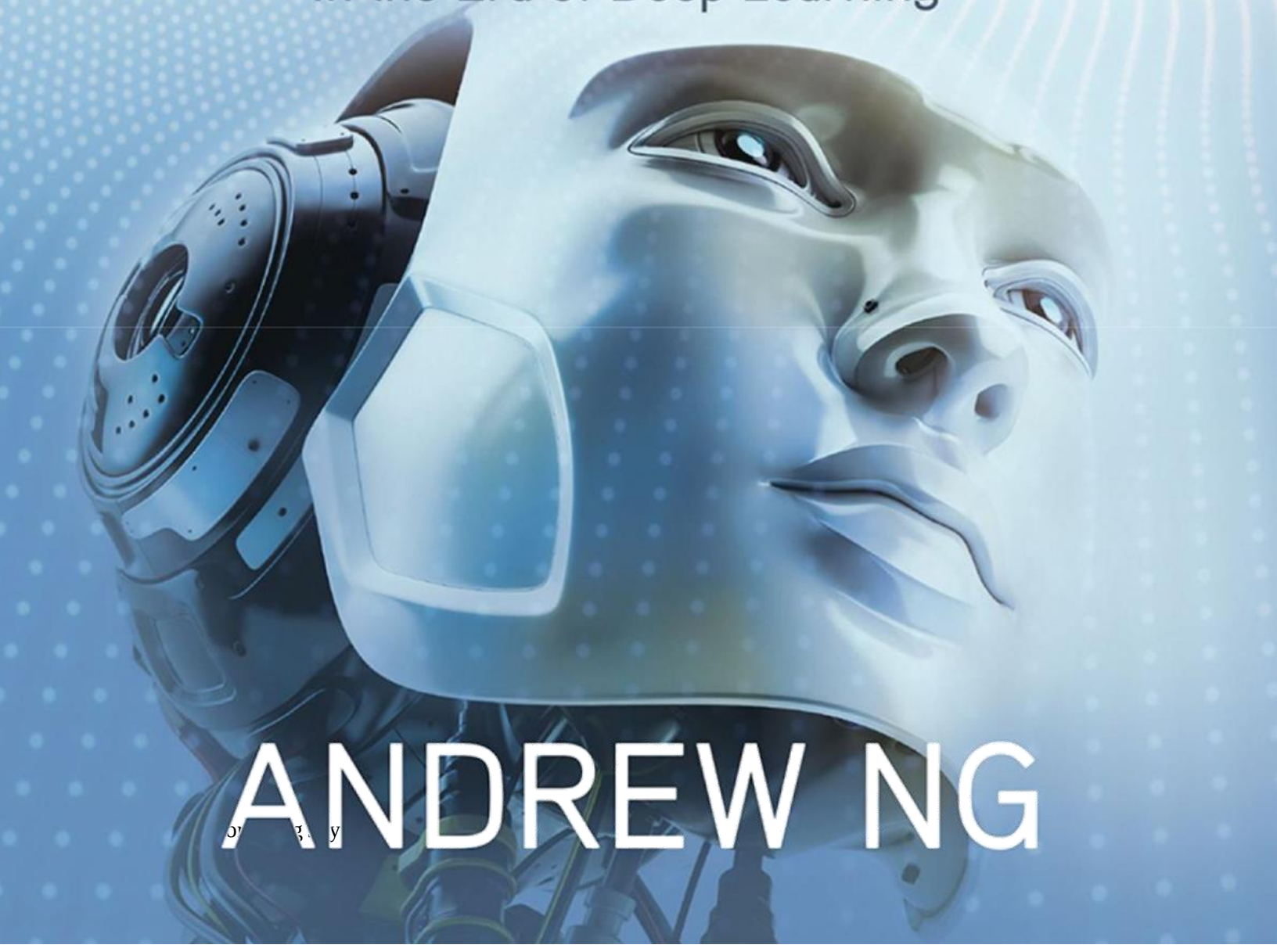
Draft Version

MACHINE LEARNING YEARNING

Technical Strategy for AI Engineers, In the Era of Deep Learning



ANDREW NG

Machine Learning Yearning is a
deeplearning.ai project.

