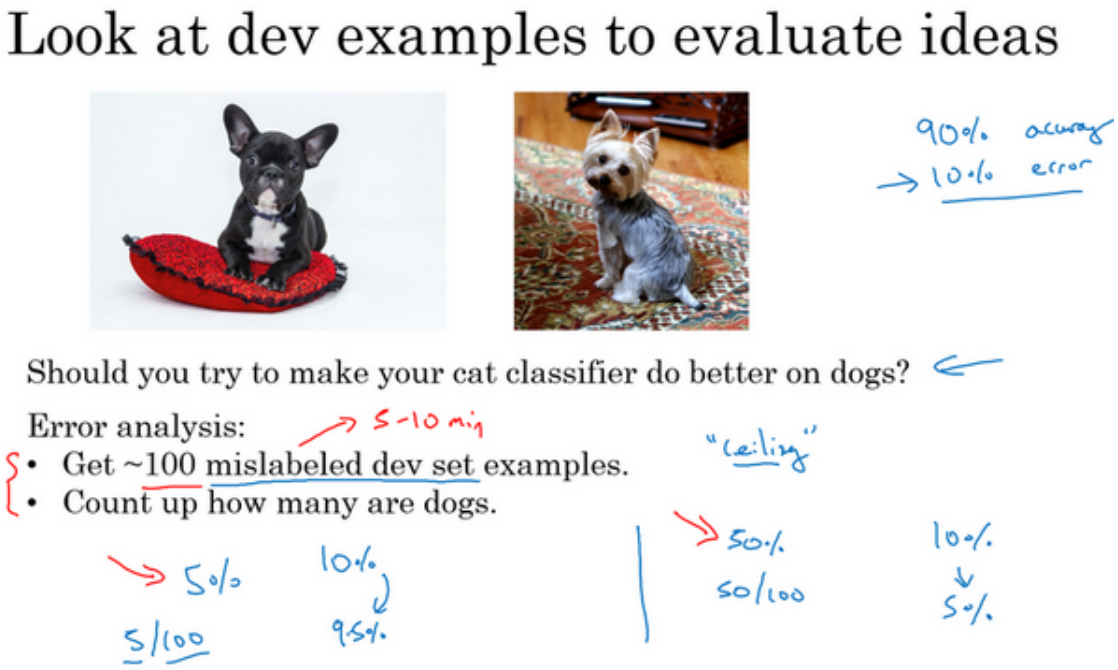
2.1 进行误差分析（Carrying out error analysis）

你好，欢迎回来，如果你希望让学习算法能够胜任人类能做的任务，但你的学习算法还没有达到人类的表现，那么人工检查一下你的算法犯的错误也许可以让你了解接下来应该做什么。这个过程称为错误分析，我们从一个例子开始讲吧。



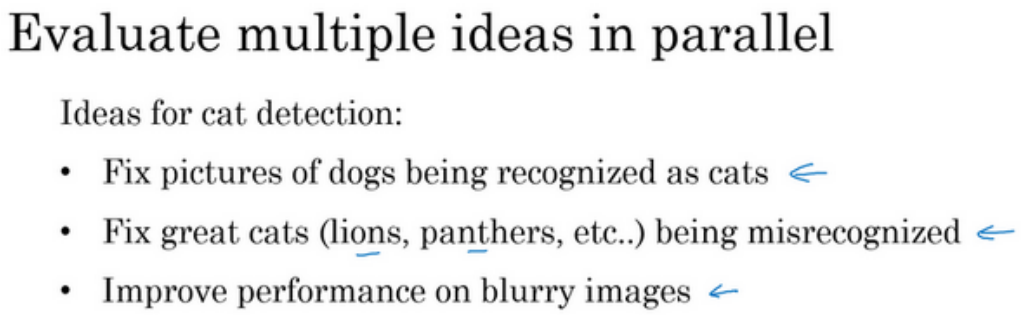
假设你正在调试猫分类器，然后你取得了 90%准确率，相当于 10%错误，在你的开发集上做到这样，这离你希望的目标还有很远。也许你的队员看了一下算法分类出错的例子，注意到算法将一些狗分类为猫，你看看这两只狗，它们看起来是有点像猫，至少乍一看是。所以也许你的队友给你一个建议，如何针对狗的图片优化算法。试想一下，你可以针对狗，收集更多的狗图，或者设计一些只处理狗的算法功能之类的，为了让你的猫分类器在狗图上做的更好，让算法不再将狗分类成猫。所以问题在于，你是不是应该去开始做一个项目专门处理狗？这项目可能需要花费几个月的时间才能让算法在狗图片上犯更少的错误，这样做值得吗？或者与其花几个月做这个项目，有可能最后发现这样一点用都没有。这里有个错误分析流程，可以让你很快知道这个方向是否值得努力。

