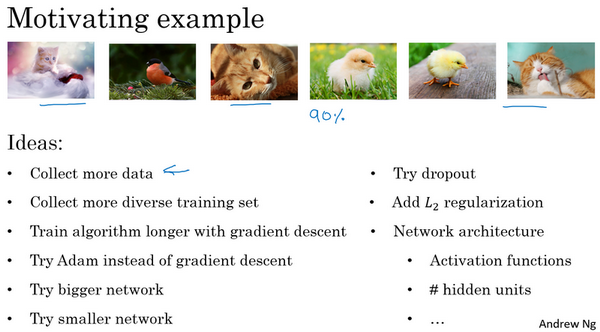
# 机器学习策略

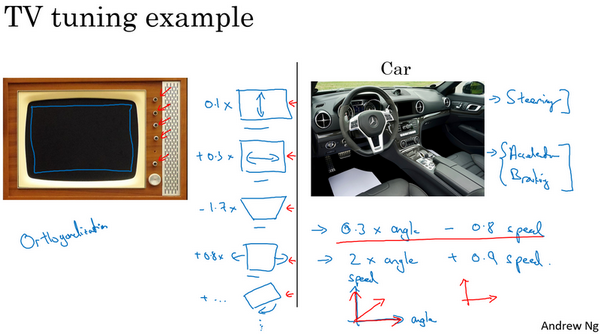
## 为什么是ML策略

我们从一个简单的案例开始谈论什么是机器学习策略。我们正在训练一个猫片分类器，经过一段时间的调整，系统达到了百分之90的准确率，但对我们的应用程序来说还不够好。如图所示，我们有很多种不同的方法可以尝试，

但是使用错误的策略，会导致浪费很多时间，并且对效果毫无增益。我们应该要使用一种策略，来决定优先使用哪种方法来处理问题。

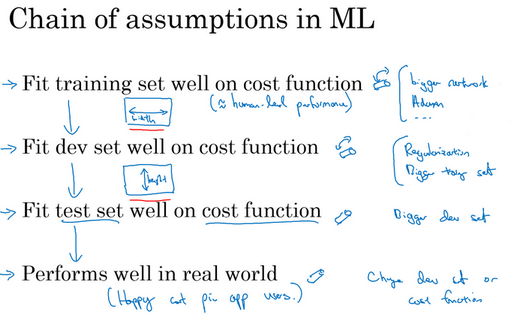
## 正交化

搭建机器学习系统过程中，我们遇到一个很重大的挑战是，我们有很多种超参数可以调试，我们可以注意到，那些经验丰富的专家在调试时会遵循这样一个步骤，对于某个调整，只达到一种效果。这种调整方法我们称之为正交化。

如图所示，若一样操作只调整一个目标，我们可以很容易调整好，但若一样操作调整单个目标，我们就难以调整好。

那么这和机器学习有什么关系呢？要弄好一个机器学习系统，我们通常需要调整好系统的旋钮。我们要确保这四件事，首先我们必须确保系统在训练集上得到的结果不错，对于某些应用，这意味着要在训练集上达到人类的表现的程度。

