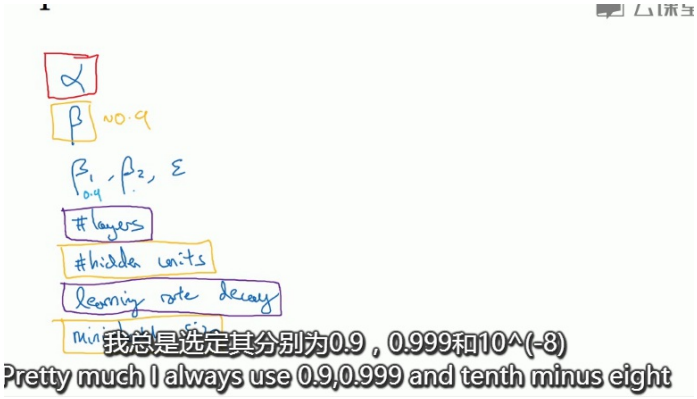
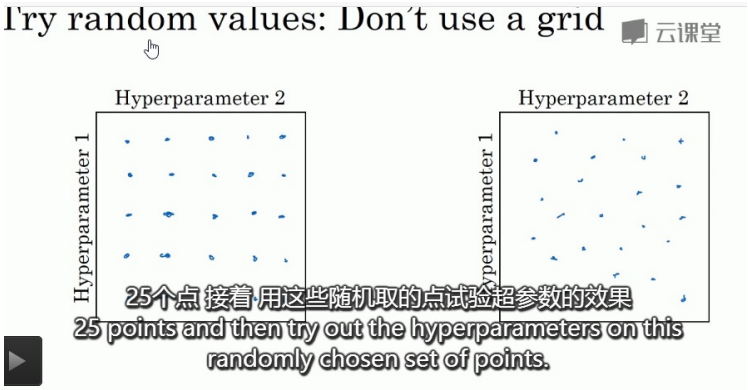
# 第二篇第三周

# 1、调试处理：

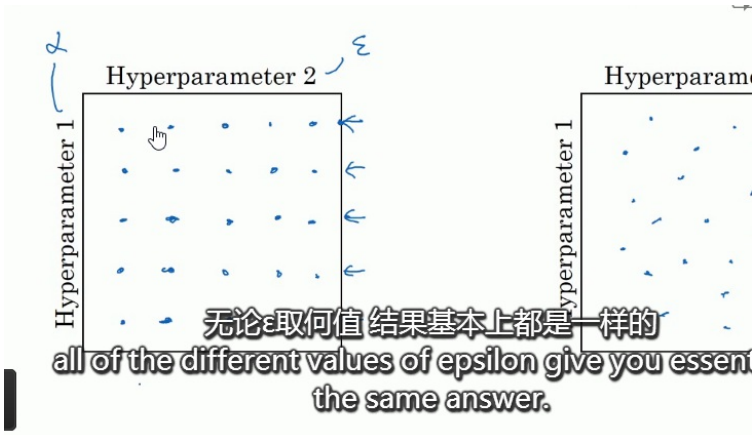
学习率、指数加权平均（0.9）、mini-batch的大小，以确保最优算法运行有效，还要调试隐藏层单元，还有就是层数和学习率的衰减率有时也会产生影响。当使用Adam算法时，吴恩达从不调试β1，β2，ε，分别选择如下：




最重要的参数是学习率α，调节超参数的技巧：

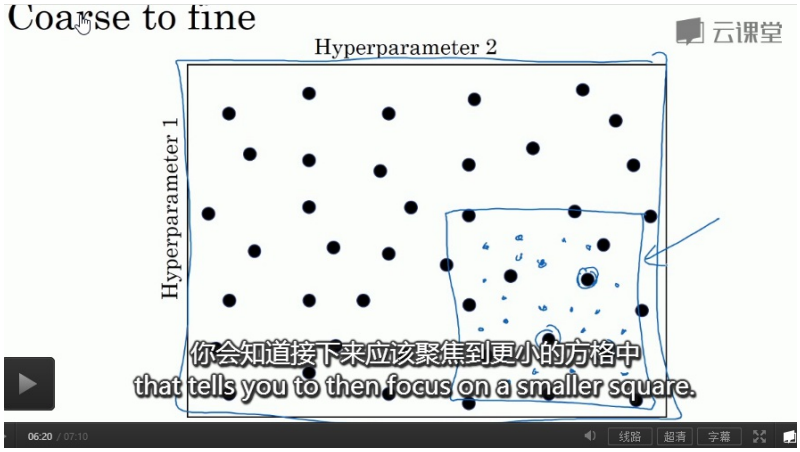



这样做的原因在于，你很难提前知道哪个超参数最重要，正如你之前看到的，一些超参数的确要比其他的更重要，假设超参数1是α学习速率，取一个极端的例子，假设超参数2，是Adam算法中ε的值，这种情况下，α的取值很重要，而ε的取值无关紧要：