这是我建议你做的，首先，收集一下，比如说 100 个错误标记的开发集样本，然后手动检查，一次只看一个，看看你的开发集里有多少错误标记的样本是狗。现在，假设事实上，你的 100 个错误标记样本中只有 5%是狗，就是说在 100 个错误标记的开发集样本中，有 5 个是狗。这意味着 100 个样本，在典型的 100 个出错样本中，即使你完全解决了狗的问题，

你也只能修正这 100 个错误中的 5 个。或者换句话说，如果只有 5%的错误是狗图片，那么如果你在狗的问题上花了很多时间，那么你最多只能希望你的错误率从 10%下降到 9.5%，对吧？错误率相对下降了 5%（总体下降了 0.5%，100 的错误样本，错误率为 10%，则样本为 1000），那就是 10%下降到 9.5%。你就可以确定这样花时间不好，或者也许应该花时间，但至少这个分析给出了一个上限。如果你继续处理狗的问题，能够改善算法性能的上限，对吧？在机器学习中，有时我们称之为性能上限，就意味着，最好能到哪里，完全解决狗的问题可以对你有多少帮助。

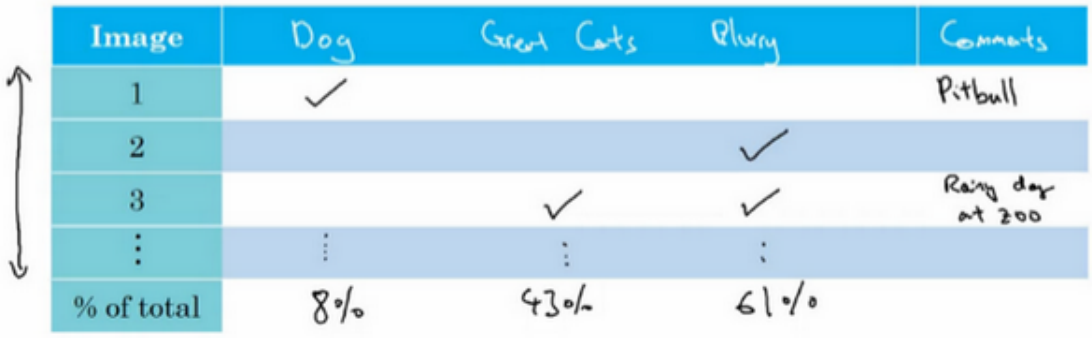
但现在，假设发生了另一件事，假设我们观察一下这 100 个错误标记的开发集样本，你发现实际有 50 张图都是狗，所以有 50%都是狗的照片，现在花时间去解决狗的问题可能效果就很好。这种情况下，如果你真的解决了狗的问题，那么你的错误率可能就从 10%下降到 5%了。然后你可能觉得让错误率减半的方向值得一试，可以集中精力减少错误标记的狗图的问题。

我知道在机器学习中，有时候我们很鄙视手工操作，或者使用了太多人为数值。但如果你要搭建应用系统，**那这个简单的人工统计步骤，错误分析，可以节省大量时间，可以迅速决定什么是最重要的，或者最有希望的方向。**实际上，如果你观察 100 个错误标记的开发集样本，也许只需要 5 到 10 分钟的时间，亲自看看这 100 个样本，并亲自统计一下有多少是狗。根据结果，看看有没有占到 5%、50%或者其他东西。这个在 5 到 10 分钟之内就能给你估计这个方向有多少价值，并且可以帮助你做出更好的决定，是不是把未来几个月的时间投入到解决错误标记的狗图这个问题。

