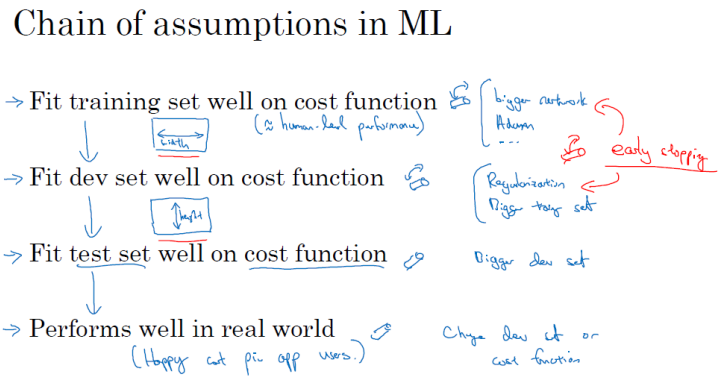
1 机器学习策略

构建好一个机器学习系统并获得一些初步结果时，为得到最令人满意的结果，后续往往还需要进行大量的改进。如前面优化神经网络中所述，改进的方法多种多样，可能是收集更多的数据，或者是进行正则化，或者是采用不同的优化算法。

想要找准改进的方向，使一个机器学习系统更快更有效地工作，需要学习一些在构建机器学习系统时常用到的策略。

1.1 正交化

正交化（Orthogonalization）是确保修改一个系统中的某个算法指令或者组件时，不会产生或传播副作用到系统种的其他组件的一种系统设计属性。它使得验证一个算法独立于另一个算法时变得更加容易，同时也能减少设计和开发的时间。其核心在于每次调整只会影响模型某一方面的性能，而对其他功能没有影响。这种方法有助于更快更有效地进行机器学习模型的调试和优化。



当设计一个监督学习系统，需做到符合下面四个假设且它们是正交的：

- 建立的模型在训练集上表现良好；
  - 训练集上表现不够好–尝试采用更大的神经网络或者换一种更好的优化算法；
- 建立的模型在验证集上表现良好；
  - 验证集上表现不够好–尝试进行正则化处理或者加入更多训练数据；
- 建立的模型在测试集上表现良好；
  - 测试集上表现不够好–尝试采用更大的验证集进行验证；
- 建立的模型在实际应用中表现良好。
  - 现实应用中表现不够好–可能是因为测试集没有设置正确或者成本函数评估出错。

面对遇到的各种问题，正交化能够帮助我们更为精准有效地解决问题。

一个反例是早停止法（Early Stopping）。如果早期停止，虽然可以改善验证集的拟合表现，但是对训练集的拟合就不太好。因为对两个不同的“功能”都有影响，所以早停止法不具有正交化。虽然也可以使用，但是用其他正交化控制手段来进行优化会更简单有效。

1.2 找准目标

1.2.1 单一数字评估指标

构建机器学习系统时，通过设置一个量化的**单值评价指标（single-number evaluation metric）**，可以使我们根据这一指标比较不同超参数对应的模型的优劣，从而选择最优的那个模型。

例如，对于二分类问题，常用的评价指标是**精确率（Precision）**和**召回率（Recall）**。假设我们有 A 和 B 两个分类器，其两项指标分别如下：

	1	0
1	True Positive	False Positive
0	False Negative	True Negative

其中横轴为实际的值 y，纵轴为预测值 $\hat{y}$

准确率： $P = \frac{TP}{TP+FP}$

召回率： $R = \frac{TP}{TP+FN}$

实际应用中，我们通常使用综合了精确率和召回率的单值评价指标 F1 Score 来评价模型的好坏。F1 Score 其实就是精准率和召回率的**调和平均数（Harmonic Mean）**，比单纯的平均数效果要好。

