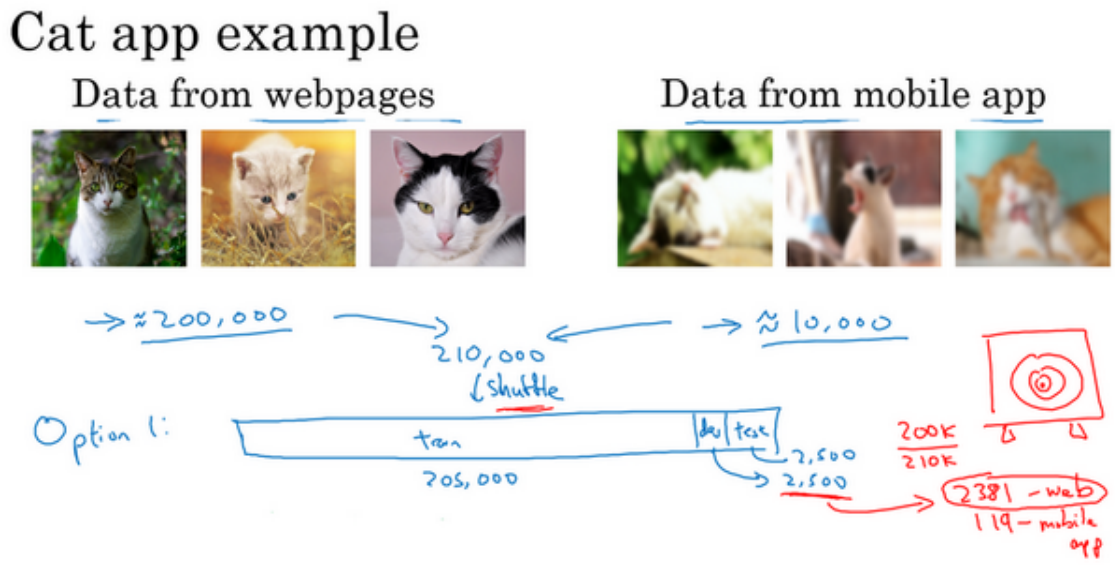
2.4 使用来自不同分布的数据进行训练和测试（Training and testing on different distributions）

深度学习算法对训练数据的胃口很大，当你收集到足够多带标签的数据构成训练集时，算法效果最好，这导致很多团队用尽一切办法收集数据，然后把它们堆到训练集里，让训练的数据量更大，即使有些数据，甚至是大部分数据都来自和开发集、测试集不同的分布。在深度学习时代，越来越多的团队都用来自和开发集、测试集分布不同的数据来训练，这里有一些微妙的地方，一些最佳做法来处理训练集和测试集存在差异的情况，我们来看看。

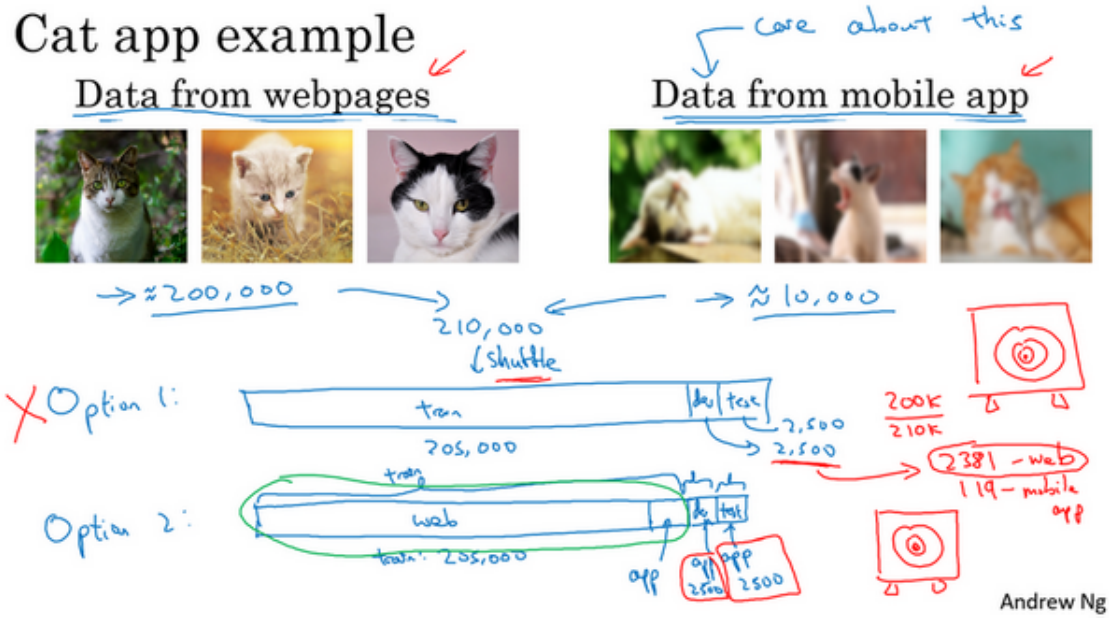
假设你在开发一个手机应用，用户会上传他们用手机拍摄的照片，你想识别用户从应用中上传的图片是不是猫。现在你有两个数据来源，一个是你真正关心的数据分布，来自应用上传的数据，比如右边的应用，这些照片一般更业余，取景不太好，有些甚至很模糊，因为它们都是业余用户拍的。另一个数据来源就是你可以用爬虫程序挖掘网页直接下载，就这个样本而言，可以下载很多取景专业、高分辨率、拍摄专业的猫图片。如果你的应用用户数还不多，也许你只收集到 10,000 张用户上传的照片，但通过爬虫挖掘网页，你可以下载到海量猫图，也许你从互联网上下载了超过 20 万张猫图。而你真正关心的算法表现是你的最终系统处理来自应用程序的这个图片分布时效果好不好，因为最后你的用户会上传类似右边这些图片，你的分类器必须在这个任务中表现良好。现在你就陷入困境了，因为你有一个相对小的数据集，只有 10,000 个样本来自那个分布，而你还有一个大得多的数据集来自另一个分布，图片的外观和你真正想要处理的并不一样。但你又不想直接用这 10,000 张图片，因为这样你的训练集就太小了，使用这 20 万张图片似乎有帮助。但是，困境在于，这 20 万张图片并不完全来自你想要的分布，那么你可以怎么做呢？