Translator: Ying Joy https://www.yingjoy.cn
如有侵权，请联系删除; 如有错误，还请指正 E-mail: yzk.1314@outlook.com

目录(手稿)

1 为什么使用机器学习

2 如何使用本书帮助你的团队

3 预备知识和符号约定

4 数据规模的增大促进了机器学习的发展

5 开发和测试集

6 开发集和测试集应该来自同一分布

7 开发集和测试集需要多大？

8 建立一个单一的数字评估指标

9 优化指标和满足指标

10 用开发集和评估指标加速迭代

11 什么时候更改开发集和评估指标

12 小结 ：建立开发和测试集

13 快速构建一个系统，然后进行迭代

14 错误分析：查看开发集样本来评估 ideas

1 为什么使用机器学习

机器学习是很多应用程序的基础，包括 Web 搜索、垃圾邮件过滤系统、语音识别、产品推荐等等。如果你的团队正在研究一个机器学习的程序，希望本书可以帮助你快速的取得进展。

实例：构建基于猫咪图片的公司

假如你正在创建一个公司，主要业务为向爱猫人士提供许多猫咪的图片。

你使用神经网络来建立一个计算机视觉系统来检测图片中的猫咪。

不幸的是，你的神经网络学习算法的**准确性**还不够好，你需要去改进算法，你会怎么做？

你的团队可能会有许多想法，比如：

- 更多的数据： 收集更多的猫咪图片。
- 收集更**多样化**的训练集，比如，在各种角度拍的猫咪的图片，不同颜色的猫的图片，拍摄照片时相机的各种设置等等。
- 增加算法的训练时间，运行更多次数的梯度下降迭代。
- 构建一个更大的神经网络，拥有更多的层数(layers)/隐层神经元(hidden units)/参数(parameters)。

- 尝试更小的神经网络
- 尝试添加正则化（比如 L2 正则化）
- 尝试改变神经网络的架构（激活函数，隐层神经元数等）
- ...

在这些可能的方案中，如果你选择正确，你将建立一个相比原来更加完善的系统，并取得成功。如果你选择失误，那么可能会白白浪费几个月的时间。如果是你，你会怎么选择呢？

本书将会告诉你应该如何去选择。许多机器学习的问题都会留下一些线索，告诉你什么是有用的，什么是没用的。如果你可以什么学习理解这些线索获取可以帮你节省数月或数年的开发时间。

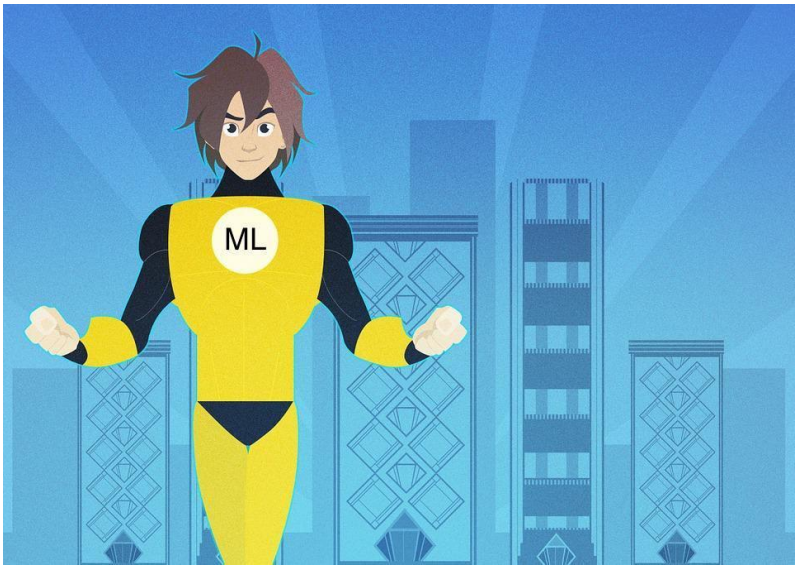
2 如何使用本书来帮助你的团队

在读完本书后，你将会对如何制定机器学习项目中的技术方案有一个深刻的理解。

但是你的队友可能不会理解为什么使用你制定的技术方案，也许你想和你的团队定义一个评估指标，如果他们不信服你，你该怎么说服他们？

这就是为什么我把章节设置的这么短的原因，这样你可以把你想让队友知道的那几页打印出来给他们看。

面对几个不同的选择，对这些选择的排序可能会对团队的生产力产生巨大的影响。通过帮助你的团队做出一个最优的选择，我希望你可以变为团队中的大佬!