F1 度量值： $F1 = \frac{2}{\frac{1}{P}+\frac{1}{R}} = \frac{2PR}{P+R}$

Classifier	Precision (p)	Recall (r)	F1 Score
A	95%	90%	92.4%
B	98%	85%	91.0%

如此，算出上图种A分类器的F1度量值为92.4%，B分类器的为91.0%，从未得知A分类器效果好些。这里F1度量值就作为了单一数字评估指标。

1.2.2 满足、优化指标

Classifier	Accuracy	Running time
A	90%	80 ms
B	92%	95 ms
C	95%	1 500 ms

有时，评判的标准不限于一个单一数字评估指标。比如上图中的几个猫分类器，想同时关心它们各自的识别准确率和运行时间，但如果把这两个指标组合成一个单一数字评估指标的话，就不太好了。这时，就需要把一个指标作为**优化指标（Optimizing Metric）**，而另外的一些的作为**满足指标（Satisficing Metric）**。

如上面所举的例子中，准确率就是一个优化指标，因为想要分类器尽可能做到正确分类，而运行时间就是一个满足指标，如果你想要分类器的运行时间不多于某个值，那你需要选择的分类器就应该是以这个值为界里面准确率最高的那个，以此作出权衡。

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。

动态改变评价指标

除了采用这些标准来评判一个模型外，也要学会在必要时及时地调整一些评判指标，甚至是更换训练数据。例如两个猫分类器A和B的识别误差分别为3%和5%，但是处于某种原因，A识别器会把色情图片误识为猫，引起用户的不适，而B不会出现这种情况，这时，识别误差大一些的B反而是更好的分类器。可以用以下公式来计算错误识别率：

$$Error: \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\hat{y}^i \neq y^i \tag{1}$$

还可以设置一个 $w^{(i)}$，当 $x^{(i)}$ 是色情图片时，$w^{(i)}$ 为 10，否则为 1，以此来区分色情图片及其他误识别的图片：

$$Error: \frac{1}{\sum w^{(i)}} \sum_{i=1}^{m} w^{(i)} \mathcal{L}\hat{y}^{i} \neq y^{i} \tag{2}$$

1.2.3 数据处理

1.2.3.1 训练/开发/测试集划分

我们一般将数据集分为训练集、验证集、测试集。构建机器学习系统时，我们采用不同的学习方法，在训练集上训练出不同的模型，然后使用验证集对模型的好坏进行评估，确信其中某个模型足够好时再用测试集对其进行测试。

因此，训练集、验证集、测试集的设置对于机器学习模型非常重要，合理的设置能够大大提高模型训练效率和模型质量。

1.2.3.2 验证集和测试集的分布

验证集和测试集的数据来源应该相同（来自同一分布）、和机器学习系统将要在实际应用中面对的数据一致，且必须**从所有数据中随机抽取**。这样，系统才能做到尽可能不偏离目标。

1.2.3.3 验证集和测试集的大小

过去数据量较小（小于 1 万）时，通常将数据集按照以下比例进行划分：

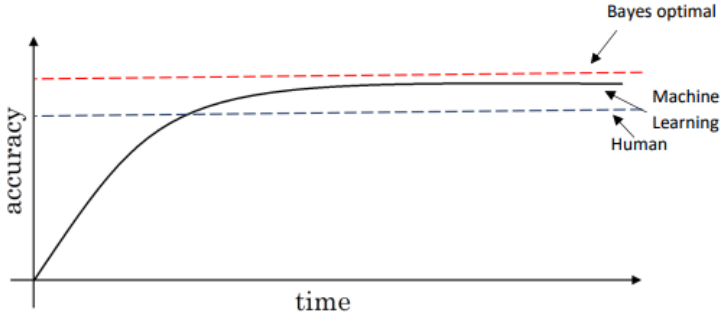
- 无验证集的情况：70% / 30%；
- 有验证集的情况：60% / 20% / 20%；

这是为了保证验证集和测试集有足够的数据。现在的机器学习时代数据集规模普遍较大，例如 100 万数据量，这时将相应比例设为 98% / 1% / 1% 或 99% / 1% 就已经能保证验证集和测试集的规模足够。

测试集的大小应该设置得足够提高系统整体性能的可信度，验证集的大小也要设置得足够用于评估几个不同的模型。应该根据实际情况对数据集灵活地进行划分，而不是死板地遵循老旧的经验。

1.3 比较人类表现

如今，设计和建立一个机器学习系统比以前变得更为简单高效，一些机器学习算法的在很多领域的表现已经可以和我们人类一决高下。所以在这些场合，比较人类和机器是很自然的，或者你要让机器模仿人类的行为。



