# 机器学习策略2

## 进行误差分析

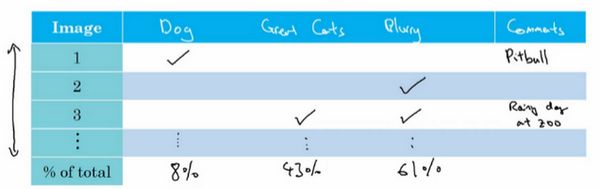
如果你的算法距离人类表现还有一定差距，那么你检查一下数据，分析一下误差，也许是个比较好的办法。

假设你在调试一个猫片分类器，取得了百分之90的准确率，但你的目标是百分百，还缺一些进展，于是你决定看一下算法分类出错的例子，发现很多错误是把狗识别成猫，于是你想了很多种办法解决问题。

但是，wait，你不应该就这么决断。因为很可能最后你花费了好几个月的时间来解决问题而一无所获。

首先，我们应该收集例如100个错误样本，然后手动检查，看看有多少个狗被分类成猫，如果是百分之5，就表示你处理完狗的问题，最多也只能上升0.5%的准确率。这个能够改善的准确率，我们称之为可改善问题的上限。

如果你发现错误样本里有一般都是狗，那我们修正狗的问题，就可以提升百分之5的准确率。

推而广之，我们要收集一些算法处理出错的数据，然后建立一张表格，如图所示，

我们计算出各分类出错占比的概率，由此就可以计算出可改善的上限。这个分析步骤的结果可以给出一个估计，是否值得去处理每个不同的错误类型。

同时，我们也需要考虑解决问题所需要的代价，如果代价过高，就需要你自行权衡了。

## 清除标记错误的数据

假如我们发现数据集中有不少数据被标记错误，那么该如何处理呢？

首先，我们观察训练集，对于训练集中随机被标记出错的概率，我们通常不需要做任何处理。神经网络有足够好的鲁棒性来处理这个问题。但是如果是系统性的错误，比如所有的狗都被标记成猫，我们就需要仔细研究这个问题了。

那么对于开发集中出现的错误呢？我们仿照误差分析中的例子，建立一张表格判断，有多少错误是来自于标记错误的，如果标记错误占总错误比例较高，提升上限高，那么就值得对此进行修正。

如果你要进行修正的话，那么最重要的一点，如果你要修正开发集的数据偏差，那测试集的也要同时进行修改。有时也可以考虑观察一下分类正确的数据，看看是不是由于标签错误才分类正确的。但是一般不这么做，因为几率比较低。

## 快速搭建你的第一个系统，并进行迭代

如果我们在开发全新的机器学习应用，那吴恩达建议首先开发出一个简陋的模型，设立出目标，并为此进行快速迭代。

比如我们有很多技术可以改进语音识别技术，但是我们要做的不是先采用这些技术，而是先搭建一个模型，跑一下，看一下效果如何，再对算法的错误进行优化。如果目标错了，之后改也可以。建立一个系统，进行一下误差分析，你就能知道下一步应该做什么。我建议你们快速建立你的第一个系统，然后迭代。

不过如果你在这个应用程序领域有很多经验，这个建议适用程度要低一些。

还有一种情况适应程度更低，当这个领域有很多可以借鉴的学术文献，处理的问题和你要解决的几乎完全相同，所以，比如说，人脸识别就有很多学术文献，如果你尝试搭建一个人脸识别设备，那么可以从现有大量学术文献为基础出发，一开始就搭建比较复杂的系统。

但是如果你要解决一个全新的问题，那不用想太多，干就完了。

## 使用来自不同分布的数据进行训练和测试

假设我们搭建一个猫片识别的应用，我们的用户上传的可能是模糊的，随意拍摄的照片，而你训练集中所使用的可能是从网络上下载的，精确度很高的照片，由此就产生了数据分布不匹配的问题。

你用户最终关心的是他们上传的图片在你系统上的表现。

假如你有20w张照片来自于网络下载，1w张照片来自于接近的真实分布，那你该如何划分数据集呢。

比较好的做法是，让这20w张照片全部归属于训练集，这1w张照片可以全部划入验证集和测试集，也可以拿出一些划入训练集。确保开发集和测试集是相同分布的。并且尽可能接近于真实情境下的分布。

## 数据不匹配时的方差和偏差分析

当你训练集和开发集不匹配时，两者误差差距较大，你就不能简单的说是方差引起的，也有可能是开发集中的数据对于模型来说难以识别。解决办法就是从训练集中拿出一块，作为训练-开发集，并在训练完后和训练集进行误差评估，如果差距较大，才能说明存在方差问题。而如果差距较小，就说明开发集中的误差是由于数据不匹配引起的。

如果开发集和测试集差距较大，就说明发生过拟合了，需要使用更大的开发集。

如果开发集表现得比训练集更好，可能是由于开发集的数据比训练集中的更容易识别。

## 处理数据不匹配问题

处理数据不匹配问题并没有什么通用的方法，以下介绍一些小技巧。进行一下误差分析，查看是由那种不匹配引起的难以识别，并对训练集进行预处理，使训练集更趋近于真实分布。比如进行人工数据合成。

假设你进行一个车载语音识别系统的开发，你的训练集是无噪音的，而测试集和开发

集是有噪音的，由此造成的数据不匹配你可以通过给训练集中人工加入车辆噪音来进行处理。当然，你要当心过拟合问题，因为一个一小时的噪音被加入到每个训练集中，就被重复了上万次，很容易使模型对此产生过拟合，你应该使用尽可能不同的噪音加入。

## 迁移学习

神经网络很强大的一点是，有时从一个任务学到的经验，可以迁移到其他任务上。比如你已经训练了一个猫狗分类的神经网络，你想要搭建一个识别X光照片的模型，这个神经网络学到的低层次图片特征其实是可以迁移到新的神经网络去的。因为低层次特征很多是类似的。我们要做的就是把神经网络的前面的层次冻结，并使用新的数据来重新训练最后几个层次的参数。

这取决于你的数据多大，你的数据很小就训练比较少的层次，数据多就可以多训练一些层次，数据很多，比这个预训练的神经网络数据还多的话，就没必要迁移学习了。

这个重新训练最后几层参数的过程叫做微调。

如果你需要的任务没有那么多数据，并且之前有类似的任务的话，很适合用迁移学习来完成。

## 多任务学习

使用一个神经网络来执行多种任务，比如多个物体的目标检测。

有先决条件，各种任务类型类似，并且数据量接近。适用范围比较窄，多用在目标检测中。

比如输入一张图片，最后输入一个4列的向量，每列为0或1，使用逻辑回归生成，用来表示是否存在这个物体。如此，你输入一张图片，就可以得到是否存在四种物体的信息。这种方式要比搭建四个网络进行学习要好，尤其是你的数据量足够的情况下。

## 端对端的机器学习

端对端的机器学习，即通过训练一个模型，直接得到从X到Y的映射，不需要中间任何的特征工程。这需要海量的数据来完成。

比如说使用深度学习来进行翻译任务，直接给出英文，就能够得到对应法文的翻译。

而大部分情况下，是将一个任务拆成多个子任务来完成的，比如说人脸检测，就包含了了人脸对照和目标检测两个过程。

如果数据量足够的情况下，使用端对端的机器学习可能可以得到更好的效果。