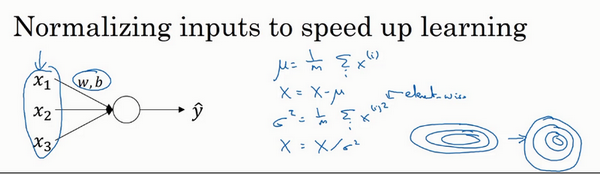
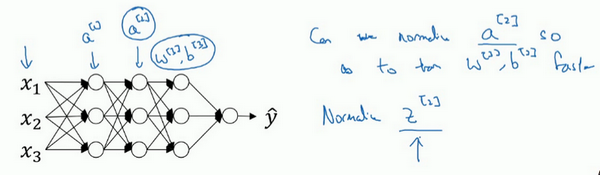
### 3.4 归一化网络的激活函数（Normalizing activations in a network）

在深度学习兴起后，最重要的一个思想是它的一种算法，叫做**Batch**归一化，由**Sergey loffe**和**Christian Szegedy**两位研究者创造。**Batch**归一化会使你的参数搜索问题变得很容易，使神经网络对超参数的选择更加稳定，超参数的范围会更加庞大，工作效果也很好，也会是你的训练更加容易，甚至是深层网络。让我们来看看**Batch**归一化是怎么起作用的吧。



当训练一个模型，比如**logistic**回归时，你也许会记得，归一化输入特征可以加快学习过程。你计算了平均值，从训练集中减去平均值，计算了方差，接着根据方差归一化你的数据集，在之前的视频中我们看到，这是如何把学习问题的轮廓，从很长的东西，变成更圆的东西，更易于算法优化。所以这是有效的，对**logistic**回归和神经网络的归一化输入特征值而言。



那么更深的模型呢？你不仅输入了特征值，而且这层有激活值，这层有激活值等等。如果你想训练这些参数，比如，，那归一化的平均值和方差岂不是很好？以便使，的训练更有效率。在**logistic**回归的例子中，我们看到了如何归一化，，，会帮助你更有效的训练和。

所以问题来了，对任何一个隐藏层而言，我们能否归一化值，在此例中，比如说的值，但可以是任何隐藏层的，以更快的速度训练，，因为是下一层的输入值，所以就会影响，的训练。简单来说，这就是**Batch**归一化的作用。尽管严格来说，我们真正归一化的不是，而是，深度学习文献中有一些争论，关于在激活函数之前是否应该将值归一化，或是否应该在应用激活函数后再规范值。**实践中，经常做的是归一化，所以这就是我介绍的版本，我推荐其为默认选择，那下面就是Batch归一化的使用方法。**



在神经网络中，已知一些中间值，假设你有一些隐藏单元值，从到，这些来源于隐藏层，所以这样写会更准确，即为隐藏层，从1到，但这样书写，我要省略及方括号，以便简化这一行的符号。所以已知这些值，如下，你要计算平均值，强调一下，所有这些都是针对层，但我省略及方括号，然后用正如你常用的那个公式计算方差，接着，你会取每个值，使其规范化，方法如下，减去均值再除以标准偏差，为了使数值稳定，通常将作为分母，以防的情况。

图片包含 文字, 白板

描述已自动生成

所以现在我们已把这些值标准化，化为含平均值0和标准单位方差，所以的每一个分量都含有平均值0和方差1，但我们不想让隐藏单元总是含有平均值0和方差1，也许隐藏单元有了不同的分布会有意义，所以我们所要做的就是计算，**我们称之为，，这里和是你模型的学习参数**，所以我们使用梯度下降或一些其它类似梯度下降的算法，比如**Momentum**或者**Nesterov**，**Adam**，你会更新和，正如更新神经网络的权重一样。

图片包含 物体

描述已自动生成

请注意和的作用是，你可以随意设置的平均值，事实上，如果，如果等于这个分母项（中的分母），等于，这里的这个值是中的，那么的作用在于，它会精确转化这个方程，如果这些成立（），那么。

