要优化Deep Learning的神经网络首先要搞清楚问题出在哪里，总体上分为两部分：针对Training data上性能的优化和针对Testing data的性能优化。上篇笔记介绍针对Training data上性能优化的一种方法：替换激活函数（New activation function）；本文将介绍优化Training data上性能的另一种方法：Adaptive Learning Rate。

假设用Cross entropy作为Loss function，则我们要求解的是最优参数：

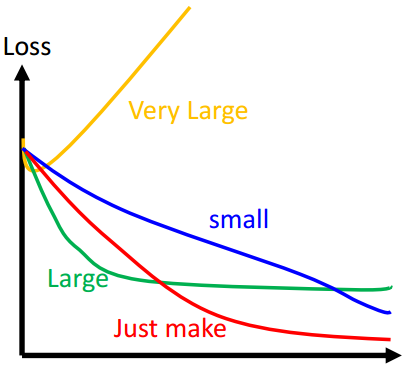


利用梯度下降法做Training的迭代公式如下：

随机的给参数一个初始值：





在做参数优化时，我们随着迭代次数的增加一直在更新参数，如图1所示（假设参数只有一个维度）：

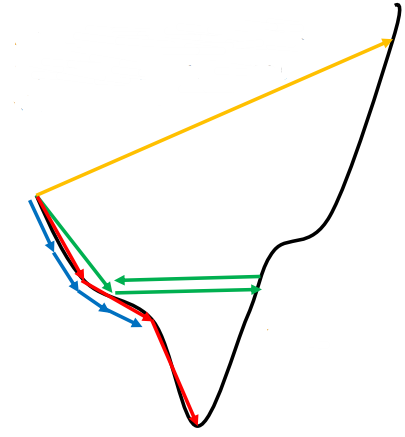


图1（a） 参数在Loss上的变化 图1（b） Loss随迭代次数的变化

如图1所示，假设我们设置了4种学习步长，分别对应蓝色、红色、绿色和黄色。很明显，设置太小，学习速度太慢，虽然最终能到一个最优点，但迭代次数太多；设置较大，学习速度很快，但并没有得到全局最优点；设置过大，Loss一下就飞出去了，相比设置较为合理；对应如右图所示，在实际的调参过程中，我们可以根据右边图中Loss的变化趋势来判断设置是否合理。

如果我们在整个训练过程中值保持不变，显然就容易出现图1中的各种情况，而实际中我们希望一开始训练速度快一点，而到后期接近最优参数时能将训练的step放慢一点。这个想法就是Adaptive Learning Rate。

在实际操作中，一般是几个epochs改变一次Learning Rate。

1. **vanilla Gradient descent**

第t次epochs迭代，就减小倍，即：





vanilla Gradient descent由于只与第t次epochs相关，还是相对固定的；实际上在梯度比较大的时候，参数学习的步长还是较大的，而且与之前的梯度之间没有建立关系。

**2、Ada-grad**

Ada-grad是在vanilla Gradient descent上做的改进：

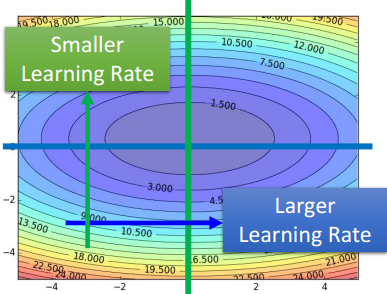


其中是之前所有梯度的平方和求平均后再开根号：



当较大时， 变小；当较小时， 变大；形成一种反差。





如左图：

1）当横坐标维度上的较小时，learning rate： 就较大；



2）当纵坐标维度上的较大时，

learning rate： 就较小；



图2 loss function

**3、RMSProp**

很多时候，Error surface并非像图2那样在单一维度上梯度变化是一致的（变化大或者变化小），而是时大时小的，如图3。

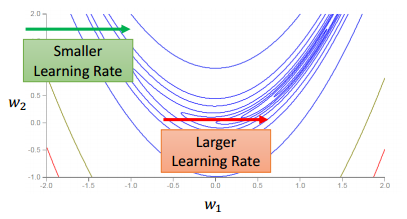


图3 Error surface

这个时候，步长的变化就不会像**Ada-grad**那样简单的加权平均之前所有的，而是只给之前的一个权重，根据权重来选择更相信之前的值还是现在的值。



**3、Momentum**

**Momentum**的思想来源于物理规律，如图4所示，从上坡上滚下来的小球，可能在A、B或C点停下来，三点处的梯度都接近或等于零，但是并不是最优的点，实际上在小球在势能的助力下还会向前一时刻运动的方向滚动，在local minima的地方也会继续朝右运动，有可能借助势能就会跨过小坡从而越过这个local minima的点。

local minima可能是一个影响比较大的问题，但是对参数超多的Deep Learning而言或许并不是什么问题，因为Deep Learning有成千上万的参数，可以认为参数有成千上万的维度，如果Error surface上有一个local minima，则要求这个点的每个维度都是山谷的位置，这个成立的概率是很低的，所以Error surface上是不会有太多的local minima的；这个观点是Facebook的机器学习大牛Yann Lecun在07年的一篇论文中提出来的。



图4 Momentum示意图

具体迭代方式（以一个参数举例）：

1）初始参数，代表前一个时间点运动的方向，初始为；

2）计算梯度，则当前时刻移动方向：

，

更新参数：

3）依此类推，第t个时刻移动方向：



更新参数：

代表惯性的大小，越大惯性越大，反之亦然。

**4、Adam**

Adam实际上就是RMSprop +Momentum，详情可参考论文《Adam:A Method for Stochastic Optimization》，论文中算法描述如图5。

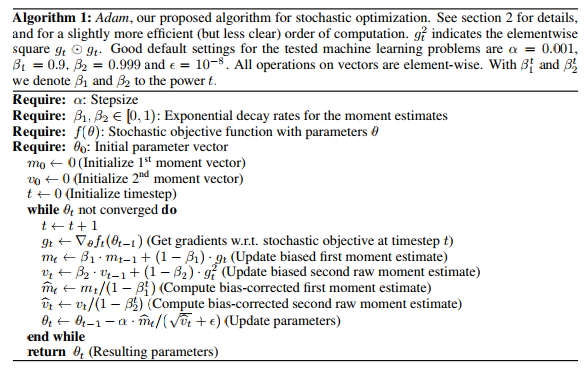


图5 Adam算法实现

针对Training data上性能差的主要优化手段已经介绍完了，大体上就是替换激活函数和调整迭代步长，但具体的细分方法比较多，这两篇博文也只是将主要的方法介绍下，更多的方法可通过相关文献获得。



2017.08.31.