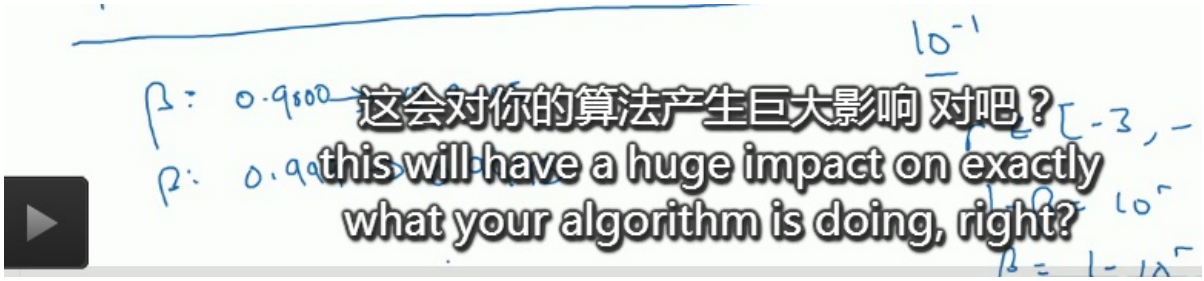

所以你知道有25中模型，但是进行试验的α只有5个，我认为这是很重要的，对比而言，如果你随机取值，你会试验25个独立的α值。

可以先进行粗略的搜索，然后聚焦到更小的方格中，也就是从粗到细的搜索，，

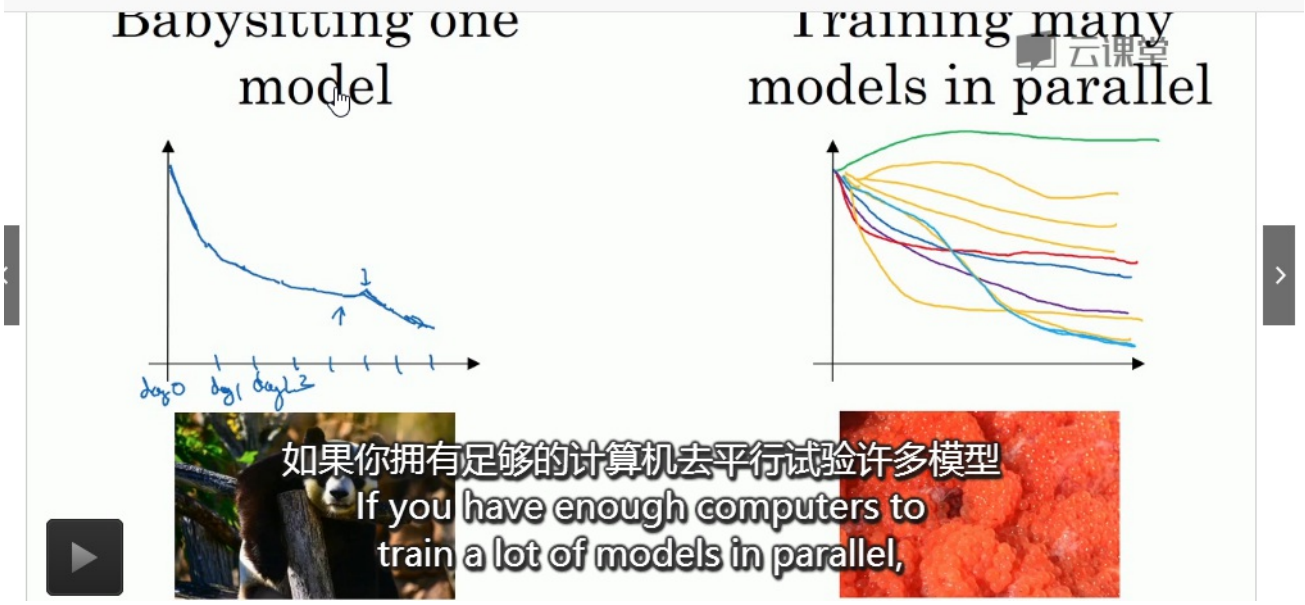


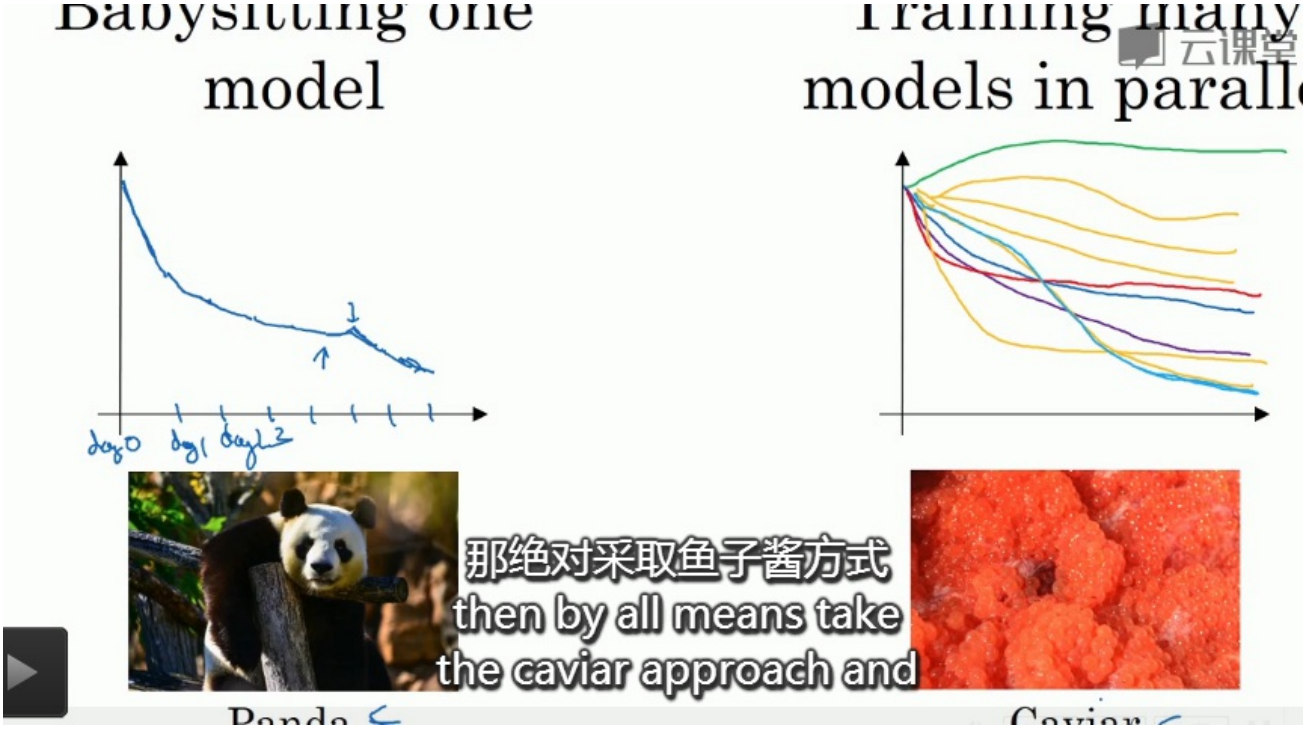

## 2、不采用线性取值的原因：

接近1的时候，一点小小的变化都可能对算法的影响很大。



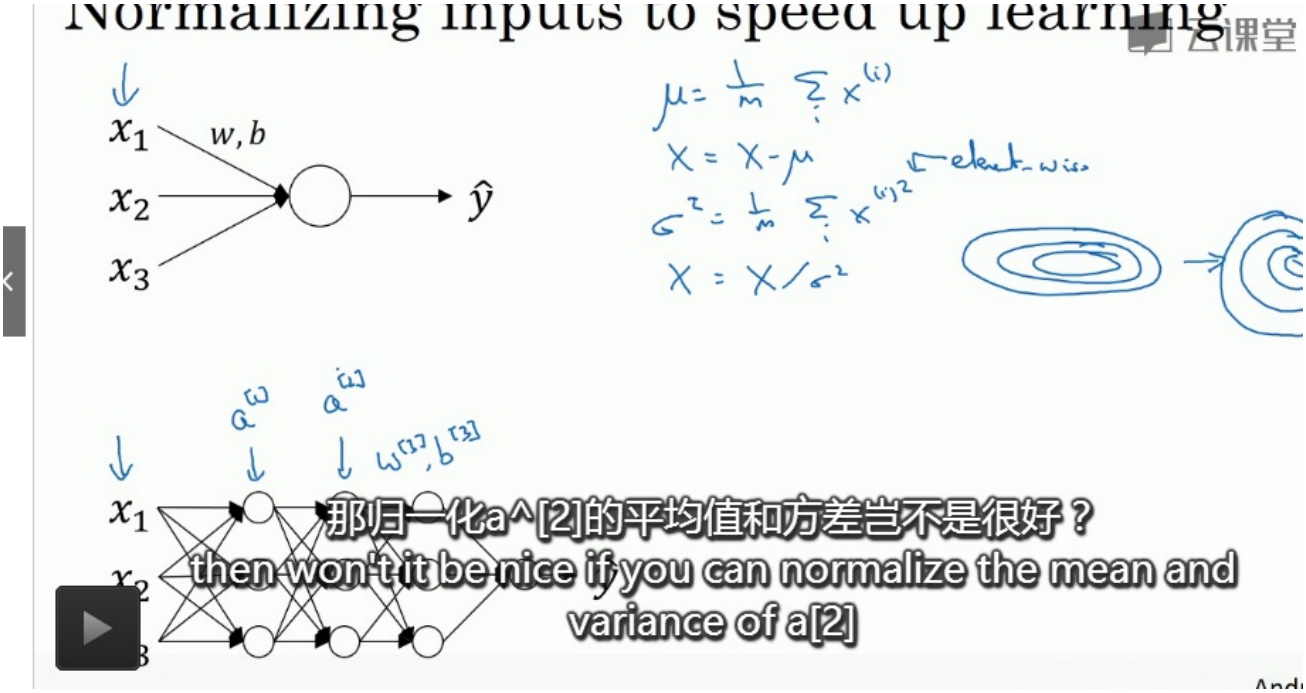

## 3、





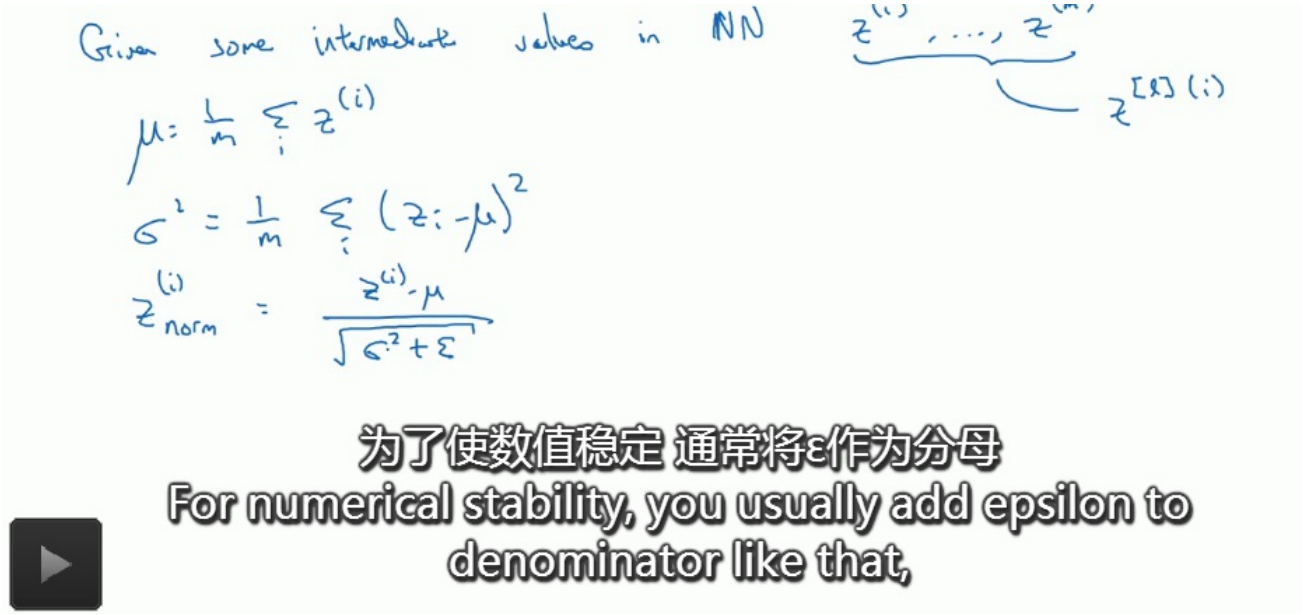

# 4、Batch归一化

batch归一化会使得你的参数搜索变得很容易，使神经网络对超参数的选择更加稳定，超参数的范围会变得更强大，工作效果也很好，也容易使你很容易训练，甚至是深层网络，我们还记得，归一化输入可以加快学习过程，根据方差归一化数据集，在深层神经网络中还可以做如下工作：

