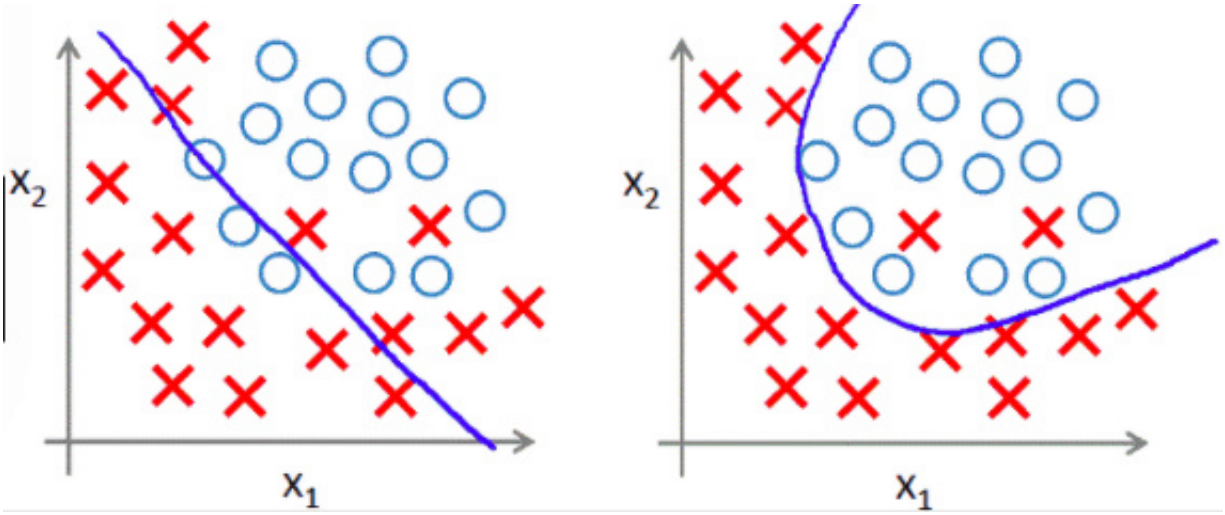
训练卷积网络

前面我们主要讲了网络结构，现在该讲如何训练我们的网络来达到最好的效果，在此之前，我们先讲两个概念，过拟合和欠拟合。

过拟合和欠拟合

欠拟合

欠拟合非常简单，就是模型训练次数还不够，或者是说模型太简单了，没有办法很好的拟合真是的数据，一般一开始的模型就是一个欠拟合的模型，然后我们不断优化希望能够逼近真实的模型。下面就是欠拟合的简单例子



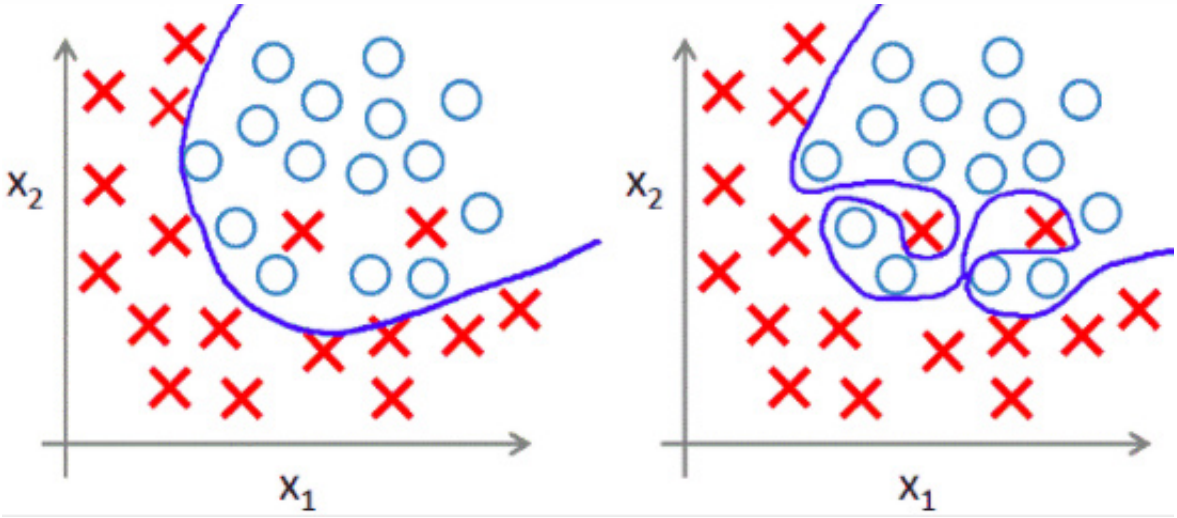
左边是欠拟合，右边是训练好之后的结果，可以看到，模型继续训练能够得到更好的结果。

