# 深度学习的实践层面

## 1 训练 验证 测试集

正确配置训练，验证，测试数据集能够很大程度上帮助我们创建高校的神经网络。训练神经网络时，我们要做出很多决策，如神经网络应该分多少层，每层包含多少个隐藏单元，学习率是多大，以及使用哪些激活函数。

在创建新应用的过程中，我们不可能一开始就准测预测出这些超参数。机器学习是一个高度迭代的过程，通常在项目启动的时候，我们用一个浅显的想法搭建一个简陋的模型，并且尝试这个模型。然后我们根据模型的结果，不断改变策略，一直到找到最优方案。

而通过创建训练，验证，测试集，有利于提高这种迭代开发的效率。

我们将数据集分为这三个部分，接下来，我们执行训练算法，使用训练集进行训练，接着通过验证集选择最好的模型，最后，我们在测试集上进行无偏差评估，来评测模型的真实情况。

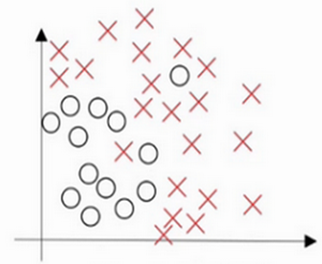
在机器学习的早期阶段，我们通常将数据三七分，百分之30测试集，百分之70训练集，如果加上验证集，就是百分之60训练集，百分之20验证集，百分之20训练集。在只有几万条以内的数据情况下，这样的划分是非常合理的。但是在大数据时代，现在的数据量可能是百万级别，于是验证集和测试集的占比会更小。为了迅速判断哪种算法更有效，我们只需要百分之一左右的数据便足以进行评估。如果我们有一百万条数据，其中一万条作为验证集，一万条作为测试集。对于数据过百万的应用，验证集和测试集各占0.25%即可。

现代深度学习的另一个趋势是越来越多的人在训练和测试集分布不匹配的情况下进行训练，但是这样是不好的。我们要求验证集和测试集来自于同一分布，才能更好的评估模型。

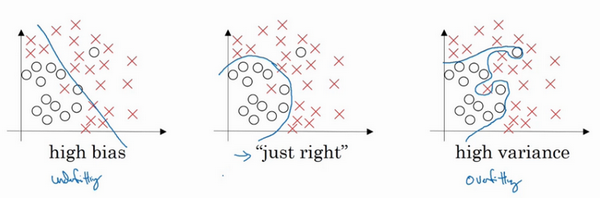
最后一点，你不搞测试集也没什么问题。

## 2.偏差，方差

偏差和方差这两个概念易学难精。在深度学习中，一个趋势是对偏差方差的权衡研究甚浅。深度学习很少权衡这二者，我们总是分别考虑这两种误差，接下来我们一探究竟。

如图，假设这是一个数据集，如果给这个数据集拟合一条直线，就可能得到一个逻辑回归拟合，这并不能很好的拟合该数据，这就是高偏差的情况。我们称为欠拟合。如果我们用一个非常深层的神经网络，就可能非常适合于这个数据集，但我们可能会过度拟合这个数据集，这就是过拟合。

在两者之间，就有刚好拟合的情况，这时方差和偏差都很低。如图所示。



我们可以通过以下一些指标来研究方差和偏差，我们以猫咪识别为例。

如图所示，

理解偏差方差的关键数据在于训练集误差和测试集误差，为了说明情况，我们假设人眼是不会出错的。假定训练集误差是1%，而测试集误差是11%，可以看出训练集数据拟合的非常好，而测试集拟合的很差，这种情况下，就是发生了过拟合，我们称之为高方差。假定测试集误差和训练集误差都是15%，这时说明算法没有很好的拟合数据，我们称之为欠拟合，即高偏差。而还有一种情况是，训练集误差为15%，而测试集误差为30%，这时就是方差和偏差都很高。而训练集和测试集中都表现良好，便是低方差，低偏差。

我们通过和最优分类器比较，就可以判断是否高偏差，和验证集/测试集和训练集比较，就可以判断是否有高偏差。

## 3.机器学习基础

现在我们要来简要介绍一些解决误差的方法，以下的方法，可能有用也可能没用。

当我们训练完初始模型后，我们首先要知道偏差高不高，如果偏差很高，甚至无法拟合数据集，我们要做的就是选择一个新网络，增加更多的单元和层次，或者花费更多的时间来训练网络，或者使用更优化的算法。采用更大的网络通常会有所帮助，延长训练时间不一定有用。我们要一直尝试这些方法，直到网络至少能够拟合训练集。

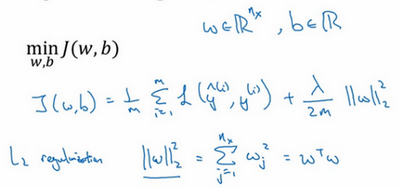
如果方差高，最好的方法就是使用更多的数据，如果无法，就要使用过拟合的方法。

需要说明的一点是，高方差和高偏差是两种不同的情况，解决方法也不能混用。比如说如果高偏差，你使用再多数据也无法解决问题。

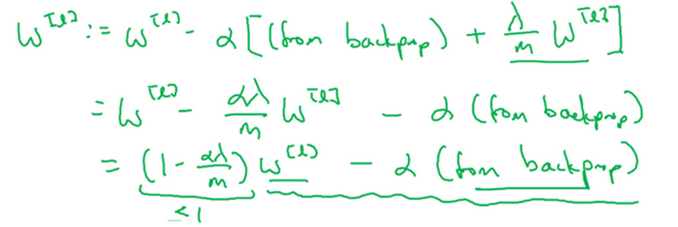
第二点，在机器学习以及深度学习的早期阶段，我们通常要面对方差与偏差的权衡，但现在我们有了更强大的方法能够只减少一方面的误差，而不影响另一方面。只要正则适度，使用一个较大的神经网络就可以同时减少方差和偏差。这也是深度学习不关心方差偏差权衡的原因。