在训练集上有良好的表现后，我们希望能在开发，测试集上都有良好的表现，最后，我们希望系统在实际表现中能令人满意。

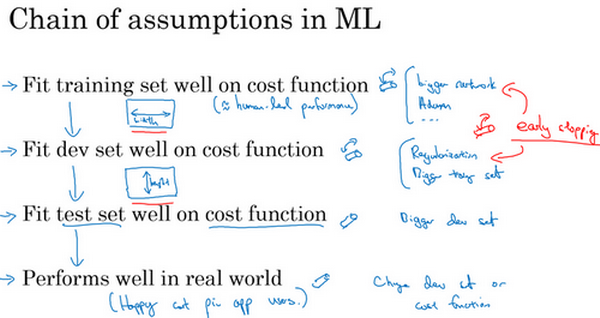


如果我们在训练集上的表现很差，我们需要系统能良好的拟合训练集，我们可能选择的方法有训练一个更大的神经网络，或者切换到更好的优化方法，比如说Adam。

当模型对开发集拟合很差，我们有一些其他工具来进行调整。比如说增大训练集。

当解决完第二个问题后，我们来处理第三个问题。我们可能是需要使用更大的开发集。

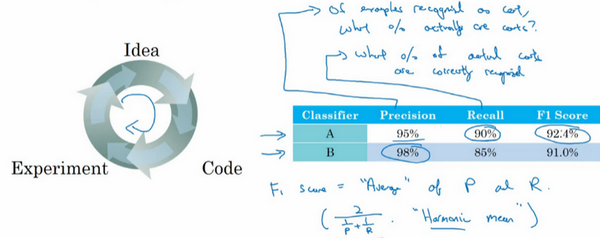
最后，如果在用户反馈阶段表现得不好，这可能意味着样本分布不正确，或者我们设置的损失函数有问题。我们要确保开发集和真实世界来自同一分布。



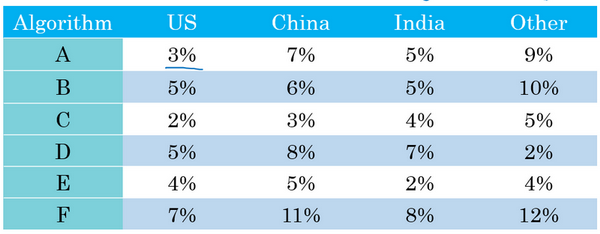
当训练神经网络时，通常不建议使用early stopping，因为这意味着不正交化。

## 单一数字评估指标

当我们在调整超参数，或者使用不同的算法时，使用一个单一实数作为评估指标会让我们的开发迭代快很多。

看这个查全率和查准率的例子，

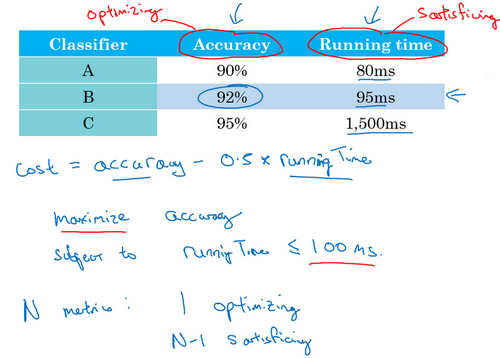
有两个评估指标，我们并不能一目了然的观察到哪一个算法更好。所以我们引入F1 Score，来综合评测这个算法的优劣情况。

我们来看另一个例子，这是我们的算法在不同地区的表现情况。

可以看到，这里有多个指标，并不能让我们对算法的优劣情况直接估计，一个可能的解决办法是使用各地区的平均值作为评估指标。

## 满足和优化指标

要把我们所顾及的所有事情都设置成单一实数指标并不容易，这时候我们需要设立满足指标和优化指标。

假设我们很看重猫片分类器的准确率，也就是F1-Score,但除了准确率以外，我们还需要考虑运行时间，就是需要多长时间来分类一种图，如图所示。

我们对时间并无优化要求，而需要准确率越高越好，因此我们对时间进行一个特殊设置，只要时间能够满足这个要求，就不需要追求最短时间。比如我们设置时间为100ms，在这种情况下，分类器C不满足要求被舍弃，分类器B就是在这种情况下的最优分类器。

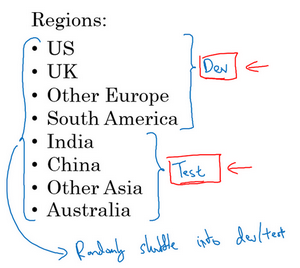
所以更一般地说，如果你要考虑N个指标，有时候选择其中一个指标做为优化指标是合理的。所以你想尽量优化那个指标，然后剩下N-1个指标都是满足指标，意味着只要它们达到一定阈值，例如运行时间快于100毫秒，但只要达到一定的阈值，你不在乎它超过那个门槛之后的表现，但它们必须达到这个门槛。

