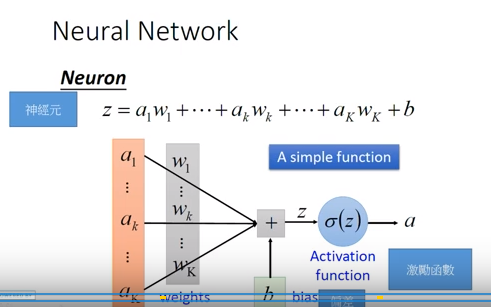
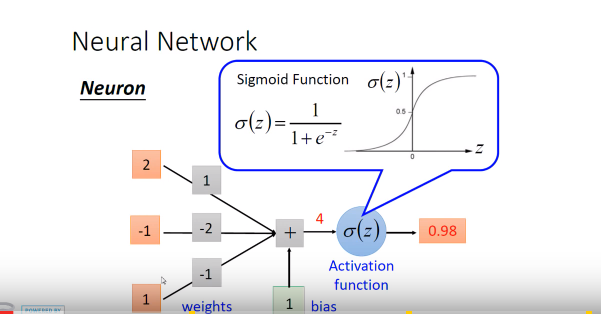
通过Youtube上李宏毅教授的课程对神经网络有了一个全览性的认识

架构一个深度学习网络大概分为三步：建立神经元、设定学习目标、开始学习

可以做到辨识、益智游戏对抗。

一个神经元在做的事情：

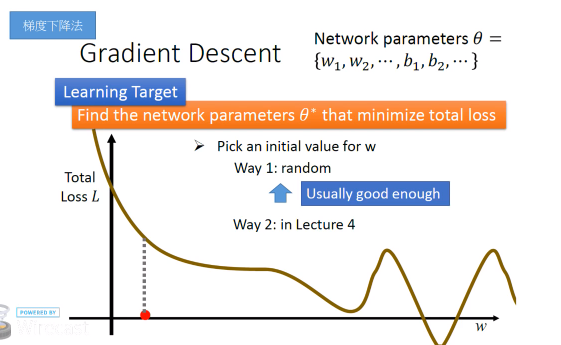




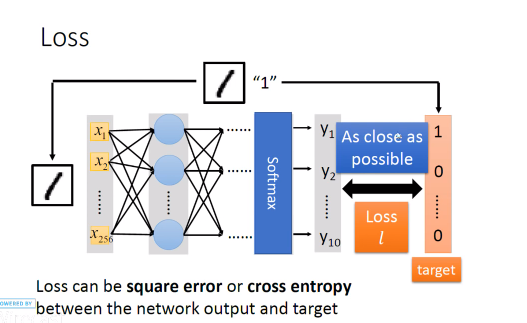
通过单个神经元链接起来进行多次运算，最终得到一个值。

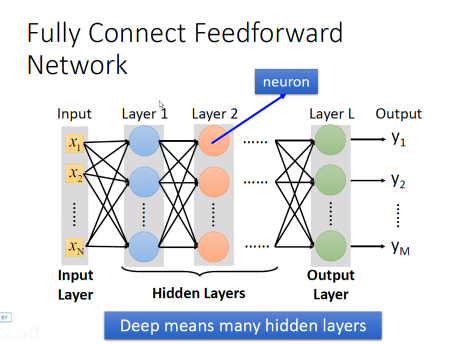
但是由于我们不可能最初就能获得偏差值为0的结果，查找如何计算能让偏差值最小，需要通过计算的修正不断学习。

由于想要获得全局的最佳解在计算上和存储上代价过大，所以现实意义上是不考虑全局最佳解的，以拿到区域最佳解为目的，通过不断的计算获得区域最佳解（Total Lose最小）。