通过对和合理设定，规范化过程，即这四个等式，从根本来说，只是计算恒等函数，通过赋予和其它值，可以使你构造含其它平均值和方差的隐藏单元值。

图片包含 文字, 白板

描述已自动生成

所以，在网络匹配这个单元的方式，之前可能是用，等等，现在则会用取代，方便神经网络中的后续计算。如果你想放回，以清楚的表明它位于哪层，你可以把它放这。

所以我希望你学到的是，归一化输入特征是怎样有助于神经网络中的学习，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中的深度隐藏层。你应用Batch归一化了一些隐藏单元值中的平均值和方差，不过训练输入和这些隐藏单元值的一个区别是，你也许不想隐藏单元值必须是平均值0和方差1。

图片包含 文字, 天空

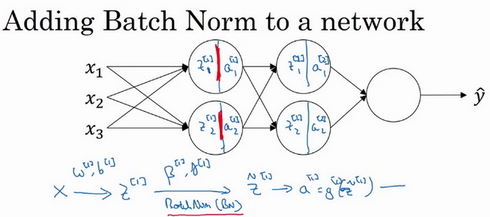
描述已自动生成

比如，如果你有**sigmoid**激活函数，你不想让你的值总是全部集中在这里，你想使它们有更大的方差，或不是0的平均值，以便更好的利用非线性的**sigmoid**函数，而不是使所有的值都集中于这个线性版本中，这就是为什么有了和两个参数后，你可以确保所有的值可以是你想赋予的任意值，或者它的作用是保证隐藏的单元已使均值和方差标准化。那里，均值和方差由两参数控制，即和，学习算法可以设置为任何值，所以它真正的作用是，使隐藏单元值的均值和方差标准化，即有固定的均值和方差，均值和方差可以是0和1，也可以是其它值，它是由和两参数控制的。

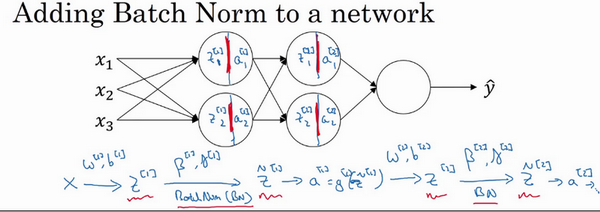
我希望你能学会怎样使用**Batch**归一化，至少就神经网络的单一层而言，在下一个视频中，我会教你如何将**Batch**归一化与神经网络甚至是深度神经网络相匹配。对于神经网络许多不同层而言，又该如何使它适用，之后，我会告诉你，**Batch**归一化有助于训练神经网络的原因。所以如果觉得**Batch**归一化起作用的原因还显得有点神秘，那跟着我走，在接下来的两个视频中，我们会弄清楚。

### 3.5 将 Batch Norm 拟合进神经网络（Fitting Batch Norm into a neural network）

你已经看到那些等式，它可以在单一隐藏层进行**Batch**归一化，接下来，让我们看看它是怎样在深度网络训练中拟合的吧。



假设你有一个这样的神经网络，我之前说过，你可以认为每个单元负责计算两件事。第一，它先计算，然后应用其到激活函数中再计算，所以我可以认为，每个圆圈代表着两步的计算过程。同样的，对于下一层而言，那就是和等。所以如果你没有应用**Batch**归一化，你会把输入拟合到第一隐藏层，然后首先计算，这是由和两个参数控制的。接着，通常而言，你会把拟合到激活函数以计算。**但Batch归一化的做法是将值进行Batch归一化，简称BN，此过程将由和两参数控制，这一操作会给你一个新的规范化的值（），然后将其输入激活函数中得到，即。**



现在，你已在第一层进行了计算，此时**Batch**归一化发生在的计算和之间，接下来，你需要应用值来计算，此过程是由和控制的。与你在第一层所做的类似，你**会将进行Batch归一化，现在我们简称BN，这是由下一层的Batch归一化参数所管制的，即和，现在你得到，再通过激活函数计算出等等。**

所以需要强调的是**Batch**归一化是发生在计算和之间的。直觉就是，**与其应用没有归一化的值，不如用归一过的，这是第一层（）。**第二层同理，与其应用没有规范过的值，不如用经过方差和均值归一后的。所以，你网络的参数就会是，，和等等，我们将要去掉这些参数。但现在，想象参数，到，，我们将另一些参数加入到此新网络中，，，等等。对于应用**Batch**归一化的每一层而言。需要澄清的是，请注意，这里的这些（，等等）和超参数没有任何关系，下一张幻灯片中会解释原因，后者是用于**Momentum**或计算各个指数的加权平均值。**Adam**论文的作者，在论文里用代表超参数。**Batch**归一化论文的作者，则使用代表此参数（，等等），但这是两个完全不同的。我在两种情况下都决定使用，以便你阅读那些原创的论文，但**Batch**归一化学习参数，等等和用于**Momentum**、**Adam**、**RMSprop**算法中的不同。