这里有一种选择，你可以做的一件事是将两组数据合并在一起，这样你就有 21 万张照片，你可以把这 21 万张照片随机分配到训练、开发和测试集中。为了说明观点，我们假设你已经确定开发集和测试集各包含 2500 个样本，所以你的训练集有 205000 个样本。现在

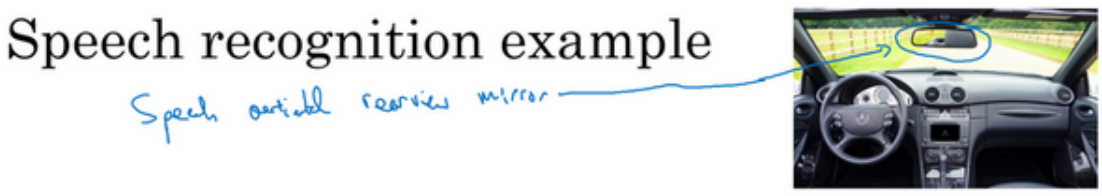
这么设立你的数据集有一些好处，也有坏处。好处在于，你的训练集、开发集和测试集都来自同一分布，这样更好管理。但坏处在于，这坏处还不小，就是如果你观察开发集，看看这 2500 个样本其中很多图片都来自网页下载的图片，那并不是你真正关心的数据分布，你真正要处理的是来自手机的图片。



所以结果你的数据总量，这 200,000 个样本，我就用$200k$缩写表示，我把那些是从网页下载的数据总量写成$210k$，所以对于这 2500 个样本，数学期望值是：$2500 \times \frac{200k}{210k} = 2381$，有 2381 张图来自网页下载，这是期望值，确切数目会变化，取决于具体的随机分配操作。但平均而言，只有 119 张图来自手机上传。要记住，设立开发集的目的是告诉你的团队去瞄准的目标，而你瞄准目标的方式，你的大部分精力都用在优化来自网页下载的图片，这其实不是你想要的。所以我真的不建议使用第一个选项，因为这样设立开发集就是告诉你的团队，针对不同于你实际关心的数据分布去优化，所以不要这么做。

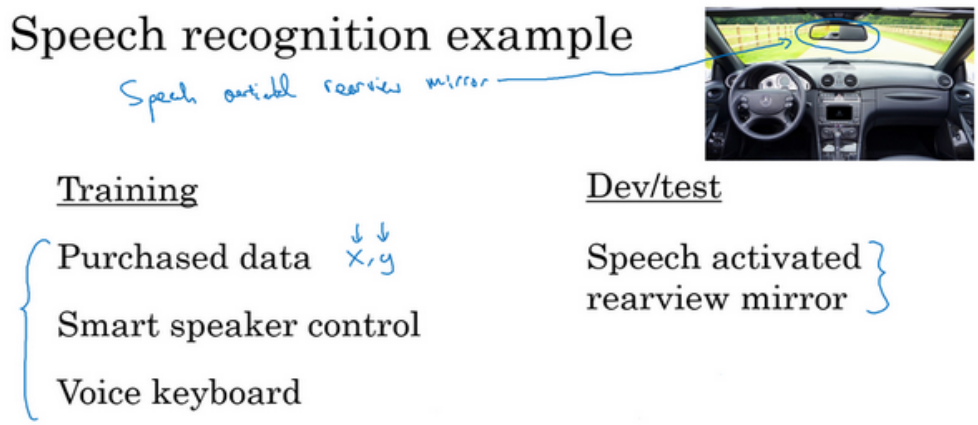


我建议你走另外一条路，就是这样，训练集，比如说还是 205,000 张图片，我们的训练集是来自网页下载的 200,000 张图片，然后如果需要的话，再加上 5000 张来自手机上传的图片。然后对于开发集和测试集，这数据集的大小是按比例画的，你的开发集和测试集都是手机图。而训练集包含了来自网页的 20 万张图片，还有 5000 张来自应用的图片，开发集就是 2500 张来自应用的图片，测试集也是 2500 张来自应用的图片。这样将数据分成训练集、开发集和测试集的好处在于，现在你瞄准的目标就是你想要处理的目标，你告诉你的团队，我的开发集包含的数据全部来自手机上传，这是你真正关心的图片分布。我们试试搭建一个学习系统，让系统在处理手机上传图片分布时效果良好。缺点在于，当然了，现在你的训练集分布和你的开发集、测试集分布并不一样。但事实证明，这样把数据分成训练、开发和测试集，在长期能给你带来更好的系统性能。我们以后会讨论一些特殊的技巧，可以处理 训练集的分布和开发集和测试集分布不一样的情况。



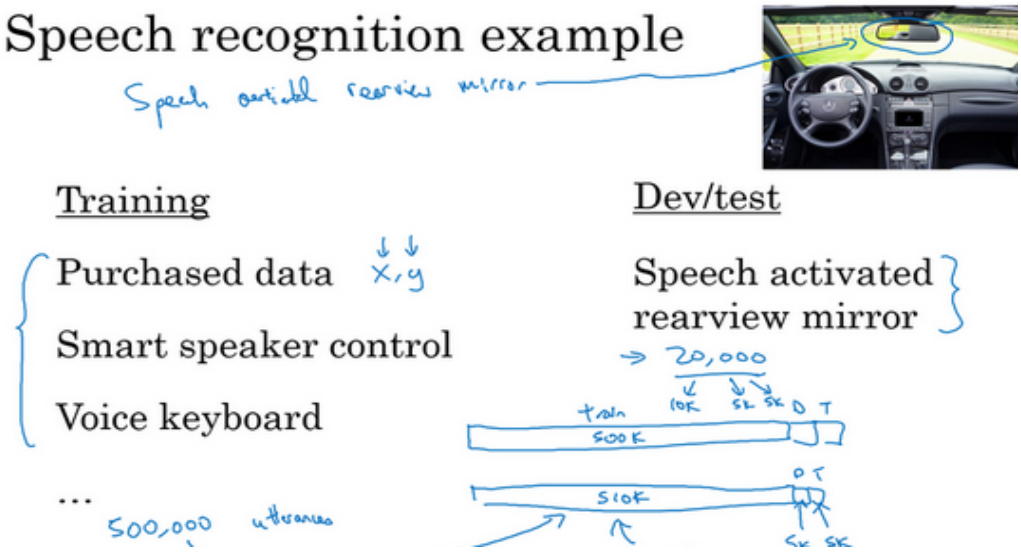
我们来看另一个样本，假设你正在开发一个全新的产品，一个语音激活汽车后视镜，这在中国是个真实存在的产品，它正在进入其他国家。但这就是造一个后视镜，把这个小东西换掉，现在你就可以和后视镜对话了，然后只需要说：“亲爱的后视镜，请帮我找找到最近的加油站的导航方向”，然后后视镜就会处理这个请求。所以这实际上是一个真正的产品，假设现在你要为你自己的国家研制这个产品，那么你怎么收集数据去训练这个产品语言识

