机器学习纪元

--在深度学习浪潮中,为AI攻城狮提供指路明灯

目录：

机器学习策略之源头

1. 为何选择机器学习策略

机器学习是无数重要应用程序的基础，包括网络搜索,垃圾邮件过滤,语音识别,商品推荐等等.假如你或者你的团队正在制作机器学习应用程序，并且您想要获得快速进展。本书将帮助你做到这一点。

例子：从一张猫图创业

假如你正在建立一家初创公司,为撸猫患者提供无尽的猫咪图片.你使用神经网络来构建计算机视觉系统来检测图片中的猫。但悲惨的是，你的学习算法的准确性还不够好。你急需改善你的猫咪探测器。你该怎么办？

你的团队有很多预案：

1. 获得更多数据：收集更多的猫图
2. 收集更多样化的训练集：例如，不同位置的猫，不同颜色的猫，不同相机视角下的猫...
3. 用更长的时间来训练，运行更多的梯度下降，以接近全局最优
4. 试用一个更大的神经网络，有更多的层,隐层,参数...
5. 试用一个更小的神经网络
6. 尝试加入正则化参数（例如L2正则项）
7. 调整网络结构(激活函数，隐层数量...)

... ...

在诸多选择中，如果选择得当，你会建立一个优秀的猫图平台，走上人生巅峰。如果选择失误，可能白白浪费几个月。你该何去何从？

这本书将会解决你的疑问。大多数机器学习问题中已经留下了诸多线索，告诉你什么才是有用的尝试，什么才是无用的尝试。学到这些线索，将会节省你数月甚至是数年的开发时间。

1. 如何使用这本书来帮助你的团队

看完本书后，你将对如何为机器学习项目设定技术方向有深入的了解。但是你的队友可能不明白你为什么推荐这个特定的方向。也许你希望你的团队定义一个单一数量的评估指标，但他们不信服。你如何说服他们呢？

这就是为什么我将章节缩短的原因：这样你就可以打印出来，让你的队友阅读你所需要的1-2页。

事物优先级的小变化将会使你的团队有更高的研发效率。通过帮助你的团队进行这样一些改变，我希望你能成为团队中的超级英雄！



1. 先决条件和标识

如果您在Coursera上学过机器学习课程，例如我的机器学习MOOC，或者您有应用监督式学习的经验，您将能够理解本文。

我假设你了解监督学习：使用已标记的训练样本(x,y)来学习一个从x映射到y的函数。监督学习算法包括线性回归,逻辑回归和神经网络。机器学习有很多形式，但今天大部分机器学习的实用价值都来自监督式学习。

我会经常提到神经网络(也称为深度学习).你只需要了解他们遵循本文的基本知识。

如果您不熟悉这里提到的概念，请在Coursera的机器学习课程http://ml-class.org观看前三周的视频。

1. 量级推动机器学习进程发展

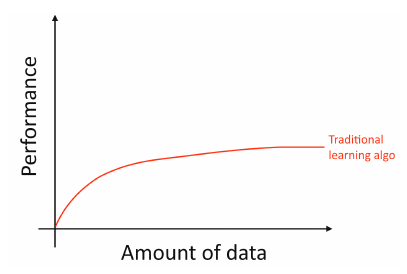
深度学习(神经网络)的很多想法都已经被提出了数十年。为什么现在这些开始火爆了呢？

最近进展的两个最大推动力是：

数据的可获得性。人们现在花费大量的时间在数字设备(笔记本电脑,移动设备)上.他们的产生的大量数据，可供我们提供给我们的学习算法。

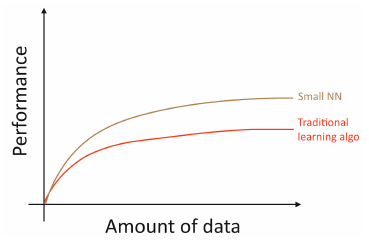
计算规模。我们也仅仅是在几年前才能开始着手培养足够大的神经网络来利用我们现有的庞大数据集。