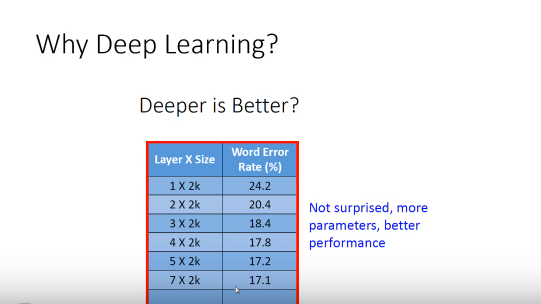


机器学习并不是机器能做到全览性的认知，而是通过算法对目标较优化地进行猜测，得出最大概率的情况。





为什么要深度学习？



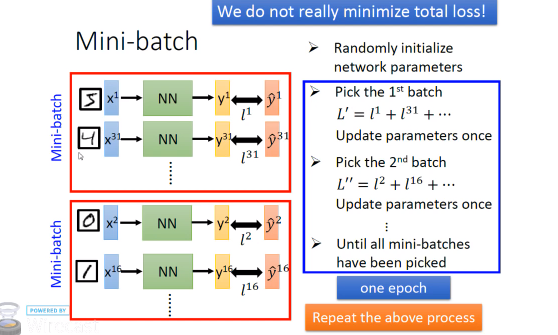
越深错误率越低，层数越深参数越多，准确率越高。

为什么不直接选择层数极限来进行计算？

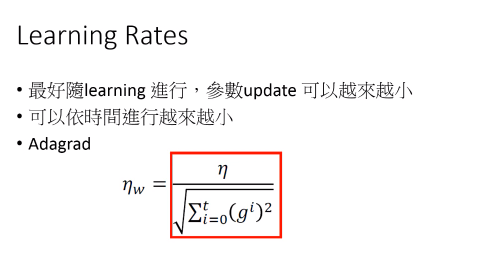
考虑到部分Sample数量较少，通过分步能够将这个情况绕过，避免出现某个参数的Sample数量较少。

Mini-batch

将数据进行分组来进行，能获得较好的准确度和稳定性，但是容易产生单一性。

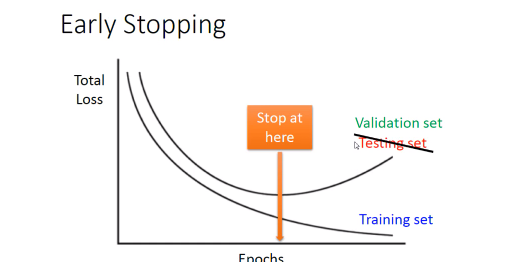


设置一个合适的学习效率是非常重要的，能帮助机器能够快速获得准确率高的结果。



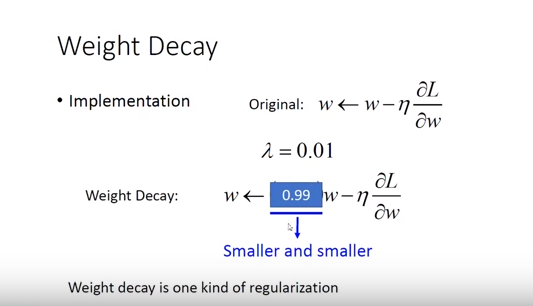
避免Overfitting 由于机器训练时接收的同类型的数据过多，但是测试时数据与训练时有略微差别，那么有可能会出现无法识别或者是误差过大的情况。

