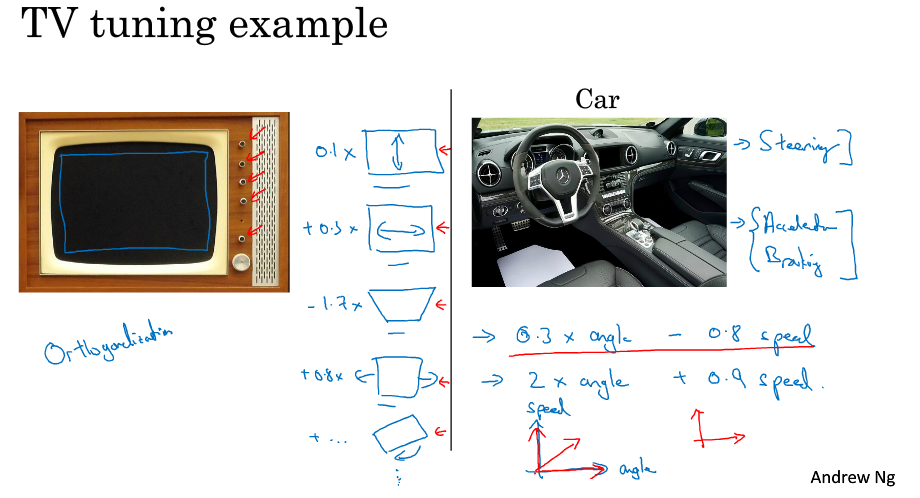
1. 机器学习策略（1）
   1. 为什么是ML策略

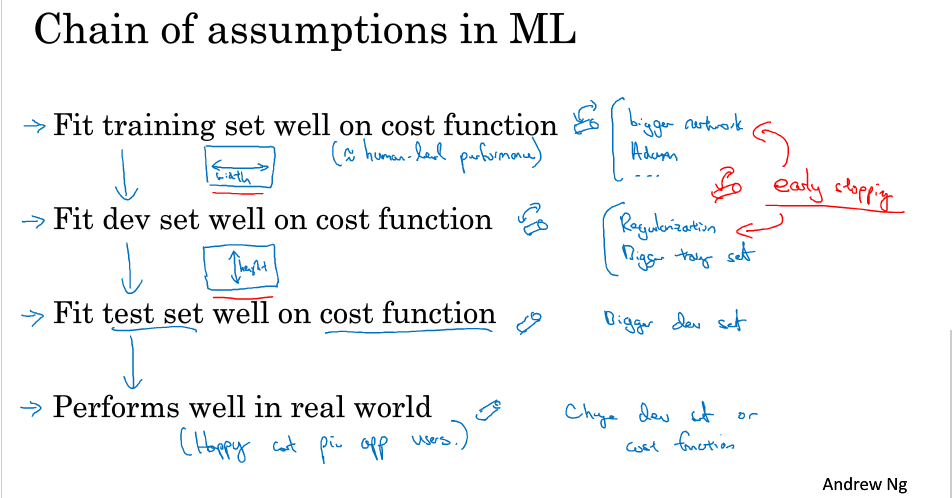
本次课程将会基于机器学习策略教你如何构建你自己的机器学习项目，通过本次课程学到更为快捷高效的构建机器学习系统的方法，那究竟什么是机器学习策略呢？让我们从一个贴近生活的实例入手，假设你正在开发一个猫分类模型，捣鼓了一段时间系统的准确率到了90%，但是并没有达到你的预期。可能你的脑海中会涌现许多提升准确率的思路，比如 你可能会想 去收集更多训练数据，你还可能会说是不是训练样本多样性不够，训练集应该包含更多不同姿势的猫，或者更多样化的反例，或者使用梯度下降再将算法训练更长时间，或者换一种优化算法比如Adam优化算法，或者尝试更复杂或更简单的神经网络，或者你想试试dropout和L2正则化，或者你想改变网络架构，比如更换激活函数，改变隐藏单元的个数等方法，当我们试图优化深度学习系统时总会有很多方法可以去尝试，问题在于如果你选择的方法不正确，很有可能花了6个月的时间最终一事无成。那么如果有快捷高效的方法分辨出在之前提到的或者没有提到的各种想法中，哪些值得进行尝试 哪些可以直接放弃将会对你有很大帮助。学习到一些基本套路也就是一些分析机器学习问题的方法，这些方法会让你避免南辕北辙。

* 1. 正交化 Orthogonalization

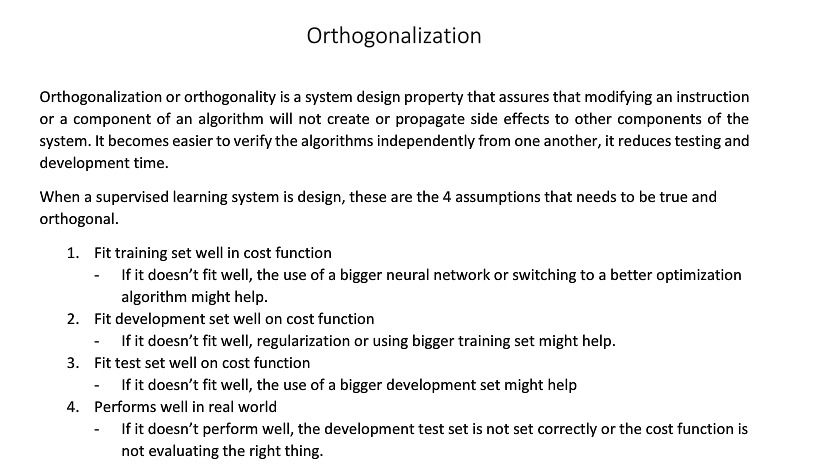
打造机器学习最大的一个挑战就是有太多选择可以尝试，太多事可以调整，比如说有很多超参数可以去调。发现衣蛾现象，机器学习厉害的那些人很有洞察力，知道靠校正什么来达到某一个效果。这个过程就叫做正交化，Orthogonalization。比如举个电视的例子，可能有很多的旋钮来调整，但是我们希望每个旋钮都有自己特定的功能，可以完成某种效果，而不是相互之间产生影响，从而校正起来非常的难。同理，车有三种控制器是一样的道理，我们的方向盘，油门和刹车都有自己专门的作用，如果必须调好某两个同时决定一个性能，那么很难控制好。