详细来讲，即使你拥有大量的数据，通常的老式学习算法，例如logistic回归，将会慢慢平稳。它的学习曲线趋于变平，即使给更多数据，它也将止步于此时的最优。

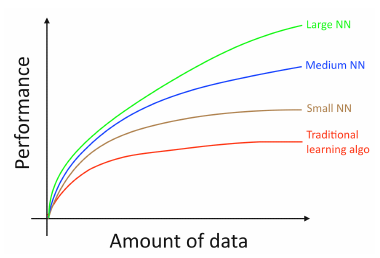


这就好像旧算法不知道如何处理我们现在拥有的所有数据一样。

如果你在同一监督学习任务上训练一个小型神经网络(NN)，你可能会获得更佳的表现：



此时，“小NN”是指只有少量隐藏单元/层/参数的神经网络。 最后，如果你训练越来越大的神经网络，你可以获得更好的表现：



该图显示了神经网络在小数据集中的效果更好。 这种效应不如神经网络在巨大数据集体系中表现良好的效果一致。 在小数据体系中，取决于这些功能是如何手工设计的，传统算法可能会或可能不会做得更好。 例如，如果您有20个培训示例，则使用逻辑回归还是神经网络可能无关紧要; 特征的手工设计将比算法的选择产生更大的影响。 但是如果你有100万个例子，我会赞成神经网络。

因此，当您(i)训练一个非常大的神经网络时，您可以获得最佳性能，以便您处于上面的绿色曲线上;(ii)有大量的数据。

很多其他细节如网络结构也很重要，且已经有很多创新。然而，此时提高算法性能的更可靠的方法仍然是：(i)训练更大的网络(ii)获得更多的数据。

如何完成（i）和（ii）的过程非常复杂。本书将详细讨论细节。我们将从对传统学习算法和神经网络都有用的一般策略开始，并讲述构建深度学习系统的最现代策略。

# 设置开发和测试集

1. 你的开发和测试集

让我们回到我们之前的猫图示例：您运营一个移动应用程序，并且用户正在向您的应用程序上传许多不同内容的图片。你想自动找到猫的图片。

您的团队通过下载不同网站上的猫（正面示例）和非猫（反面示例）的图片获得大量训练集。他们将数据集70％/ 30％分解为训练和测试集。使用这些数据，他们构建了一个在训练和测试集上运行良好的猫检测器。

但是当您将此分类器部署到移动应用程序中时，您会发现性能非常差！



发生了什么？

您发现用户上传的图片与构成您的训练集的网站图片有着不同的外观：用户正在上传使用手机拍摄的照片，这些照片往往分辨率较低，模糊不清，并且照明理想度较低。 由于您的训练/测试集是由网站图片组成的，因此您的算法没有很好地泛化并关心智能手机图片的实际分布。

在现代大数据时代之前，机器学习中的一个常见规则是使用随机70％/ 30％的分割形式来组成您的训练和测试集。这种做法可行，但在越来越多的应用程序中，如果培训分布（我们上面示例中的网站图像）与您最终关心的分布（手机图像）不同，这是一个糟糕的主意。

我们通常定义：

训练集---运行你的学习算法。

开发集---用来调参，选择特征并针对学习算法做出其他决定。有时也称为交叉验证集。

测试集---使用它来评估算法的性能，但不会根据此对参数等做出调整。

你可以定义一个开发集（开发集）和测试集，你的团队会尝试很多想法，比如不同的学习算法参数，看看哪些最好。开发和测试集可以让您的团队快速了解您的算法做得如何。

换句话说，开发和测试集的目的是引导您的团队走向机器学习系统的最重要变化。

所以，你要做到以下所说：

选择的开发和测试集要反映您期望在未来获得的数据，并希望做得更好。

换句话说，你的测试集不应该只是可用数据的30％，特别是如果你期望你的未来数据(移动应用程序图像)在本质上与你的训练集(网站图像)不同。

如果您尚未推出移动应用程序，那么您可能还没有任何用户，因此可能无法获取准确反映您将来需要做得更好的数据。但是你仍然可以尝试这个方法。例如，请你的朋友拍下手机照片并发给你。一旦您的应用程序启动，您可以使用实际用户数据更新您的开发/测试集。