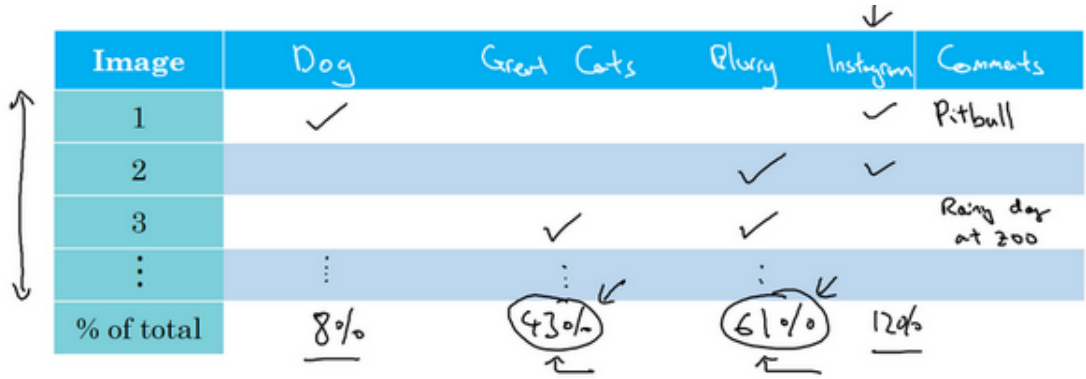


在本幻灯片中，我们要描述一下如何使用错误分析来评估某个想法，这个样本里狗的问题是否值得解决。有时你在做错误分析时，也可以同时并行评估几个想法，比如，你有几个改善猫检测器的想法，也许你可以改善针对狗图的性能，或者有时候要注意，那些猫科动物，如狮子，豹，猎豹等等，它们经常被分类成小猫或者家猫，所以你也许可以想办法解决这个错误。或者也许你发现有些图像是模糊的，如果你能设计出一些系统，能够更好地处理模糊

图像。也许你有些想法，知道大概怎么处理这些问题，要进行错误分析来评估这三个想法。



我会做的是建立这样一个表格，我通常用电子表格来做，但普通文本文件也可以。在最左边，人工过一遍你想分析的图像集，所以图像可能是从 1 到 100，如果你观察 100 张图的话。电子表格的一列就对应你要评估的想法，所以狗的问题，猫科动物的问题，模糊图像的问题，我通常也在电子表格中留下空位来写评论。所以记住，在错误分析过程中，你就看看算法识别错误的开发集样本，如果你发现第一张识别错误的图片是狗图，那么我就在那里打个勾，为了帮我自己记住这些图片，有时我会在评论里注释，也许这是一张比特犬的图。如果第二张照片很模糊，也记一下。如果第三张是在下雨天动物园里的狮子，被识别成猫了，这是大型猫科动物，还有图片模糊，在评论部分写动物园下雨天，是雨天让图像模糊的之类的。**最后，这组图像过了一遍之后，我可以统计这些算法(错误)的百分比，或者这里每个错误类型的百分比，有多少是狗，大猫或模糊这些错误类型。所以也许你检查的图像中 8%是狗，可能 43%属于大猫，61%属于模糊。这意味着扫过每一列，并统计那一列有多少百分比图像打了勾。**



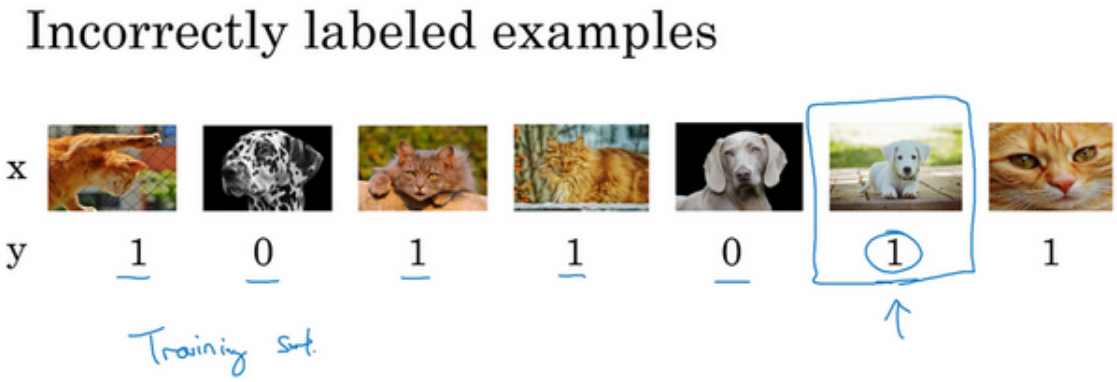
在这个步骤做到一半时，有时你可能会发现其他错误类型，比如说你可能发现有 **Instagram** 滤镜，那些花哨的图像滤镜，干扰了你的分类器。在这种情况下，实际上可以在错误分析途中，增加这样一列，比如多色滤镜 **Instagram** 滤镜和 **Snapchat** 滤镜，然后再过一遍，也统计一下那些问题，并确定这个新的错误类型占了多少百分比，这个分析步骤的结果可以给出一个估计，是否值得去处理每个不同的错误类型。

例如，在这个样本中，有很多错误来自模糊图片，也有很多错误类型是大猫图片。所以这个分析的结果不是说你一定要处理模糊图片，这个分析没有给你一个严格的数学公式，告诉你应该做什么，但它能让你对应该选择那些手段有个概念。它也告诉你，比如说不管你对狗图片或者 **Instagram** 图片处理得有多好，在这些例子中，你最多只能取得 8%或者 12%的性能提升。而在大猫图片这一类型，你可以做得更好。或者模糊图像，这些类型有改进的潜力。**这些类型里，性能提高的上限空间要大得多。所以取决于你有多少改善性能的想法，比如改善大猫图片或者模糊图片的表现。**也许你可以选择其中两个，或者你的团队成员足够多，也许你把团队可以分成两个团队，其中一个想办法改善大猫的识别，另一个团队想办法改善模糊图片的识别。但这个快速统计的步骤，你可以经常做，最多需要几小时，就可以真正帮你选出高优先级任务，并了解每种手段对性能有多大提升空间。