为了让监督式学习系统表现好，经常需要调整系统的按钮，必须要保证下面四个点，1，通常要确保至少要在训练集上表现的很好，在训练集上的效果必须超过某个门槛。2，希望在开发集dev set上也表现好。3，在测试集上表现的好。4，在现实世界也表现好。回到电视调整的例子，如果画面太宽太窄，你希望有一个旋钮来调整，而不是小心翼翼的调整五个按钮，还可能搞乱其他方面，只想要一个旋钮来调整电视宽度。同样的道理，如果你的算法在训练集上不够准，那么需要一个旋钮，或者一组特定的旋钮来调整你的算法，根据深度学习经验表明，这个旋钮应该是：训练一个更大的网络或者用更好的优化算法比如Adam优化算法等等。相反地，如果发现在dev set不够准，对开发集的拟合不好，那就有另一组旋钮来调整，也就是说在训练集上效果好，在dev set上效果不怎么样，那旋钮就应该是正则化或者使用更大的训练集。我们想要满足第二个条件的同时也对第一个条件尽量不造成影响。然后你希望在测试集上效果也好，如果在测试集上效果不好，在开发集上效果好，那需要调整的旋钮可能是更大的开发集，说明你对开发集过度拟合了，所以需要往回退一步，使用更大的开发集。最后如果在测试集上效果好，但无法给你猫图片应用用户提供良好的体验，这意味着你需要回去改变开发集或成本函数，因为如果根据某个成本函数，系统在测试集上做的很好，但无法反映你的算法在现实世界中的表现，这意味着你的开发集分布设置不正确，要么你的成本函数测量的指标不正确。如果你对细节不太明白，没关系但是你需要对正交化的过程有个概念，要非常清楚到底是问题中的哪一个，知道可以调整哪些东西尝试解决哪个问题。



当我使用机器学习的时候，一般不采用早期停止，这个技巧也还不错，很多人也会用，但就正交化概念而言，早期停止有点难以分析，因为这个旋钮的操作会同时影响你对训练集的拟合，因为早期停止必定对训练集的拟合不太好，而且它同时还能改善对开发集的表现，所以这个旋钮没有正交化，因为它同时影响两件事情，不是说这样就不用，而是如果你有更多的正交化控制比如上图中的操作，用这些手段调整网络会简单不少。



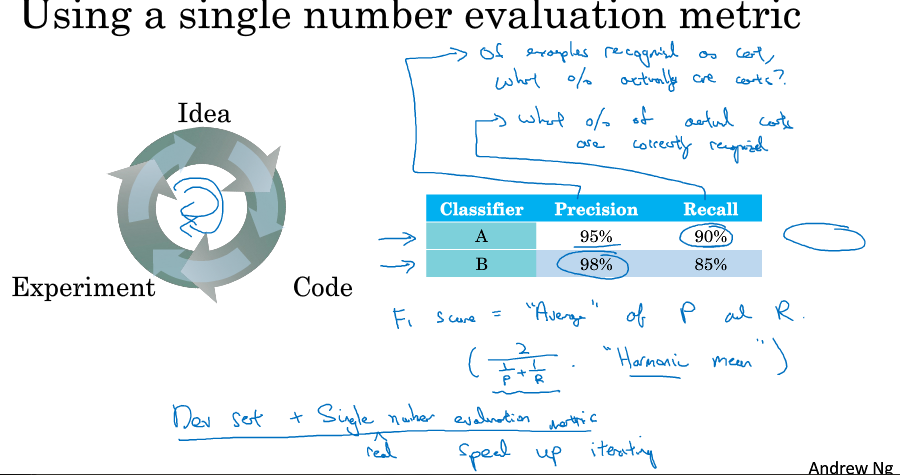
所以对正交化的理解就是，原本有两个互成角度的向量，你修改其中一个指标，它的投影在另一个向量上也会有影响，而如果两个向量正交化，变成指直角，那么修改其中一个因素，对另一个因素的影响则会很小或者没有影响，所以这就是机器学习中的一个策略，正交化。

* 1. 单一数字评估指标

无论你是选择超参调优或是选择不同的机器学习算法，还是在构建机器学习系统时尝试不同的配置项，你都会发现如果你有单一的量化评估指标，可以让你知道新的方法比上一次是更好还是更糟，那么整个进程会加快很多。所以团队启动一个机器学习项目时，常常被建议需要一个单一的数字评估指标。

举个例子，你构建了某个分类器A，通过调整超参和训练集或者其他什么方法，你训练得到了一个新的分类器B，这时一个合理的评估你分类器好坏的方法就是去看它的精确率（查准率）和召回率（查全率）。简单来说，查准率就是识别出是猫图集合中有多少真的是猫，因此分类器A有95%的精准率 就意味着A如果把一张图片分类为猫 那么95%可能它就是猫。查全率就是训练集中所有的猫图中，有多少被真正的识别出来了。事实证明你常常要在这两者之间做权衡，需要同时关注这两个指标，你要的是当分类器说它是猫时

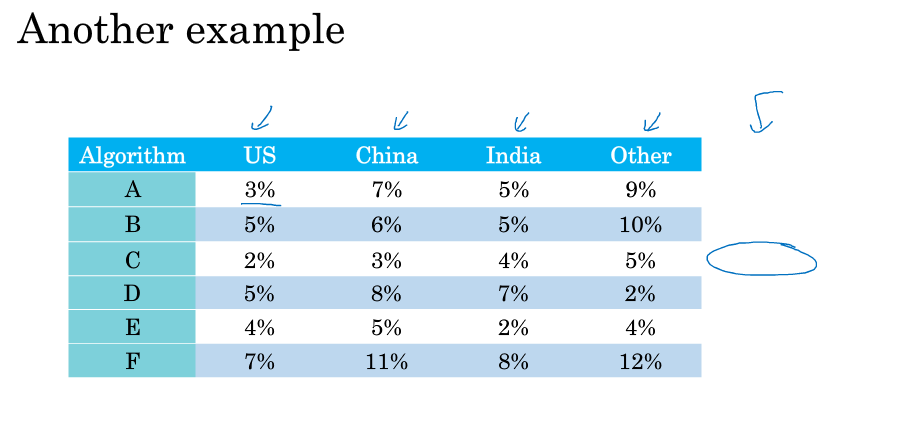
它有很高的可能性是猫，而对于所有真正是猫的图片，你同样希望分类器可以把绝大多数都识别出来。所以使用查准率和查全率是比较好的，问题在于把它们都作为评估指标时，如果某个分类器其中一个指标好，另一个差点，就不知道究竟哪个分类器好了。并且如果你尝试许多不同的算法 就会有许多超参数，你就不止从两个分类器中选择，可能需要从十几个中快速地选出最好的那个。