如果你真的没有办法获得接近未来预期的数据，也许你可以从网站图片开始。但是你应该意识到导致系统不能很好地泛化的风险。

在开发大型的开发和测试集上投入多少资金，需要你的判断来决定。但不要认为你的培训分布与你的测试分布相同。试着挑选能够反映出你最终想要表现得好的测试例子，而不是随意选择你用来训练的任何数据。

1. 你的开发和测试集应该来自同一分布

根据您最大的市场，您可以将您的猫咪应用图像数据分为四个区域：(i)美国，(ii)中国，(iii)印度和(iv)其他。为了得到一个开发集和一个测试集，我们可以随机将这两个片段分配给开发集，另外两个分配给测试集，对吧？在开发集中有美国和印度; 中国和其他在测试集。



一旦您定义了开发和测试集，您的团队将专注于提高算法在开发集上的性能。因此，开发集应该反映出你最想改进的任务：在四个地区都做得很好，而不仅仅是两个。

开发和测试集具有不同的分布将会带来第二个问题：你的团队可能在开发集上构建了一些运行良好的东西，但发现它在测试集上表现不佳。我已经预见到这种结果将会使人感到非常沮丧和浪费精力。避免让这发生在你身上。

作为一个例子，假设你的团队开发了一套在开发集上运行良好的系统，而测试集却并非如此。如果你的开发集和测试集来自同一分布，那么你会非常清楚问题出在了哪里：对开发集过拟合了。常规解决方案是增加开发集数据量。

但是如果开发集和测试集来自不同的分布，那么问题就不那么清晰了。问题的起源可能是：

1. 开发集过拟合。
2. 测试集比开发集更难。所以你的算法可能做得和预期的一样好，已经没有可能再有重大改进了。
3. 测试集可能并不比开发集更难，只是特征不同罢了。所以在开发集适应良好的算法在测试集上表现较差，无可厚非。在此例中，你基于开发集所做的大部分努力和适应都相当于白费功夫了。

在机器学习应用上工作很困难。不匹配的开发和测试集引入了很多的不确定性，你无法确定是否提高开发集性能就能提高算法在测试集上的性能。如果开发和测试集不匹配，就很难弄清楚到底哪些因素起作用，哪些不起作用，从而很难分辨哪些工作应该被优先考虑。

如果你在处理第三方基准问题，他们的创建者可能指定了不同分布的开发和测试集。运气，而并非技术，将会对你研究用来比较开发和测试集是否来自同一分布的基准起着更大作用。这是一个重要的研究问题，即开发学习算法，在一个分布上训练，然后能够很好地推广到另一个分布。但是，如果你的目标是在特定的机器学习应用上取得进展，而不是进行研究。那么我建议你选择相同分布的开发和测试集。这将会使你的团队更有效率。

1. 开发/测试集的大小如何确定

开发集应该大到能够检测到你正在尝试的算法之间的差异。例如，如果分类器A具有90.0％的准确度而分类器B具有90.1％的准确度，则100个示例的开发集将不能够检测到这个0.1％的差异。且与我见过的其他机器学习问题相比，100个示例的开发集非常的小。开发集被设置为1000-10000都是很常见的。通过10000个示例，你将很可能检测到这0.1%的改进。

理论上，还可以测试算法的变化是否会在开发集上产生统计上的显着差异。在实践中，大多数团队不关心这个（除非他们正在发表学术研究论文），而且我通常没有发现对测量中期进展有用的统计显着性检验。

对于成熟和重要的应用(例如广告投放,网络搜索和产品推荐)，我也看到有为了0.01%提升而去努力的团队，因为它直接影响到了公司的利润。在这种情况下，开发集可能远远超过10000，以便能够检测到更小的改进。

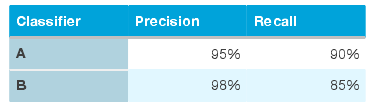
那么测试集的大小呢？它应该大到能让我们相信系统整体性能的可用性。一种流行的启发式方式是将数据的30%用于测试集。这适用于你只有少量示例的情况下--例如100到10000个示例。但是在大数据时代，我们现有的机器学习问题，有时候会有十亿多个例子，分配给开发/测试集的数据的比例尽管在减少，但是开发/测试集中数据的绝对数量却在增加。我们没有必要去建立超出算法性能评估所需的数据量的开发/测试集。

