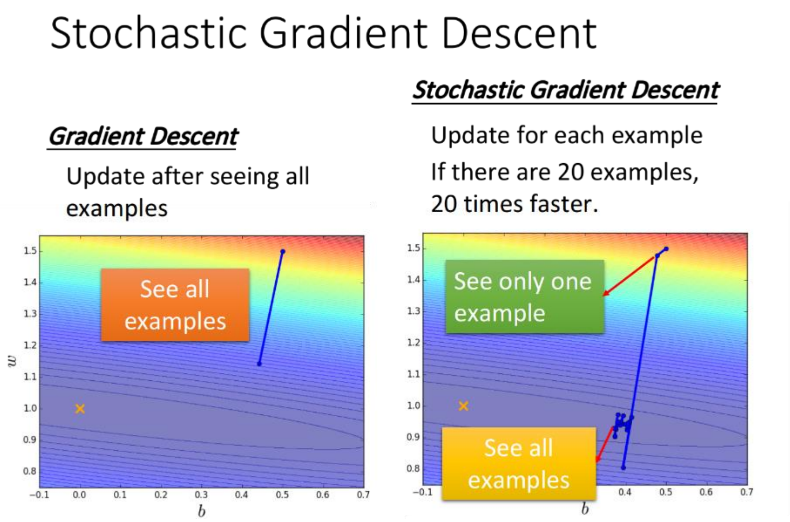
在原来的 Gradient Descent 里面，做法是看完全部训练数据，在一次过计算所有数据的 Loss 然后求 Average。但是在 SGD 里面，做法是对每个单一的数据进行 Loss 的计算然后就直接 Update 参数，没有看完全部数据。

$$Loss\ Function\ for\ SGD = \left(Y - \hat{Y}\right)^2$$

$$Y \rightarrow Predicted\ Value\ for\ 1\ Data$$

$$\hat{Y} \rightarrow Actual\ Value\ for\ 1\ Data\ (Ground\ Truth)$$

SGD 的优点



当看完 20 个数据，Gradient Descent 只会 Update 一次，而 SGD 每看一次就会 Update 一次，所以当看完 20 个数据的时候，SGD 已经 Update 了 20 次。虽然 SGD 每次的 Update 都比较散乱但是速度比 Gradient Descent 还快很多。

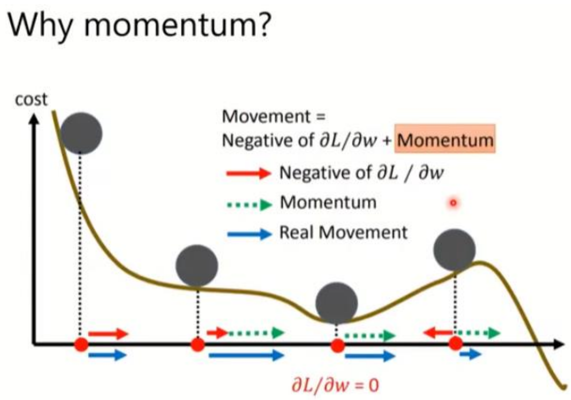
SGD 的缺点

SGD 因为频繁的 Update，所以导致 Loss 会有严重的动荡，但是不影响 Converge。

SGDM (Stochastic Gradient Descent with Momentum) 1986

在训练的过程中，会遇到的情况是在某一个 timestamp 有可能算到接近 0 的 Gradient 但是这个点不是最低点，这时候如果使用 SGD 的话就会卡在那个点很难找到更好的 Minimum Point。而 SGDM 因为有一个 Previous Movement 的值，这时候就会推动继续往前走。

还有一种说法就是 SGDM 的 step 会比较不那么崎岖，可以更快的 Converge，也能够把 Learning Rate 稍微提高让训练更快。



上图可以解释 SGDM 的好处，就是比较不会卡在一个 Local Minimum 的点。再上图算出$\frac{\partial L}{\partial w} = 0$的时候，因为有 Momentum 所以会继续往下走。

使用 SGDM 的时候会定义一个参数 Movement 为 0 先。开始计算后，会 Update Movement

$$Movement\ V\_d\theta = 0$$

$$Compute\ Gradient\ at\ \theta^0$$

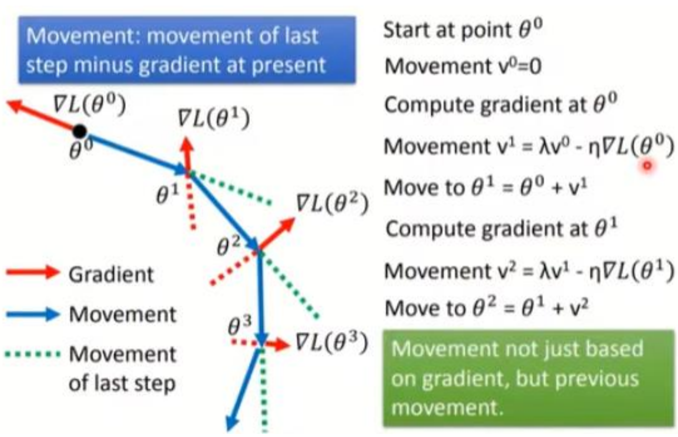
$$Update\ Movement \rightarrow V_{d\theta} = \beta V_{d\theta} + (1-\beta)\nabla L(\theta^0)$$