别模块呢？



嗯，也许你已经在语音识别领域上工作了很久，所以你有很多来自其他语音识别应用的数据，它们并不是来自语音激活后视镜的数据。现在我讲讲如何分配训练集、开发集和测试集。对于你的训练集，你可以将你拥有的所有语音数据，从其他语音识别问题收集来的数据，比如这些年你从各种语音识别数据供应商买来的数据，今天你可以直接买到成x,y对的数据，其中x是音频剪辑，y是听写记录。或者也许你研究过智能音箱，语音激活音箱，所以你有一些数据，也许你做过语音激活键盘的开发之类的。

举例来说，也许你从这些来源收集了 500,000 段录音，对于你的开发集和测试集也许数据集小得多，比如实际上来自语音激活后视镜的数据。因为用户要查询导航信息或试图找到通往各个地方的路线，这个数据集可能会有很多街道地址，对吧？“请帮我导航到这个街道地址”，或者说：“请帮助我导航到这个加油站”，所以这个数据的分布和左边大不一样，但这真的是你关心的数据，因为这些数据是你的产品必须处理好的，所以你就应该把它设成你的开发和测试集。



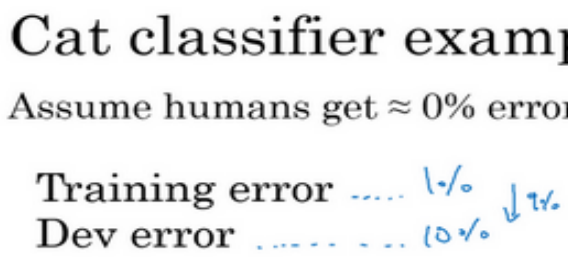
在这个样本中，你应该这样设立你的训练集，左边有 500,000 段语音，然后你的开发集和测试集，我把它简写成D和T，可能每个集包含 10,000 段语音，是从实际的语音激活后视镜收集的。或者换种方式，如果你觉得不需要将 20,000 段来自语音激活后视镜的录音全部放进开发和测试集，也许你可以拿一半，把它放在训练集里，那么训练集可能是 51 万段语音，包括来自那里的 50 万段语音，还有来自后视镜的 1 万段语音，然后开发集和测试集也许各自有 5000 段语音。所以有 2 万段语音，也许 1 万段语音放入了训练集，5000 放入开发集，5000 放入测试集。所以这是另一种将你的数据分成训练、开发和测试的方式。这样你的训练集大得多，大概有 50 万段语音，比只用语音激活后视镜数据作为训练集要大得多。

所以在这个视频中，你们见到几组样本，让你的训练集数据来自和开发集、测试集不同的分布，这样你就可以有更多的训练数据。在这些样本中，这将改善你的学习算法。

现在你可能会问，是不是应该把收集到的数据都用掉？答案很微妙，不一定都是肯定的答案，我们在下段视频看看一个反例。

2.5 数据分布不匹配时的偏差与方差的分析（Bias and Variance with mismatched data distributions）

估计学习算法的偏差和方差真的可以帮你确定接下来应该优先做的方向，但是，当你的训练集来自和开发集、测试集不同分布时，分析偏差和方差的方式可能不一样，我们来看为什么。



我们继续用猫分类器为例，我们说人类在这个任务上能做到几乎完美，所以贝叶斯错误率或者说贝叶斯最优错误率，我们知道这个问题里几乎是 0%。所以要进行错误率分析，你通常需要看训练误差，也要看看开发集的误差。比如说，在这个样本中，你的训练集误差是 1%，你的开发集误差是 10%，如果你的开发集来自和训练集一样的分布，你可能会说，这里存在很大的方差问题，你的算法不能很好的从训练集出发泛化，它处理训练集很好，但处理开发集就突然间效果很差了。

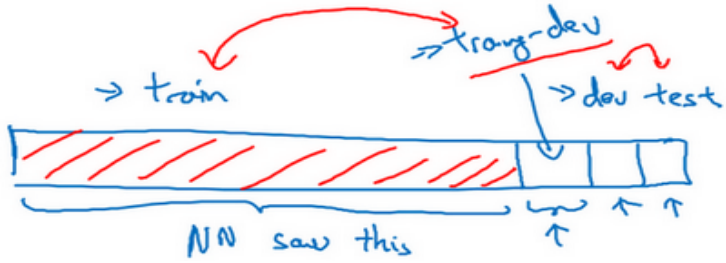
但如果你的训练数据和开发数据来自不同的分布，你就不能再放心下这个结论了。特别是，也许算法在开发集上做得不错，可能因为训练集很容易识别，因为训练集都是高分辨率图片，很清晰的图像，但开发集要难以识别得多。所以也许软件没有方差问题，这只不过反映了开发集包含更难准确分类的图片。所以这个分析的问题在于，当你看训练误差，再看开发误差，有两件事变了。首先算法只见过训练集数据，没见过开发集数据。第二，开发集数据来自不同的分布。而且因为你同时改变了两件事情，很难确认这增加的 9%误差率有多少是因为算法没看到开发集中的数据导致的，这是问题方差的部分，有多少是因为开发集数据就是不一样。

为了弄清楚哪个因素影响更大，如果你完全不懂这两种影响到底是什么，别担心我们马上会再讲一遍。但为了分辨清楚两个因素的影响，定义一组新的数据是有意义的，我们称之为训练-开发集，所以这是一个新的数据子集。我们应该从训练集的分布里挖出来，但你不会用来训练你的网络。

