# 机器学习笔记

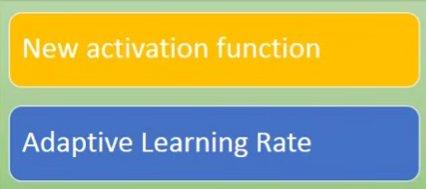
李宏毅教授机器学习(视频地址：https://www.bilibili.com/video/BV1JE411g7XF?p=3&vd\_source=0a27b8c0284e7cd3b72bc0c89478f2d7)

# Tips for Training DNN

1.不是所有不好的performance都源自于overfitting

要先看training data上的情况，如果在training data上layer多的model的错误率本来就比layer少的model的错误率高，那么这个情况就不是overfitting，它本来就training地不好。

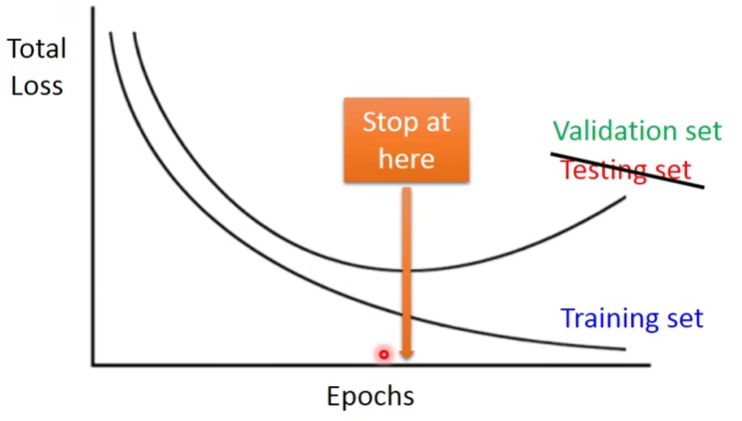
2.在training data上想要得到好的结果

有两种方法可以使得training data上的结果更好，，第一种是设置更有效的function，第二种是调整我们的learning rate，

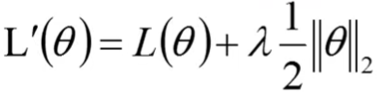
3.在testing data上想要得到好的结果

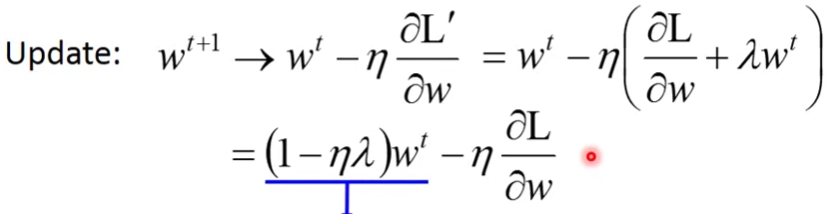


改进testing data上的结果有三种方法，第一种是早停止程序，



如图所示，在黄色箭头指的位置validation set的loss就开始上升了，所以我们应该在这个位置就停止程序。