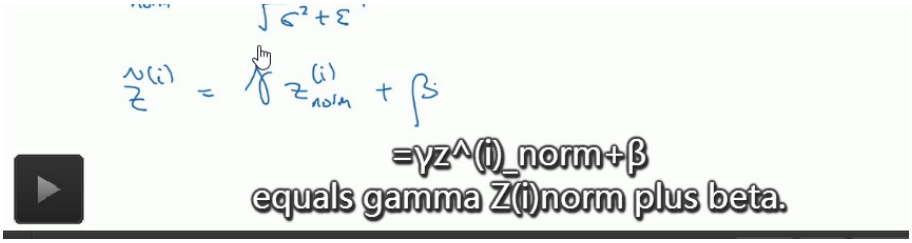


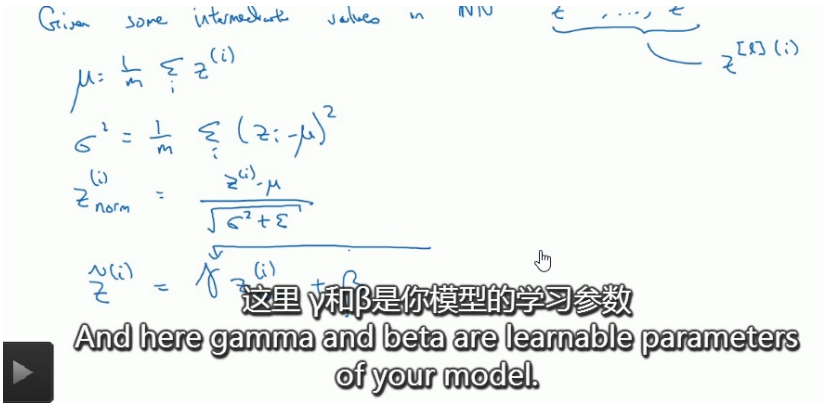

以便使得W[3],b[3]训练更加有意义，会影响W[3],b[3]的计算，这就是batch归一化的作用，严格来说，我们归一化的不是a[2]而是z[2]。

在神经网络中，已知一些中间值，假设你有一些隐藏单元值，从Z[1]到Z[m]，z为隐藏层，先计算所有z层的平均值，然后是所有Z层共同的方差，然后就可以标准化（归一化）每一层了：

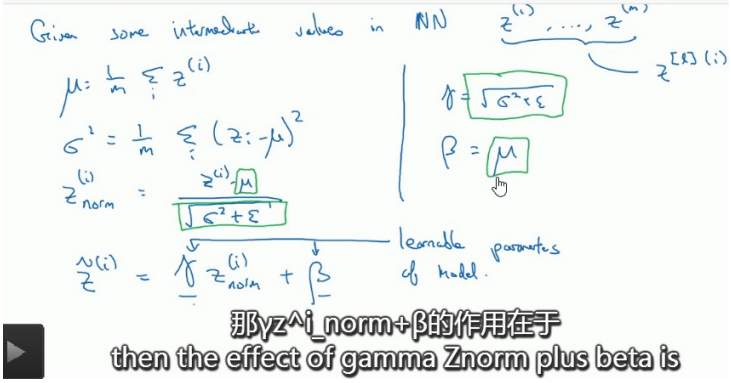


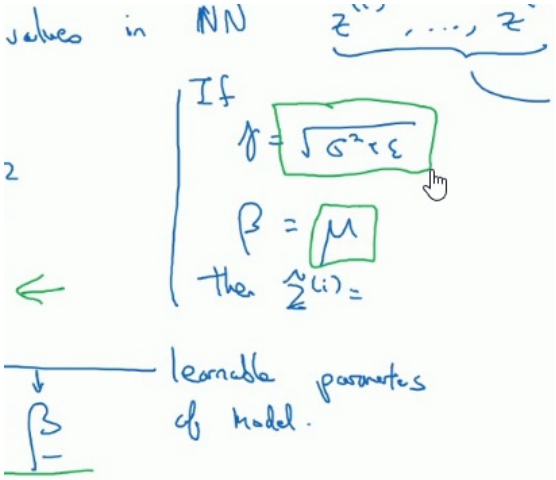

加上ε作为分母，以防止σ为零的情况，所以现在，我们把这些Z值都标准化了，化为含平均值为0和标准单位方差，所以Z的每个分量都含有平均值为0和方差1，但是我们不想让隐藏单元总是含有平均值为0和标准单位方差，也许隐藏单元有了不同的分布会有意义，所以我们需要做的运算就是计算Z^hat:




所以我们使用梯度下降，或者类似梯度下降的算法，你会更新上面那两个参数，正如更新神经网络的权重一样：

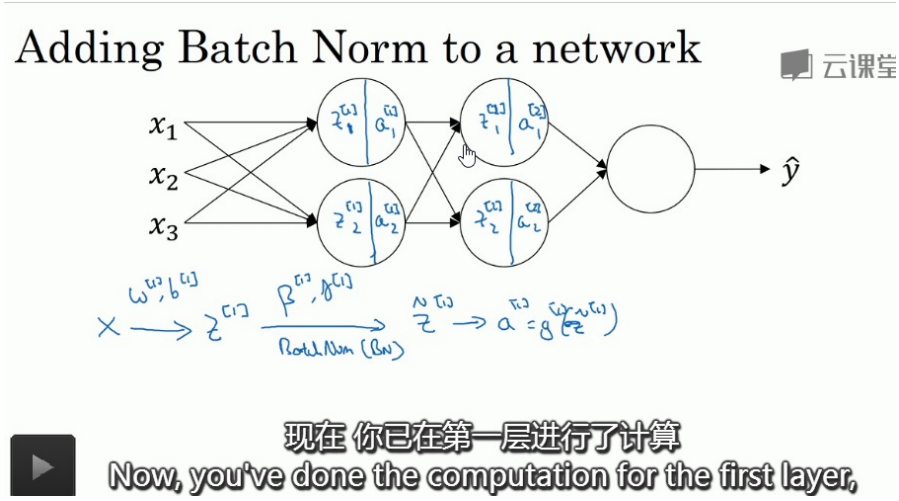




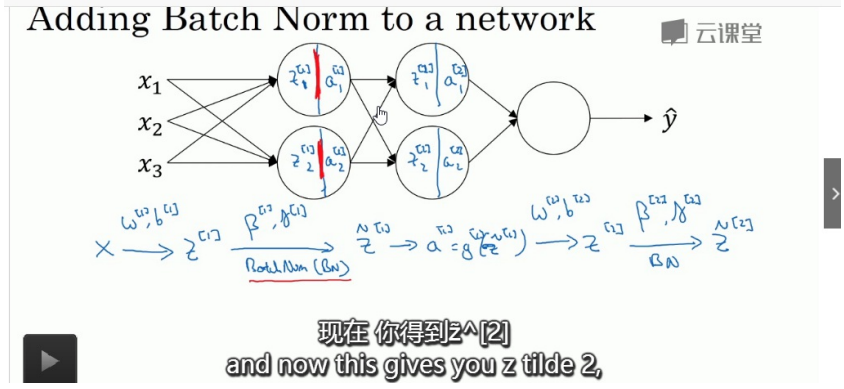

那么z^(i)=z^(i)

batch归一化的作用是它适用的归一化过程不只是输入层，甚至同样适用于神经网络的深度隐藏层。你可以使用batch归一化一些隐藏层单元值中的平均值和方差，不过训练输入和这些隐藏层单元值的一个区别是：你也许不想这些隐藏层单元的平均值为0，方差为1，假设你是使用Sigmoid函数，你并不想使得它落在0附近，也就是比较线性的那块区域，你想充分利用好非线性的Sigmoid函数，而不是使所有的值集中在线性版本中，这就是为什么有了γ和β之后，你可以确保所有的Z[i]的值可以是你想要赋予的任意值。或者他的作用是确保隐藏层单元已使均值和方差标准化，这里均值和方差由两个参数控制，即γ和β，学习算法可以设置为任意值。

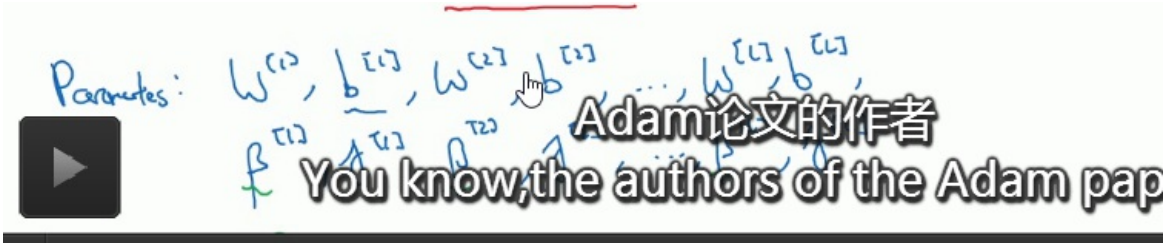
# 5、Batch归一化拟合进神经网络




第一层计算如上图的笔记所示，此时，这项Batch归一化发生在Z的计算和a之间，接下来，你需要应用a[1]值计算Z[2]，此过程是由W[1]和b[1]控制的，与你在第一层所做的类似，你会将Z[2]进行Batch归一化，我们现在简称BN，如下图蓝色笔记：

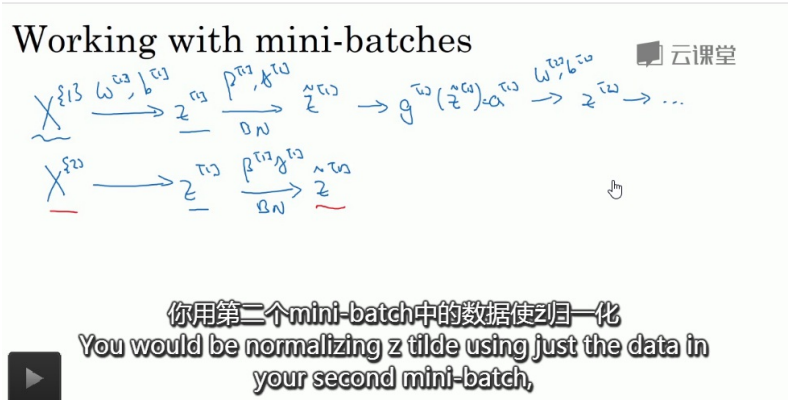



再通过激活函数计算a[2]，所以，需要强调的是，Batch归一化发生在计算Z和a之间，参数如下：

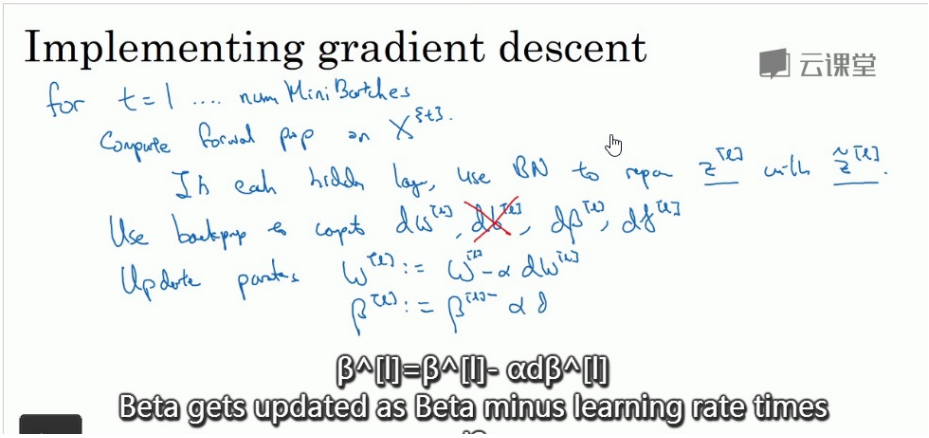



有了参数，你就可以选择你想要的优化算法，比如梯度下降等等，更新参数过程和之前的还是一样子，如果使用深度学习框架，把batch归一化步骤应于batch归一化层，只需要一个函数就解决问题，知道这个过程你就可以更好的理解代码。