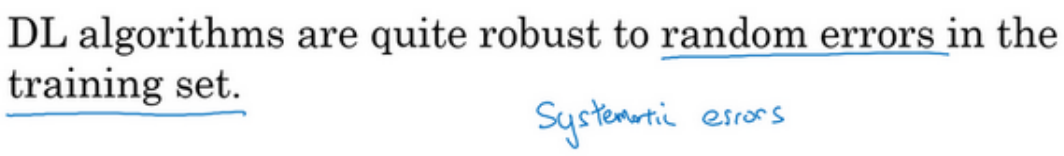
所以总结一下，进行错误分析，你应该找一组错误样本，可能在你的开发集里或者测试集里，观察错误标记的样本，看看假阳性（false positives）和假阴性（false negatives），统计属于不同错误类型的错误数量。在这个过程中，你可能会得到启发，归纳出新的错误类型。如果你过了一遍错误样本，然后说，天，有这么多 **Instagram** 滤镜或 **Snapchat** 滤镜，这些滤镜干扰了我的分类器，你就可以在途中新建一个错误类型。总之，通过统计不同错误标记类型占总数的百分比，可以帮你发现哪些问题需要优先解决，或者给你构思新优化方向的灵感。在做错误分析的时候，有时你会注意到开发集里有些样本被错误标记了，这时应该怎么做呢？我们下一个视频来讨论。

2.2 清除标注错误的数据（Cleaning up Incorrectly labeled data）

你的监督学习问题的数据由输入x和输出标签 y 构成，如果你观察一下你的数据，并发现有些输出标签 y 是错的，你的数据有些标签是错的，是否值得花时间去修正这些标签呢？



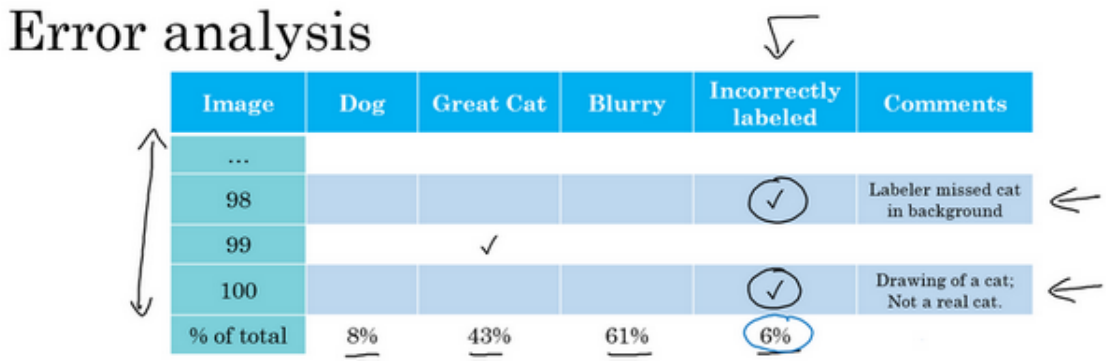
我们看看在猫分类问题中，图片是猫，$y=1$；不是猫，$y=0$。所以假设你看了一些数据样本，发现这（倒数第二张图片）其实不是猫，所以这是标记错误的样本。我用了这个词，“标记错误的样本”来表示你的学习算法输出了错误的 y 值。但我要说的是，对于标记错误的样本，参考你的数据集，在训练集或者测试集 y 的标签，人类给这部分数据加的标签，实际上是错的，这实际上是一只狗，所以 y 其实应该是 0，也许做标记的那人疏忽了。如果你发现你的数据有一些标记错误的样本，你该怎么办？



首先，我们来考虑训练集，事实证明，深度学习算法对于训练集中的随机错误是相当健壮的（**robust**）。只要你的标记出错的样本，只要这些错误样本离随机错误不太远，有时可能做标记的人没有注意或者不小心，按错键了，如果错误足够随机，那么放着这些错误不管可能也没问题，而不要花太多时间修复它们。

当然你浏览一下训练集，检查一下这些标签，并修正它们也没什么害处。有时候修正这些错误是有价值的，有时候放着不管也可以，只要总数据集总足够大，实际错误率可能不会太高。我见过一大批机器学习算法训练的时候，明知训练集里有x个错误标签，但最后训练出来也没问题。