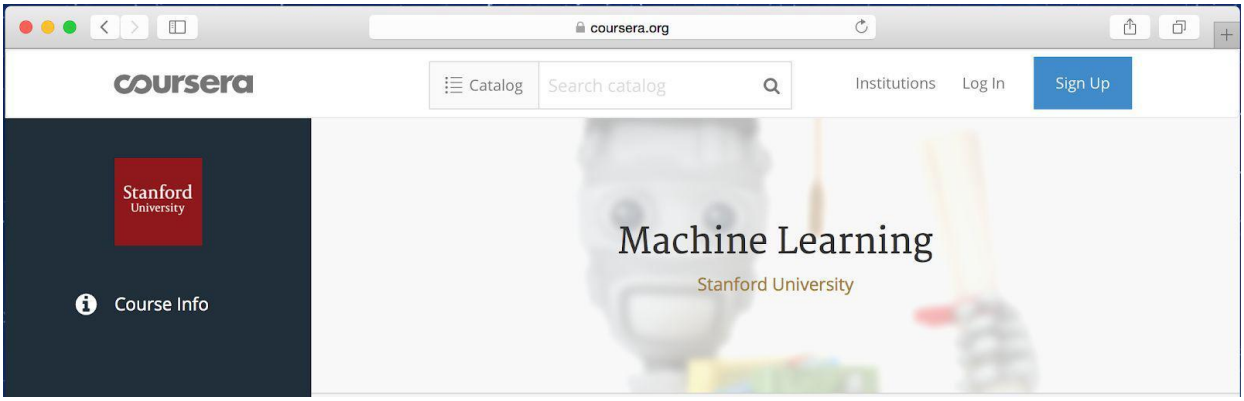
3 预备知识和符号约定

如果你已经学习了机器学习课程（如我在 Coursera 上的 Machine Learning MOOC），或者你拥有应用监督式学习的经验，你应该可以理解下面的内容。

我假设你熟悉**监督式学习(supervised learning)**: 使用带有标签的训练样例(x, y)学习一个从x映射到y的函数。监督式学习包括**线性回归(linear regression)**，**逻辑回归(logistic regression)**和**神经网络(neural networks)**。机器学习的形式有很多种，但今天大多数的机器学习应用都是监督式学习。

我会经常提到**神经网络(neural networks)(也被称为“深度学习(deep learning)”)**，而你只需要对他们有一个基本的了解即可。

如果你对这里提到的概念不是很熟悉，你可以看一下 Coursera 上面的 Machine Learning 课程（课程网址： http://ml-class.org）前三周的视频。



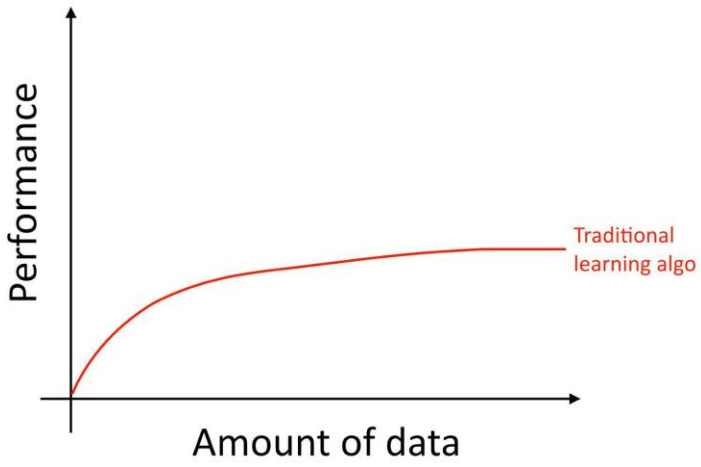
4 数据规模的增大促进了机器学习的发展

深度学习（神经网络）中许多的想法都已经存在了几十年。为什么今天这些想法火起来了呢？

促进机器学习发展的因素主要有两个：

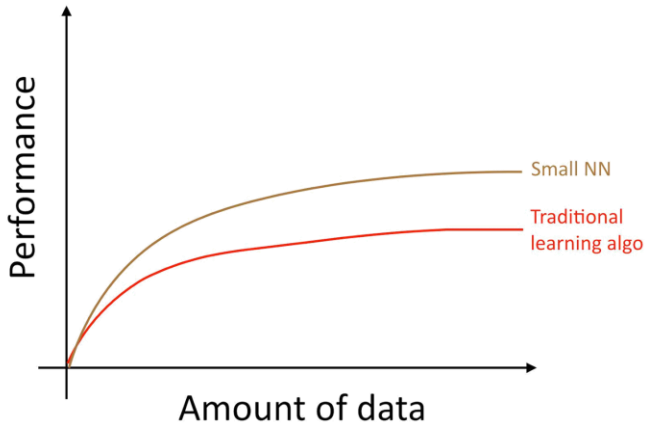
- **数据量越来越多。**如今人们在数字设备（电脑，移动设备）上所花费的时间相比以前多得多，这些活动产生了大量的数据，我们可以使用这些数据来训练我们的算法。
- **计算能力的提升。**人类几年前才开始训练神经网络，而且这些神经网络都足够大，可以将现在的大数据作为输入。

具体来说，如果你使用的是传统的机器学习算法（如：逻辑回归），即使你拥有更大的数据量，也会出现“高原效应(plateaus)”。也就是说即使你给它更多的数据，它的学习曲线也会变得平坦(flattens out)，算法就不会再有很明显的提升了：