图片包含 文字, 白板

描述已自动生成

所以现在，这是你算法的新参数，接下来你可以使用想用的任何一种优化算法，比如使用梯度下降法来执行它。

举个例子，对于给定层，你会计算，接着更新参数为。你也可以使用**Adam**或**RMSprop**或**Momentum**，以更新参数和，并不是只应用梯度下降法。

即使在之前的视频中，我已经解释过**Batch**归一化是怎么操作的，计算均值和方差，减去均值，再除以方差，如果它们使用的是深度学习编程框架，通常你不必自己把**Batch**归一化步骤应用于**Batch**归一化层。因此，探究框架，可写成一行代码，比如说，在**TensorFlow**框架中，你可以用这个函数**（tf.nn.batch\_normalization）**来实现**Batch**归一化，我们稍后讲解，但实践中，你不必自己操作所有这些具体的细节，但知道它是如何作用的，你可以更好的理解代码的作用。但在深度学习框架中，**Batch**归一化的过程，经常是类似一行代码的东西。

所以，到目前为止，我们已经讲了**Batch**归一化，就像你在整个训练站点上训练一样，或就像你正在使用**Batch**梯度下降法。

图片包含 物体

描述已自动生成

实践中，**Batch归一化通常和训练集的mini-batch一起使用**。你应用**Batch**归一化的方式就是，你用第一个**mini-batch**()，然后计算，这和上张幻灯片上我们所做的一样，应用参数和，使用这个**mini-batch**()。接着，继续第二个**mini-batch**()，接着**Batch**归一化会减去均值，除以标准差，由和重新缩放，这样就得到了，而所有的这些都是在第一个**mini-batch**的基础上，你再应用激活函数得到。然后用和计算，等等，**所以你做的这一切都是为了在第一个mini-batch()上进行一步梯度下降法。**

图片包含 设备

描述已自动生成

类似的工作，你会在第二个**mini-batch**（）上计算，然后用**Batch**归一化来计算，所以**Batch**归一化的此步中，你用第二个**mini-batch**（）中的数据使归一化，这里的**Batch**归一化步骤也是如此，让我们来看看在第二个**mini-batch**（）中的例子，在**mini-batch**上计算的均值和方差，重新缩放的和得到，等等。



然后在第三个**mini-batch**（）上同样这样做，继续训练。

**现在，我想澄清此参数的一个细节。先前我说过每层的参数是和，还有和，请注意计算的方式如下，，但Batch归一化做的是，它要看这个mini-batch，先将归一化，结果为均值0和标准方差，再由和重缩放，但这意味着，无论的值是多少，都是要被减去的，因为在Batch归一化的过程中，你要计算的均值，再减去平均值，在此例中的mini-batch中增加任何常数，数值都不会改变，因为加上的任何常数都将会被均值减去所抵消。**



所以，如果你在使用**Batch**归一化，其实你可以消除这个参数（），或者你也可以，暂时把它设置为0，那么，参数变成，然后你计算归一化的，，你最后会用参数，以便决定的取值，这就是原因。

图片包含 设备

描述已自动生成

所以总结一下，因为**Batch**归一化超过了此层的均值，这个参数没有意义，所以，你必须去掉它，由代替，这是个控制参数，会影响转移或偏置条件。

最后，请记住**的维数，因为在这个例子中，维数会是**，**的尺寸为**，如果是l层隐藏单元的数量，**那和的维度也是**，因为这是你隐藏层的数量，你有隐藏单元，所以和用来将每个隐藏层的均值和方差缩放为网络想要的值。

图片包含 设备

描述已自动生成

让我们总结一下关于如何用**Batch**归一化来应用梯度下降法，假设你在使用**mini-batch**梯度下降法，你运行到**batch**数量的**for**循环，你会在**mini-batch** 上应用正向**prop**，每个隐藏层都应用正向**prop**，用**Batch**归一化代替为。接下来，它确保在这个**mini-batch**中，值有归一化的均值和方差，归一化均值和方差后是，然后，你用反向**prop**计算和，及所有l层所有的参数，和。尽管严格来说，因为你要去掉，这部分其实已经去掉了。最后，你更新这些参数：，和以前一样，，对于也是如此。