通过Early Stopping

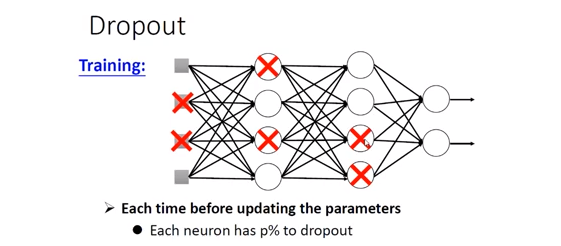


出现过大差距时就停止

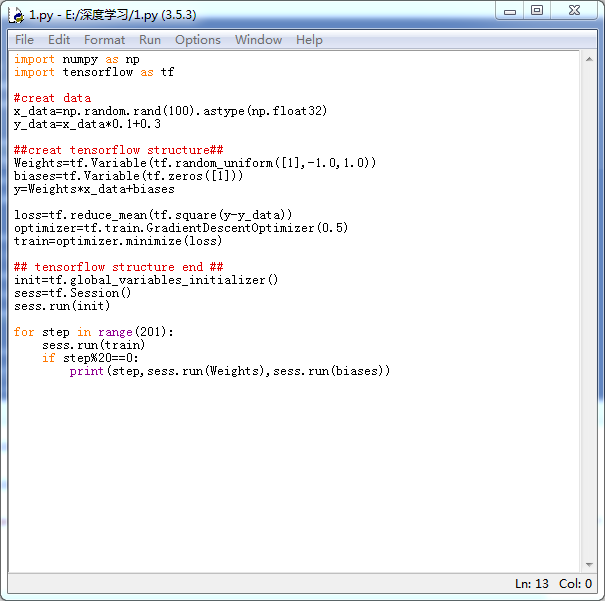
Weight Decay



Dropout



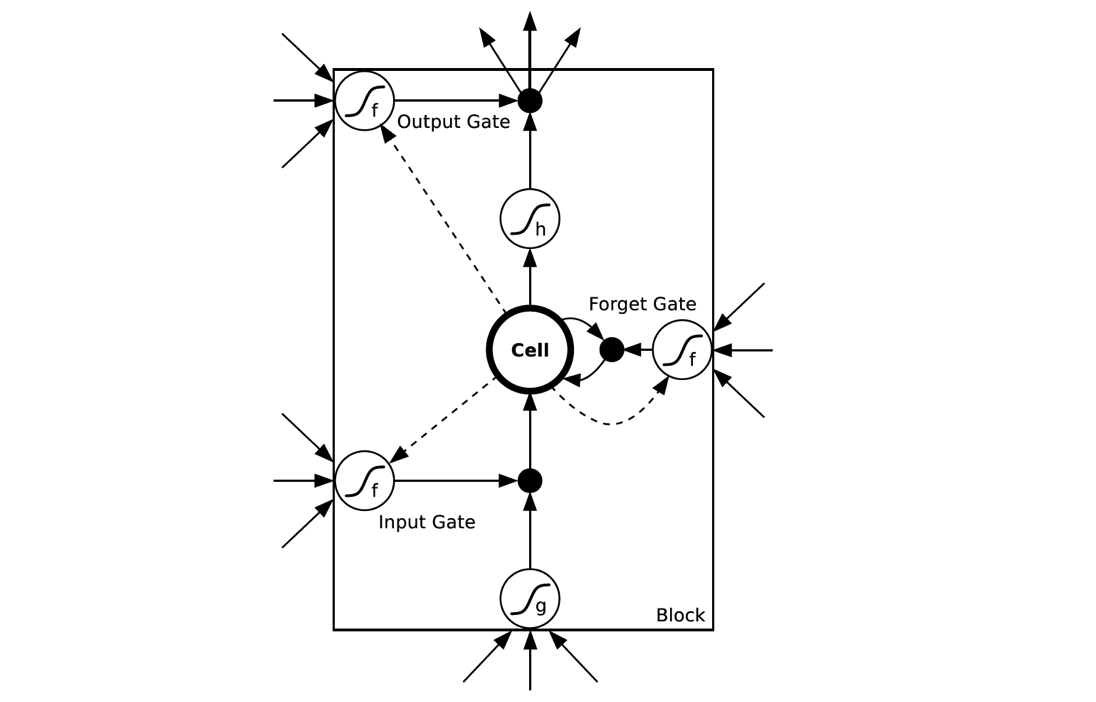
跑了一个神经网络的小程序，大致了解了基本过程。



# LSTM（Long-Short Term Memory）

原生的RNN会遇到一个很大的问题，叫做 The vanishing gradient problem for RNNs，也就是后面时间的节点对于前面时间的节点感知力下降，也就是忘事儿，这也是NN在很长一段时间内不得志的原因，网络一深就没法训练了，**[深度学习](http://lib.csdn.net/base/deeplearning" \o "深度学习知识库" \t "http://blog.csdn.net/Dark_Scope/article/details/_blank)**那一套东西暂且不表，RNN解决这个问题用到的就叫LSTM，简单来说就是你不是忘事儿吗？我给你拿个小本子把事记上，好记性不如烂笔头嘛，所以LSTM引入一个核心元素就是Cell。

与其说LSTM是一种RNN结构，倒不如说LSTM是RNN的一个魔改组件，把上面看到的网络中的小圆圈换成LSTM的block，就是所谓的LSTM了。那它的block长什么样子呢？