Batch归一化经常和mini-batch一起使用：：




过程如下，也是先正向传播，然后反向传播更新参数。：

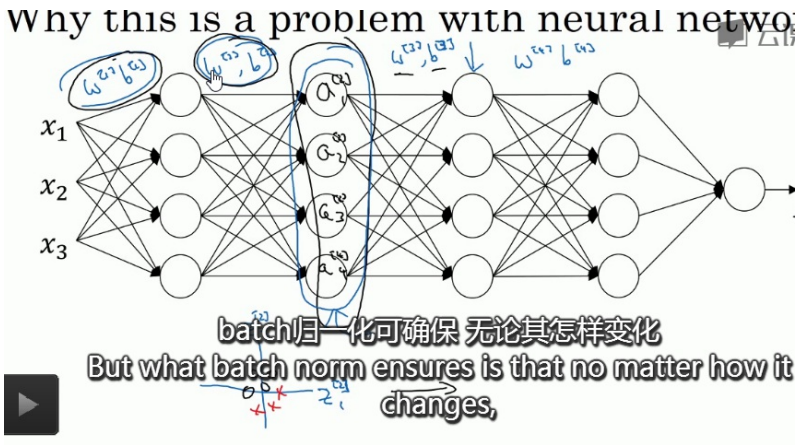


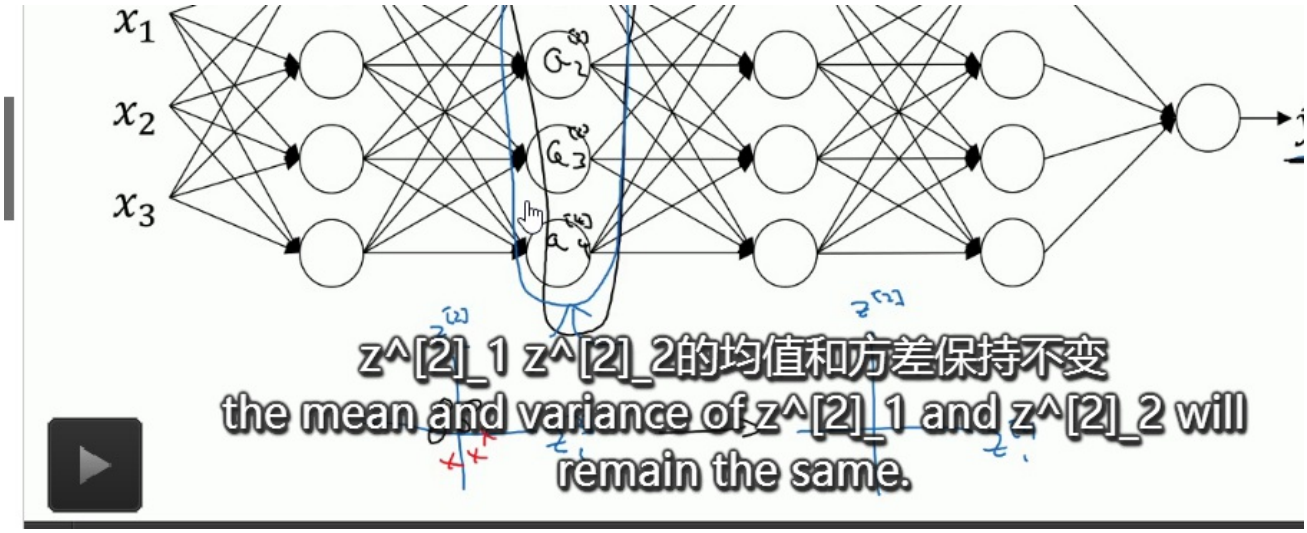

# 6、为什么Batch归一化会其作用：

其实跟前面的归一化原理类似，但是这里不仅仅是对于输入值，还对于隐藏层单元的值，这只是Batch归一化作用的冰山一角。

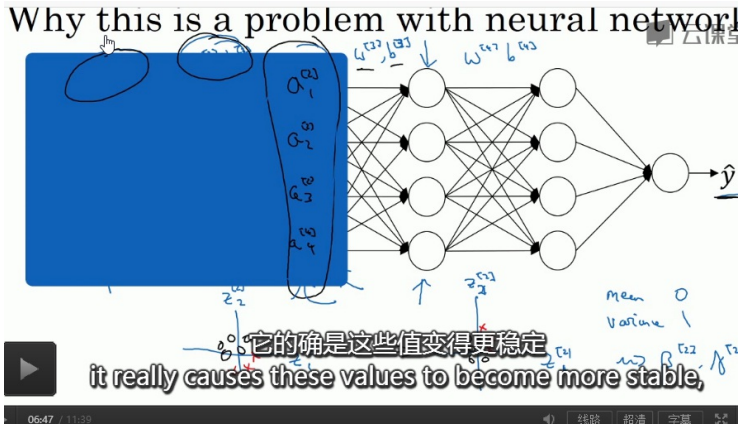
Batch归一化有效的另外一个重要原因是，它可以使权重，比你的网络更滞后或者更深层，比如第十层的权重更加能够经受得住变化，相比于神经网络中前层的权重，比如层一。

batch归一化做的，就是减少这些隐藏层分布变化的数量，如果是绘制这些隐藏层的单元值的分布，Batch归一化要求的是Z[1],z[2]可变：




但是方差和均值保持不变，并不一定是0或者1，也可以是γ和β决定的值，Batch归一化减少了输入值改变的问题：

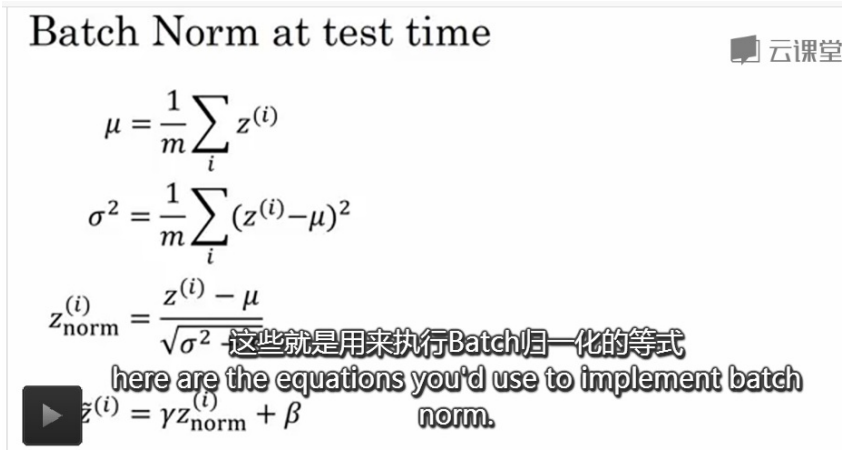



神经网络之后的层就会有更加坚实的基础，即使输入分布改变了一些，它会改变的更少，它做的是，当前层保持学习，当前层改变时，迫使后层，适应的程度变小了，你可以这样想，他减弱了前层参数的作用与后层参数的作用之间的联系，它使得神经网络的每一层都可以自己学习，稍稍独立于其他层，有助于加速整个网络的学习，重点Batch归一化的意思是，尤其从神经网络后一层的角度而言，前层不会左右移动很多，因为它们被同样的均值和方差限制，所以会使后层的学习工作变得更加容易一些。