欠拟合的解决办法非常简单，进行充分的训练，或者增加模型的复杂度，一般欠拟合不是我们会遇到的主要问题。

过拟合

过拟合是我们会遇到主要的问题，通俗来讲，过拟合就是在训练数集上表现良好，但是在测试数据集上表现不好，因为模型完全学会了训练数据集上的特征，而这些特征是有偏差的，并不是一般化的特征，所以遇到新的数据集就表现不够好了。举个简单的例子，你在考试之前课后习题都会做，但是考试遇到没见过的题就不会了，这就是典型的过拟合，没有办法提取出更一般化的特征。

现在深度学习的模型都十分复杂，同时由于 GPU 高效的计算性能，使得我们非常容易充分训练，这个时候我们很大程度上会出现过拟合，因为模型非常复杂，而数据集是有限的，所以很可能学习到局部的特征。



这就是过拟合的一张图示，左边是正确的拟合，右边是过拟合，可以看到右边的模型在训练集上学习到了一些局部的特性，虽然能够把训练集全部分对，但是这种分类方法却是错误的。

后面的课程我们会教给大家一些主流的方法来减小过拟合的影响。

数据增强

前面我们已经讲了几个非常著名的卷积网络的结构，但是单单只靠这些网络并不能取得 state-of-the-art 的结果，现实问题往往更加复杂，非常容易出现过拟合的问题，而数据增强的方法是对抗过拟合问题的一个重要方法。

2012 年 AlexNet 在 ImageNet 上大获全胜，图片增强方法功不可没，因为有了图片增强，使得训练的数据集比实际数据集多了很多'新'样本，减少了过拟合的问题，下面我们来具体解释一下。

常用的数据增强方法

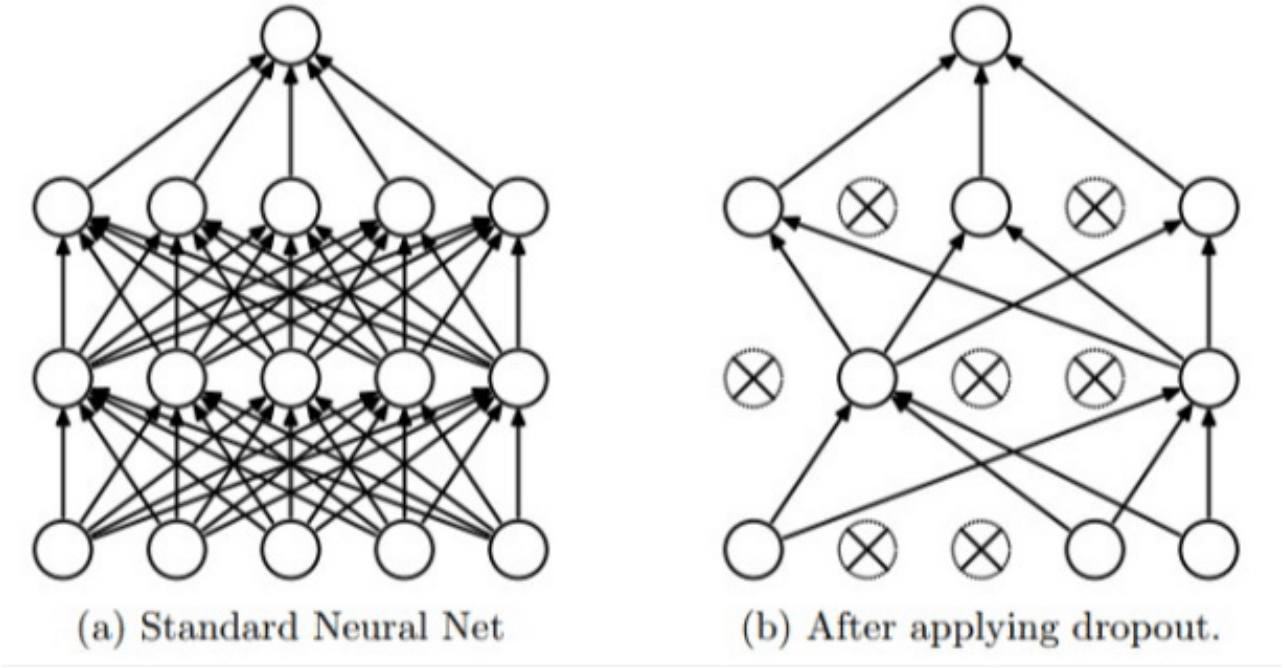
常用的数据增强方法如下： 1.对图片进行一定比例缩放 2.对图片进行随机位置的截取 3.对图片进行随机的水平和竖直翻转 4.对图片进行随机角度的旋转 5.对图片进行亮度、对比度和颜色的随机变化