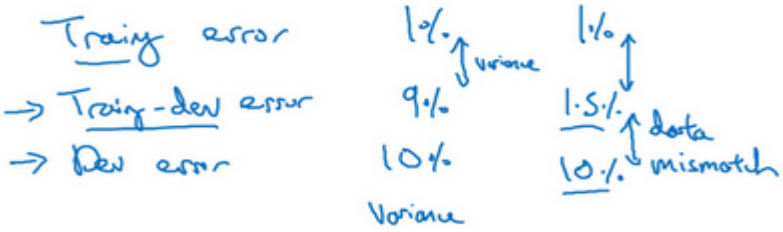
Training-dev set: Same distribution as training set, but not used for training

我的意思是我们已经设立过这样的训练集、开发集和测试集了，并且开发集和测试集来自相同的分布，但训练集来自不同的分布。

我们要做的是随机打散训练集，然后分出一部分训练集作为训练-开发集（training-dev），就像开发集和测试集来自同一分布，训练集、训练-开发集也来自同一分布。

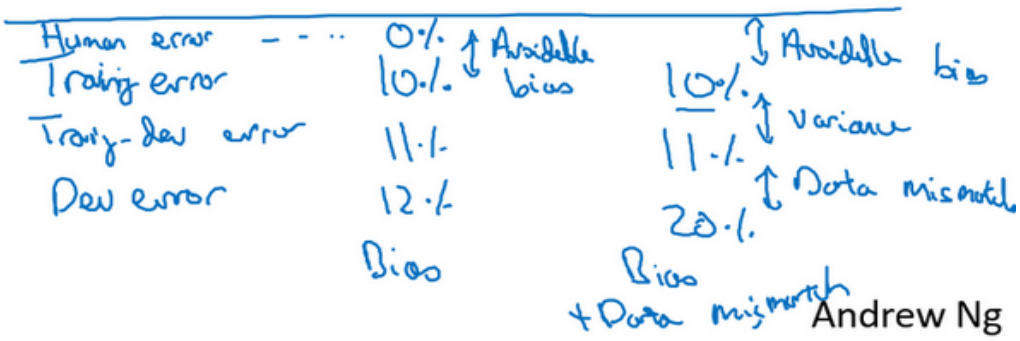


但不同的地方是，现在你只在训练集训练你的神经网络，你不会让神经网络在训练-开发集上跑后向传播。为了进行误差分析，你应该做的是看看分类器在训练集上的误差，训练-开发集上的误差，还有开发集上的误差。



比如说这个样本中，训练误差是 1%，我们说训练-开发集上的误差是 9%，然后开发集误差是 10%，和以前一样。你就可以从这里得到结论，当你从训练数据变到训练-开发集数据时，错误率真的上升了很多。而训练数据和训练-开发数据的差异在于，你的神经网络能看到第一部分数据并直接在上面做了训练，但没有在训练-开发集上直接训练，这就告诉你，算法存在方差问题，因为训练-开发集的错误率是在和训练集来自同一分布的数据中测得的。所以你知道，尽管你的神经网络在训练集中表现良好，但无法泛化到来自相同分布的训练-开发集里，它无法很好地泛化推广到来自同一分布，但以前没见过的数据中，所以在这个样本中我们确实有一个方差问题。

我们来看一个不同的样本，假设训练误差为 1%，训练-开发误差为 1.5%，但当你开始处理开发集时，错误率上升到 10%。现在你的方差问题就很小了，因为当你从见过的训练数据转到训练-开发集数据，神经网络还没有看到的数据，错误率只上升了一点点。但当你转到开发集时，错误率就大大上升了，所以这是数据不匹配的问题。因为你的学习算法没有直接在训练-开发集或者开发集训练过，但是这两个数据集来自不同的分布。但不管算法在学习什么，它在训练-开发集上做的很好，但开发集上做的不好，所以总之你的算法擅长处理和你关心的数据不同的分布，我们称之为数据不匹配的问题。

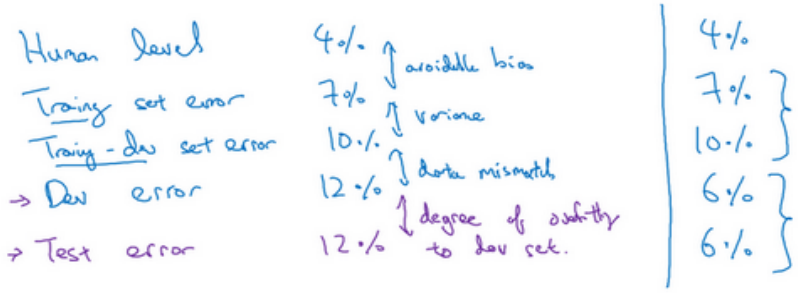


我们再来看几个样本，我会在下一行里写出来，因上面没空间了。所以训练误差、训练-开发误差、还有开发误差，我们说训练误差是 10%，训练-开发误差是 11%，开发误差为 12%，要记住，人类水平对贝叶斯错误率的估计大概是 0%，如果你得到了这种等级的表现，那就真的存在偏差问题了。存在可避免偏差问题，因为算法做的比人类水平差很多，所以这里的偏差真的很高。

最后一个例子，如果你的训练集错误率是 10%，你的训练-开发错误率是 11%，开发错误率是 20%，那么这其实有两个问题。第一，可避免偏差相当高，因为你在训练集上都没有做得很好，而人类能做到接近 0%错误率，但你的算法在训练集上错误率为 10%。这里方差似乎很小，但数据不匹配问题很大。所以对于这个样本，我说，如果你有很大的偏差或者可避免偏差问题，还有数据不匹配问题。

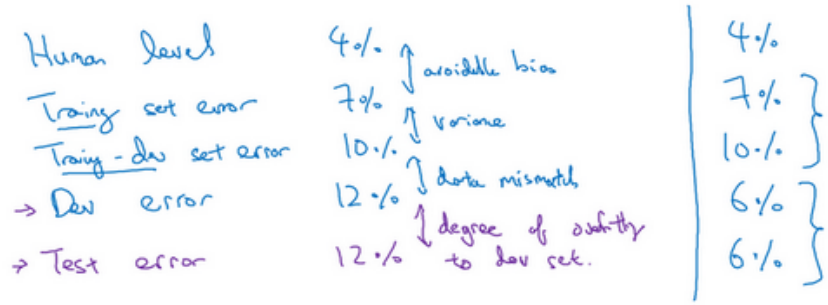
我们看看这张幻灯片里做了什么，然后写出一般的原则，我们要看的关键数据是人类水平错误率，你的训练集错误率，训练-开发集错误率，所以这分布和训练集一样，但你没有直接在上面训练。根据这些错误率之间差距有多大，你可以大概知道，可避免偏差、方差数据不匹配问题各自有多大。