我这里先警告一下，**深度学习算法对随机误差很健壮，但对系统性的错误就没那么健壮了。**所以比如说，如果做标记的人一直把白色的狗标记成猫，那就成问题了。因为你的分类器学习之后，会把所有白色的狗都分类为猫。但随机错误或近似随机错误，对于大多数深度学习算法来说不成问题。

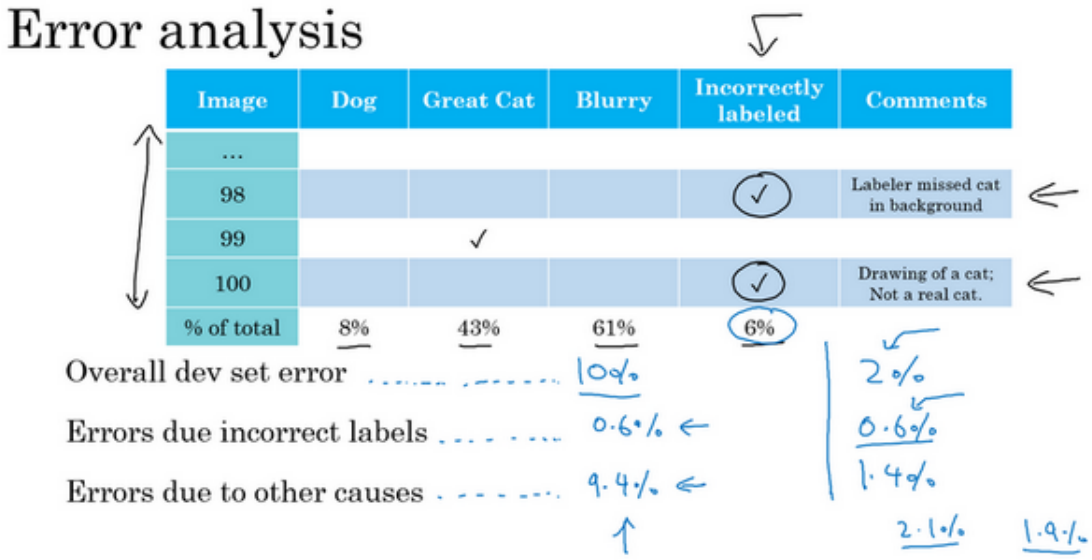


现在，之前的讨论集中在训练集中的标记出错的样本，那么如果是开发集和测试集中有这些标记出错的样本呢？如果你担心开发集或测试集上标记出错的样本带来的影响，他们一般建议你在错误分析时，添加一个额外的列，这样你也可以统计标签 $y = 1$错误的样本数。所以比如说，也许你统计一下对 100 个标记出错的样本的影响，所以你会找到 100 个样本，其中你的分类器的输出和开发集的标签不一致，有时对于其中的少数样本，你的分类器输出和标签不同，是因为标签错了，而不是你的分类器出错。所以也许在这个样本中，你发现标记的人漏了背景里的一只猫，所以那里打个勾，来表示样本 98 标签出错了。也许这张图实际上是猫的画，而不是一只真正的猫，也许你希望标记数据的人将它标记为$y = 0$，而不是 $y = 1$，然后再在那里打个勾。当你统计出其他错误类型的百分比后，就像我们在之前的视频中看到的那样，你还可以统计因为标签错误所占的百分比，你的开发集里的 y 值是错的，这就解释了为什么你的学习算法做出和数据集里的标记不一样的预测 1。

所以现在问题是，是否值得修正这 6%标记出错的样本，我的建议是，**如果这些标记错误严重影响了你在开发集上评估算法的能力，那么就应该去花时间修正错误的标签。但是，如果它们没有严重影响到你用开发集评估成本偏差的能力，那么可能就不应该花宝贵的时间去处理。**

我给你看一个样本，解释清楚我的意思。所以我建议你看 3 个数字来确定是否值得去人工修正标记出错的数据，我建议你看看整体的开发集错误率，在我们以前的视频中的样本，我们说也许我们的系统达到了 90%整体准确度，所以有 10%错误率，那么你应该看看错误标记引起的错误的数量或者百分比。所以在这种情况下，6%的错误来自标记出错，所以 10%的 6%就是 0.6%。也许你应该看看其他原因导致的错误，如果你的开发集上有 10%错误，其

中 0.6%是因为标记出错，剩下的占 9.4%，是其他原因导致的，比如把狗误认为猫，大猫图片。所以在这种情况下，我说有 9.4%错误率需要集中精力修正，而标记出错导致的错误是总体错误的一小部分而已，所以如果你一定要这么做，你也可以手工修正各种错误标签，但也许这不是当下最重要的任务。