1. 为你的团队建立一个单一评估指标用于优化

分类准确性是单一评估指标的一个例子：你可以在开发集(或测试集)上运行分类器，然后获取关于正确分类的示例个数。根据这个度量，如果分类器A获得97％的准确度，并且分类器B获得90％的准确度，那么我们判断分类器A是优越的。

相比之下，精确率和召回率不是一个单一评估指标：它给出了两个数字来评估你的分类器。拥有多个评估指标使得比较算法变得更加困难。假设你的算法表现如下：

猫分类器的精确率是开发(或测试)集中图像的一小部分，即将其标记为猫的图片中到底有多少只猫的比例。其召回率是开发(或测试)集中所有是猫的图像到底有多少被识别(召回)了出来的比例。在高精确率和高召回率之间经常进行权衡。

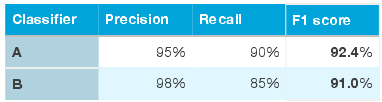


在这里，两个分类器都没有明显的优势，所以它不会立即指导你选择哪一个。

在开发过程中，你的团队将尝试许多有关算法体系结构，模型参数，特征选择等想法。使用单一评估指标(如准确度)可以让你根据此指标对素有模型进行排序，快速决定什么起作用最大。

如果你真的关心精确率和召回率，我推荐使用一个标准方法将它们合并为一个数字。例如，你可以取精确率和召回率的和的平均值作为一个数字。或者，你可以计算’F1值’，这是通过计算它们的调和均值，比单一的计算均值效果更优。

If you want to learn more about the F1 score, see https://en.wikipedia.org/wiki/F1\_score. It is the “geometric mean” between Precision and Recall, and is calculated as 2/((1/Precision)+(1/Recall)).



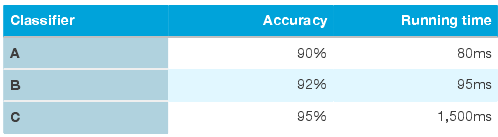
当你在大量的分类器中进行选择时，使用单一评估指标可以加快你做出决定的能力。它在所有这些分类器中给出了明确的偏好排序，因此可以十分明确地进行挑选。

作为最后一个例子，假设你分别在四个关键市场追踪猫分类器的准确性: (i)美国, (ii)中国, (iii)印度和(iv)其他。这里我们给出四个指标。通过对这四个数字进行平均或加权平均，你将得到一个单一的数字度量。取平均或者加权平均是将多个指标合并为一个的最常用的方法之一。

1. 优化和满足指标

这是组合多个评估指标的另一种方法。

假设你关系学习算法的准确性和运行时间，你需要从下列三个分类器中作出选择：



通过将准确性和运行时间放入单个公式中来导出某个单一评估指标似乎不太自然。例如：

Accuracy - 0.5\*Running Time

你可以用以下方法来进行计算：首先，定义什么是”可接受的”运行时间。可以说任何在100ms内运行的东西都是可以接受的。然后，从满足运行时间标准的分类器中，找出具有最大的准确性的。在这里，运行时间是一个”满足指标”-你的分类器必须要在这个指标下表现良好，即它应满足最多需要100ms运行时间。准确性是”优化指标”。

如果你具有N中不同的指标，例如模块的二进制文件的大小(这对移动应用程序非常重要，因为用户不想下载大型应用程序),运行时间和准确性，你可以设置其中的N-1个指标作为”满足指标”。也就是说，你只是要求他们达到这些指标中的某些确定的程度。然后将最后一个指标定义为”优化指标”。例如，为二进制文件大小和运行时间设置可接受的阈值，并尝试根据这些限制来优化准确性。