上图展示了随着时间的推进，机器学习系统和人的表现水平的变化。一般的，当机器学习超过人的表现水平后，它的进步速度逐渐变得缓慢，最终性能无法超过某个理论上限，这个上限被称为**贝叶斯最优误差（Bayes Optimal Error）**。

贝叶斯最优误差一般认为是理论上可能达到的最优误差，换句话说，其就是理论最优函数，任何从 x 到精确度 y 映射的函数都不可能超过这个值。例如，对于语音识别，某些音频片段嘈杂到基本不可能知道说的是什么，所以完美的识别率不可能达到 100%。

因为人类对于一些自然感知问题的表现水平十分接近贝叶斯最优误差，所以当机器学习系统的表现超过人类后，就没有太多继续改善的空间了。

也因此，只要建立的机器学习模型的表现还没达到人类的表现水平时，就可以通过各种手段来提升它。例如采用人工标记过的数据进行训练，通过人工误差分析了解为什么人能够正确识别，或者是进行偏差、方差分析。

当模型的表现超过人类后，这些手段起的作用就微乎其微了。

1.3.1 可避免偏差

通过与贝叶斯最优误差，或者说，与人类表现水平的比较，可以表明一个机器学习模型表现的好坏程度，由此判断后续操作应该注重于减小偏差还是减小方差。

模型在**训练集上的误差**与**人类表现水平**的差值被称作可避免偏差（Avoidable Bias）。可避免偏差低便意味着模型在训练集上的表现很好，而训练集与验证集之间错误率的差值越小，意味着模型在验证集与测试集上的表现和训练集同样好。

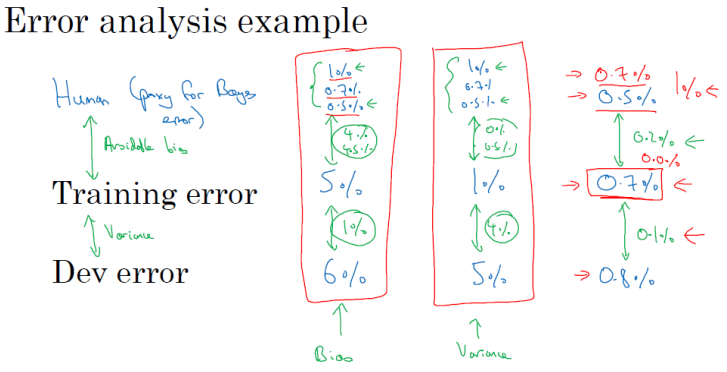
	Classification error (%)	
	Scenario A	Scenario B
Humans	1	7.5
Training error	8	8
Development error	10	10

例如上图中的两个场景下，将人的错误率和机器学习模型的错误率进行比较，看以看出在A场景下，学习算法的错误率和人的错误率的可避免偏差较大，这种情况下后续的工作就是通过之前介绍过的方法来降低训练集的错误率，以减小偏差。而在B场景下，学习算法和人的表现相当，偏可避免偏差只有0.5%，，后续的工作就应该转向尽可能地减小开发集和训练集那部分2%的方差。

如果可避免偏差大于训练集与验证集之间错误率的差值，之后的工作就应该专注于减小偏差；反之，就应该专注于减小方差。

1.3.2 理解人类表现水平

我们一般用人类水平误差（Human-level Error）来代表贝叶斯最优误差（或者简称贝叶斯误差）。对于不同领域的例子，不同人群由于其经验水平不一，错误率也不同。一般来说，我们将**表现最好的作为人类水平误差**。但是实际应用中，不同人选择人类水平误差的基准是不同的，这会带来一定的影响。



例如，如果某模型在训练集上的错误率为 0.7%，验证集的错误率为 0.8%。如果选择的人类水平误差为 0.5%，那么偏差（bias）比方差（variance）更加突出；而如果选择的人类水平误差为 0.7%，则方差更加突出。也就是说，根据人类水平误差的不同选择，我们可能因此选择不同的优化操作。