怎么这么复杂……不要怕，下文慢慢帮你缕清楚。理解LSTM最方便的就是结合上面这个图，先简单介绍下里面有几个东西：

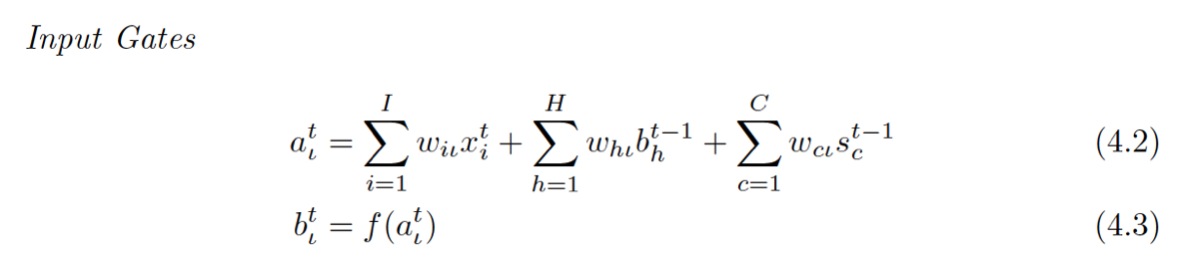
1. Cell，就是我们的小本子，有个叫做state的参数东西来记事儿的
2. Input Gate，Output Gate，在参数输入输出的时候起点作用，算一算东西
3. Forget Gate：不是要记东西吗，咋还要Forget呢。这个没找到为啥就要加入这样一个东西，因为原始的LSTM在这个位置就是一个值1，是连接到下一时间的那个参数，估计是以前的事情记太牢了，最近的就不住就不好了，所以要选择性遗忘一些东西。（没找到解释设置这个东西的动机，还望指正）

在阅读下面公式说明的时候时刻记得这个block上面有一个输出节点，下面有一个输入节点，block只是中间的隐层小圆圈~~~

## 前向传播

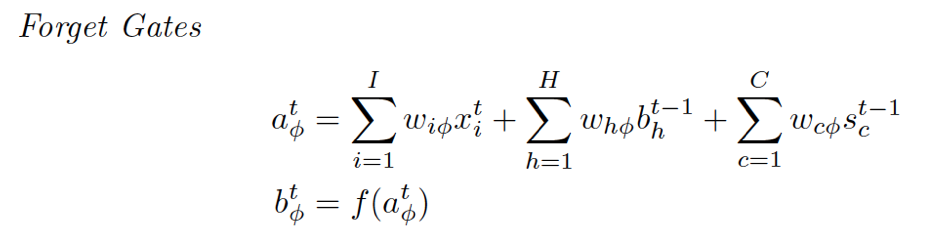
一大波公式正在路上。。。。。公式均来自Alex的论文   
我们按照一般算法的计算顺序来给出每个部分的公式：

### Input Gate



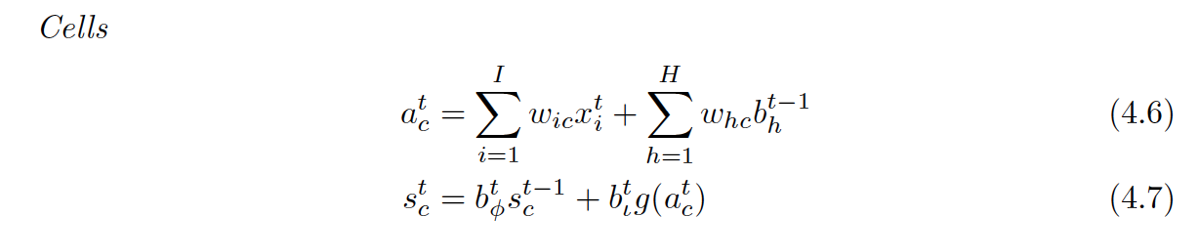
带下标L的就是跟Input Gate相关的，回去看上面那个图，看都有谁连向了Input Gate：外面的输入，来自Cell的那个虚线（虚线叫做peephole连接），这在公式立体现在4.2的第一项和第三项，计算就是普通的累积求和。那中间那个是个什么鬼？   
带H的是一个泛指，因为LSTM的一个重要特点是其灵活性，cell之间可以互联，hidden units之间可以互联，至于连不连都看你（所以你可能在不同地方看到的LSTM公式结构都不一样）所以这个H就是泛指这些连进来的东西，可以看成是从外面连进了的三条边的一部分。   
至于4.3就是简单的激活函数计算而已

