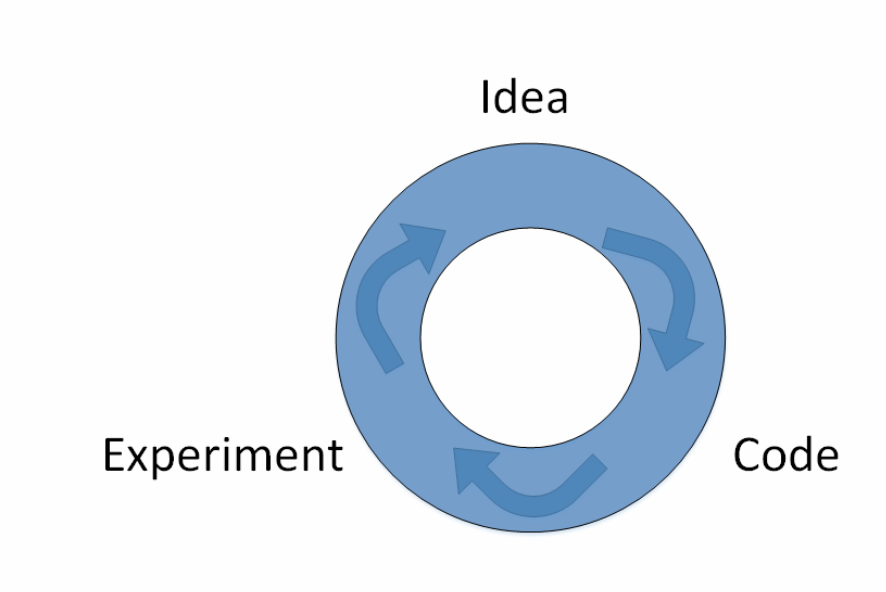
**深度学习的实际运用层面**

1. **构建一个神经网络的通常过程：**



我们先有个想法Idea，先选择初始的参数值，构建神经网络模型结构；然后通过代码Code的形式，实现这个神经网络。最后，通过实验Experiment验证这些参数对应的神经网络的表现性能。根据验证结果，我们对参数进行适当的调整优化，再进行下一次的Idea->Code->Experiment循环。通过很多次的循环，不断调整参数，选定最佳的参数值，从而让神经网络性能最优化。

1. **构建神经网络的三大参数集**

选择最佳的训练集（Training sets）、验证集（Development sets）、测试集（Test sets）对神经网络的性能影响非常重要。

通常有以下分配方式：

小数据样本时：Train/Dev/Test sets，分别为60% 20% 20%

大数据样本时：Train/Dev/Test sets，分别为98% 1% 1%，或者99% 0.5% 0.5%，样本数据量越大，相应的Dev/Test sets的比例可以设置的越低一些。

1. **偏差（Bias）和方差（Variance）**

在传统的机器学习算法中，Bias和Variance是对立的，分别对应着欠拟合和过拟合，我们常常需要在Bias和Variance之间进行权衡。而在深度学习中，我们可以同时减小Bias和Variance，构建一个最佳的神经网络模型。

**欠拟合的通常情况**：假设Train set error为15%，而Dev set error为16%，虽然二者error接近，即该算法模型对训练样本和验证集的识别都不是太好。这说明了该模型对训练样本存在欠拟合。这恰恰是high bias的表现