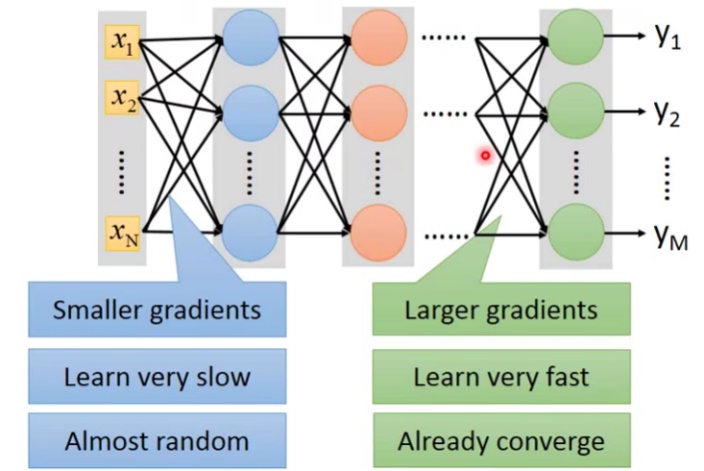
第二种方法就是regularization，令，更新参数的步骤如图所示，



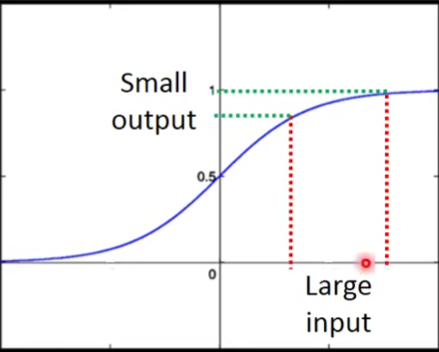
这样做每次更新参数的范围就根据参数而定，参数大的话更新就快，参数小的话更新也跟着变慢。

第三种方法就是dropout，每个神经元有p%的几率被舍弃，舍弃掉部分神经元后整个神经的结构就被改变了，然后我们就拿这个新的神经去进行training。如果在training的时候某个神经元被舍弃的概率是p%，那么在testing的时候所有的weight都要乘上(1-p)%。

4.vanishing gradient problem

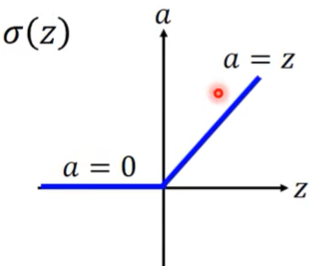


在前面的layer里面，gradient比较小，learning的速度很慢，其output几乎是随机的，而在后面的layer里面，learning的速度很快，它的结果几乎是收敛的，并且取决于前面layer的比较随机的output。



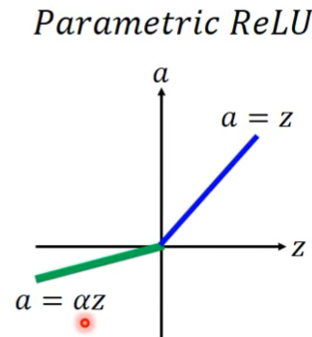
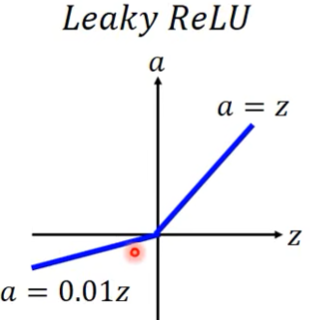
在σ()这个function里面，它会把从负无穷到正无穷的输出压缩在0到1之间，这样会导致比较大的input变化对应的output变化缺比较小，而在经过多次σ()之后，前面layer的input的变化给后面layer带来的影响就越来越小。