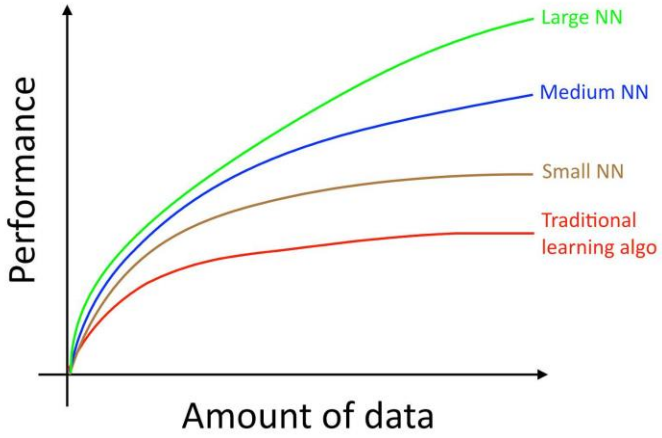


这就好像是传统算法不知道该怎么处理我们所拥有的全部数据。

如果你在面对监督学习任务时训练了一个小型的神经网络，可能你会获得相对较好效果：



这里，“小型神经网络(small NN)”是指具有较少的隐层神经元/层/参数。你训练的神经网络越大，性能就会越好。[1]



因此，如果你想获得较好的性能，你需要：

（1）训练一个较大的神经网络。

（2）拥有大量的数据。

还有很多其它的细节也是非常重要的，如神经网络的架构，在这方面的创新目前也是非常多的。但是想提高你算法的性能最可靠的方法还是：

（1）训练一个较大的神经网络。

（2）拥有大量的数据。

[1] 这个图展示了神经网络在数量较少的数据集上也能有不错的效果（前半部分）。神经网络在大数据中展现的效果很好，但是在小数据集上就不一定了。在小数据集中，可能传统算法会做的更好，这取决于特征的选择。比如，你只有 20 个训练样本，那么你使用 logistic regression 或神经网络可能没有什么区别，主要是特征的选择对算法结果造成的影响较大。但是，如果你拥有 100 万的数据量，那我更倾向使用神经网络。

完成过程（1）和（2）的是非常复杂的。本书将会详细讨论其中的一些细节问题。我们将从通用的策略开始，这些通用的策略在传统机器学习和深度学习上都很有用，并且为现代深度学习策略奠定了基础。

建立
开发集和测试集

5 开发集和测试集

让我们回到之前关于猫咪图片的例子：你开发了一个移动 APP，用户可以上传许多不同的图片到你的 APP 上，你想识别出用户上传的图片中所有包含猫咪的图片。

你的团队下载了很多图片数据集，包含猫咪图片（正样本，positive example）和非猫咪图片（负样本，negative example）。他们将这些数据划分为 70%的训练集，30%的测试集。

当使用这些进行算法训练时，效果非常不错。

但是将算法（分类器）部署到 APP 的时候，发现效果却非常的糟糕！

发生了什么？

你发现用户上传的图片和你团队下载作为训练集的图片不同。用户使用收集拍摄的图片分辨率更低，模糊，而且照明效果较差。由于你的训练/测试集来源于网站上的图片，你的算法没有很好的把智能手机图片一般化。

大数据时代之前，在机器学习中人们对数据集的一个常见划分规则为：将数据集划分为 70%/30%的训练集和测试集。但是在越来越多的应用（比如上面的例子）中，收集的数据与训练数据不同，这个时候该规则就不适用了。

我们通常定义：

- **训练集(Training set)** — 运行在算法上的数据集.
- **开发集(Dev/development set)** — 这部分数据通常用来调参，选择特征，以及对学习算法进行改进。有时也被用于**交叉验证。**
- **测试集(Test set)** — 这部分数据通常用于评估算法的性能，但不要依靠这个数据对你的算法进行调参和修改。

一旦你定义了一个开发集和测试集，你的团队就可以进行模型的建立，通过调整参数，特征选择等。从而制定最有效的机器学习算法。开发集和测试集可以很快的告诉你算法的运行情况。

换句话说，**开发集和测试集的目的是为了让你对算法进行改进，使算法效果变得更好**
所以你应该：

- 选择开发集和测试集时，主要选择可以反映未来需要获取的数据

换句话说，你的测试集不应该只是可用数据的 30%这么简单，尤其是你得到的数据（移动 app 的图像）和你训练数据（网站图像）不一样的时候。
如果你还没有运行你的移动应用，也就意味着还没有用户，可能你就无法获取到一些比较好的数据，但是你可以尝试去获取这些数据。比如：让你的朋友拍一些手机照片发给你，
一旦你的应用启动后，你可以使用实际的数据来更新你的开发/测试集。
如果你实在没有办法获取到你期望的数据，那么你可以从网站图片开始，不过这种做法可能会导致你的系统不能一般化(generalize)的很好。
有时，可能需要花费一些资金去获取比较好的数据集。切记不要认为你的训练集和测试集分布必须是一样的。尽量去选择那些可以反映真实情况的数据作为测试样本。