如果你已将梯度计算如下，你就可以使用梯度下降法了，这就是我写到这里的，但也适用于有**Momentum**、**RMSprop**、**Adam**的梯度下降法。与其使用梯度下降法更新**mini-batch**，你可以使用这些其它算法来更新，我们在之前几个星期中的视频中讨论过的，也可以应用其它的一些优化算法来更新由**Batch**归一化添加到算法中的 和 参数。

图片包含 文字, 人员

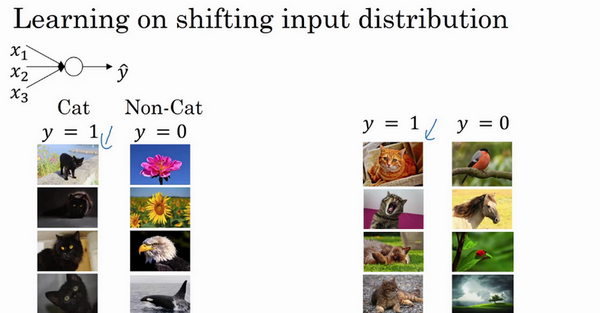
描述已自动生成

### 3.6 Batch Norm 为什么奏效？（Why does Batch Norm work?）

为什么**Batch**归一化会起作用呢？

一个原因是，你已经看到如何归一化输入特征值，使其均值为0，方差1，它又是怎样加速学习的，有一些从0到1而不是从1到1000的特征值，通过归一化所有的输入特征值，以获得类似范围的值，可以加速学习。所以**Batch**归一化起的作用的原因，直观的一点就是，它在做类似的工作，但不仅仅对于这里的输入值，还有隐藏单元的值，这只是**Batch**归一化作用的冰山一角，还有些深层的原理，它会有助于你对**Batch**归一化的作用有更深的理解，让我们一起来看看吧。

**Batch**归一化有效的第二个原因是，它可以使权重比你的网络更滞后或更深层，比如，第10层的权重更能经受得住变化，相比于神经网络中前层的权重，比如第1层，为了解释我的意思，让我们来看看这个最生动形象的例子。



这是一个网络的训练，也许是个浅层网络，比如**logistic**回归或是一个神经网络，也许是个浅层网络，像这个回归函数。或一个深层网络，建立在我们著名的猫脸识别检测上，但假设你已经在所有黑猫的图像上训练了数据集，如果现在你要把此网络应用于有色猫，这种情况下，正面的例子不只是左边的黑猫，还有右边其它颜色的猫，那么你的**cosfa**可能适用的不会很好。

图片包含 文字, 天空

描述已自动生成

如果图像中，你的训练集是这个样子的，你的正面例子在这儿，反面例子在那儿（左图），但你试图把它们都统一于一个数据集，也许正面例子在这，反面例子在那儿（右图）。你也许无法期待，在左边训练得很好的模块，同样在右边也运行得很好，即使存在运行都很好的同一个函数，但你不会希望你的学习算法去发现绿色的决策边界，如果只看左边数据的话。

图片包含 文字

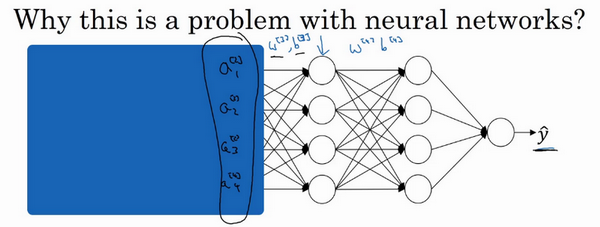
描述已自动生成

所以使你数据改变分布的这个想法，有个有点怪的名字“**Covariate shift**”，想法是这样的，如果你已经学习了到 的映射，如果的分布改变了，那么你可能需要重新训练你的学习算法。这种做法同样适用于，如果真实函数由到 映射保持不变，正如此例中，因为真实函数是此图片是否是一只猫，训练你的函数的需要变得更加迫切，如果真实函数也改变，情况就更糟了。