$$Update\ \theta \rightarrow \theta^1 = \theta^0 + \eta V^1$$

$$\theta \rightarrow weight\ \&\ bias\ (y = wx + b)$$

$$然后继续循环计算 Gradient \rightarrow Update\ Movement \rightarrow Update\ \theta$$

$$\beta \rightarrow Movement 的 Scale，一般在 0.9 左右$$



上图是在课堂上的 Slide，能够帮助了解 SGDM。

Adaptive Learning Rates

通常 Learning Rate 是随着参数的 Update 会越来越小。刚开始训练的时候离最低点比较远，这时候可以 Update 大步一点，当训练一定次数过后，参数已经比较靠近最低点的地方，这时候需要小的 Learning Rate 才能更好的 Converge。

$$\frac{1}{t} deay \rightarrow \eta^t = \frac{\eta}{\sqrt{t+1}}$$

$$\eta \rightarrow Learning\ Rate\ (是一个t\ dependent\ 的函数)$$

$$t \rightarrow 参数Update的次数$$

参数 Update 的次数越多，Learning Rate 就会越来越小。

Adagrad 2011

$$Gradient\ Descent \rightarrow w^{t+1} = w^t - \eta^t g^t$$

$$Adagrad \rightarrow w^{t+1} = w^t - \frac{\eta^t}{\sigma^t} g^t$$

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

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

$$g^t \rightarrow Value\ of\ Partial\ Derivative\ (Gradient)$$

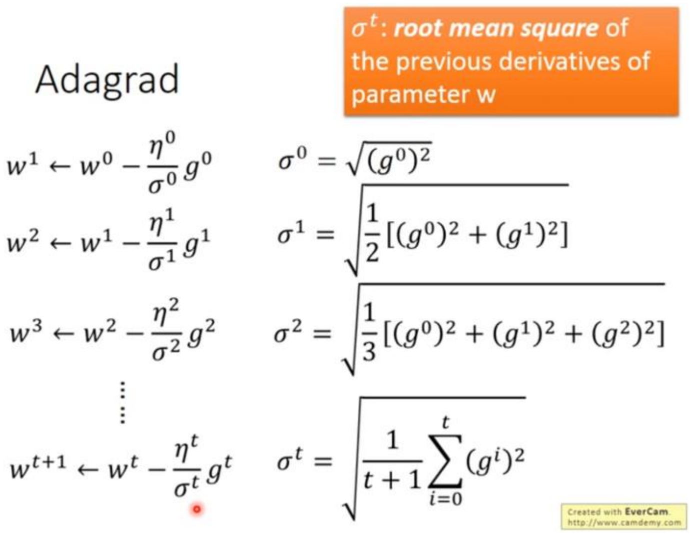
$$\sigma^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g^i)^2} \rightarrow t+1\ \ 因为\, t\ start\ from\ 0$$

$$\sigma^t \rightarrow 过去所有 Partial\ Derivative\ 的\, Root\ Mean\ Square$$

$$Simplified\ Adagrad \rightarrow w^{t+1} = w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$$

$$w^{t+1} = w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2} \rightarrow (Second\ Derivative)} g^t \rightarrow First\ Derivative$$

Second Derivative 的想法是让 Update 的参数能够走到最好的一步。通过计算发现，最好的步数是 $\frac{|First\ Derivative|}{Second\ Derivative}$。Adagrad 就是再模仿这个算法。



每一次 Update Parameter 的时候 Learning Rate 都会不同，因为除上了σ^t。但是随着 Update 的次数越来越多，Adagrad 这个方法 Update 参数会越来越小，导致训练时间需要比较长。

RMSProp (Root Mean Square Propagation) 2013

RMSProp 与 Adagrad 的差别只在于 Scale Down Learning Rate 的方法不一样。RMSProp 接用了 Momentum 的概念。这是为了解决 Adagrad 卡着的问题因为 Adagrad 算法的 Learning Rate 在计算到一定次数的时候会变得非常小导致卡着没进展。还有就是使用了 Momentum 的概念会让模型更快的 Converge，步伐不会这么崎岖，也能够把 Learning Rate 稍微提高让训练更快。

$$\theta^t = \theta^t - \frac{\eta}{\sqrt{S_{d\theta}^t}} d\theta^t$$

$$S_{d\theta} = \beta S_{d\theta} + (1-\beta) d\theta^2$$

要注意的是$\sqrt{S_{d\theta}^t}$ 不会是 0。

Adam (Adaptive Moment Estimation)

Adam 就是把 SGDM 与 RMSProp 的概念融合在一起。

SGDM

$$\theta_t = \theta_{t-1} - \eta m_t$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_{t-1}$$

RMSProp

$$\theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{v_t}} g_{t-1}$$

$$v_1 = g_0^2$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_{t-1}^2$$

Adam

$$Adam \to \theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\widehat{v_t} + \varepsilon}} \widehat{m_t}$$

$$Bias\ Correction \to \widehat{m_t} = \frac{m_t}{1-\beta_1^t} \to \widehat{v_t} = \frac{v_t}{1-\beta_2^t}$$

$$\eta \to Learning\ Rate\ (Need\ to\ Tune)$$

$$B_1 \to 0.9$$

$$\beta_2 \to 0.999$$

$$\varepsilon \to 10^{-8}\ (Avoid\ indeterminate \to something\ divide\ by\ 0)$$

初始化的时候，因为β_1和β_2的值较小，会导致一开始的 Update Step 比较大，所以需要做 Bias Correction 来确保可靠性。