### Forget Gate



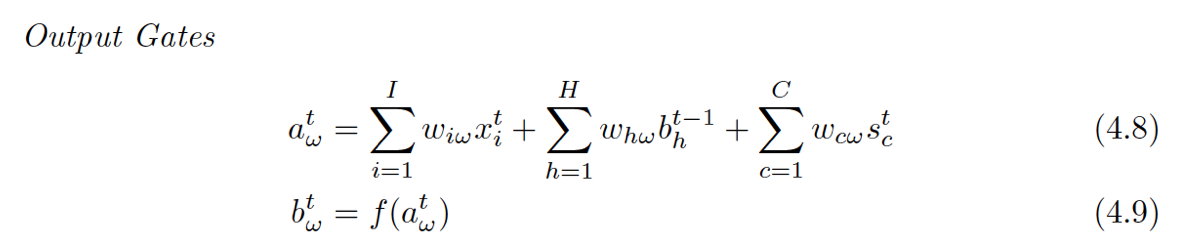
### 再回去看那个图，连到Forget Gate都有哪些：输入层的输入、泛指的输入、来自cell的虚线，这个和Input Gate就是一回事嘛

### Cells



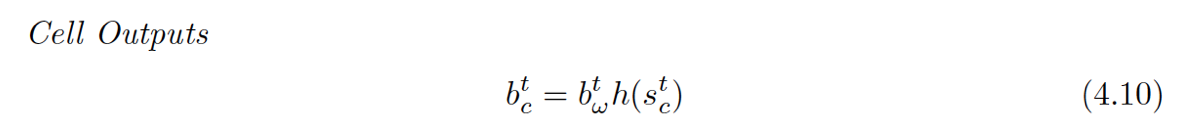
还是老样子，回去看都有啥连到了Cell（这里的cell不是指中间那个Cell，而是最下面那个小圆圈，中间的Cell表示的其实是那个状态值S[c][t]）：输入层的输入，泛指的输入。（这体现在4.6式中）   
再看看中间的那个Cell状态值都有谁连过去了：这次好像不大一样，连过去的都是经过一个小黑点汇合的，从公式也能体现出来，分别是：ForgetGate\*上一时间的状态 + InputGate\*Cell激活后的值

### Output Gate



老样子，看谁连到了Output Gate：跟其他几个Gate好像完全一样嘛~咦，4.8那个S[c][t]为啥是t，以前都是t-1啊。   
这里我也没找到相关的原因，可以理解为在计算OG的时候，S[c][t]已经被计算出来了，所以就不用使用上一时间的状态值了（同样动机不明~~这就是设定好嘛。。。）