对于这个问题有一个改进方法就是用ReLU。其函数图形如图所示



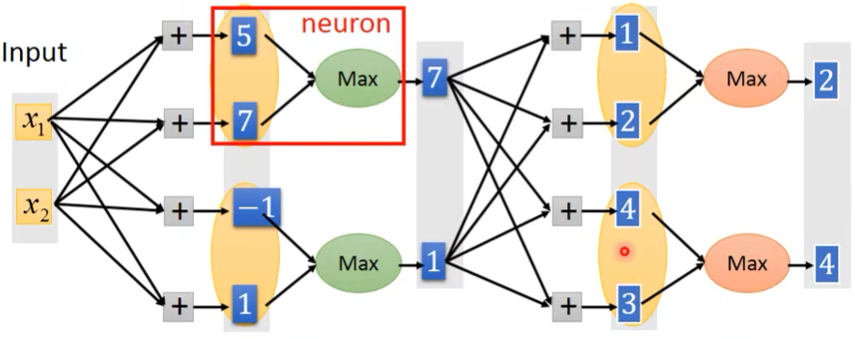
这个函数的速度相比σ()来说很快，且等同于无穷多个σ()叠加的效果。

还有一些这种函数的变式，比如下图，在输入小于0时令输出不完全为0。



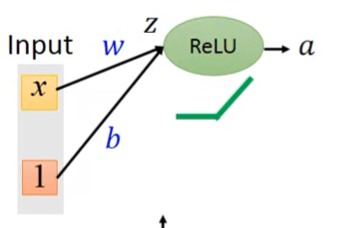
Maxout

ReLU是一种特殊的Maxout。Maxout和maxpooling很像，其操作如图所示



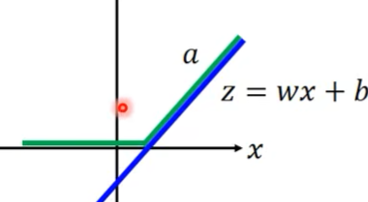
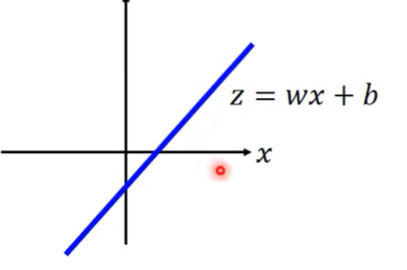
输出时选取最大的进行输出。

为什么说ReLU是一种特殊的Maxout呢，如图所示。



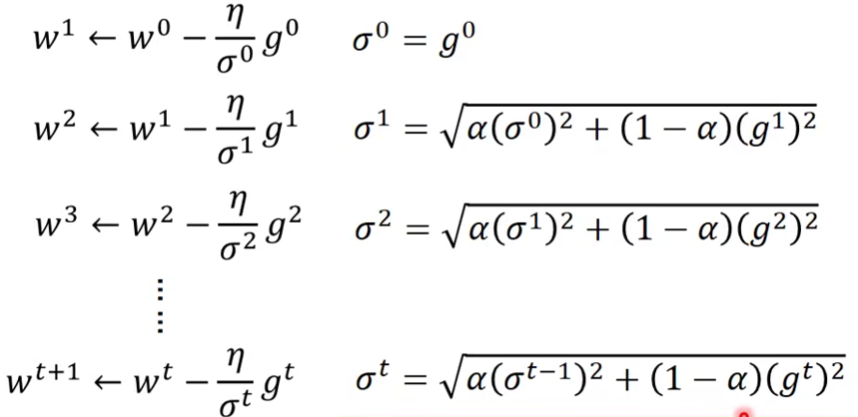
令z=wx+b，a是z通过ReLU之后得到的值。

z的函数就是一条直线的形式，而a就和Maxout一样，在z小于0的时候a的输出是0，z大于0的时候a=z。



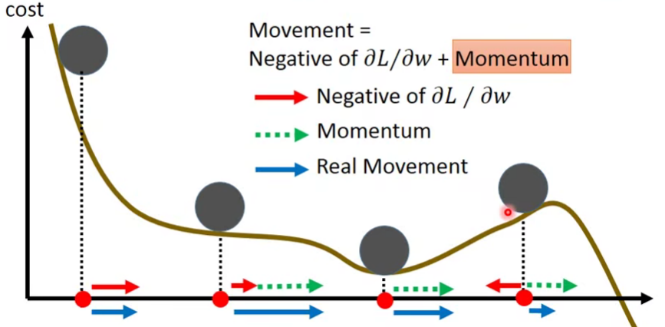
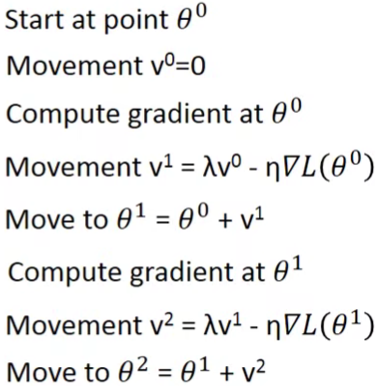
Adagrad的改进——RMSProp

其步骤如图所示，



Gradient descent的改进——加入momentum

做gradient descent的时候我们很容易卡在local minimum的地方，我们用“惯性”的方法就有可能可以跳出这个local minimum，其做法如图所示，movement v1取决于上一步的方向和距离以及当前gradient告诉我们该走的方向和距离。



加入momentum后我们有可能可以跳出local minimum的位置，但如果“惯性”不够的话依然有可能无法跳出。