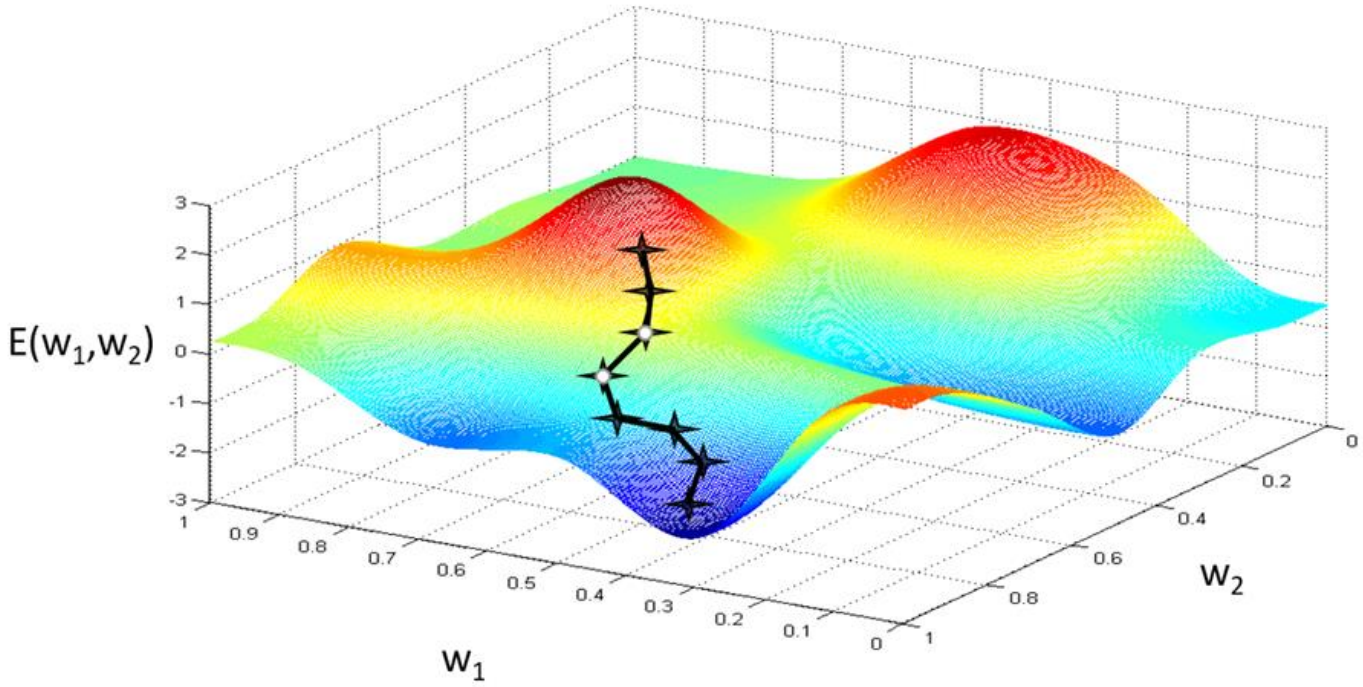
Optimizer：

tf.train.GradientDescentOptimizer
tf.train.AdadeltaOptimizer
tf.train.AdagradOptimizer
tf.train.AdagradDAOptimizer
tf.train.MomentumOptimizer
tf.train.AdamOptimizer
tf.train.FtrlOptimizer
tf.train.ProximalGradientDescentOptimizer
tf.train.ProximalAdagradOptimizer
tf.train.RMSPropOptimizer

各种优化器对比：
标准梯度下降法：
标准梯度下降先计算所有样本汇总误差，然后根据总误差来更新权值
随机梯度下降法：
随机梯度下降随机抽取一个样本来计算误差，然后更新权值
批量梯度下降法：
批量梯度下降算是一种折中的方案，从总样本中选取一个批次（比如一共有10000个样本，随机选取100个样本作为一个batch），然后计算这个batch的总误差，根据总误差来更新权值。