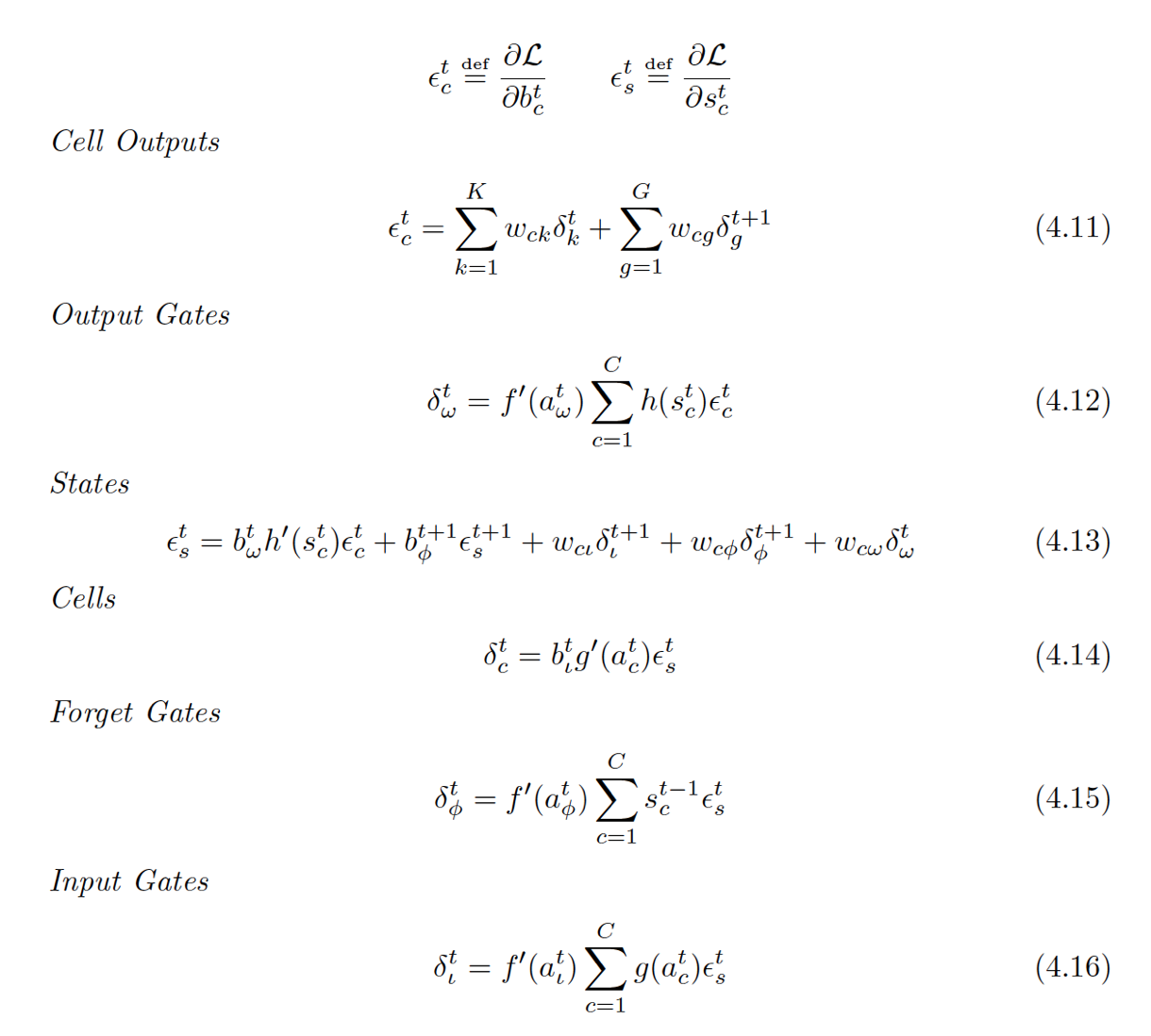
### 最后最后的输出



小黑点，用到了激活后的状态值和Output Gate的结果。   
一定按照图的连接来捋一捋，公式还是非常清晰的。

## 后向传播

又一波公式来袭。。。。。。



这次就只贴公式了，因为要每个都讲一下实在是太费功夫了，记住一个要点就是『看上面的图！！』，看看每个要求偏导的东西都有谁会反向传回东西给它，可以看到最复杂的就是4.13了，因为这是对那个状态值求导，它不光连向了三个门（公式后三项，两个本下一时刻，FG是本时刻的），还连向了最后的输出b[c][t]（公式第一项）以及下一时刻的自己（公式第二项），反向传播公式推导用到的唯一数学工具就是链式法则，你要觉得求偏导看不懂，就把它拆成链看就好了。

还有一点，记得最后的Loss Function是每一时间的一个求和，所以当你算当前层输出层传回来的残差都时候就可以忽略其它东西了，举个例子：4.11是对b[c][t]求偏导，而b[c][t]是正向传播LSTM block的输出，输出到谁了？当前层的输出层，下一层的Hidden Layer，这两个东西的最后的Loss function是分开的，彼此之间没有关系，所以公式里是两部分相加。4.11中的G和之前的H一样，也是泛指，因为它不一定只输出到下一时间的自己，可能还会到下一时间的其他隐层unit，G代表什么纯看你怎么确定的网络结构。