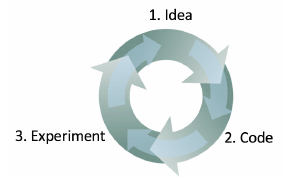
作为最后一个例子，假设你在构建一个硬件设备，该设备使用麦克风来监听用户说特定的”唤醒词”，然后唤醒系统。例子包括Amazon Echo倾听”Alexa”; Apple Siri倾听”Hey Siri”; Android听”Okay Google”;和百度应用程序在倾听”Hello Baidu”。您关心的是假阳性率-即使没有人说唤醒词时系统醒来的频率-以及假阴性率-当有人说唤醒词时它没有醒来的次数。该系统性能的一个合理目标是最大程度地减少假阴性率(优化指标)，但每运行24小时不能有一个以上的假阳性误报(满足指标)。

如果你的团队按照这些指标进行优化，他们将会取得更快的进展。

1. 通过开发集和指标来加速迭代

在解决新问题时，事先找到最适合的方法是非常困难的。即使有经验的机器学习研究人员通常会在试验很多想法之后才能发现令人满意的东西。在建立机器学习系统时，我会经常：

1. 首先去想如何构建系统。
2. 用代码实现想法。
3. 进行实验，测试这个想法有多奏效。(通常我的第一个想法不起作用！)基于这些学习，回过头来产生更多想法，并继续迭代。



这是一个迭代过程。你在这个循环上进展越快，你的项目进展就越快。这就体现了为什么开发/测试集和度量指标那么重要：每次当你尝试一个想法时，在开发集上衡量指标能够让你快速知道是否正朝着正确的方向前进。

相反，假设你没有特定的开发集和度量指标。因此，每当你的团队开发出新的猫分类器时，你都必须将其整合到你的应用中，并在运行几个小时后才能获知是否新分类器有所改进。这会非常的慢！另外，如果你的团队将分类器的准确度从95.0％提高到95.1％，那么在运行应用的过程中可能无法检测到这0.1%的提升。然而，通过逐步累积这些0.1%的改进效果，你的系统将会取得更大的进步。拥有开发集和度量指标，可以让你快速发现哪些想法能够成功实现小型（或大型）改进，因此可以让你快速决定要继续提炼的内容以及要抛弃的内容。

1. 何时更改开发/测试集和指标

当开始一个新项目时，我试着快速选择开发/测试集，因为这给了团队一个明确定义的目标。

我通常会要求我的团队在不到一周的时间内提供初始开发/测试集和初始指标 - 几乎不会更长。最好能提出一些不完美的东西，并迅速行动起来，而不是长时间的思考。但是这一周的时间限并不适用于成熟的应用程序。 例如，反垃圾邮件是一种成熟的深度学习应用。 我看到在已经成熟的系统上工作的团队花费数月的时间来获得更好的开发/测试集。

如果您后来意识到您的初始开发/测试集或度量指标没有达到要求，那么一定要尽快更改它们。 例如，如果你的开发集+度量指标将分类器A排序在分类器B之上，但你的团队认为分类器B对于你的产品来说确实优越，那么这可能表示你需要更改开发/测试集或你的评估指标。

开发集/度量指标有三个主要可能原因导致将分类器A的评分更高：

1. 你实际想要运营的数据和开发/测试集数据的分布不同。

假设你的初始开发/测试集主要是成年猫的图片。你上线你的猫应用，然后发现用户比预期中上传更多的小猫图片。所以，开发/测试集并不代表你需要应用的数据的实际分布。在这种情况下，需要更新你的开发/测试集使其更具代表性。



1. 对开发集过拟合了。

在开发集上重复评估想法的过程会导致你的算法逐渐对于开发集”过拟合”。当完成开发后，你将会在测试集上评估你的系统。如果你发现你的开发集性能比你的测试集性能好的多，那么这是你已经过拟合的一个迹象。在这种情况下，更换一个新的开发集吧。

如果你需要跟踪团队的进度，你还可以定期评估你的系统-例如每周一次或每月一次-在测试集上。但不要使用测试集对算法进行任何决断，包括是否回滚到前一周的系统。如果你这样做了，你将开始过拟合你的测试集，并且不能再依靠它来对你的系统性能进行完全无偏的估计(如果你发布研究论文或者做出重要的商业决策时，这条能够用到)。