**过拟合的通常情况**：假设Train set error为1%，而Dev set error为11%，即该算法模型对训练样本的识别很好，但是对验证集的识别却不太好。这说明了该模型对训练样本可能存在过拟合。

**最好的情况**：再假设Train set error为0.5%，而Dev set error为1%，即low bias和low variance，是最好的情况

**最差的情况**：假设Train set error为15%，而Dev set error为30%，说明了该模型既存在high bias也存在high variance

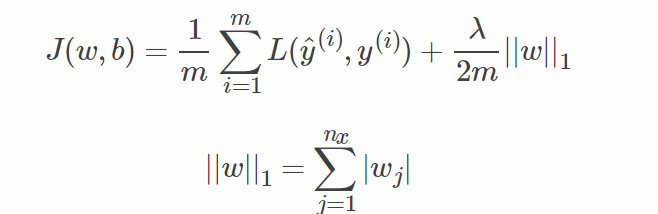
1. **偏如何避免出现高偏差和高方差**

**减少高偏差的方法：**通常是增加神经网络的隐藏层个数、神经元个数，训练时间延长，选择其它更复杂的NN模型等。

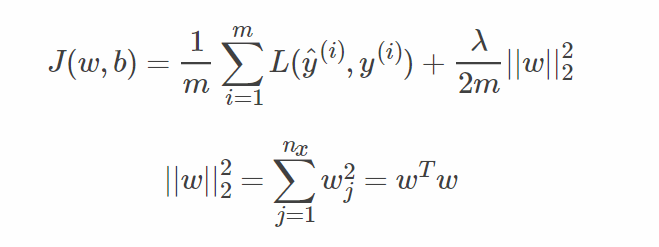
**减少高方差的方法：**通常是增加训练样本数据，进行正则化，选择其他更复杂的NN模型等。

1. **正则化-解决过拟合**

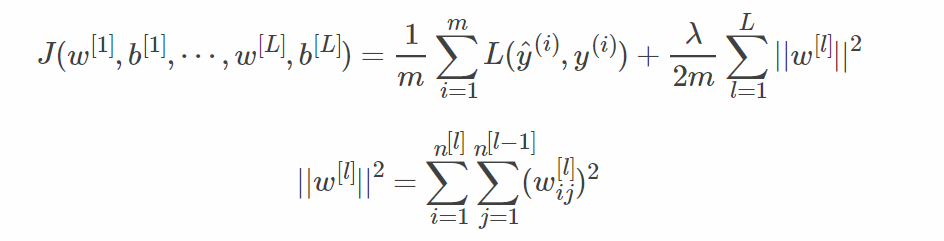
**L1正则化：**

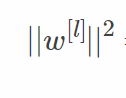
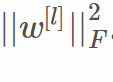


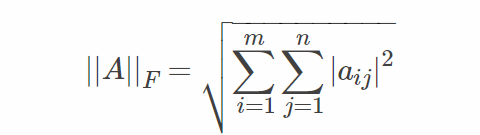
**L2正则化：**



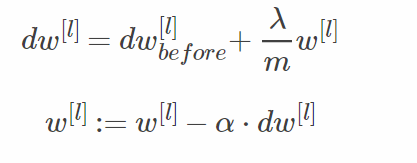
**在深度学习模型中，L2正则化的表达式为**：



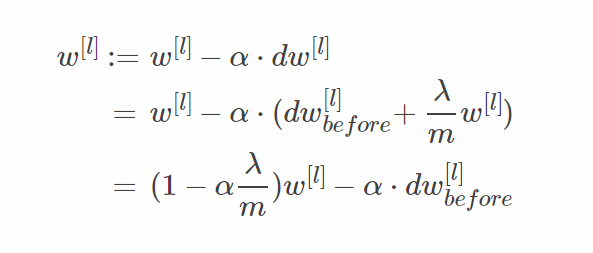
通常，我们把称为Frobenius范数，记为。一个矩阵的Frobenius范数就是计算所有元素平方和再开方，如下所示：

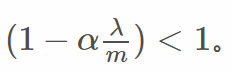


值得注意的是，由于加入了正则化项，梯度下降算法中的dw[l]计算表达式需要做如下修改：



L2正则化也被称做权重衰减。这是因为，由于加上了正则项，dw[l]有个增量，在更新w[l]的时候，会多减去这个增量，使得w[l]比没有正则项的值要小一些。不断迭代更新，不断地减小。



其中，

**正则化防止过拟合原因：**

假如我们选择了非常复杂的神经网络模型。在未使用正则化的情况下，我们得到的可能是过拟合。但是，如果使用L2正则化，当λ很大时，w[l]≈0。w[l]近似为零，意味着该神经网络模型中的某些神经元实际的作用很小，可以忽略。从效果上来看，其实是将某些神经元给忽略掉了。这样原本过于复杂的神经网络模型就变得不那么复杂了，而变得非常简单化了，整个简化的神经网络模型变成了一个逻辑回归模型。问题就从高方差变成了高偏差了。