## 训练 开发 测试集划分

如何划分训练，开发，测试集，也能大大的影响开发速度。

在机器学习的工作流程中，我们尝试很多思路，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，然后再选择一个，然后不断迭代去改善开发集的性能，直到最后你得到一个能够令你满意的成果，然后你再用测试集去评估。

举个例子，还是猫片分类，我们的分类器要在以下这些区域运行，那么我们该如何划分这三个集呢？



其中一种可能的做法是，随机选择其中的四个区域作为开发集，再用其中的其他四个区域作为测试集。实际上，这个想法非常糟糕，因为测试集和开发集来自于不同分布。测试集和开发集分布不同，相当于改变目标，因此在开发集上表现好的数据，在测试集中可能会表现很差。

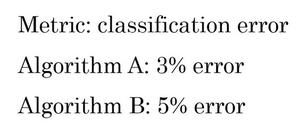
我建议的是你将所有数据随机洗牌，放入开发集和测试集，所以开发集和测试集都有来自八个地区的数据，并且开发集和测试集都来自同一分布，这分布就是你的所有数据混在一起。

在设计开发集和测试集时，数据的分布要尽可能和需要应用的情况来自于同一分布。不管那些数据是什么，都要随机分配到开发集和测试集上。

## 开发集和测试集的大小

在深度学习时代，开发集和测试集占比的情况和传统的机器学习并不相同。在传统机器学习算法中，我们使用百分之60的数据作为训练集，百分之20的数据作为开发集，百分之20的数据作为测试集。在传统机器学习算法中，我们的样本数量大概为几千到一万。但在深度学习时代，我们拥有几十万几百万的数据集，此使，一万个样本大小的测试集就足以对我们的系统做出精准评估了。这个比例远远少于百分之20.

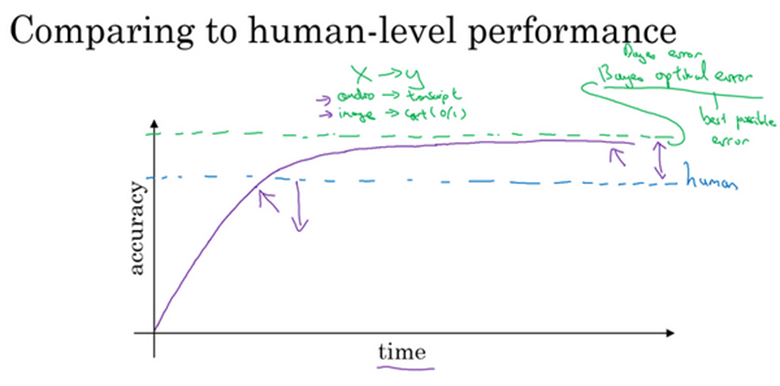
## 什么时候该改变开发/测试集指标

在项目进行的过程中，我们可能会意识到我们的目标设置错了，在这种情况下，我们应该要移动我们的目标。我们仍然以猫片分类器为例，如图所示。算法A只有百分之3的错误率，算法B百分之5的错误率，这么看来也许算法A更加优秀。但是当我们在实际情况中使用这些算法时，我们会发现由于某些原因，算法A把很多色情图片都识别为猫了，这让我们的用户感到十分不适。相反，算法B虽然错误率稍微高一点，但是却更少的色情图片，这种情况下我们的用户通常会偏好算法B.