所以当有两个指标时，快速找出哪个分类器最好是比较困难的。因此解决方法就是与其使用两个数字，查准率和查全率来选择分类器，不如找一个新的评估指标 它兼顾这两个指标在机器学习领域，标准的方法是使用 F1分数。

P表示查准率，R表示查全率，这个F1被称为P和R的调和平均数，通俗的说你可以理解为是这两种指标的某种平均，只是不是使用算术平均值，根据一计算，立马可以得到F1A>F1B，所以A分类器更好。

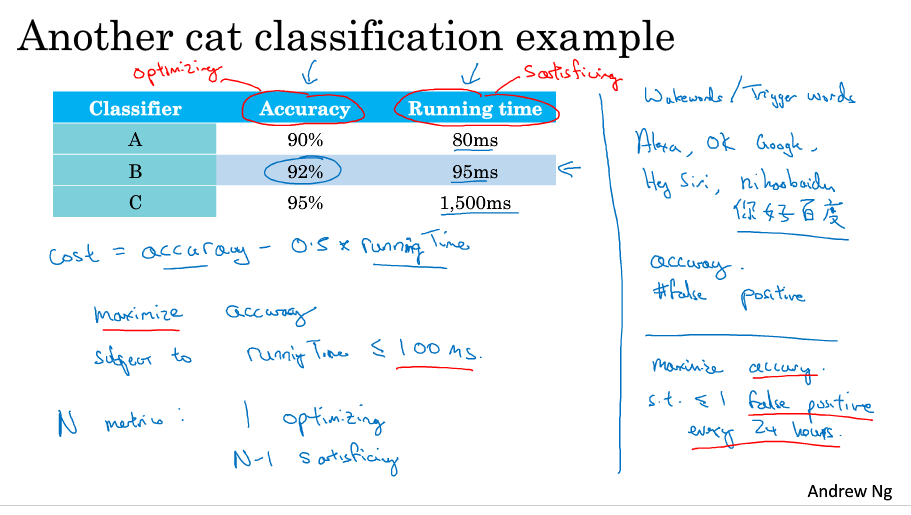
许多机器学习的团队都有一个明确定义的验证集，用来侧量查准率和查全率再加上一个单一量化评估指标，有时称为单实数评估指标，这个评估指标可以帮你快速判断哪个分类器更好。所以一个好的验证集和单一量化评估指标可以提高迭代的效率。



又比如上图的这个例子，建议除了追踪4个不同地区的表现，应该同时关注它们的平均值，假设平均表现是一个合理的单一量化评估指标，通过计算平均值很快的看出似乎算法C的错误率最低，然后你就可以继续对于它做改进了。你必须选择一个算法并且不断的对它进行迭代，在进行机器学习类工作的时候，常常先有一个想法，再实现它，然后你想要知道你的想法是否有效。

* 1. 满足和优化指标

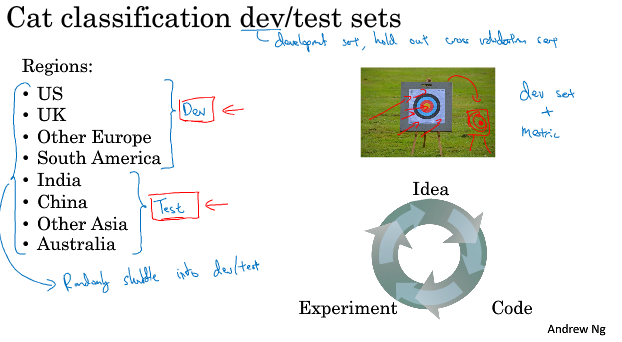
要把你想要顾及的指标组合成单实数指标有时并不容易，所以我们就需要建立满足指标和优化指标。比如你决定重点关注猫分类器的分类准确率(accuracy)它可以是F1分数或者别的什么精确度度量，但同时你还关注运行时间，也就是对一个图像分类需要多少时间。你当然可以将准确率和运行时间组合成一个整体评价指标(overall evaluation metric)，比如说，整体代价=准确率-0.5\*运行时间，但是将将准确率和运行时间用这样的公式整合看起来有些刻意，这就像二者的线性加权。还可以这么做，你可能想要选择一个分类器，它在确保运行时间的前提下提供最大准确率，那么运行时间就是满足指标，只要满足运行时间小于100ms，在用户接受范围内，最大程度想要的最高的准确率，所以准确率就是优化指标。



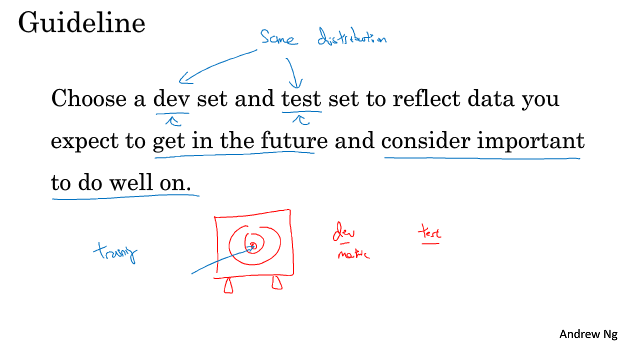
通过定义优化指标和满足指标，你就有了挑选"最优"分类器的明确方向。更一般的说 如果你有N个关心的指标，有时候选择其中的一个加以优化是个合理的策略,你想要它的表现尽可能的好。那么剩下的N-1就是满足指标，意味着他们只要达到某种阈值(threshold)就可以了。一旦这些指标到达阈值，就不用再优化它们了。大多数情况下满足指标内容都会优于最低标准，这样一来你就有了一个几乎自动的，快速评价模型和选择"最佳"模型的方法，现在这些评价指标必须在训练集/开发集/测试集上评估计算。所以另一件事需要做的就是建立训练集/开发集/测试集。

* 1. 训练集/开发集/测试集划分

建立训练集、开发集、测试集的方式大大的影响你或你的团队在建立机器学习应用方面取得进展的速度，学习如何设立这些数据集，让你的团队效率最大化。这节内容几种讨论设置dev set开发集（有时也叫保留交叉验证集）和test set测试集，然后机器学习中的工作流程是你尝试很多思路，用训练集训练不同的模型，然后用开发集来评估不同的思路，然后选择一个再不断迭代去改善开发集的性能，直到最后得到一个令你满意的成本，再用测试集去计算。