### **Dropout-防止过拟合：**

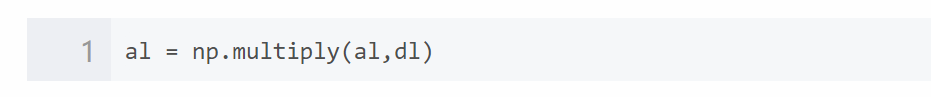
### 是指在深度学习网络的训练过程中，对于每层的神经元，按照一定的概率将其暂时从网络中丢弃。也就是说，每次训练时，每一层都有部分神经元不工作，起到简化复杂网络模型的效果，从而避免发生过拟合。

**Dropout实现方法-Inverted dropout：**

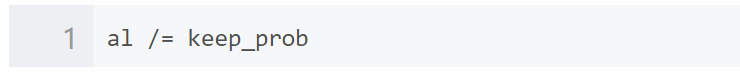
### 假设对于第l层神经元，设定保留神经元比例概率keep\_prob=0.8，即该层有20%的神经元停止工作。dl为dropout向量，设置dl为随机矢量，其中80%的元素为1，20%的元素为0。在python中可以使用如下语句生成dropout矢量：



然后，第l层经过dropout，随机删减20%的神经元，只保留80%的神经元，其输出为：



最后，还要对al进行scale up处理，即：



之所以要对al进行scale up是为了保证在经过dropout后，al作为下一层神经元的输入值尽量保持不变。

**最后注意的是：**

使用dropout训练结束后，在测试和实际应用模型时，不需要进行dropout和随机删减神经元，所有的神经元都在工作。

**为什么dropout能防止过拟合：**

dropout通过每次迭代训练时，随机选择不同的神经元，相当于每次都在不同的神经网络上进行训练，能够防止过拟合。

还可以从权重w的角度来解释为什么dropout能够有效防止过拟合。对于某个神经元来说，某次训练时，它的某些输入在dropout的作用被过滤了。而在下一次训练时，又有不同的某些输入被过滤。经过多次训练后，某些输入被过滤，某些输入被保留。这样，该神经元就不会受某个输入非常大的影响，影响被均匀化了。

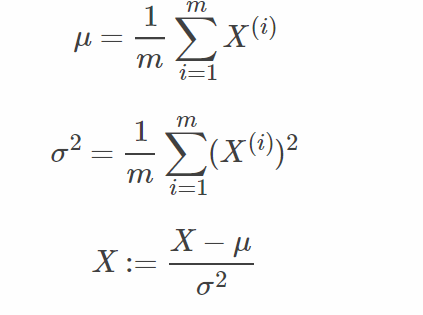
**使用dropout注意：**

1. 不同隐藏层的dropout系数keep\_prob可以不同。一般来说，神经元越多的隐藏层，keep\_out可以设置得小一些。实际应用中，不建议对输入层进行dropout，如果输入层维度很大，例如图片，那么可以设置dropout，但keep\_out应设置的大一些。
2. 值得注意的是dropout是一种正则化技巧，用来防止过拟合的，最好只在需要regularization的时候使用dropout。
3. 使用dropout的时候，可以通过绘制cost function来进行debug，看看dropout是否正确执行。一般做法是，将所有层的keep\_prob全设置为1，再绘制cost function，即涵盖所有神经元，看J是否单调下降。下一次迭代训练时，再将keep\_prob设置为其它值。

**其它防止过拟合方法：**制造更多样本，例如图片识别问题中，可以对已有的图片进行水平翻转、垂直翻转、任意角度旋转、缩放或扩大等等。

**early stopping-防止过拟合：**一个神经网络模型随着迭代训练次数增加，train set error一般是单调减小的，而dev set error 先减小，之后又增大。也就是说训练次数过多时，模型会对训练样本拟合的越来越好，但是对验证集拟合效果逐渐变差，即发生了过拟合。因此，迭代训练次数不是越多越好，可以通过train set error和dev set error随着迭代次数的变化趋势，选择合适的迭代次数，即early stopping。

1. **标准化输入-归一化操作**



之所以要对输入进行标准化操作，主要是为了让所有输入归一化同样的尺度上，方便进行梯度下降算法时能够更快更准确地找到全局最优解。