Dropout

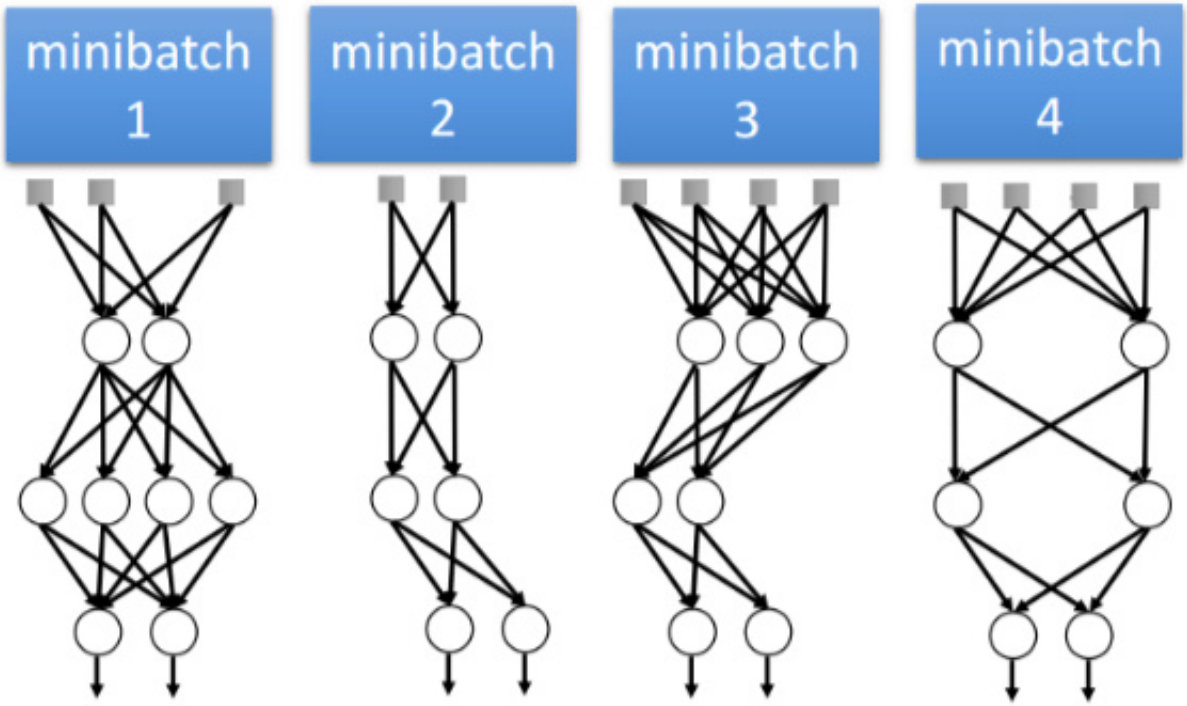
前面我们讲了数据增强，看到了数据增强对模型结果具有非常明显的改善，对于改善过拟合有非常明显的效果，除了数据增强之外，改善过拟合的办法还有 dropout。

dropout

dropout 的灵感来源于人脑的神经元发育，从幼年时候发展到青年时期，人脑的神经元会将一些不使用的逐渐去掉，而发展一些新的神经元，dropout 依据这个原理，在训练的时候以概率 p 保留每个神经元，也就是说在训练的时候，每次都会有神经元被随机设置为 0，如下图