6 开发集和测试集应该来自同一分布

根据市场情况，由于存在不同地区的用户，你可以把你的猫咪 APP 图片数据分为四个区域：

(1) 美国

(2) 中国

(3) 印度

(4) 其它地区

为了生成一个开发集和测试集，你可能会随机的分配两个地区的数据到开发集中，另外两个作为测试集。比如：来自美国和印度的数据作为一类，来自中国和其它地区的数据作为另一类。

一旦你这样划分了数据集，你的团队可能会主要关注提高在开发集的上的性能。开发集应该能够正确的反映出你的整体情况，而不是局部情况。比如这里主要提升了 APP 在美国和印度区域的性能，而没有考虑到中国和其他地区。

其次，开发集和测试集如果来自不同分布还会导致另一个问题：你团队进行开发后会发现，算法在开发集（美国，印度）上的效果会非常好，但是到了测试集（中国，其它地区）上就会变得很差。我曾经看到过很多人都是因为这个问题导致白费了很多努力，所以我不希望这发生在你的身上。

举一个例子，假设你团队开发的系统在开发集上的效果非常好，但是在测试集上却表现的非常糟糕。如果你的开发集和测试集来自于同种分布，那么你可以立刻判断，你的算法在开发集上过拟合了。比较简单的解决办法是输入更多的数据进行算法性能提升。

但是如果开发集和测试集来自不同分布，那么你可能就比较不好找原因了，可能会出现以下错误：

1. 你的算法在开发集上过拟合了
2. 测试集比开发集更难识别，所以算法输出的结果可能就没预期那么好了，而且进行改进比较困难

3. 测试集不一定比开发集更难识别，只是它们来自不同分布。所以在开发集上表示好的算法并不能在测试集上表现良好。（如，美国和印度猫咪的数据可能就无法反映出中国和其他区域猫咪的数据）这种情况下，你之前对算法进行改进的努力可能都白费了。

机器学习的应用本身是非常困难的。如果开发集和测试集还没有来自同一个分布，那么你将会浪费很多时间在你的算法上。甚至你不知道你该做什么，不该做什么。

如果你面对的是第三方基准测试（benchmark）的问题，可能开发集和测试集来源于不同的分布，这种时候只有运气对你算法产生的影响最大。当然，如果开发集和测试集在同一分布，那么你的算法应该能够很好的进行推广和拓展。如果你开发的应用是针对特定的方向的话，我建议在选择开发集和测试集的时候让它们在同一分布。

7 开发集和测试集应该多大？

开发集应该足够大，大到可以检测出不同算法之间的差异。比如：如果分类器 A 的精度为 90.0%，分类器 B 精度为 90.1%。如果你的开发集只有 100 条，那么你可能检测不出这 0.1%的差异，与其它机器学习的问题相比，100 条数据很小，常见的开发集数据规模在 1000 到 10000 条之间。数据量越高，模型之间的差异越明显。[2]

对于一些成熟的重要应用来说（如：广告推荐，网页推荐，产品推荐等）。我经常看到团队在为 0.01%的性能提升而奋斗，因为这直接影响到了公司的利润。在这种情况下，开发集的数据量可能远远超过 10000 条，只为了对算法进行改进。

测试集要多大？它也应该足够大，大到你有一个很高自信度去对系统的整体性能进行评估。这里有一个方法：将 30%的数据用于测试。在你拥有一个中等（100 到 10000 个样本）数据量的情况下，它的效果不错。但是在大数据的时代下，我们面对的机器学习问题数据量可能会超过 10 亿条样本，开发集与测试集之间的比例一直在减小，但是开发与测试集的绝对数量在增加。在给开发集和数据集分配时，没必要过多的进行分配。

2 理论上，如果一个算法的变化差异符合统计学上的某种变化，那么我们可以进行测试。在实践中，大多数团队都会这样做（除非它们发表论文）。而我没有发现用于统计意义上的测试。

8 建立一个单一数字的评估指标

分类准确率是**单一数字评估指标(single-number evaluation metric)**的示例：你在你的开发集（或测试集）上运行你的分类器,然后得到样本分类正确的比例（fraction）（单个数字），根据这个指标，如果分类器 A 的准确率为 97%，分类器 B 的准确率为 90%，那么我们认为分类器 A 更好。