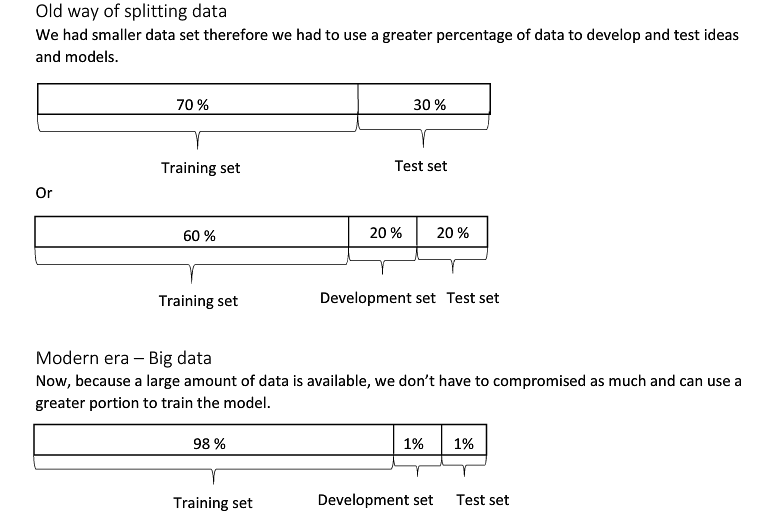
举个例子，你有一个猫分类器，然后在图中这些区域运营，那你应该如歌设立开发集和测试集呢？其中一个做法是，选择前四个，或可随机选四个区域的数据构成开发集，其他四个区域的数据构成测试集，事实证明，这样选择很糟糕，因为你的开发集和测试集来自不同的分布。建议开发集和测试集应该来自同一分布，建立一个开发集+单实数评估指标，就像是定下目标，告诉团队那就是你要瞄准的靶心，因为一旦建立了这样的开发集和指标，团队就可以快速迭代，然后尝试不同的想法，做实验，针对开发集上的指标优化。不然可能出现情况，在开发集上花了很多时间，效果也很好，最后在测试集上应用时，数据差异可能很大，最后发现几乎崩溃。



为了避免这种情况，就该将所有数据随机洗牌，放入开发集和测试集，所有开发集和测试集数据都来自八个地区的数据，并且来自同一分布。 建议在设立开发集和测试集时，要选择这样的数据，能够反映你未来会得到的数据，并且认为可以得到好结果的数据，不管你选择什么数据，一旦你算法表现不错，就多收集类似的数据，而且都要随机分配到开发集和测试集上。还没说如何设立训练集，这节主要是如何设立开发集和指标，定义要瞄准的目标，希望通过同一分布中设立开发集和测试集，设立训练集的方式则会影响你逼近那个目标有多快，

* 1. 开发集和测试集的大小

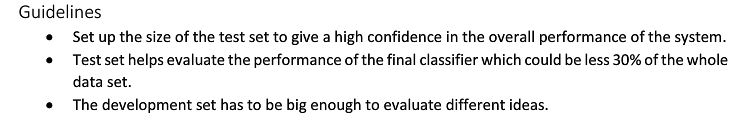
也许听说过机器学习中的一条经验法则，就是将你拥有的所有数据，照70/30的比例，割成训练集和测试集，需要建立训练集，发集和测试集，可以60%的数据作为训练集，20%作为开发集，20%为测试集。早期的机器学习时代，这曾经是非常合理的做法。但是在现代机器学习中，们往往需要处理更大量的数据，假如你有100万个训练样例，那么一种合理的做法是，使用98%的数据作为训练集，1%作为开发集，1%作为测试集。D和T分别表示开发(Dev)和测试(Test)集。因为深度学习算法极度需要数据，拥有大量数据的问题中，训练集会占有更高的比例。



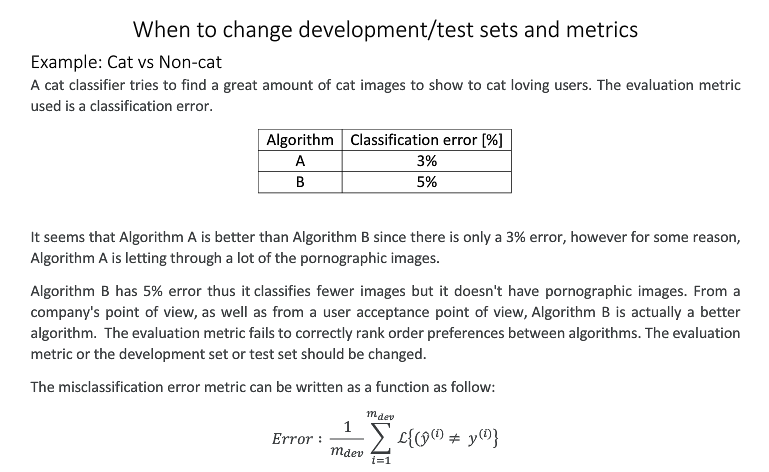
测试集的作用是在系统开发完成后帮我们评估最终系统的性能，因此 测试集的大小只要足够能保证对系统整体性能评估的高置信度即可，除非你需要对最终系统性能有非常精确的测量，否则你的测试集并不需要上百万个样例。或许你认为1万个样例就可以提供足够的置信度来评估性能，者也可能需要10万个或更多才够。这个数目可能远小于总数据量的30%，这取决于你所拥有的总数据量。

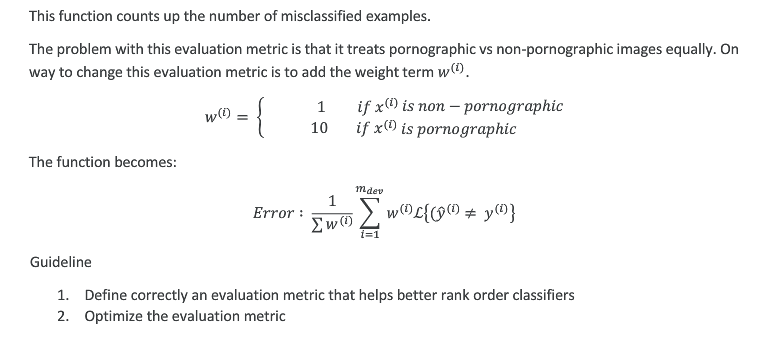
在某些应用场景中可能你并不需要对最终系统，体性能的评估具有很高的置信度，也许你只需要训练集和开发集，那么我认为没有测试集也是可以的。事实上有时候人们说，他们划分了训练集和测试集，但实际上却是在测试集上进行迭代，此并不存在测试集。他们只是划分了训练集和开发集，有测试集，果在这个数据集上进行调优，那么将这个所谓的测试集，称为开发集会更加合适。有时候所谓的测试集，际上是开发集，如果我们只关心有用于训练的数据和用于调优的数据，然后直接上线最终的系统而不太关心它的实际性能，我认为也可以只使用训练集和开发集，而不使用测试集。然而在建议大家在开发系统时不要舍弃测试集。但是如果你有一个很大的开发集，相信不会对开发集过拟合得太厉害，那么只使用训练集和开发集，也不是完全不合理。