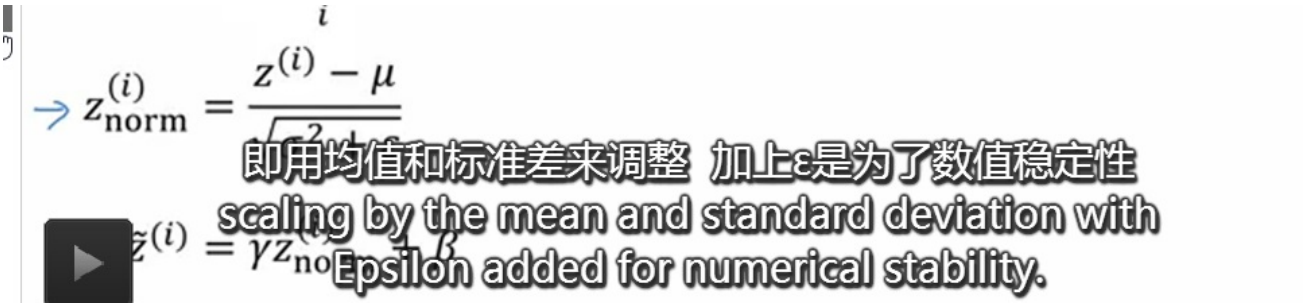
另外一个作用是：它有轻微正则化效果，因为mini-batch并不是在全体数据上，所以会有噪声，这些噪声会传递下去，它有点像dropout，给隐藏层单元添加了噪声，这迫使后面的单元，不过分依赖任何一个隐藏层单元，因为添加的噪声微小，所以并不是巨大的正则化效果，所以有轻微正则化的效果，你可以将batch归一化和dropout一起使用，可以得到dropou更强大的正则化效果，另外，使用更大的mini-batch会减少噪声的存在，也会较小正则化的效果，最后需要注意的是，batch归一化一次只能处理一个mini-batch的数据，它在mini-batch上计算均值和方差。

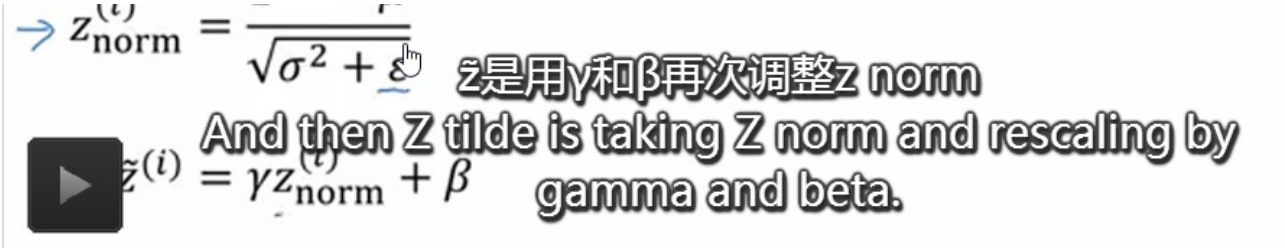
# 7、测试时的Batch归一化：

batch归一化将你的数据以mini-batch的形式逐一处理，但在测试时，你可能需要对每个样本逐一处理：



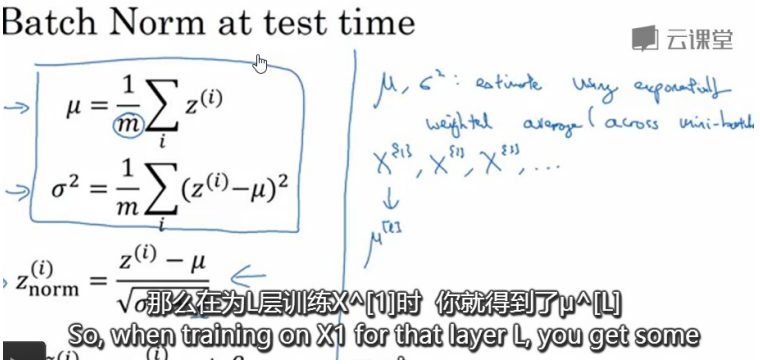

在一个mini-batch中，你将mini-batch求和，计算均值，所以这里你只要把一个mini-batch的样本都加起来，然后计算方差，Znorm:



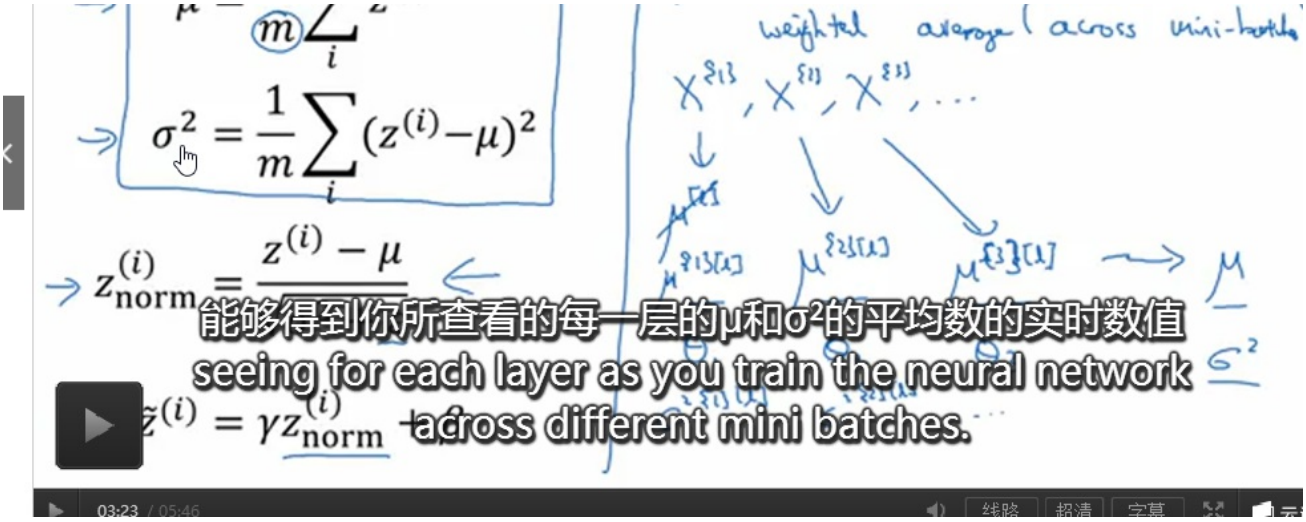




注意：σ2和μ是在整个mini-batch上进行计算的。

测试时，你需要单独计算σ2和μ，你需要用一个指数加权平均来估算，这个平均数涵盖了所有的mini-batch：




每一个batch都得到一个μ，你对某一层隐藏层Z均值的估值，同样的你可以用指数加权平均来追踪，你在这一层的第一个mini-batch中所见到的σ2的值，第二个mini-batch中所见到的σ2的值：