1. 指标只是评估并非项目的优化方向

假设对于你的猫应用，你的指标是分类精确度。这个指标目前将分类器A评为优于分类器B。但是，你尝试了两种算法，并且发现分类器A偶尔允许色情图片通过，尽管分类器A更准确，偶尔的色情图像留下的坏印象意味着它的表现是不可接受的。你将怎么做？

在这里，该度量指标未能确定算法B实际上比算法A更加适用于你的产品。所以，你不能再相信这个指标能够选择出最好的算法。现在是改变评估指标的时候了。例如，你可以将指标更改为严重惩罚通过色情图片。我强烈选择一个新的度量指标，并使用新的度量指标明确定义团队的新目标，而不是在没有可信的度量指标的情况下工作太久，然后转而通过手动选择分类器。

在项目中更改开发/测试集或评估指标是非常常见的。拥有初始的开发/测试集和指标能够帮助你快速迭代。如果你发现开发/测试集或指标不再指引你的团队向正确的方向发展，那不是什么大不了的事情！只要改变它们，并且确保你的团队知道新的方向。

1. 小贴士：建立开发和测试集

从你将要运用的真实数据的分布中挑选开发和测试集。这可能与你的训练集数据的分布不一致。

如果可能，请从同一分布中选择开发和测试集。

为你的团队选择一个单一评估指标进行优化。如果你有多个指标想要关心，尝试将它们组合成一个公式(例如平均多个误差度量指标)或定义满足指标和优化指标。

机器学习是一个高速迭代的过程：你可以尝试许多想法，然后找到你满意的一个。

拥有开发/测试集和单一评估指标可以帮助您快速评估算法，从而更快地迭代。

从一个全新的应用程序开始，尝试快速建立开发/测试集和度量指标，例如在不到一周的时间内完成。 在成熟的应用程序上花费更长时间通常没有问题。

当你有大量数据时，旧的启发式的70%/30%的拆分方式将不适合。通常开发和测试集可能远远少于30%的数据。

你的开发集应该足够大，以检测算法的准确率时能看到有意义的变化，但不一定要非常大。您的测试集应该足够大，以便您对系统的最终性能有一个充分的估计。

如果您的开发集和指标不再指向正确的方向，请立即改变它们：（i）如果你过拟合开发集，请获取更多开发集数据。（ii）如果你关心的实际数据的分布与开发/测试集分布不同，请获取新的开发/测试集数据。（iii）如果你的指标不再衡量对你而言最重要的内容，请更换指标。

# 贝叶斯误差分析

1. 误差分析：查看开发集示例来评估想法

当你玩你的猫的应用程序，你会注意到几个例子，它错误将狗分为猫。有些狗看起来像猫！



一个团队成员建议加入第三方软件，这将使系统在狗身上做得更好。这些变化需要一个月的时间，团队成员非常热情。你应该让他们继续吗？

在投入一个月的时间来完成这项任务之前，我建议您首先估计一下它能够真正提高多少系统的准确性。然后你可以更理性地决定这是否值得一个月的开发时间，或者如果你最好在其他任务上使用这段时间。

详细来说，下面这些可以帮助你：

1. 获得开发集中被误分类的100个示例。
2. 手动查看这些示例，并计算它们中的哪部分是狗图像。

查看错误分类示例的过程称为“误差分析”。在这个例子中，如果你发现错误分类图像只有5％是狗，那么无论你对狗问题做出多少改进，你都不会摆脱5％以上的修正率。换句话说，5%是第三方软件能修正的上限(意味着最大可能数量)。因此，如果你的整个项目目前有90%的准确率(10%的误差)，那么这种改进最大可能导致90.5%的准确率(或9.5%的误差,比原始10%误差小了5%的误差总数)。

相反，如果你发现误差的50%都是狗，那么你可以更有信心，这个第三方插件将对项目产生重大影响。他可以将准确率从90%提高到95%(误差相对减少50%,从10%降低到了5%)。

这种简单的误差分析的计数程序为你提供了一种快速估算加入第三方插件(能对狗更好地进行处理)后的可能价值。它为决定是否进行这项投资提供了量化基础。

误差分析通常可以帮助你找出不同方向的前景。我曾看到许多工程师不愿意进行误差分析。直接跳入并实现一些想法，而不管该想法是否值得是令人兴奋的。这是一个常见的错误：这可能会导致你的团队只花费一个月的时间才会意识到它没有什么好处。