Bias/variance on mismatched training and dev/test sets



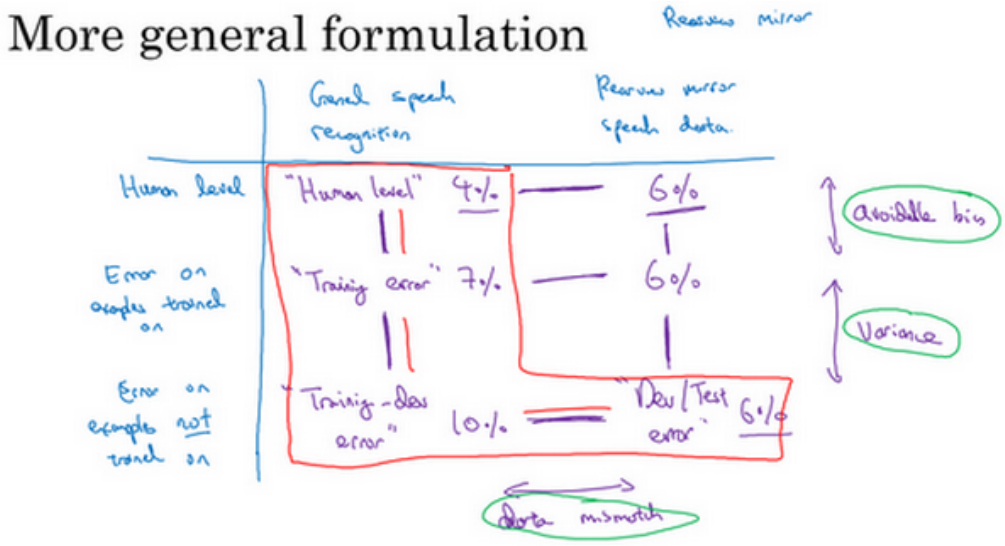
我们说人类水平错误率是 4%的话，你的训练错误率是 7%，而你的训练-开发错误率是 10%，而开发错误率是 12%，这样你就大概知道可避免偏差有多大。因为你知道，你希望你的算法至少要在训练集上的表现接近人类。而这大概表明了方差大小，所以你从训练集泛化推广到训练-开发集时效果如何？而这告诉你数据不匹配的问题大概有多大。技术上你还可以再加入一个数字，就是测试集表现，我们写成测试集错误率，你不应该在测试集上开发，因为你不希望对测试集过拟合。但如果你看看这个，那么这里的差距就说明你对开发集过拟合的程度。所以如果开发集表现和测试集表现有很大差距，那么你可能对开发集过拟合了，所以也许你需要一个更大的开发集，对吧？要记住，你的开发集和测试集来自同一分布，所以这里存在很大差距的话。如果算法在开发集上做的很好，比测试集好得多，那么你就可能对开发集过拟合了。如果是这种情况，那么你可能要往回退一步，然后收集更多开发集数据。现在我写出这些数字，这数字列表越往后数字越大。

Bias/variance on mismatched training and dev/test sets



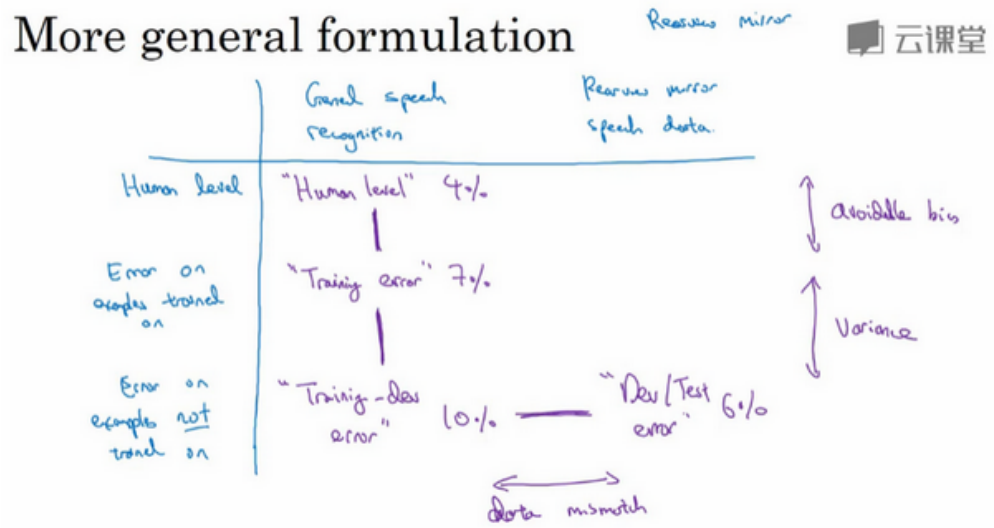
这里还有个例子，其中数字并没有一直变大，也许人类的表现是 4%，训练错误率是 7%，训练-开发错误率是 10%。但我们看看开发集，你发现，很意外，算法在开发集上做的更好，也许是 6%。所以如果你见到这种现象，比如说在处理语音识别任务时发现这样，其中训练数据其实比你的开发集和测试集难识别得多。所以这两个（7%，10%）是从训练集分布评估

的，而这两个（6%，6%）是从开发测试集分布评估的。所以有时候如果你的开发测试集分布比你应用实际处理的数据要容易得多，那么这些错误率可能真的会下降。所以如果你看到这样的有趣的事情，可能需要比这个分析更普适的分析，我在下一张幻灯片里快速解释一下。



所以，我们就以语音激活后视镜为例子，事实证明，我们一直写出的数字可以放到一张表里，在水平轴上，我要放入不同的数据集。比如说，你可能从一般语音识别任务里得到很多数据，所以你可能会有一堆数据，来自小型智能音箱的语音识别问题的数据，你购买的数据等等。然后你收集了和后视镜有关的语音数据，在车里录的。所以这是表格的x轴，不同的数据集。在另一条轴上，我要标记处理数据不同的方式或算法。