这种问题只会发生在模型表现很好，接近人类水平误差的时候才会出现。人类水平误差给了我们一种估计贝叶斯误差的方式，而不是像之前一样将训练的错误率直接对着 0% 的方向进行优化。我们要做的知识准确理解我们的目标是什么，这样才能制定出下一步要优化的方案。

1.3.3 超越人类水平

当机器学习模型的表现超过了人类水平误差时，很难再通过人的直觉去判断模型还能够往什么方向优化以提高性能。

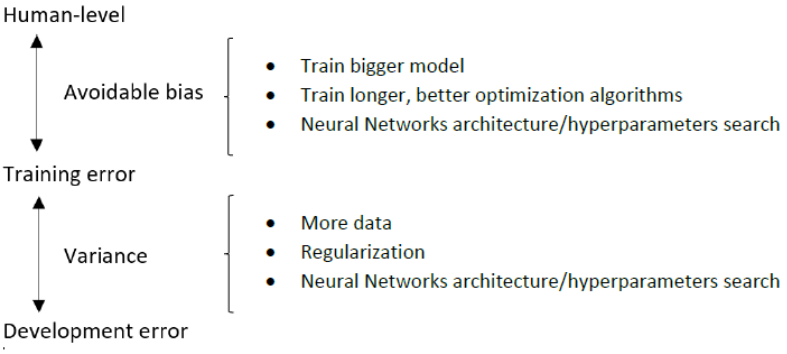
There are many problems where machine learning significantly surpasses human-level performance, especially with structured data:

- Online advertising
- Product recommendations
- Logistics (predicting transit time)
- Loan approvals

1.3.4 改善模型的表现

想让一个监督学习算法达到使用程度，应该做到以下两点：

- 算法对训练集的拟合很好，可以看作可避免偏差很低；
- 推广到验证集和测试集效果也很好，即方差不是很大。



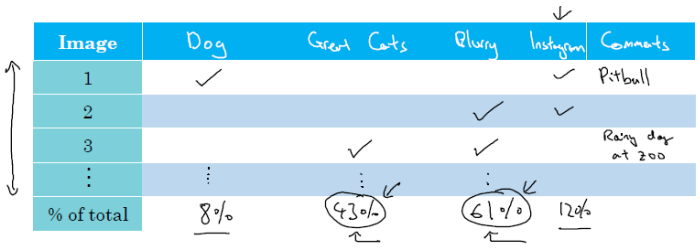
根据正交化的思想，我们有一些措施可以独立地优化二者之一。

1.4 错误分析

通过人工检查机器学习模型得出的结果中出现的一些错误，有助于深入了解下一步要进行的工作。这个过程被称作**错误分析（Error Analysis）**。

比如对于一个猫分类器，在开发组里你已经取得了90%的识别准确率，还存在10%的出错率，而且还发现分类器会将一些看起来像猫的狗的图片误识别为猫，这时就不是立即盲目地转向去做一个能够精确识别出狗的算法，而是先进行错误分析。可以先把学习算法标签错误的图片找出来，然后进行人工检查，假如100张错误标签的图片中有5张是狗的图片，那么也就表明你的学习算法的10%的错误中大致只有5%来自于狗，也就是0.5%的错误来自于狗，这表明改善狗的识别问题并不能给你的学习算法带来多大提升。但如果100张错误标签的图片中有50张是狗的图片，也就是5%的错误是狗产生的，那么对于狗的识别就是你要解决的问题了。这样，先通过少量的时间去分析问题，再决定后面的要进行大方向。

这种人工检查看似简单而愚笨，但却是十分必要的，因为这项工作能够有效避免花费大量的时间与精力去做一些对提高模型性能收效甚微的工作，让我们专注于解决影响模型准确率的主要问题。



在对输出结果中分类错误的样本进行人工分析时，可以建立一个表格来记录每一个分类错误的具体信息，例如某些图像是模糊的，或者是把狗识别成了猫等，并统计属于不同错误类型的错误数量。这样，分类结果会更加清晰。