相比之下，精度（查准率）（Precision）和召回率（查全率）（Recall）[3]就不是一个单一数字的评估指标：它给出两个数字来评估分类器。拥有多个评估指标使得算法之间的比较更加困难，假设你的算法表现如下：

分类器	精度	召回率
A	95%	90%
B	98%	85%

如上所示，这两个分类器的性能差不多，这就导致我们无法轻松的选择最好的那个。

在开发期间，你的团队会尝试大量关于算法架构，参数调整，特征选择等方面的想法。使用单一数字评估指标（如精度）使得你可以根据其在该指标上的表现快速对所有模型进行排序，从而绝对哪一个最好。

如果你真的即关心精度也关心召回率，我推荐你使用一个标准方法来把他们组合成一个单一的数字。例如你可以使用它们的平均值。或者你可以计算 F1 值（F1 score），这是一种基于平均值改善的方法，比简单的取平均值的效果要好。[4]

---

[3] 猫咪分类器的精度是指在开发集（或测试集）中检测出所有猫咪图片中有多少比例是真正的含有猫咪。它的召回率是指在开发集（或测试集）中所有真正的猫咪图片中有多少比例被检测出来了。在高精度和高召回率之间通常是权衡的。

[4] 如果你想了解更多关于 F1 值的信息，请见： https://en.wikipedia.org/wiki/F1_score 它是 基于精度和召回率的“几何平均（geometric mean）”定义的，计算公式为： 2/((1/精度)+(1/召回率)).

分类器	精度	召回率	F1 值
A	95%	90%	92.4%
B	98%	85%	91.0%

当你面对大量的分类器时，使用单一数字评估更加方便和快速的让你选择出最好的分类器。

在最后一个例子中，假如你得到了分别在四个主要市场（（1）美国，（2）中国，（3）印度和（4）其他地区）猫咪分类器的准确率。这里提供了四个指标。通过对这四个数据进行平均或加权平均，最终得到一个单一数字度量。取平均值或加权平均值是合并多个指标的常见方法之一。

9 优化指标和满足指标

这里有组合多个评价指标的另一个方法。

假设你同时关系算法的精度和运行时间。你需要在如下分类器中进行选择:

分类器	精度	运行时间
A	90%	80ms
B	92%	95ms
C	95%	1,500ms

如果将精度和运行时间按照下面的公式进行组合可能看起来不太自然:

精度 - 0.5*运行时间

你可以这样做：首先定义一个可接受(acceptable)的运行时间。例如任何运行时间在 100ms 内都是可以接受的。然后再在满足运行时间要求的分类器中选择精度最高的。在这里运行时间就是一个“满足指标(satisficing metric)”，你的分类器只要在这个指标上表现的足够好即可，这意味着你的算法最多耗时 100ms，而准确率是一个“优化指标(optimizing metric)”。

如果你正在权衡 N 个不同的标准，比如模型文件的大小（这对移动 APP 很重要，因为大多数用户都不想下载过大的 APP）、运行时间和准确率。你可以考虑将其中 N-1 个标准设置为“满足指标”，然后将最后一个指标定义为“优化指标”。如，你将模型文件的大小，和运行时间设置为一个可接受的阈值，然后在这些约束下不断优化你的算法准确度。

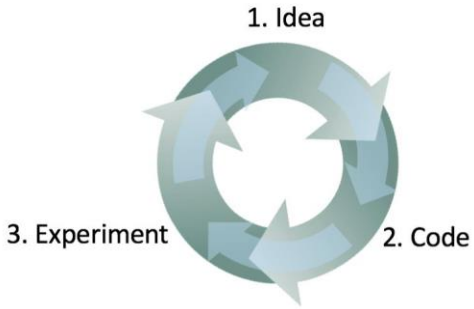
在最后一个例子中，假设你正在构建一个硬件设备，该设备使用麦克风监听用户说出的某个词作为特定的“唤醒词语（wakeword）”，从而唤醒系统。例如：Amazon Echo 监听“Alexa”；苹果 Siri 监听“Hey Siri”；Android 监听“Okay Google”；百度 app 监听“Hello Baidu”。你同时关心假正例的比例（the false positive rate）——当没有人说出唤醒词系统唤醒的频率， 和假反例的比例（the false negative rate）——有人说出唤醒词但是系统没有唤醒的频率。 该系统性能的一个合理目标是最大限度的减少误报率（优化指标），同时满足每 24 个小时操作不会出现一个假正例（满足指标）。