首先，人类水平，人类处理这些数据集时准确度是多少。然后这是神经网络训练过的数据集上达到的错误率，然后还有神经网络没有训练过的数据集上达到的错误率。所以结果我们上一张幻灯片说是人类水平的错误率，数字填入这个单元格里（第二行第二列），人类对这一类数据处理得有多好，比如来自各种语音识别系统的数据，那些进入你的训练集的成千上万的语音片段，而上一张幻灯片中的例子是 4%。这个数字（7%），可能是我们的训练错误率，在上一张幻灯片中的例子中是 7%。是的，如果你的学习算法见过这个样本，在这个样本上跑过梯度下降，这个样本来自你的训练集分布或一般的语音识别数据分布，你的算法在训练过的数据中表现如何呢？然后这就是训练-开发集错误率，通常来自这个分布的错误率会高一点，一般的语音识别数据，如果你的算法没在来自这个分布的样本上训练过，它的表现如何呢？这就是我们说的训练-开发集错误率。



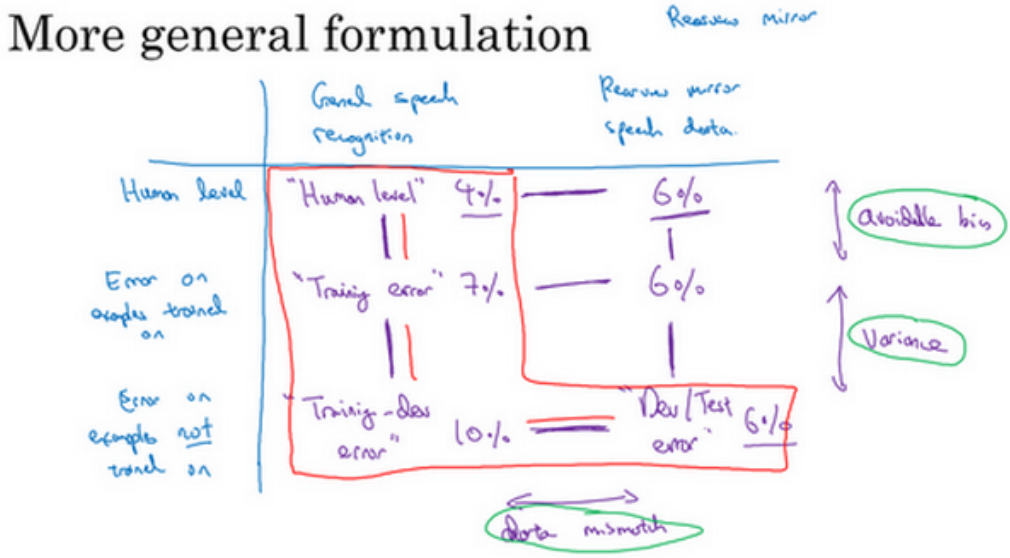
如果你移到右边去，这个单元格是开发集错误率，也可能是测试集错误，在刚刚的例子中是 6%。而开发集和测试集，实际上是两个数字，但都可以放入这个单元格里。如果你有来自后视镜的数据，来自从后视镜应用在车里实际录得的数据，但你的神经网络没有在这些数据上做过反向传播，那么错误率是多少呢？

我们在上一张幻灯片作的分析是观察这两个数字之间的差异（**Human level 4%**和 **Training error 7%**），还有这两个数字之间（**Training error 7%**和 **Training-dev error 10%**），这两个数字之间（**Training-dev error 10%**和 **Dev/Test dev 6%**）。这个差距（**Human level 4%**和 **Training error 7%**）衡量了可避免偏差大小，这个差距 **Training error 7%**和 **Training-dev error 10%**）衡量了方差大小，而这个差距（**Training-dev error 10%**和 **Dev/Test dev 6%**）衡量了数据不匹配问题的大小。

事实证明，把剩下的两个数字（**rearview mirror speech data 6%**和 **Error on examples trained on 6%**），也放到这个表格里也是有用的。如果结果这也是 6%，那么你获得这个数字的方式是你让一些人自己标记他们的后视镜语音识别数据，看看人类在这个任务里能做多好，也许结果也是 6%。做法就是，你收集一些后视镜语音识别数据，把它放在训练集中，让神经网络去学习，然后测量那个数据子集上的错误率，但如果你得到这样的结果，好吧，那就是说你已经在后视镜语音数据上达到人类水平了，所以也许你对那个数据分布做的已经不错了。

当你继续进行更多分析时，分析并不一定会给你指明一条前进道路，但有时候你可能洞察到一些特征。比如比较这两个数字（**General speech recognition Human level 4%**和 **rearview mirror speech data 6%**），告诉我们对于人类来说，后视镜的语音数据实际上比一般语音识别更难，因为人类都有 6%的错误，而不是 4%的错误，但看看这个差值，你就可以了解到偏差

和方差，还有数据不匹配这些问题的不同程度。所以更一般的分析方法是，我已经用过几次了。我还没用过，但对于很多问题来说检查这个子集的条目，看看这些差值，已经足够让你往相对有希望的方向前进了。但有时候填满整个表格，你可能会洞察到更多特征。

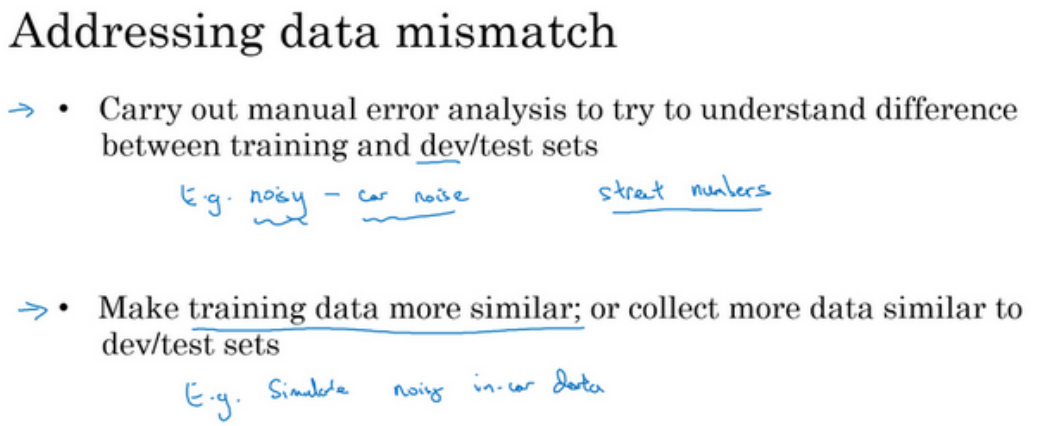


最后，我们以前讲过很多处理偏差的手段，讲过处理方差的手段，但怎么处理数据不匹配呢？特别是开发集、测试集和你的训练集数据来自不同分布时，这样可以用更多训练数据，真正帮你提高学习算法性能。但是，如果问题不仅来自偏差和方差，你现在又有了这个潜在的新问题，数据不匹配，有什么好办法可以处理数据不匹配的呢？实话说，并没有很通用，或者至少说是系统解决数据不匹配问题的方法，但你可以做一些尝试，可能会有帮助，我们在下一个视频里看看这些尝试。