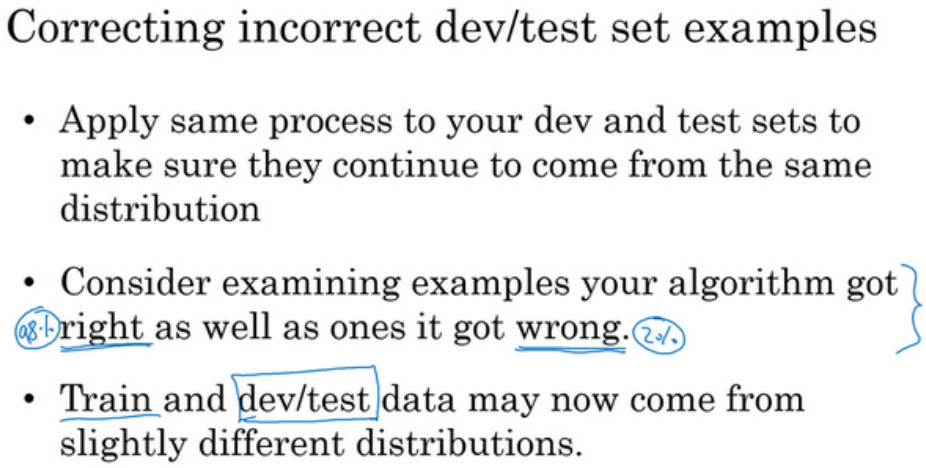
我们再看另一个样本，假设你在学习问题上取得了很大进展，所以现在错误率不再是 10%了，假设你把错误率降到了 2%，但总体错误中的 0.6%还是标记出错导致的。所以现在，如果你想检查一组标记出错的开发集图片，开发集数据有 2%标记错误了，那么其中很大一部分，0.6%除以 2%，实际上变成 30%标签而不是 6%标签了。有那么多错误样本其实是因为标记出错导致的，所以现在其他原因导致的错误是 1.4%。当测得的那么大一部分的错误都是开发集标记出错导致的，那似乎修正开发集里的错误标签似乎更有价值。

Goal of dev set is to help you select between two classifiers A & B.

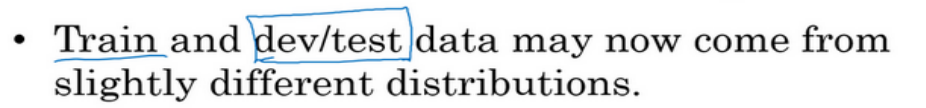
如果你还记得设立开发集的目标的话，**开发集的主要目的是，你希望用它来从两个分类器*A*和*B*中选择一个。**所以当你测试两个分类器*A*和*B*时，在开发集上一个有 2.1%错误率，另一个有 1.9%错误率，但是你不能再信任开发集了，因为它无法告诉你这个分类器是否比这个好，因为 0.6%的错误率是标记出错导致的。那么现在你就有很好的理由去修正开发集里的错误标签，因为在右边这个样本中，标记出错对算法错误的整体评估标准有严重的影响。而左边的样本中，标记出错对你算法影响的百分比还是相对较小的。

现在如果你决定要去修正开发集数据，手动重新检查标签，并尝试修正一些标签，这里还有一些额外的方针和原则需要考虑。**首先，我鼓励你不管用什么修正手段，都要同时作用到开发集和测试集上，我们之前讨论过为什么，开发和测试集必须来自相同的分布。**开发集

确定了你的目标，当你击中目标后，你希望算法能够推广到测试集上，这样你的团队能够更高效的在来自同一分布的开发集和测试集上迭代。如果你打算修正开发集上的部分数据，那么最好也对测试集做同样的修正以确保它们继续来自相同的分布。所以我们雇佣了一个人来仔细检查这些标签，但必须同时检查开发集和测试集。



其次，我强烈建议你要考虑同时检验算法判断正确和判断错误的样本，要检查算法出错的样本很容易，只需要看看那些样本是否需要修正，但还有可能有些样本算法判断正确，那些也需要修正。如果你只修正算法出错的样本，你对算法的偏差估计可能会变大，这会让你的算法有一点不公平的优势，我们就需要再次检查出错的样本，但也需要再次检查做对的样本，因为算法有可能因为运气好把某个东西判断对了。在那个特例里，修正那些标签可能会让算法从判断对变成判断错。这第二点不是很容易做，所以通常不会这么做。通常不会这么做的原因是，如果你的分类器很准确，那么判断错的次数比判断正确的次数要少得多。那么就有 2%出错，98%都是对的，所以更容易检查 2%数据上的标签，然而检查 98%数据上的标签要花的时间长得多，所以通常不这么做，但也是要考虑到的。