一旦你的团队按照评估指标进行优化，那你们肯定可以更快的取得进展。

10 用开发集和评估指标来加速迭代

对于一个新问题，事先是很难知道用什么方法解决它是最合适的。即使机器学习经验丰富的研究员也需要尝试许多，才能得到令自己满意的东西。在构建机器学习系统时，我经常会：

1. 首先有一些如何构建系统的想法(**idea**)。
2. 用代码实现这些 **idea**。
3. 进行实验(experiment)，来告诉我 我的 **idea** 工作的如何。（通常我前几个 **idea** 效果不是很好）但是基于这些想法的结果，我会返回去产生更多的 **idea**。并不断的迭代。



这是一个迭代过程，你循环的越快，你的进展也就越快。这就是开发集和评估指标是非常重要的原因：每当你尝试一个新的 **idea** 时，在开发集上衡量 **idea** 的表现，可以很清楚的知道你是否朝着正确的方向前进。

相反，如果你没有特定的开发集或评估指标。那么每一次团队开发出一个新的猫咪分类器时，你必须把它移植到你的 APP 中，然后经过几个小时的体验来感受新分类器的效果是否有提升。这非常慢！此外，如果分类器只有 0.1%的提升话，人为可能会感受不到这个提升。你的系统通过不断积累 0.1%的提升从而得到一个很大的提升。有一个开发集和评估指标，可以让你很快的检测出那些想法给你的系统带来了提升，你就可以快速的决定可以对哪些想法进一步的完善，哪些想法可以舍弃。

11 什么时候更改开发/测试集和评估指标

当开始一个新项目的时候，我会试图快速的选择开发/测试集，因为这样可以给团队一个很明确的目标。

我通常会要求我的团队在不到一周的时间内（几乎不会超过这个时间）提供一个初始的开发/测试集和评估指标，并且提出一个不太完美的方案迅速行动起来，这比花更多的时间去思考更好。但在一些比较成熟的应用上，一周时间可能不够，比如：反垃圾邮件（anti-spam）是一个成熟的深度学习应用。我见过一些团队会花费数月的时间在已完成的成熟系统上，去获得更好的开发集/测试集。

如果之后发现你最初的开发/测试集或评价指标与目标有些偏差，那么请修改它们。例如：如果你的开发集和评估指标在分类器 A 上表现的效果比 B 好，但是你的团队认为 B 在实际的产品中表现的更加优越，这可能表示你需要重新更改你的开发/测试集或者你的评价指标。

有三个主要原因导致开发集或评估指标错误的认为分类器 A 的效果更好：

1. 你的实际数据与开发数据来自不同分布.

假设你的初始开发/测试集主要是一些成年猫咪的图片。而在 APP 上，用户上传了比预期多得多的幼年猫咪图片。所以导致你的开发集与测试集不在同一分布。在这种情况下，更新数据集是最好的选择。

2. 算法在开发集上过拟合了

你把你的评估标准设置的过高，在开发集上反复评估导致算法过拟合，当完成开发后，如果你的算法在开发集上的效果明显高于在测试集上的效果，这就意味着你的算法在开发集上过拟合了。这种情况下，更新你的开发集。如果你需要跟踪团队进度，你也可以在测试集上定期对你的系统进行评估——每周或每月进行一次。但不要使用测试集来对你的算法进行改变。包括是否回滚到上一周的系统。如果你这样做，你的算法可能会在测试集上过拟合，并且不能在依靠它来对系统进行评估。(如果你发表研究论文或者进行一个很重要的商业决策，就需要注意这一点).

3. 评估指标衡量的并不是项目优化所需要的东西

假设对于你的猫咪 APP，你的评估指标是分类准确率。在该指标下，分类器 A 优于分类器 B。但是假设你尝试了这两种算法，发现分类器 A 偶尔会允许敏感图片通过。那么即使分类器 A 的精度优越于分类器 B，偶尔让敏感图片通过，这是无法接受的。你需要做什么呢？这里，该评估指标不能辨别出对产品而言算法 B 比算法 A 好这一事实。所以，这时候你就不能相信这个指标可以帮你选择出最好的分类器，你需要重新选择评估指标。例如，你可以改变评估指标，当对敏感图片分类错误时对算法进行严厉"惩罚"。我强烈建议你选择一个新的评估指标，并用新的标准来为团队明确定义一个新的指标。而不是在一个不可信的指标下处理太长的时间。

在项目中改变开发/测试集或者评估指标是很常见的。拥有一个初始的开发/测试集和评估指标能帮你快速迭代你的 idea。如果你发现你的开发/测试集或评估指标没有正确的引导你的团队前进，你可以随时更改它们。