手动检查100个示例并不需要很长时间。即使你每张图片花费一分钟，你也会在两小时内完成。这两个小时可以为你节省一个月的浪费，而且时间花费的非常值得。

“误差分析”是指检测被你的算法误分类的开发集示例的过程，这可以帮助我们了解误差的根本原因。这可以帮助你优先考虑其他项目(如本例中)，或是激发别的方向，这个我们将在下一节进行讨论。下面的几个章节还会介绍执行误差分析的最佳实践。

1. 在误差分析期间并行评估多个想法

你的团队有几个想法来改进猫探测器：

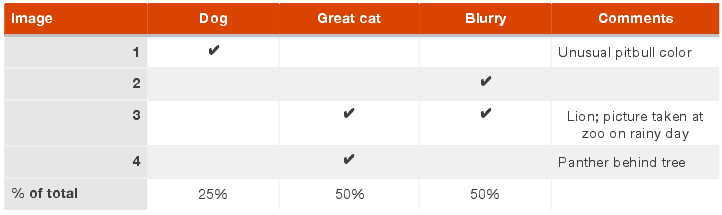
解决狗被识别为猫的问题。

解决大猫(狮子,黑豹等)被识别为家猫(宠物猫)的问题。

改进系统在模糊图片上的检测性能

... ...

你可以并行有效地评估所有这些想法。我通常会创建一个电子表格，并在查看100个误分类的开发集图片时将其填上，同时记下可能帮助记住具体示例的注释。用一个四个小例子组成的演示说明这个过程，你的电子表格可能看起来像这样：



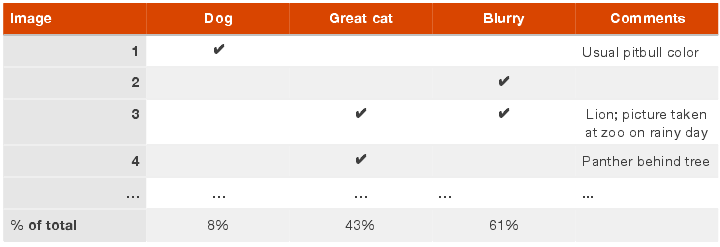
上面的图像#3勾选了Great Cat和Blurry列：可以将一个示例与多个类别关联。这就是为什么底部的百分比加起来不一定是100％。

虽然我已经将这个过程描述为首先制定类别(狗,大猫,模糊)，然后通过示例进行分类，实际上，一旦你开始查看示例，你可能会受到启发来提出新的错误类别。例如，也许在浏览了十几张照片后，你会发现很多错误都出现在由Instagram过滤器预处理后的图片。你可以返回并在电子表格中添加一个新的”Instagram”列。手动查看算法出错的例子，并想想自己作为人类是否能够正确标记此图像，通常会激发你想出新的错误类别和解决方案。

最有帮助的错误类别将是你最想改进的错误类别。例如你想”撤销”Instagram过滤器来恢复原始图像，那么Instagram类别将有所帮助。但是，你不必局限于你已有改进想法的错误类别：这个过程的目标是你对哪些领域最有希望关注的直觉。

误差分析是一个迭代过程。你甚至可以从没有任何类别开始。通过查看图像，你可能会想到一些错误类别。然后在继续查看和手动分类后，你可能会受到启发创建新的错误类别，然后根据新的类别重新检查图像，等等。

假设你完成了对100个开发集的示例进行误差分析，并得到这个：



你现在知道了解决狗错误的插件最多可以消除8%的错误。处理大猫或模糊图像错误可能会给准确率带来更大提升。因此，你可以选择后两个类别中的一个来处理。如果你的团队有足够人手并行研究多个方向，你可以让一些人在大猫错误上工作，另一些人在模糊图像上工作。

误差分析不会产生一个严格的数学公式，来告诉你什么应该是最优先的任务。你还必须考虑你期望在不同类别上取得的进展以及解决每个问题所需的工作量。