W：要训练的参数

J(W)：代价函数

$\nabla_W J(W)$：代价函数的梯度

η：学习率

SGD：

$$W = W - \eta \cdot \nabla_W J(W; x^{(i)}; y^{(i)})$$

Momentum：

γ：动力，通常设置为0.9

$$v_t = \gamma v_{t-1} + \eta \nabla_W J(W)$$
$$W = W - v_t$$

当前权值的改变会受到上一次权值改变的影响，类似于小球向下滚动的时候带上了惯性。这样可以加快小球的向下的速度。

NAG（Nesterov accelerated gradient）：

$$v_t = \gamma v_{t-1} + \eta \nabla_W J(W - \gamma v_{t-1})$$
$$W = W - v_t$$

NAG在TF中跟Momentum合并在同一个函数tf.train.MomentumOptimizer中，可以通过参数配置启用。
在Momentun中小球会盲目地跟从下坡的梯度，容易发生错误，所以我们需要一个更聪明的小球，这个小球提前知道它要去哪里，它还要知道走到坡底的时候速度慢下来而不是又冲上另一个坡。γvt−1会用来修改W的值，计算W−γvt−1可以表示小球下一个位置大概在哪里。从而我们可以提前计算下一个位置的梯度，然后使用到当前位置。

Adagrad：

i：代表第i个分类

t：代表出现次数

ϵ：的作用是避免分母为0，取值一般为1e-8

η：取值一般为0.01

$g_{t,i} = \nabla_W J(W_i)$

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{\sum_{t'=1}^{t}(g_{t',i})^2+\epsilon}} \odot g_t$$

它是基于SGD的一种算法，它的核心思想是对比较常见的数据给予它比较小的学习率去调整参数，对于比较罕见的数据给予它比较大的学习率去调整参数。它很适合应用于数据稀疏的数据集（比如一个图片数据集，有10000张狗的照片，10000张猫的照片，只有100张大象的照片）。
Adagrad主要的优势在于不需要人为的调节学习率，它可以自动调节。它的缺点在于，随着迭代次数的增多，学习率也会越来越低，最终会趋向于0。

RMSprop：

RMS（Root Mean Square）是均方根的缩写。

γ：动力，通常设置为0.9

η：取值一般为0.001

$E[g^2]_t$：表示前t次的梯度平方的平均值

$g_t = \nabla_W J(W)$

$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g^2_t$

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{E[g^2]_t+\epsilon}} \odot g_t$$

RMSprop借鉴了一些Adagrad的思想，不过这里RMSprop只用到了前t-1次梯度平方的平均

值加上当前梯度的平方的和的开平方作为学习率的分母。这样RMSprop不会出现学习率越来越低的问题，而且也能自己调节学习率，并且可以有一个比较好的效果。

Adadelta：

$$g_t = \nabla_W J(W)$$

$$\Delta W_t = -\frac{\eta}{\sqrt{E[g^2]_t+\epsilon}} \odot g_t$$

$$\Delta W_t = -\frac{\eta}{RMS[g]_t} \odot g_t$$

$$W_{t+1} = W_t - \frac{RMS[\Delta W]_{t-1}}{RMS[g]_t}$$

使用Adadelta我们甚至不需要设置一个默认学习率，在Adadelta不需要使用学习率也可以达到一个非常好的效果。

Adam：

β_1：一般取值0.9

β_2：一般取值0.999

ε：避免分母为0，一般取值10^{-8}

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

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

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{V}_t = \frac{v_t}{1-\beta_2^t}$$

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{\hat{v}_t}+\varepsilon}\hat{m}_t$$

就像Adadelta和RMSprop一样Adam会存储之前衰减的平方梯度，同时它也会保存之前衰减的梯度。经过一些处理之后再使用类似Adadelta和RMSprop的方式更新参数。