最后，如果你进入到一个开发集和测试集去修正这里的部分标签，你可能会，也可能不会去对训练集做同样的事情，还记得我们在其他视频里讲过，修正训练集中的标签其实相对没那么重要，你可能决定只修正开发集和测试集中的标签，因为它们通常比训练集小得多，你可能不想把所有额外的精力投入到修正大得多的训练集中的标签，所以这样其实是可以的。我们将在本周晚些时候讨论一些步骤，用于处理你的训练数据分布和开发与测试数据不同的情况，对于这种情况学习算法其实相当健壮，你的开发集和测试集来自同一分布非常重要。但如果你的训练集来自稍微不同的分布，通常这是一件很合理的事情，我会在本周晚些

时候谈谈如何处理这个问题。

最后我讲几个建议：

首先，深度学习研究人员有时会喜欢这样说：“我只是把数据提供给算法，我训练过了，效果拔群”。这话说出了很多深度学习错误的真相，更多时候，我们把数据喂给算法，然后训练它，并减少人工干预，减少使用人类的见解。**但我认为，在构造实际系统时，通常需要更多的人工错误分析，更多的人类见解来架构这些系统，尽管深度学习的研究人员不愿意承认这点。**

其次，不知道为什么，我看一些工程师和研究人员不愿意亲自去看这些样本，也许做这些事情很无聊，坐下来看 100 或几百个样本来统计错误数量，但我经常亲自这么做。当我带领一个机器学习团队时，我想知道它所犯的错误，我会亲自去看看这些数据，尝试和一部分错误作斗争。我想就因为花了这几分钟，或者几个小时去亲自统计数据，真的可以帮你找到需要优先处理的任务，**我发现花时间亲自检查数据非常值得，**所以我强烈建议你们这样做，如果你在搭建你的机器学习系统的话，然后你想确定应该优先尝试哪些想法，或者哪些方向。

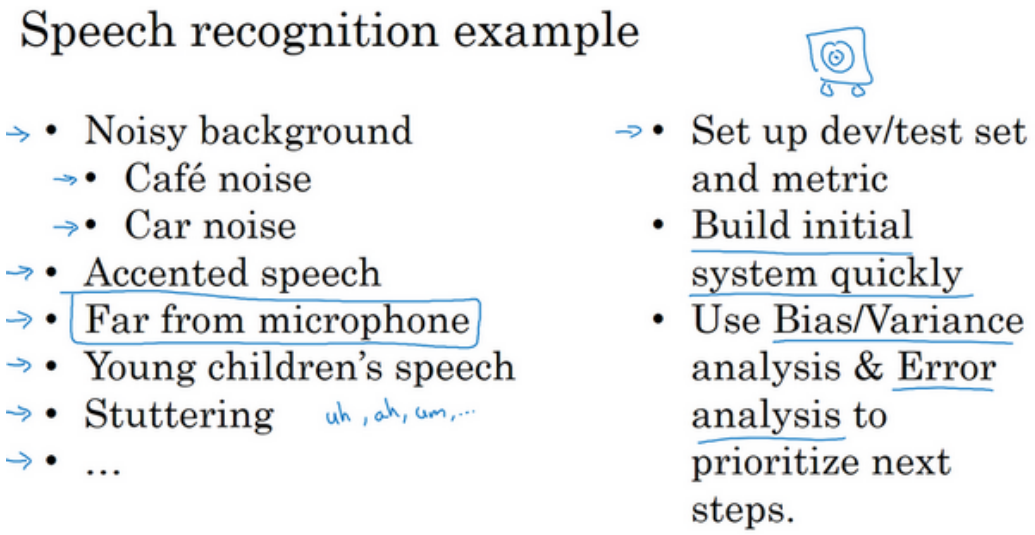
这就是错误分析过程，在下一个视频中，我想分享一下错误分析是如何在启动新的机器学习项目中发挥作用的。