左边是标准的神经网络，稠密连接，而右边就是使用了 dropout 的稀疏连接。我们可以看到每次训练的时候就会有某些神经元没有参与训练，所以在每个 batch 进行训练的时候模型都会有微小的区别，比如



这个时候就会出现一个新的问题，测试的时候怎么办呢？如果我们使用 dropout，那么测试得到的结果就是随机的，没有确定性，而不使用 dropout 网络的输出肯定跟使用 dropout 的结果不一样。

比如使用 dropout 之前的输入是 x，那么使用完 dropout 之后输出的期望就是 $px + (1-p)0 = px$，也就是说 $x \to px$。为了保证结果相同，非常简单，对输出做一下缩放，乘上 $\frac{1}{p}$ 就可以了。

pytorch 中使用 dropout 的方法非常简单，使用 `nn.Dropout(p)` 就行了，p 表示丢弃的概率，默认是 0.5。对于 dropout 在训练和测试时候的表现不同，只需要将模型改变为训练模式 `net = net.train()` 和测试模式 `net = net.eval()` 就行了。

tensorflow 中使用 dropout 的方法非常简单，使用 `tf.nn.dropout` 就行了，p 表示丢弃的概率，默认是 0.5。在训练时, p 可以设置为小于 1 的数, 在测试时, 设置 p 为 1.0 即可.

小练习：尝试在之前讲的网络结构的全连接部分添加 dropout，看看有什么改变

一般 dropout 会用在全连接层，但是由于现在卷积网络逐渐去掉全连接层，所以现在 dropout 在卷积网络中使用的非常的少，同时有更好的技术进行替代，就是下面要讲的正则化。

正则化

前面我们讲了数据增强和 dropout，而在实际使用中，现在的网络往往不使用 dropout，而是用另外一个技术，叫正则化。

正则化是机器学习中提出来的一种方法，有 L1 和 L2 正则化，目前使用较多的是 L2 正则化，引入正则化相当于在 loss 函数上面加上一项，比如

$$f = loss + \lambda \sum_{p \in params} ||p||_2^2 \tag{1}$$

就是在 loss 的基础上加上了参数的二范数作为一个正则化，我们在训练网络的时候，不仅要最小化 loss 函数，同时还要最小化参数的二范数，也就是说我们会对参数做一些限制，不让它变得太大。

如果我们对新的损失函数 f 求导进行梯度下降，就有

$$\frac{\partial f}{\partial p_j} = \frac{\partial loss}{\partial p_j} + 2\lambda p_j \tag{2}$$

那么在更新参数的时候就有

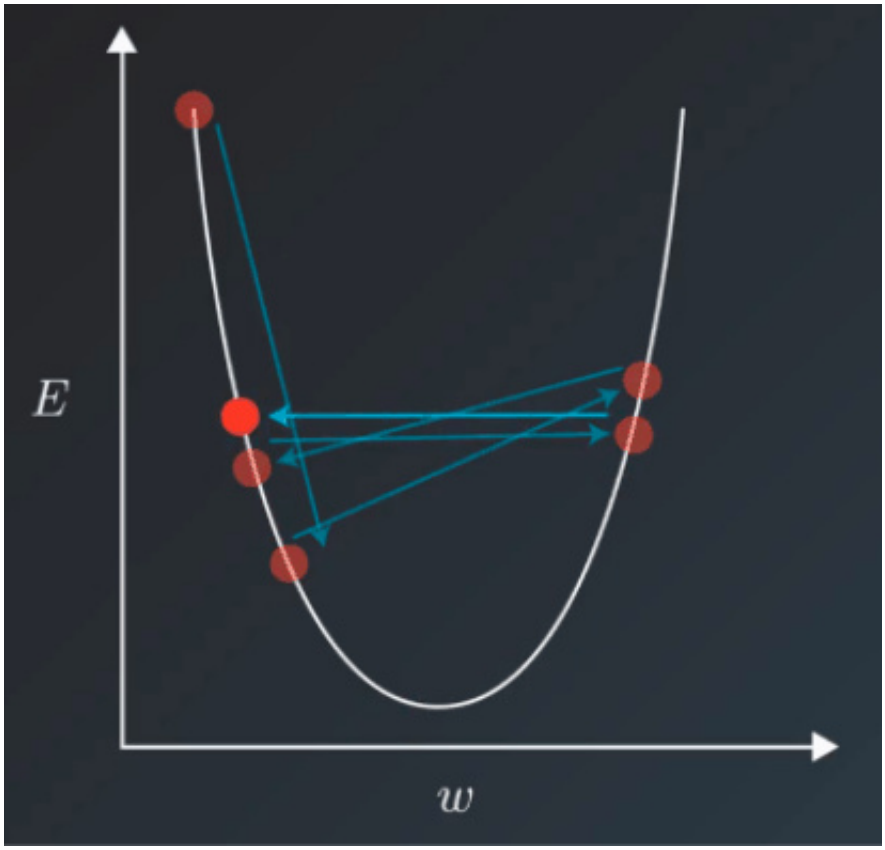
$$p_j \to p_j - \eta(\frac{\partial loss}{\partial p_j} + 2\lambda p_j) = p_j - \eta\frac{\partial loss}{\partial p_j} - 2\eta\lambda p_j \tag{3}$$