## 4.正则化

正则化是解决深度学习中高方差的有效方法。我们只需要正则化参数ω，如图所示。



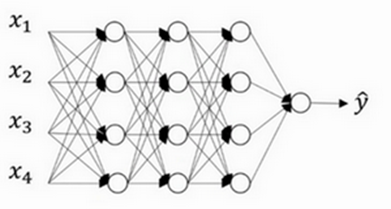
这称为L2正则化，因为我们使用了L2范数加入损失函数中。是正则化参数，我们通常使用验证集来选择这个参数。选择一个合适的参数，我们就能有效的防止过拟合。我们可以通过反向传播的过程来实现正则化，如图所示。



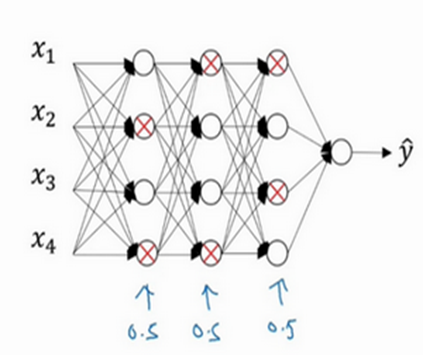
由此可知，不论ω是什么，我们都试图使它更小，因此L2正则化也被叫做权重衰减。

## 5.dropout正则化

除了L2正则化以外，还有一个非常强大的正则化方法叫做dropout，接下来我们来看看它的原理。



假设我们在训练如图所示的神经网络，并且出现了过拟合，我们便可以采用dropout方法来处理。Dropout会遍历这个神经网络的每一层，并设置一定的概率，使其中的某些节点随即消失。我们由此得到一个规模更小的神经网络，并由此通过反向传播方法进行训练。



这就是dropout执行过程中的示意。我们每次训练的网络都是原来大神经网络中的子集。

如何实现dropout呢?我们将讲授最常用的inverted dropout，反向随即失活方法。为了便于理解，我们使用一个三层神经网络来做说明。首先要定义一个向量d，d^([3])表示一个三层的dropout向量：

d3 = np.random.rand(a3.shape[0],a3.shape[1])

然后观察它是否小于某一常数，我们称为keep-prob，这个常数表示这层中的每个节点各自被留下的概率。然后我们将处理这层的激活值，即a3\*=d3，我们通过这个来消除a3中的某些节点。最后，我们将a3/keep-prob，这可以使得a3的期望值不改变。A3的期望值不变，使得我们在测试阶段更容易对数据进行测试。

我们在测试阶段不使用dropout方法，但可以得到十分接近的期望值。

## 6.理解dropout

Dropout可以随机删除节点，为什么删除节点有这么大作用呢？

直观上理解，不要依赖任何一个特征，因为任何一个特征可能很快就被删除了。

因此使用了dropout方法的神经网络会一点一点增加各个单元节点的权重，由此产生收缩平方范数的效果。

在选择keep-prob参数时，我们会在有更多节点的层次上使用更小的keep-prob，以便应用更好的dropout

Dropout在计算机视觉领域用的比较多，而在其他领域上用的比较少。

## 7.使用其他正则化方法

除了dropout和L2，我们还可以使用其他正则化方法。

1. 数据扩增，我们可以通过扩增数据的方法来减少方差。如翻转，裁剪，为图片增加噪点等方法。
2. Early stopping 在训练过程中，我们可以绘制验证集的误差，在验证集误差上升前停止训练。

但是需要指出的是，我们在机器学习策略中常使用正交化的方法，即我们只需要不停优化代价函数J，过拟合之类的问题之后通过其他方法再处理，不要将不同的处理方法重叠，但是early stopping违反了这一原则，因为我们其实还能够将偏差继续减小的。

## 8.归一化输入

我们在机器学习课程中已经系统研究过归一化输入，在此不再进行赘述。即将输入数据的方差置为1，而均值置为0，有利于更快执行梯度下降算法。

## 9. 梯度消失与梯度爆炸

在神经网络的训练中，我们常会遇到的一点情况就是梯度消失和梯度爆炸，导数梯度有的时候会变得特别大或者特别小。假设我们在进行一个极深的神经网络，参数传递到最后，值就会是原来的指数形式，如有l层，初始的参数x，到最后一层经过一系列相乘，就会变成x^(l-1),如果x大于1，就会变成很大的数，x小于1，就会趋近于0。这种情况在很长一段时间内我们都无法解决，还好，我们现在有了新的初始化权重的方式。

## 10.随机初始化权重

面对梯度消失与梯度爆炸的问题，我们有了一个不太完美的解决方案，虽然不能彻底解决，但是也很有用。

在一个节点中，，为了预防z值过大或过小，你可以看到n越大，你希望w越小，因为z是wx的和，如果你把很多此类项相加，希望每项值更小，最合理的方法就是设置，因此，我们要做的就是设置权重，

如果你用relu激活函数，那么使用会更好。除此之外，还有其他一些其他略微不同的系数。

通过这种方法，我们可以让w不大不小，从而避免梯度爆炸和梯度消失。

在实验超参数时，这个系数只是作为优先级最低的一种，因此不用过于纠结选择哪种系数。