经验法则是：确保开发集的大小足以达到其目的，就是帮我们评估不同的想法，我们能更好地从A或B中做出选择，而测试集的目的是对最终的分类器进行评估。只要让测试集的大小足以满足这一目的即可，这可能远小于总数据量的30%。

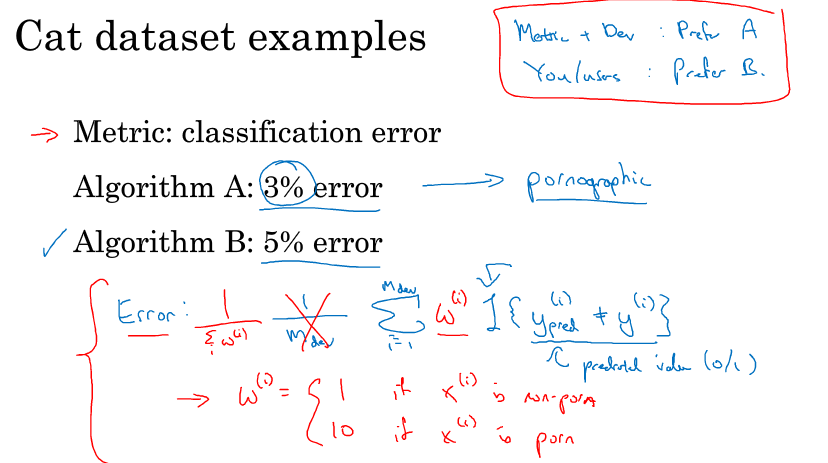


* 1. 什么时候改变开发集，测试集以及指标





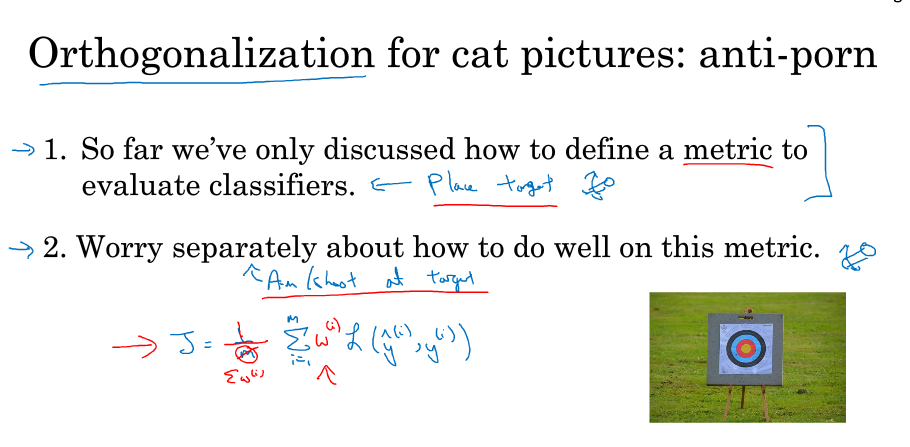
我们已经学习了如何设置开发集和评估指标，这就好比确定你的团队要瞄准的靶子，可有时候项目进行到一半，可能会发现靶子放错了位置，这时候就应该移动你的靶子。例如在下面例子中 当你的评估指标，无法正确地对算法的优劣进行排序时，就像下图猫分裂例子中错误地判断算法B更优秀，这时你就应该，修改评估指标，可能也要修改开发集或测试集。



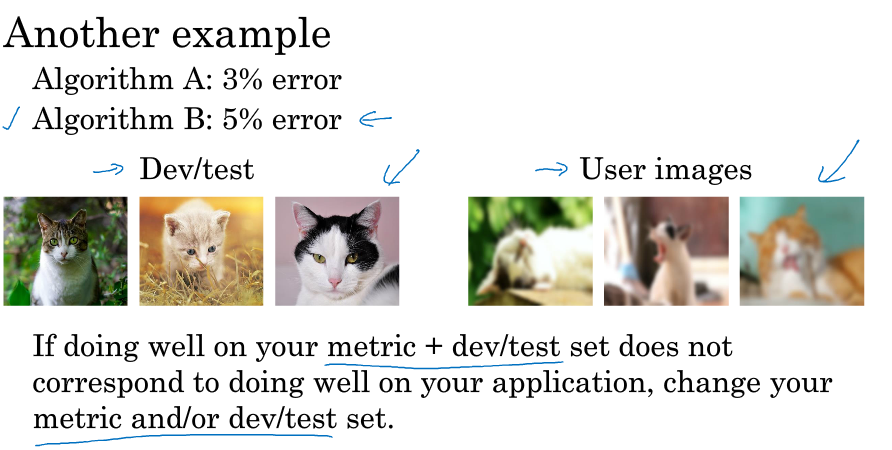
关于使用的分类误差指标，可以换种计算方式：

代表验证集的样本数量。这个公式统计了分类错误的样例的数量，I那个函数是返回样本标记错误的个数。但是在这个公式里，你将猫图和色情图片一视同仁，都错就错了，记录数加1，但事实上色情图分错更不能容忍，所以改善上述式子的一个当时就是在这个评估指标里面加一个权重，比如说如果是色情图分类错误，则乘10，赋予更大的权重。

这里的，如果要归一化常数，前面的就变成，从而令误差在0到1之间。这里权重的细节不重要，实际上要实现这个权重，必须检查一遍开发集和测试集，手动将图片标记出来，这样才能实现这个权重函数，需要记住的是当你发现评估指标，无法对算法的优劣给出正确的排序时，那么就需要考虑定义一个新的评估指标。这里的例子只是定义评估指标的一种方法，评价指标的目的是为了能够准确地告诉你，给出两个分类器，哪一个更适合你的应用。