可以看到 $p_j - \eta\frac{\partial loss}{\partial p_j}$ 和没加正则项要更新的部分一样，而后面的 $2\eta\lambda p_j$ 就是正则项的影响，可以看到加完正则项之后会对参数做更大程度的更新，这也被称为权重衰减(weight decay)。

注意正则项的系数的大小非常重要，如果太大，会极大的抑制参数的更新，导致欠拟合，如果太小，那么正则项这个部分基本没有贡献，所以选择一个合适的权重衰减系数非常重要，这个需要根据具体的情况去尝试，初步尝试可以使用 `1e-4` 或者 `1e-3` 。

学习率衰减

对于基于一阶梯度进行优化的方法而言，开始的时候更新的幅度是比较大的，也就是说开始的学习率可以设置大一点，但是当训练集的 loss 下降到一定程度之后，，使用这个太大的学习率就会导致 loss 一直来回震荡，比如



这个时候就需要对学习率进行衰减已达到 loss 的充分下降，而是用学习率衰减的办法能够解决这个矛盾，学习率衰减就是随着训练的进行不断的减小学习率。

微调进行迁移学习

前面我们介绍了如何训练卷积神经网络进行图像分类，可能你已经注意到了，训练一个卷积网络是特别耗费时间的，特别是一个比较深的卷积网络，而且可能因为训练方法不当导致训练不收敛的问题，就算训练好了网络，还有可能出现过拟合的问题，所以由此可见能够得到一个好的模型有多么困难。

有的时候，我们的数据集还特别少，这对于我们来讲无异于雪上加霜，因为少的数据集意味着非常高的风险过拟合，那么我们有没有办法在某种程度上避免这个问题呢？其实现在有一种方法特别流行，大家一直在使用，那就是微调(fine-tuning)，在介绍微调之前，我们先介绍一个数据集 ImageNet。

ImageNet

ImageNet 是一个计算机视觉系统识别项目，是目前世界上最大的图像识别数据库，由斯坦福大学组织建立，大约有 1500 万张图片，2.2 万中类别，其中 ISLVRC 作为其子集是学术界中使用最为广泛的公开数据集，一共有 1281167 张图片作为训练集，50000 张图片作为验证集，一共是 1000 分类，是目前测试网络性能的标杆。

我们说的这个数据集有什么用呢？我们又不关心这个数据集，但是对于我们自己的问题，我们有没有办法借助 ImageNet 中的数据集来提升模型效果，比如我们要做一个猫狗分类器，但是我们现在只有几百张图片，肯定不够，ImageNet 中有很多关于猫狗的图片，我们如果能够把这些图片拿过来训练，不就能够提升模型性能了吗？

但是这种做法太麻烦了，从 ImageNet 中寻找这些图片就很困难，如果做另外一个问题又要去找新的图片，所以直接找图片并不靠谱，那么有没有办法能够让我们不去找这些图片，又能使用这些图片呢？

非常简单，我们可以使用在 ImageNet 上训练好的网路，然后把这个网络在放到我们自己的数据集上进行训练不就好了。这个方法就叫做微调，这十分形象，相当于把一个已经很厉害的模型再微调到我们自己的数据集上来，也可称为迁移学习。

迁移学习的方法非常简单，将预训练的模型导入，然后将最后的分类全连接层换成适合我们自己问题的全连接层，然后开始训练，可以固定卷积层的参数，也可以不固定进行训练，最后能够非常有效的得到结果。