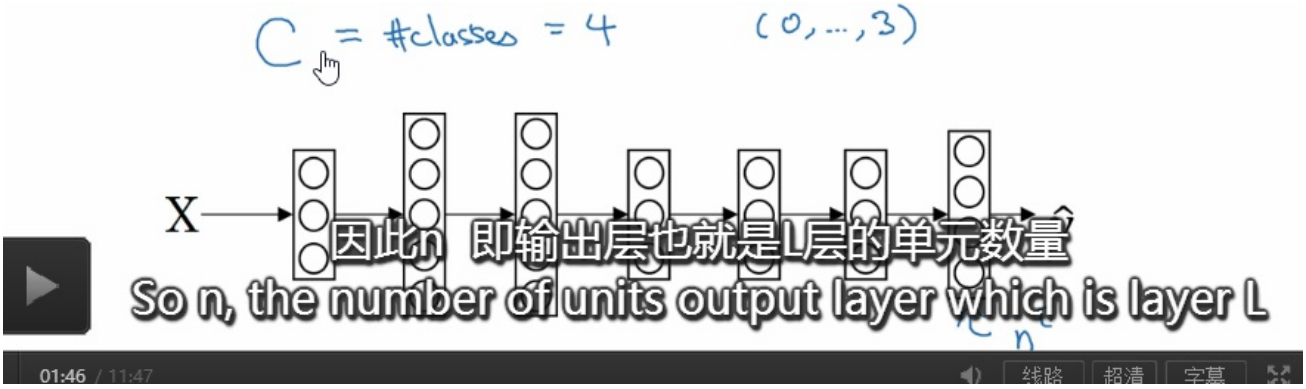

测试的时候，σ2和μ用指数加权平均就可以了。

总结一下：在训练时，σ2和μ是在整个mini-batch上计算出来的，包含了一定数量的样本，但是在测试时，你可能需要逐一处理样本，方法是根据你的训练集估算σ2和μ，估算的方法有很多，你可以在最终的网络中运行整个训练集，来得到σ2和μ，但是在实际应用中，我们通常应用指数加权平均，来追踪在训练过程中你看到的σ2和μ的值，然后用测试中σ2和μ的值，来进行你需要的隐藏层单元Z值的调整，如果你用的是深度学习框架，会有默认的方式估计σ2和μ的值，使用batch归一化，你可以训练更深的网络，让你的学习算法运行的更快。

# 8、Softmax回归

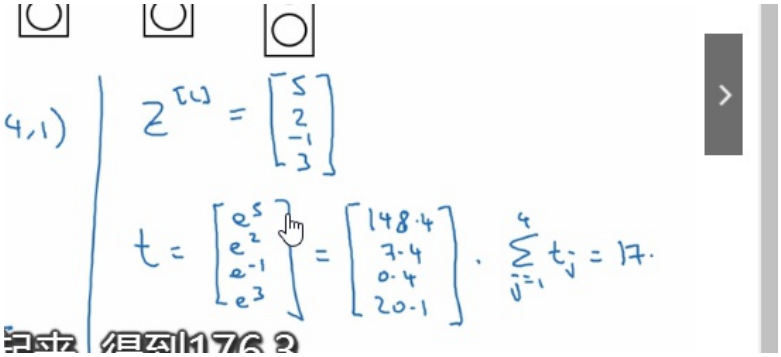
可以用于进行多分类问题，例子：


如果不属于上面的某一类，我们把它归为其他类别，或者叫做类0，用C表示你的输入会被分入的类别总个数，这个例子C=4，（0,1,2,3）在这个例子中，我们建立一个神经网络，其输出层为4：