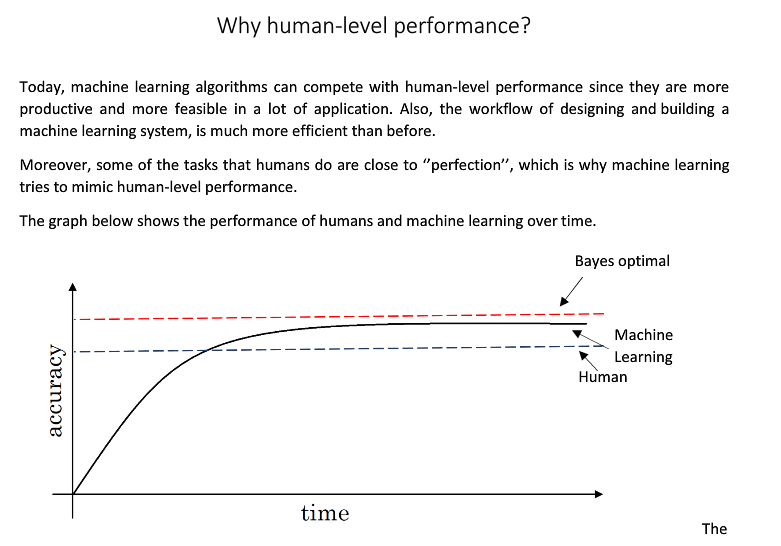
  再次强调如何定义代价函数J并不是重点，重点是这种正交化的思想：放置靶子是第一步，瞄准和射击靶子是另一个独立的步骤，单独地进行。换句话说，我建议大家，将定义指标看成是一步，在定义了指标之后，再考虑如何在这个指标上做好，有时可能需要修改神经网络所优化的代价函数J。

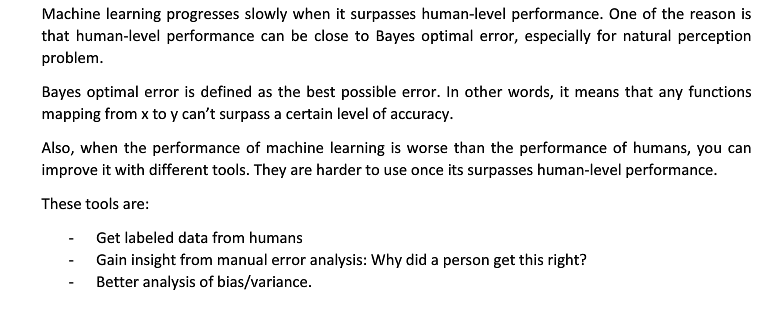


指导方针是如果在你的指标上，以及在当前开发集和测试集的分布上表现得很好，不能对应于在你真正关心的应用场景上也表现得很好，这时就需要修改指标 和/或开发集和测试集。换句话说，当我们发现在具有非常高质量图片的开发集和测试集上进行评估，无法正确预测你的应用的实际表现情况，因为你的应用实际需要处理的是低质量的图片，那么就应该修改你的开发集和测试集，让你的数据能够更好地反应，实际中你真正关心的数据的情况。

建议是即便你无法定义一个完美的评估指标和开发集，你也应该尽快将它们确定下来 以此来驱动你们团队的迭代速度，如果之后发现选的不好，你有了更好的想法你完全可以再进行修改。

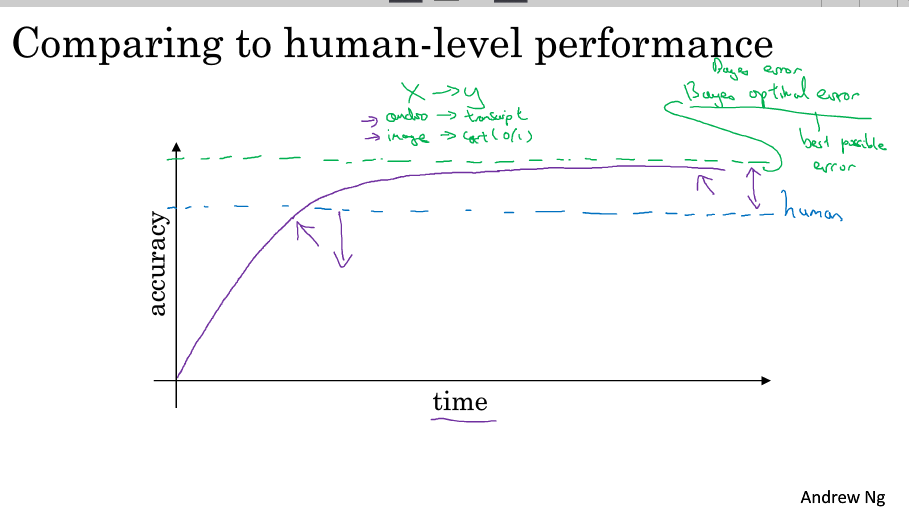
* 1. 为什么是人的表现



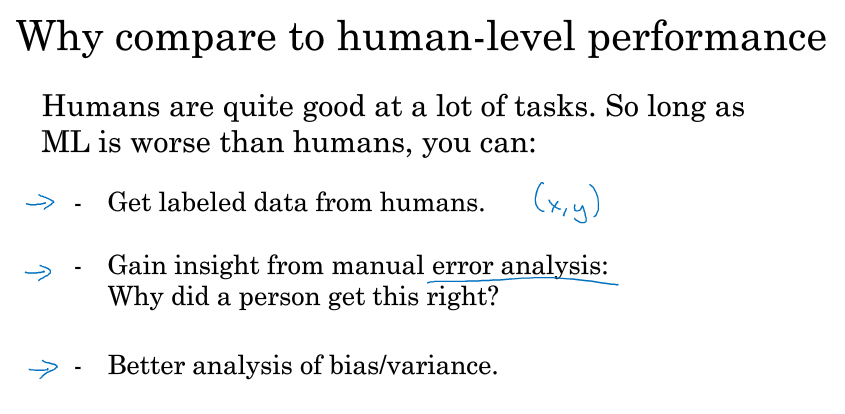


过去的几年中，越来越多的团队一直在谈论将机器学习系统与人类的表现进行比较，为什么这么说呢，我认为有两个主要原因：第一是随着深度学习的发展，机器学习算法的效果迅速提高，使它应用在很多新领域，算法和人相比也更有竞争力；第二，在某些领域用机器学习系统解决问题的效率, 比用人工来解决问题的效率要高。所以，人们自然要比较机器学习和人类表现孰优孰劣。让我们来看看几个例子：