2.3 快速搭建你的第一个系统，并进行迭代（Build your first system quickly, then iterate）

如果你正在开发全新的机器学习应用，我通常会给你这样的建议，你应该尽快建立你的第一个系统原型，然后快速迭代。

让我告诉你我的意思，我在语音识别领域研究了很多年，如果你正在考虑建立一个新的语音识别系统，其实你可以走很多方向，可以优先考虑很多事情。

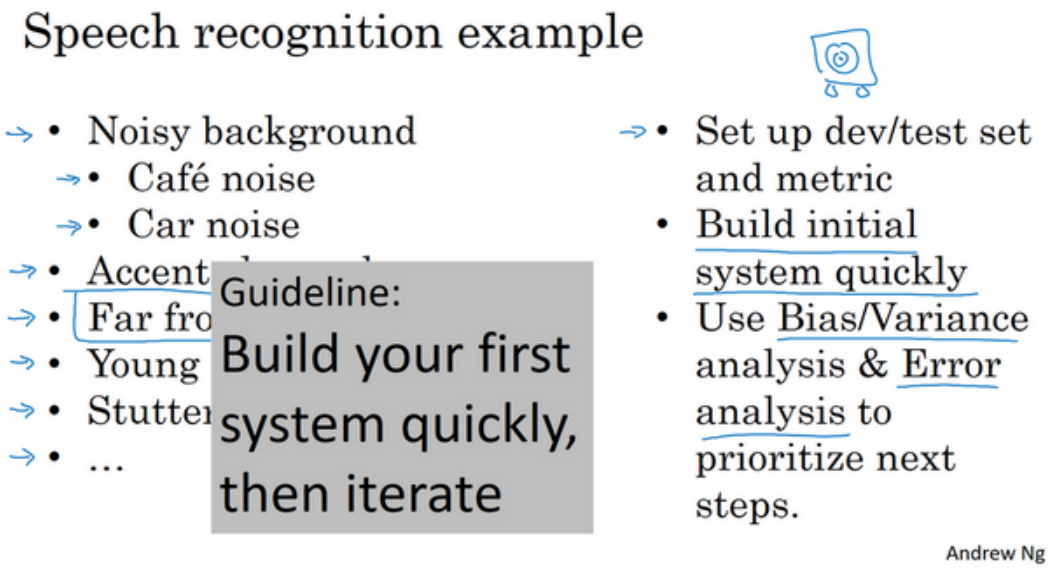
比如，有一些特定的技术，可以让语音识别系统对嘈杂的背景更加健壮，嘈杂的背景可能是说咖啡店的噪音，背景里有很多人在聊天，或者车辆的噪音，高速上汽车的噪音或者其他类型的噪音。有一些方法可以让语音识别系统在处理带口音时更健壮，还有特定的问题和麦克风与说话人距离很远有关，就是所谓的远场语音识别。儿童的语音识别带来特殊的挑战，挑战来自单词发音方面，还有他们选择的词汇，他们倾向于使用的词汇。还有比如说，说话人口吃，或者说了很多无意义的短语，比如“哦”，“啊”之类的。你可以选择很多不同的技术，让你听写下来的文本可读性更强，所以你可以做很多事情来改进语音识别系统。



一般来说，对于几乎所有的机器学习程序可能会有 50 个不同的方向可以前进，并且每个方向都是相对合理的可以改善你的系统。但挑战在于，你如何选择一个方向集中精力处理。即使我已经在语音识别领域工作多年了，如果我要为一个新应用程序域构建新系统，我还是觉得很难不花时间去思考这个问题就直接选择方向。所以我建议你们，如果你想搭建全新的机器学习程序，就是快速搭好你的第一个系统，然后开始迭代。我的意思是我建议你快速设立开发集和测试集还有指标，这样就决定了你的目标所在，如果你的目标定错了，之后改也

是可以的。但一定要设立某个目标，然后我建议你马上搭好一个机器学习系统原型，然后找到训练集，训练一下，看看效果，开始理解你的算法表现如何，在开发集测试集，你的评估指标上表现如何。当你建立第一个系统后，你就可以马上用到之前说的偏差方差分析，还有之前最后几个视频讨论的错误分析，来确定下一步优先做什么。特别是如果错误分析让你了解到大部分的错误的来源是说话人远离麦克风，这对语音识别构成特殊挑战，那么你就有很好的理由去集中精力研究这些技术，所谓远场语音识别的技术，这基本上就是处理说话人离麦克风很远的情况。

建立这个初始系统的所有意义在于，它可以是一个快速和粗糙的实现（quick and dirty implementation），你知道的，别想太多。初始系统的全部意义在于，有一个学习过的系统，有一个训练过的系统，让你确定偏差方差的范围，就可以知道下一步应该优先做什么，让你能够进行错误分析，可以观察一些错误，然后想出所有能走的方向，哪些是实际上最有希望的方向。



所以回顾一下，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。但如果你第一次处理某个新问题，那我真的不鼓励你想太多，或者把第一个系统弄得太复杂。我建议你们构建一些快速而粗糙的实现，然后用来帮你找到改善系统要优先处理的方向。我见过很多机器学习项目，我觉得有些团队的解决方案想太多了，他们造出了过于复杂的系统。我也见过有限团队想的不够，然后造出

过于简单的系统。平均来说，我见到更多的团队想太多，构建太复杂的系统。

所以我希望这些策略有帮助，如果你将机器学习算法应用到新的应用程序里，你的主要目标是弄出能用的系统，你的主要目标并不是发明全新的机器学习算法，这是完全不同的目标，那时你的目标应该是想出某种效果非常好的算法。所以我鼓励你们搭建快速而粗糙的实现，然后用它做偏差/方差分析，用它做错误分析，然后用分析结果确定下一步优先要做的方向。