图片包含 运输

描述已自动生成

“**Covariate shift**”的问题怎么应用于神经网络呢？试想一个像这样的深度网络，让我们从这层（第三层）来看看学习过程。此网络已经学习了参数和，从第三隐藏层的角度来看，它从前层中取得一些值，接着它需要做些什么，使希望输出值接近真实值。



让我先遮住左边的部分，从第三隐藏层的角度来看，它得到一些值，称为，，，，但这些值也可以是特征值，，，，第三层隐藏层的工作是找到一种方式，使这些值映射到，你可以想象做一些截断，所以这些参数和或和或和，也许是学习这些参数，所以网络做的不错，从左边我用黑色笔写的映射到输出值。

图片包含 运输, 手推车

描述已自动生成

现在我们把网络的左边揭开，这个网络还有参数，和，，如果这些参数改变，这些的值也会改变。所以从第三层隐藏层的角度来看，这些隐藏单元的值在不断地改变，所以它就有了“**Covariate shift**”的问题，上张幻灯片中我们讲过的。

图片包含 物体, 手表

描述已自动生成

**Batch**归一化做的，是它减少了这些隐藏值分布变化的数量。如果是绘制这些隐藏的单元值的分布，也许这是重整值，这其实是，，我要绘制两个值而不是四个值，以便我们设想为**2D**，**Batch**归一化讲的是，的值可以改变，它们的确会改变，当神经网络在之前层中更新参数，**Batch**归一化可以确保无论其怎样变化，的均值和方差保持不变，所以即使，的值改变，至少他们的均值和方差也会是均值0，方差1，或不一定必须是均值0，方差1，而是由和决定的值。如果神经网络选择的话，可强制其为均值0，方差1，或其他任何均值和方差。但它做的是，**它限制了在前层的参数更新，会影响数值分布的程度，第三层看到的这种情况，因此得到学习。**

**Batch归一化减少了输入值改变的问题，它的确使这些值变得更稳定，神经网络的之后层就会有更坚实的基础。即使使输入分布改变了一些，它会改变得更少。它做的是当前层保持学习，当改变时，迫使后层适应的程度减小了，你可以这样想，它减弱了前层参数的作用与后层参数的作用之间的联系，它使得网络每层都可以自己学习，稍稍独立于其它层，这有助于加速整个网络的学习。**

图片包含 文字, 地图

描述已自动生成

所以，希望这能带给你更好的直觉，重点是**Batch**归一化的意思是，尤其从神经网络后层之一的角度而言，前层不会左右移动的那么多，因为它们被同样的均值和方差所限制，所以，这会使得后层的学习工作变得更容易些。

**Batch**归一化还有一个作用，它有轻微的正则化效果，**Batch**归一化中非直观的一件事是，每个**mini-batch**，我会说**mini-batch**的值为，，在**mini-batch**计算中，由均值和方差缩放的，因为在**mini-batch**上计算的均值和方差，而不是在整个数据集上，均值和方差有一些小的噪声，因为它只在你的**mini-batch**上计算，比如64或128或256或更大的训练例子。因为均值和方差有一点小噪音，因为它只是由一小部分数据估计得出的。缩放过程从到，过程也有一些噪音，因为它是用有些噪音的均值和方差计算得出的。

图片包含 屏幕截图, 人员, 室内

描述已自动生成

所以和**dropout**相似，它往每个隐藏层的激活值上增加了噪音，**dropout**有增加噪音的方式，它使一个隐藏的单元，以一定的概率乘以0，以一定的概率乘以1，所以你的**dropout**含几重噪音，因为它乘以0或1。

对比而言，**Batch**归一化含几重噪音，因为标准偏差的缩放和减去均值带来的额外噪音。这里的均值和标准差的估计值也是有噪音的，所以类似于**dropout**，**Batch**归一化有轻微的正则化效果，因为给隐藏单元添加了噪音，这迫使后部单元不过分依赖任何一个隐藏单元，类似于**dropout**，它给隐藏层增加了噪音，因此有轻微的正则化效果。因为添加的噪音很微小，所以并不是巨大的正则化效果，你可以将**Batch**归一化和**dropout**一起使用，如果你想得到**dropout**更强大的正则化效果。

也许另一个轻微非直观的效果是，如果你应用了较大的**mini-batch**，对，比如说，你用了512而不是64，通过应用较大的**min-batch**，你减少了噪音，因此减少了正则化效果，这是**dropout**的一个奇怪的性质，就是应用较大的**mini-batch**可以减少正则化效果。