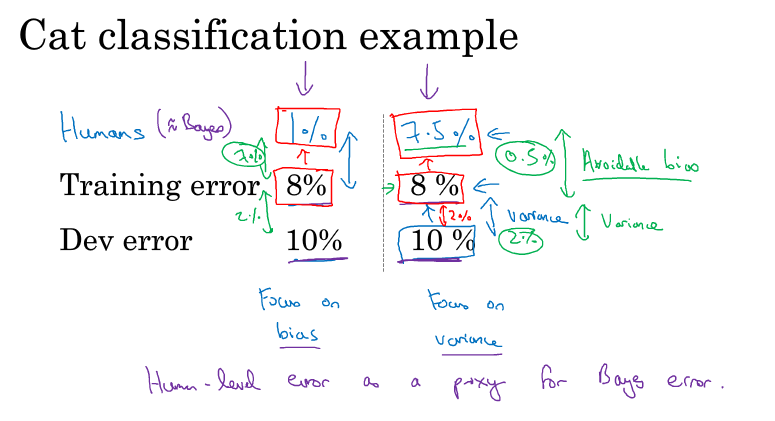
例如机器学习在一个项目上，x代表时间，纵轴是表现，机器学习算法接近人类的表现之前进展相当的快，后来该算法就优于人类的表现，准确性的提升就减慢了，增长速度越来越慢，理想状态就是能达到理论最佳水平。当你继续训练算法时，也许是越来越大模型和更多更大的数据，其效果逼近但从来不会超越一个理论值，这称为贝叶斯最优误差。所以贝叶斯最优误差就是最小的理论误差值。

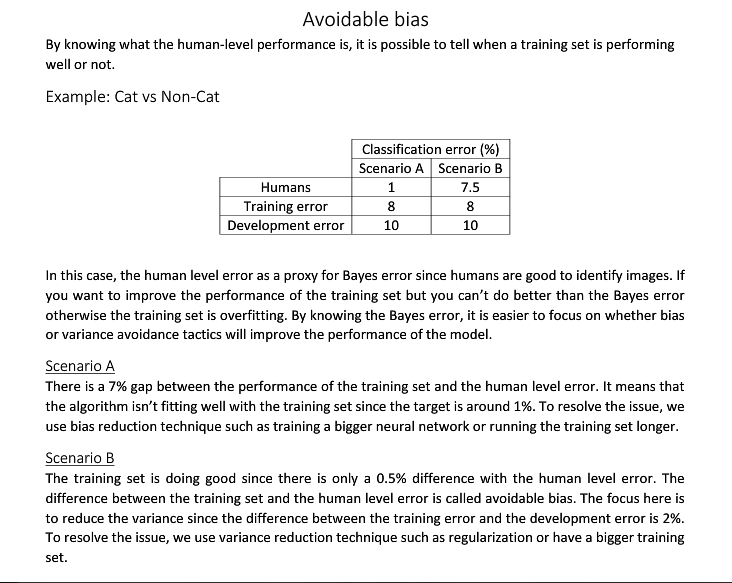


若超越了人类的表现，有时反而慢下来，我认为有两个主要原因：为什么当你超越人类级别的表现时进展往往放慢，原因之一，人类级别的表现，在许多任务中都离贝叶斯最优误差不远，人非常擅长看着图像去分辨是否有一只猫，或收听音频并写出字幕，因此，可能算法超越人类级别的表现之后并没有那么大的改善空间。第二个原因，只要你的表现还不如人类水平，那么实际上你可以用某些工具来提高，而当你超越了人类的水平后，就很难再有工具来提高了，就是这样子。



* 1. 可避免误差

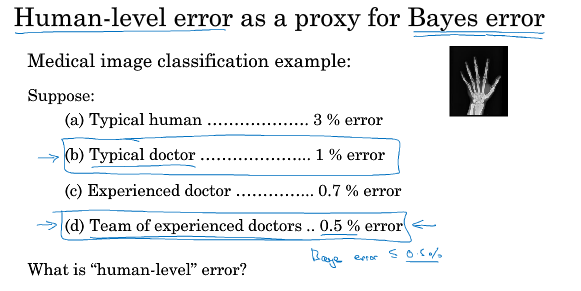




所以在上面的例子中，当你理解人类的表现水平，理解你对贝叶斯误差的估计，你就可以在不同的场景中专注于不同的策略。使用避免偏差策略还是避免方差策略。在训练时如何考虑人类水平表现在决定工作着力点，具体怎么做。

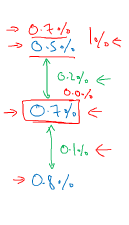
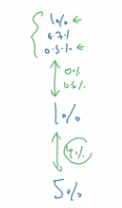
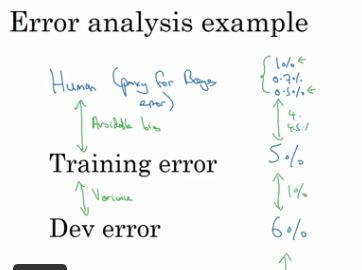
* 1. 理解人的表现

关于人类水平表现的定义？

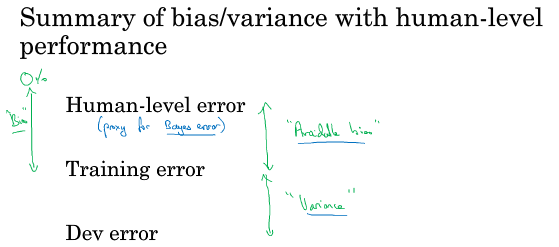


思考人类水平误差最有用的方式之一就是把它当作贝叶斯误差的代替或估计（proxy）。人类水平误差的定义：如果你想要替代或估计贝叶斯误差，那么一队经验丰富的专家讨论过后可以达到0.5%的误差，就知道贝叶斯误差小于等于0.5%。那么在这个背景下，就可以用0.5%估计贝叶斯误差，所以将人类水平定义为0.5%，你希望使用人类水平表现来分析偏差和方差的时候。当然在你研究的内容一个普通放射科医生的水平就能代表估计值，那也可以选择1%。