所以我们讲了如何使用来自和开发集、测试集不同分布的训练数据，这可以给你提供更多训练数据，因此有助于提高你的学习算法的性能，但是，潜在问题就不只是偏差和方差问题，这样做会引入第三个潜在问题，数据不匹配。如果你做了错误分析，并发现数据不匹配是大量错误的来源，那么你怎么解决这个问题呢？但结果很不幸，并没有特别系统的方法去解决数据不匹配问题，但你可以做一些尝试，可能会有帮助，我们来看下一段视频。

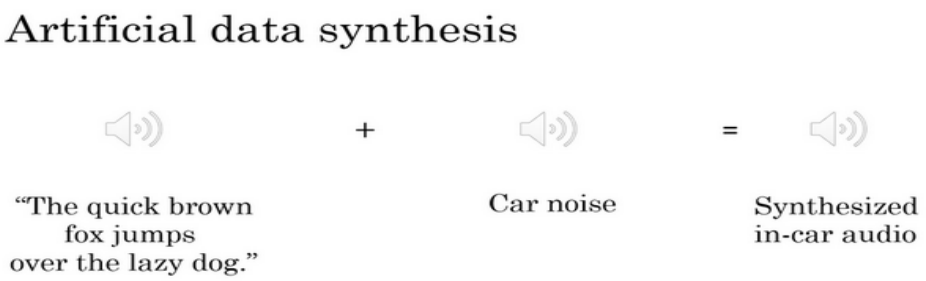
2.6 处理数据不匹配问题（Addressing data mismatch）

如果您的训练集来自和开发测试集不同的分布，如果错误分析显示你有一个数据不匹配的问题该怎么办？这个问题没有完全系统的解决方案，但我们可以看看一些可以尝试的事情。如果我发现有严重的数据不匹配问题，我通常会亲自做错误分析，尝试了解训练集和开发测试集的具体差异。技术上，为了避免对测试集过拟合，要做错误分析，你应该人工去看开发集而不是测试集。



但作为一个具体的例子，如果你正在开发一个语音激活的后视镜应用，你可能要看看……我想如果是语音的话，你可能要听一下来自开发集的样本，尝试弄清楚开发集和训练集到底有什么不同。所以，比如说你可能会发现很多开发集样本噪音很多，有很多汽车噪音，这是你的开发集和训练集差异之一。也许你还会发现其他错误，比如在你的车子里的语言激活后视镜，你发现它可能经常识别错误街道号码，因为那里有很多导航请求都有街道地址，所以得到正确的街道号码真的很重要。当你了解开发集误差的性质时，你就知道，开发集有可能跟训练集不同或者更难识别，那么你可以尝试把训练数据变得更像开发集一点，或者，你也可以收集更多类似你的开发集和测试集的数据。所以，比如说，如果你发现车辆背景噪音是主要的错误来源，那么你可以模拟车辆噪声数据，我会在下一张幻灯片里详细讨论这个问题。或者你发现很难识别街道号码，也许你可以有意识地收集更多人们说数字的音频数据，加到你的训练集里。

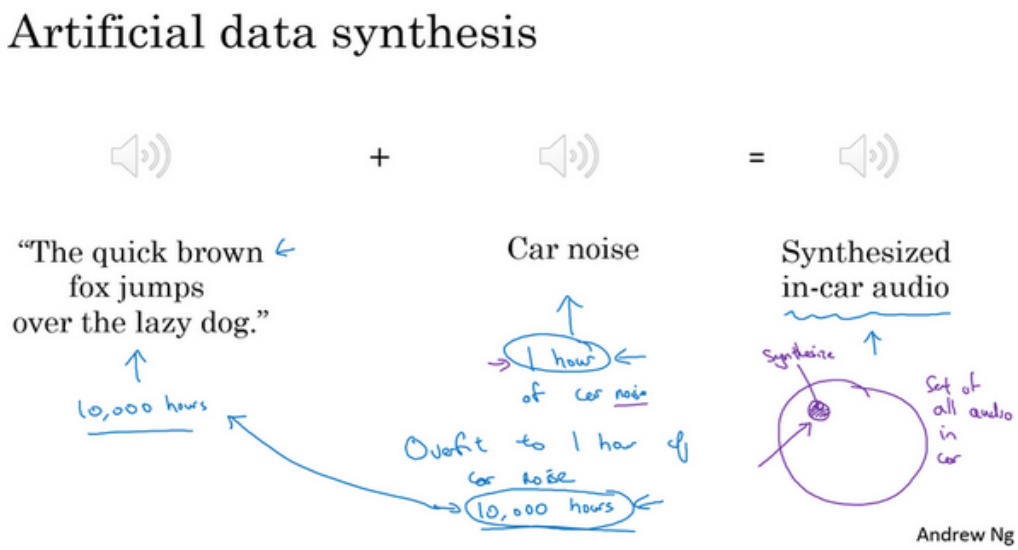
现在我知道这张幻灯片只给出了粗略的指南，列出一些你可以做的尝试，这不是一个系统化的过程，我想，这不能保证你一定能取得进展。但我发现这种人工见解，我们可以一起尝试收集更多和真正重要的场合相似的数据，这通常有助于解决很多问题。所以，如果你的目标是让训练数据更接近你的开发集，那么你可以怎么做呢？