说到这儿，我会把**Batch**归一化当成一种正则化，这确实不是其目的，但有时它会对你的算法有额外的期望效应或非期望效应。但是不要把**Batch**归一化当作正则化，把它当作将你归一化隐藏单元激活值并加速学习的方式，我认为正则化几乎是一个意想不到的副作用。

所以希望这能让你更理解**Batch**归一化的工作，在我们结束**Batch**归一化的讨论之前，我想确保你还知道一个细节。**Batch**归一化一次只能处理一个**mini-batch**数据，它在**mini-batch**上计算均值和方差。所以测试时，你试图做出预测，试着评估神经网络，你也许没有**mini-batch**的例子，你也许一次只能进行一个简单的例子，所以测试时，你需要做一些不同的东西以确保你的预测有意义。

在下一个也就是最后一个**Batch**归一化视频中，让我们详细谈谈你需要注意的一些细节，来让你的神经网络应用**Batch**归一化来做出预测。

### 3.7 测试时的 Batch Norm（Batch Norm at test time）

**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，我们来看一下怎样调整你的网络来做到这一点。

图片包含 文字

描述已自动生成

回想一下，在训练时，这些就是用来执行**Batch**归一化的等式。在一个**mini-batch**中，你将**mini-batch**的值求和，计算均值，所以这里你只把一个**mini-batch**中的样本都加起来，我用m来表示这个**mini-batch**中的样本数量，而不是整个训练集。然后计算方差，再算，即用均值和标准差来调整，加上是为了数值稳定性。是用和再次调整得到的。

请注意用于调节计算的和是在整个**mini-batch**上进行计算，但是在测试时，你可能不能将一个**mini-batch**中的6428或2056个样本同时处理，因此你需要用其它方式来得到和，而且如果你只有一个样本，一个样本的均值和方差没有意义。那么实际上，为了将你的神经网络运用于测试，就需要单独估算和，在典型的**Batch**归一化运用中，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**mini-batch**，接下来我会具体解释。

图片包含 文字

描述已自动生成

我们选择层，假设我们有**mini-batch**，，，……以及对应的值等等，那么在为层训练时，你就得到了，我还是把它写做第一个**mini-batch**和这一层的吧，（）。当你训练第二个**mini-batch**，在这一层和这个**mini-batch**中，你就会得到第二个（）值。然后在这一隐藏层的第三个**mini-batch**，你得到了第三个（）值。正如我们之前用的指数加权平均来计算，，的均值，当时是试着计算当前气温的指数加权平均，你会这样来追踪你看到的这个均值向量的最新平均值，于是这个指数加权平均就成了你对这一隐藏层的均值的估值。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**中所见的的值，以及第二个**mini-batch**中所见的的值等等。因此在用不同的**mini-batch**训练神经网络的同时，能够得到你所查看的每一层的和的平均数的实时数值。



最后在测试时，对应这个等式（），你只需要用你的值来计算，用和的指数加权平均，用你手头的最新数值来做调整，然后你可以用左边我们刚算出来的和你在神经网络训练过程中得到的和参数来计算你那个测试样本的值。

总结一下就是，在训练时，和是在整个**mini-batch**上计算出来的包含了像是64或28或其它一定数量的样本，但在测试时，你可能需要逐一处理样本，方法是根据你的训练集估算和，估算的方式有很多种，理论上你可以在最终的网络中运行整个训练集来得到和，但在实际操作中，我们通常运用指数加权平均来追踪在训练过程中你看到的和的值。还可以用指数加权平均，有时也叫做流动平均来粗略估算和，然后在测试中使用和的值来进行你所需要的隐藏单元值的调整。在实践中，不管你用什么方式估算和，这套过程都是比较稳健的，因此我不太会担心你具体的操作方式，而且如果你使用的是某种深度学习框架，通常会有默认的估算和的方式，应该一样会起到比较好的效果。但在实践中，任何合理的估算你的隐藏单元值的均值和方差的方式，在测试中应该都会有效。

**Batch**归一化就讲到这里，使用**Batch**归一化，你能够训练更深的网络，让你的学习算法运行速度更快，在结束这周的课程之前，我还想和你们分享一些关于深度学习框架的想法，让我们在下一段视频中一起讨论这个话题。