12 小结：建立开发集和测试集

- 从分布中选择开发集和测试集 ，它需要反映你将来的数据情况，并且它的效果足够好，这可能与训练的数据不在同一分布。
- 尽可能在同一分布选择你的开发集和测试集。
- 为你的团队选择一个单一数字的评估指标来进行优化。如果你关心多个指标，你可以把它们合并到一个指标中（例如：平均多个错误指标），或设定满足指标和优化指标。
- 机器学习是一个高度迭代的过：在发现你满意的结果之前需要尝试大量的 idea。
- 开发/测试集和单一数字指标可以帮助你快速的评估算法，从而迭代的更快。
- 当开始一个全新的应用时，尝试快速建立开发/测试集和评估指标，最好在一周之内，当然，如果在成熟的机器学习应用上可以花费比这更长的时间。
- 当你拥有大量数据的时候，根据 70%：30%的比例划分训练/测试集这个经验可能不太适用；开发/测试集可以 占远小于 30%的数量。
- 你的开发集应该足够大，足够检测到算法的改变，但没必要太大，只要达到你可以对你的系统整体性能有一个评估即可。
- 如果你的开发集和苹果指标没有引导你的团队往正确的方向走，请快速改变它们：
  （1）如果在开发集上过拟合了，你可以去获取更多的数据
  （2）如果你数据的实际分布和开发/测试集的分布不同，那么你需要去更新你的数据集
  （3）如果你的评估指标不能在衡量对于你来说很重要的东西，请改变它

错误分析基础

13 快速构建你的第一个系统，然后进行迭代

你想建立一个新的反垃圾邮件系统，你的团队有以下想法：

- 收集一个含有大量垃圾邮件的训练集。例如，设置一个“蜜罐”：故意发送虚假的电子邮件给已知垃圾邮件发送者，以便于能够自动收集它们发送到这些地址的垃圾邮件。
- 开发用于理解电子邮件文本内容的功能。
- 开发用于理解电子邮件 header（不清楚可以参考：https://whatismyipaddress.com/email-header）特性的功能，以显示消息所经历的一组网络服务器。
- and more.

尽管我在反垃圾邮件上已经做了大量工作，但我还是很难选择其中的一个方向，如果你不是应用领域的专家，那将更难。

所以，开始的时候不要试图设计和构建完美的系统。相反，应该快速构建和训练出一个基本系统——在短短几天的实际内[5]即使基本系统与你“最佳”系统相差很多，研究基本系统的功能仍非常具有价值：你可以很快的找到你最希望的方向的线索。接下来几章将告诉你如何去阅读这些线索。

5 这个建议是针对那些想要构建人工智能应用的读者，而不是那些想要发表学术论文的学者，稍后我会回到做研究的话题。

14 错误分析：查看开发集样本来评估 ideas

当你使用猫咪 APP 的时候，注意到一些被错误，识别成猫咪的狗样本。一些狗长的像猫！于是一个团队成员建议和第三方软件进行合作，使系统可以更好的处理狗样本。这些改变需要花一个月的时间，并且团队成员热衷于这一方案，你会让他们这样做吗？

在为这个任务投资一个月前，我建议你首先评估一下它实际上会提高多少系统 的准确率。然后你才能理性的选择是否值得花费这一个月的开发时间。

具体来说，你可以做这些事情：

1. 获取 100 个系统分类错误的样本
2. 手动查看这些样本，计算其中有多少比例是狗的图片

查看错误分类样例的这一过程称为：**错误分析（error analysis）**。在该案例中，如果你发现只有 5%的错误分类图像是狗，那么无论你在狗的问题上做多少改进，可能你都无法消除这 5%的错误。换句话说，这 5%是上述建议能够达到的改进上限。因此，整个系统当前的准确率是 90%（误差 10%），这一改进可能得到最多 90.5%的准确率（或 9.5%的错误率，比原来的错误率少 5%）。

相反，如果你发现 50%的错误图像都是狗，那么你最好找一个第三方进行合作。它可以将准确率从 90%提升到 95%（误差相对减少 50%，从 10%降到 5%）。

这种简单的错误分析的计算过程可以给你一个快速的方法来评估为“狗”的问题加入第三方软件是否值得。它为你决定是否做出这笔投资提供了一个定量的基准。

错误分析通常可以帮你找出不同想法有哪些前景。我看到很多工程师不愿意进行错误分析。相比于质疑这个想法是否值得花时间投入，直接实现一个然后查看效果可能会更好，这是一个常见的错误：可能会导致你的团队花费一个月的时间只能带来很少的收益。

手动检查 100 个样本不会花费太长的时间。即使你每分钟只看一张图，不到两小时你就可以完成，这不到两小时的时间可以为你节约一个月的白白努力时间，值得花费。

错误分析（error analysis）是指检测开发集中算法错误分类样本的过程，以便了解错误的深层原因。它不仅可以帮你重点发展你的项目，正如这个例子所述，还可以给你一些新的启发。下节将讨论该内容。接下来几个章节还将介绍一些错误分析的最佳实践。