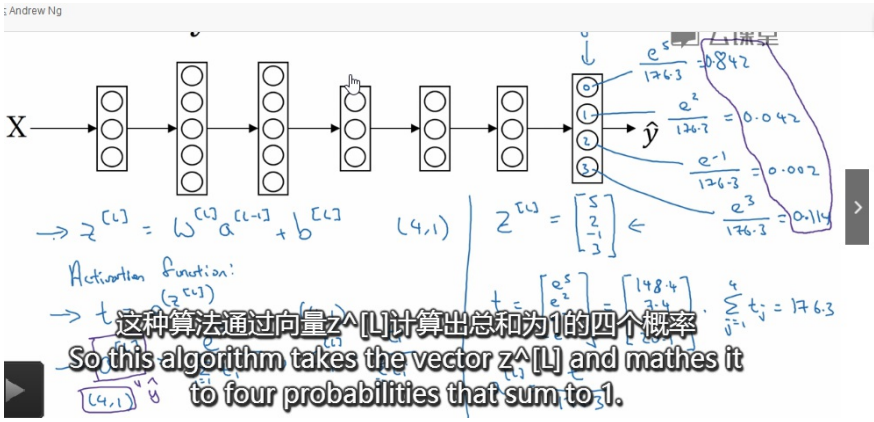



每一个输出层代表的是某一个类别的概率，在这个例子中，是一个4*1的向量，向量里面的数值加起来是

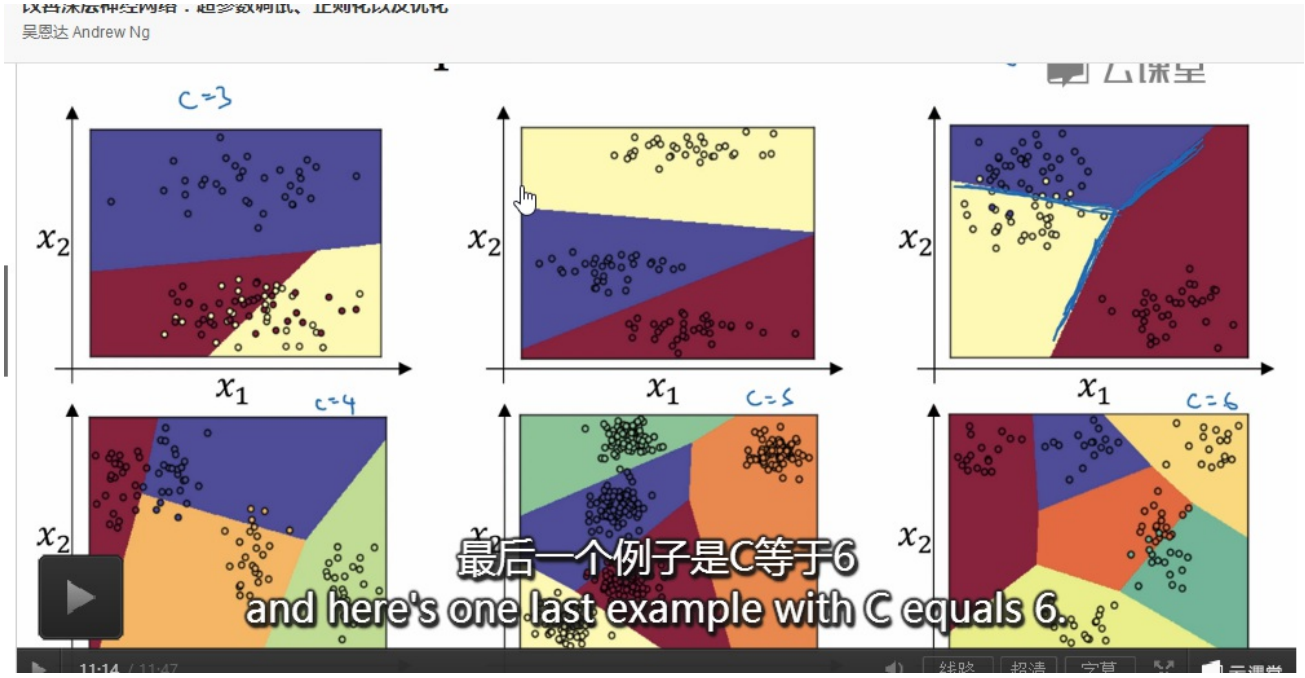
1，你用softmax,以及输出层来生成输出：



然后归一化。得到向量的总和为1,：



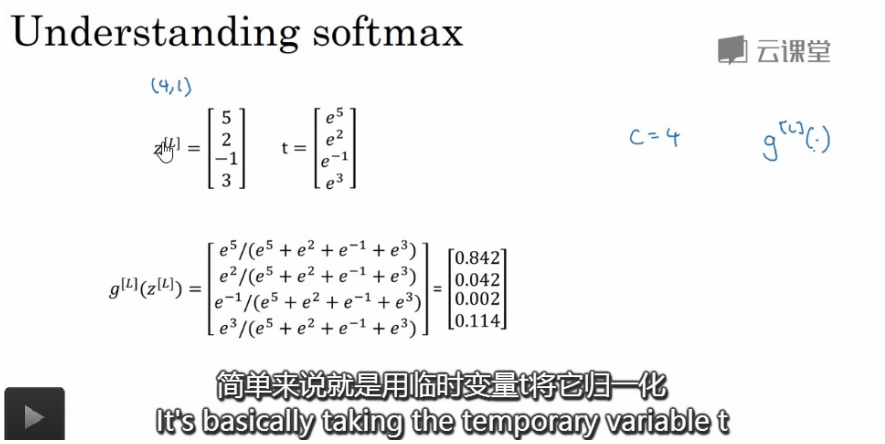

在没有隐藏层的情况下决策面是线性的：



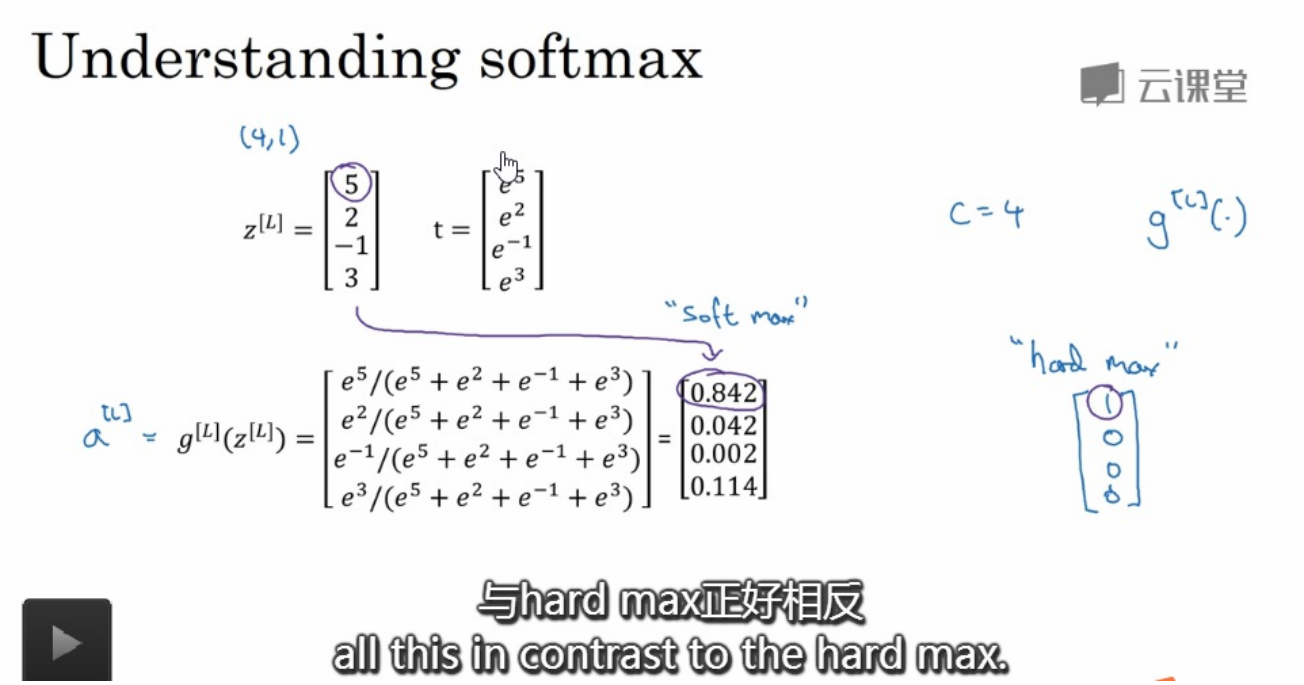

有隐藏层，就可以学习更复杂的非线性决策边界，来区分多种不同的分类，

# 9、训练一个softmax分类器

上一层的例子：

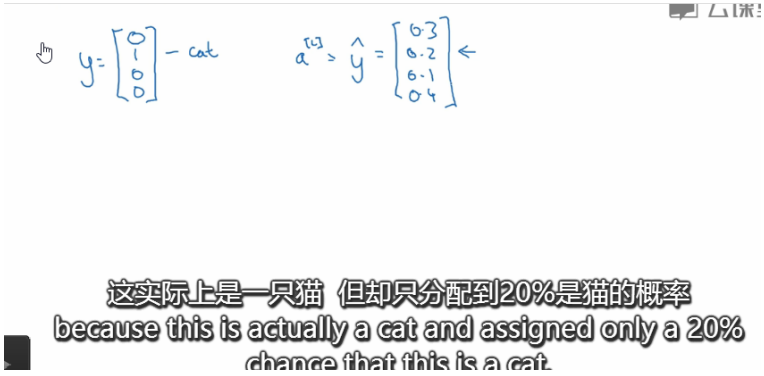



hard max是将最大的设为1，其他设置为0，

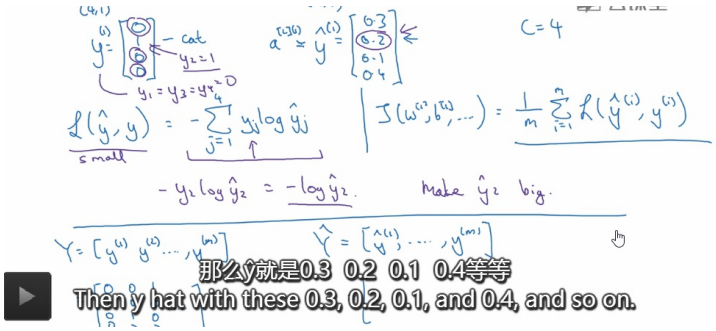



怎么训练代带softmax输出层的神经网络：

1、先定义损失函数。




所以这是个不佳的分类器了。



# 10、Tensorflow：