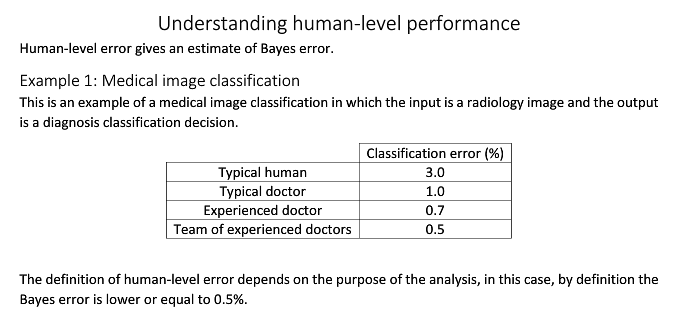
所以重点是在定义人类水平表现时，你要弄清楚你的目标所在。

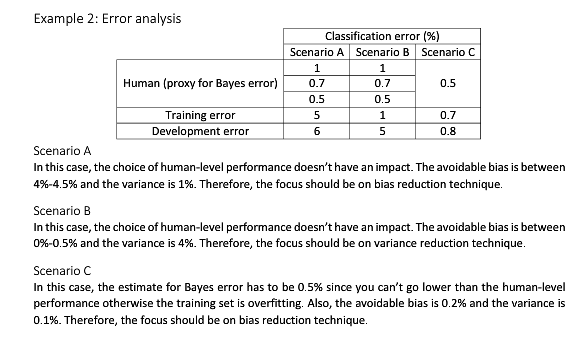


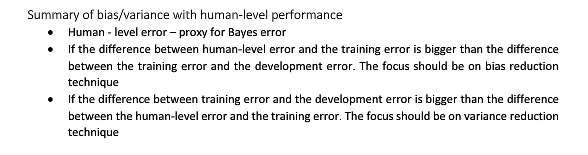
在这个例子中，无论你如何定义人类水平误差，偏差段都明显比方差问题大，所以这种情况下，你应该专注于减少偏差的技术，例如培训更大的网络。同理可得，中间数据的情况下，4%的方差怎么都比任何一种定义的可避免偏差大，所以应该专注于减少方差的工具，比如正则化，或者更大的训练集。第三种也就是最右边的情况，使用0.5%来估计贝叶斯误差关系就很大，因为测量的偏差误差是方差误差的两倍，这也许表明偏差和方差都存在问题，但是可避免偏差问题更严重，这里的0.5%就是对贝叶斯误差的最佳估计，这个例子也结识了机器学习问题上取得进展会越来越难，当你接近人类水平是进展会越来越难。除非你非常小心的估计贝叶斯误差，你可能无法知道离贝叶斯误差到底有多远，所以你应该尽量减少可避免误差。比如你只知道一个医生的水平是1%，可能很难知道是不是应该去继续拟合训练集了。这种问题只出现在你的算法已经做得很好的时候了。在左边两个例子，当你还很远离人类水平时，将优化目标放在偏差或方差上可能更容易一点。



这节内容和以前的重大区别就是以前比较的都是训练误差和0%，直接用这个值估计偏差，但实际上理论最好的值也就是贝叶斯误差了，很难达到0%。所以对于有噪声点，最佳不能达到0%的时候，很好的估计贝叶斯误差就可以帮助你更好的估计可避免偏差和方差，然后准确的选择是减少方差策略还是减少误差策略。





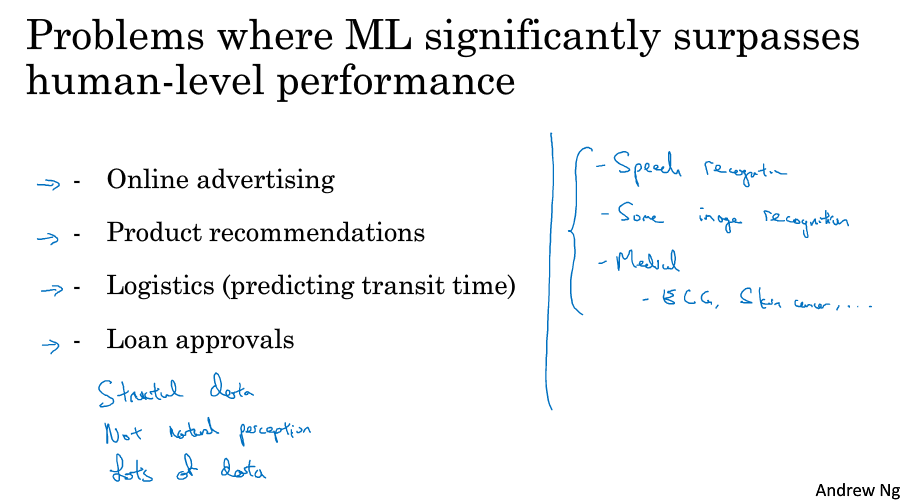


* 1. 超过人的表现

机器学习发展对于越来越多的任务，实际系统可以超越人类了。当性能接近或超过人类水平时，机器学习的进展将会越来越慢。机器学习在很多问题上，已经取得了大大超过人类水平的性能，例如我认为对于在线广告，估计用户点击某个广告的概率，学习算法的性能可能已经远超任何人类了，还有产品推荐，为用户推荐电影或者书什么的，我认为当今的网站可以做得比你最亲密的朋友还要好。或者物流，预测你从A地到B地开车需要的时间，或者预测快递车辆从A地到B地需要的时间，或预测某人是否会还贷，据此判断是否应该批准贷款申请。

如今机器学习的表现都已经大大超过了人类值得注意的是，以上四个例子

都是从结构化的数据中进行学习，都有对应的存放结构化数据的数据库，这些都不是自然感知(Natural Perception)问题，不是计算机视觉，或者语音识别，或自然语言处理的任务，人类非常擅长自然感知的任务。这使得计算机在自然感知领域超过人类水平，虽然是可能的，但是难度更大一点。最后 在上述所有问题中，团队都要能获取到海量的数据，例如上述四种应用中，最好的系统看过的数据可能比任何一个人类所能看的都多，这使得它相对容易取得超过人类水平的性能，事实上由于计算机可以检索海量的数据，所以它比人类更擅长挖掘数据中的统计规律。在一些计算机视觉和一些图像识别任务上，医学相关的任务，计算机也已经很擅长了。一件令人兴奋的事就是，甚至在自然感知领域，在某些情况下我们也能取得超过人类水平的性能，不过这会更难，因为人类很擅长这种自然感知的任务，所以取得超过人类水平的性能通常都不容易，但是基于足量的数据，已经有很多深度学习系统，在单个监督学习问题上已经超过了人类的水平。



* 1. 改善你的模型的表现

已经学过正交化、如何设置开发集和测试集、用人类水平代表贝叶斯误差、以及如何估计可避免偏差和方差，让我们把这些结合成一套准则用于改进你的学习算法。一个有监督学习算法能发挥作用意味着可以做到两件事：第一是你可以很好地拟合训练集，你可以大致认为你能得到较低的可避免偏差；需要做好的第二点是，训练集的结果可以很好地推广到开发集或者测试集，这就是说方差不太大。