这种情况下，当前使用的评估指标已经无法准确反映我们的需求了，所以我们需要改变评估指标，或者改变开发集和测试集。我们可以增大损失函数中色情图片的权重项。如何定义J并不重要，关键在于正交化的思路，把设立目标定为第一步，然后瞄准和射击目标是独立的第二步。我们把定义指标视作第一步，之后再针对指标进行相应的优化。如果当前开发集和测试集无法准确表示实际情况中的样本分布，那么我们要做的就是改变开发集和测试集。

## 为什么是人的表现

在过去几年里，机器学习从业者总是在比较机器学习系统和人类的表现，这是因为深度学习的发展使得很多机器学习系统的性能大幅度提高，足以接近人类水准。第二个原因是人类可以用自身的行为来改善机器学习系统的性能。

我们来看如图所示的这个例子，当机器学习系统表现不如人类时，系统的进展是可以很快的，

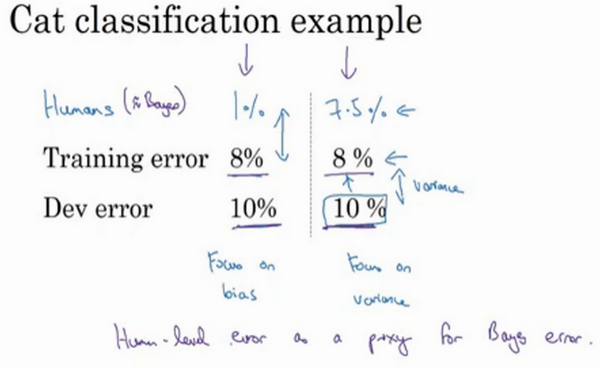
但在超过人类表现后，这个系统性能的上升会变得很缓慢。随着时间的推移，系统的性能会接近理论上的最优误差，但是无法超越。这就是所谓的贝叶斯误差。

超过人类表现后进展变慢的原因有这些，一是在很多领域中，人类的表现已经是十分接近于贝叶斯误差，所以超过人类表现后就没有多少空间可以用来改善了。另一个原因是，只要你的机器表现比人类差，你就可以使用人工标记数据集或者靠人分析数据的方法来改善机器表现。

## 可避免偏差

对于误差分析，我们需要知道人类的表现。比如说猫片识别，假设人类在这个任务上的错误率是％1，而你的机器学习系统的错误率是％10，那么，就说明你的机器学习系统欠拟合，在处理方差和偏差的问题上，我会优先处理偏差。

再看看这个例子，

假如人类的误差是百分之7.5，那么你的算法可以说是表现还好，这是观察到方差是％2，因此应该要优先处理方差问题。在这种图片识别的应用中，人类的表现和贝叶斯误差不会差距很远，所以当我们观察到训练误差为％8时，就知道我们的优化空间并不多了，而开发集上的误差差距还有％2，＞％0.5，因此我们要优先处理方差问题。即贝叶斯误差和训练误差之间的差距称为可避免偏差。

## 理解人类的表现

什么是人类的表现？在通常情况下应该要定义为人所能达到的最优表现，但是同时这也可以取决于你自己的定义，如人类的平均表现。

## 超过人的表现

当你的算法表现超过人的表现时，你就难以估计贝叶斯误差，也无法得知算法的表现是由于过拟合还是因为未达到贝叶斯误差。这是也没有足够的信息来判断是需要优先减少偏差还是方差。而且这是也无法根据人类直觉来判断使用何种方法可以改善模型表现。在有结构化数据的问题处理中，计算机超过人类的表现比较简单，如快递时间预测等。

## 改善模型的表现

我们想要让一个监督学习算法能够实用，得满足以下几点条件，一，算法对训练集拟合的很好，也就是可避免偏差很低。二是方差不是太大。在正交化的方法下，可以修正可避免偏差问题，比如训练更大的网络或者训练更久。还有一套独立的技巧可以用来处理方差问题，比如正则化或者收集更多训练数据。

如果你想提升机器学习系统的性能，我建议你们看看训练错误率和贝叶斯错误率估计值之间的距离，让你知道可避免偏差有多大。接着再观察开发集上的表现，看看开发错误率，

换句话说，你应该做多少努力让你的算法表现能够从训练集推广到开发集，算法是没有在开发集上训练的。