你可以利用的其中一种技术是人工合成数据（artificial data synthesis），我们讨论一下。在解决汽车噪音问题的场合，所以要建立语音识别系统。也许实际上你没那么多实际在汽车背景噪音下录得的音频，或者在高速公路背景噪音下录得的音频。但我们发现，你可以合成。所以假设你录制了大量清晰的音频，不带车辆背景噪音的音频，**"The quick brown fox jumps over the lazy dog"**（音频播放），所以，这可能是你的训练集里的一段音频，顺便说一下，这个句子在 AI 测试中经常使用，因为这个短句包含了从 a 到 z 所有字母，所以你会经常见到这个句子。但是，有了这个**"the quick brown fox jumps over the lazy dog"**这段录音之后，你也可以收集一段这样的汽车噪音，（播放汽车噪音音频）这就是汽车内部的背景噪音，如果你一言不发开车的话，就是这种声音。如果你把两个音频片段放到一起，你就可以合成出**"the quick brown fox jumps over the lazy dog"**（带有汽车噪声），在汽车背景噪音中的效果，听起来像这样，所以这是一个相对简单的音频合成例子。在实践中，你可能会合成其他音频效果，比如混响，就是声音从汽车内壁上反弹叠加的效果。

但是通过人工数据合成，你可以快速制造更多的训练数据，就像真的在车里录的那样，那就不需要花时间实际出去收集数据，比如说在实际行驶中的车子，录下上万小时的音频。所以，如果错误分析显示你应该尝试让你的数据听起来更像在车里录的，那么人工合成那种音频，然后喂给你的机器学习算法，这样做是合理的。

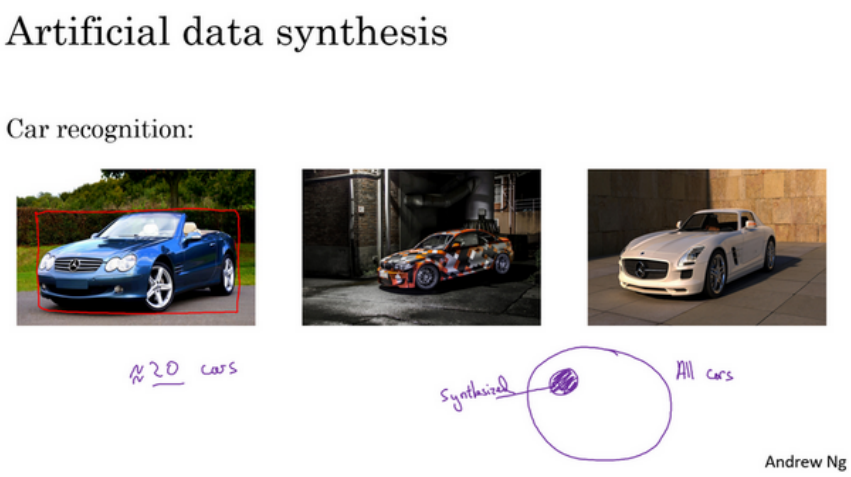
现在我们要提醒一下，人工数据合成有一个潜在问题，比如说，你在安静的背景里录得 10,000 小时音频数据，然后，比如说，你只录了一小时车辆背景噪音，那么，你可以这么做，将这 1 小时汽车噪音回放 10,000 次，并叠加到在安静的背景下录得的 10,000 小时数据。如果你这么做了，人听起来这个音频没什么问题。但是有一个风险，有可能你的学习算法对这 1 小时汽车噪音过拟合。特别是，如果这组汽车里录的音频可能是你可以想象的所有汽车噪音背景的集合，如果你只录了一小时汽车噪音，那你可能只模拟了全部数据空间的一小部分，你可能只从汽车噪音的很小的子集来合成数据。



而对于人耳来说，这些音频听起来没什么问题，因为一小时的车辆噪音对人耳来说，听起来和其他任意一小时车辆噪音是一样的。但你有可能从这整个空间很小的一个子集出发合成数据，神经网络最后可能对你这一小时汽车噪音过拟合。我不知道以较低成本收集 10,000 小时的汽车噪音是否可行，这样你就不用一遍又一遍地回放那 1 小时汽车噪音，你就有 10,000 个小时永不重复的汽车噪音来叠加到 10,000 小时安静背景下录得的永不重复的语音录音。这是可以做的，但不保证能做。但是使用 10,000 小时永不重复的汽车噪音，而不是 1 小时重复学习，算法有可能取得更好的性能。人工数据合成的挑战在于，人耳的话，人耳是无法分辨这 10,000 个小时听起来和那 1 小时没什么区别，所以你最后可能会制造出这个原始数据很少的，在一个小得多的空间子集合成的训练数据，但你自己没意识到。

这里有人工合成数据的另一个例子，假设你在研发无人驾驶汽车，你可能希望检测出这样的车，然后用这样的框包住它。很多人都讨论过的一个思路是，为什么不用计算机合成图像来模拟成千上万的车辆呢？事实上，这里有几张车辆照片（下图后两张图片），其实是用计算机合成的，我想这个合成是相当逼真的，我想通过这样合成图片，你可以训练出一个相

当不错的计算机视觉系统来检测车子。



不幸的是，上一张幻灯片介绍的情况也会在这里出现，比如这是所有车的集合，如果你只合成这些车中很小的子集，对于人眼来说也许这样合成图像没什么问题，但你的学习算法可能会对合成的这一个小子集过拟合。特别是很多人都独立提出了一个想法，一旦你找到一个电脑游戏，里面车辆渲染的画面很逼真，那么就可以截图，得到数量巨大的汽车图片数据集。事实证明，如果你仔细观察一个视频游戏，如果这个游戏只有 20 辆独立的车，那么这游戏看起来还行。因为你是在游戏里开车，你只看到这 20 辆车，这个模拟看起来相当逼真。但现实世界里车辆的设计可不只 20 种，如果你用着 20 量独特的车合成的照片去训练系统，那么你的神经网络很可能对这 20 辆车过拟合，但人类很难分辨出来。即使这些图像看起来很逼真，你可能真的只用了所有可能出现的车辆的很小的子集。

所以，总而言之，如果你认为存在数据不匹配问题，我建议你做错误分析，或者看看训练集，或者看看开发集，试图找出，试图了解这两个数据分布到底有什么不同，然后看看是否有办法收集更多看起来像开发集的数据作训练。

我们谈到其中一种办法是人工数据合成，人工数据合成确实有效。在语音识别中。我已经看到人工数据合成显著提升了已经非常好的语音识别系统的表现，所以这是可行的。但当你使用人工数据合成时，一定要谨慎，要记住你有可能从所有可能性的空间只选了很小一部分去模拟数据。

所以这就是如何处理数据不匹配问题，接下来，我想和你分享一些想法就是如何从多种类型的数据同时学习。