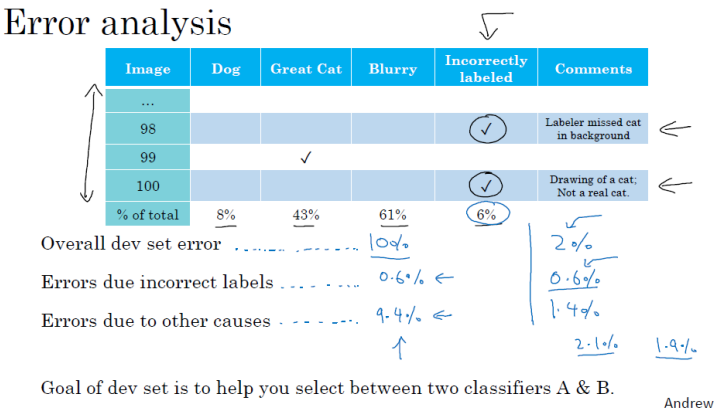
总结一下，进行错误分析时，你应该观察错误标记的例子，看看假阳性和假阴性，统计属于不同错误类型的错误数量。在这个过程中，你可能会得到启发，归纳出新的错误类型。总之，通过统计不同错误标记类型占总数的百分比，有助于发现哪些问题亟待解决，或者提供构思新优化方向的灵感。

1.4.1 修正错误标记

我们用 mislabeled examples 来表示学习算法输出了错误的 Y 值。而在做误差分析时，有时会注意到数据集中有些样本被人为地错误标记（incorrectly labeled）了，这时该怎么做？

如果是在训练集中，由于机器学习算法对于随机误差的**稳健性（Robust）**（也称作“鲁棒性”），只要这些出错的样本数量较小，且分布近似随机，就不必花费时间一一修正。

而如果出现在验证集或者测试集，则可以在进行误差分析时，通过统计人为标记错误所占的百分比，来大致分析这种情况对模型的识别准确率的影响，并比较该比例的大小和其他错误类型的比例，以此判断是否值得去将错误的标记一一进行修正，还是可以忽略。



当你决定在验证集和测试集上手动检查标签并进行修正时，有一些额外的方针和原则需要考虑：

- 在验证集和测试集上同时**使用同样的修正手段**，以保证验证集和测试集来自相同的分布；
- 同时检查判断正确和判断错误的例子（通常不用这么做）；
- 在修正验证集和测试集时，鉴于训练集的分布不必和验证/测试集完全相同，可以不去修正训练集。

1.4.2 快速搭建系统并迭代

对于每个可以改善模型的合理方向，如何选择一个方向集中精力处理成了问题。如果想搭建一个全新的机器学习系统，建议根据以下步骤快速搭建好第一个系统，然后开始迭代：

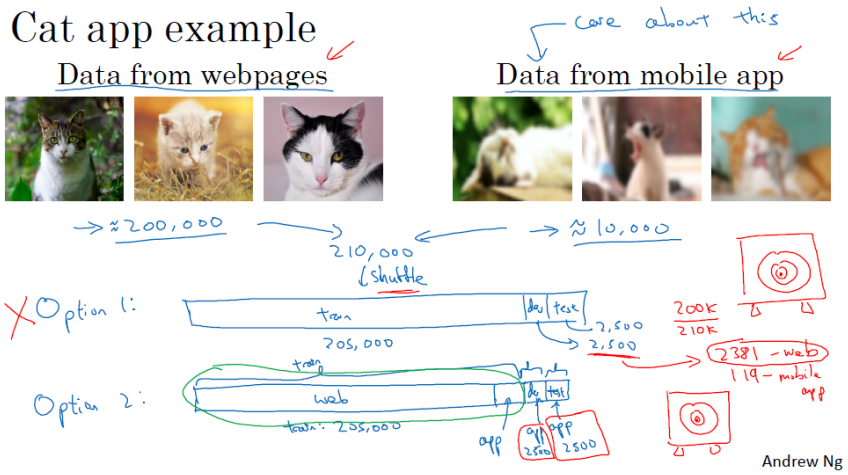
- 设置好训练、验证、测试集及衡量指标，确定目标；
- 快速训练出一个初步的系统，用训练集来拟合参数，用验证集调参，用测试集评估；
- 通过偏差/方差分析以及错误分析等方法，决定下一步优先处理的方向。

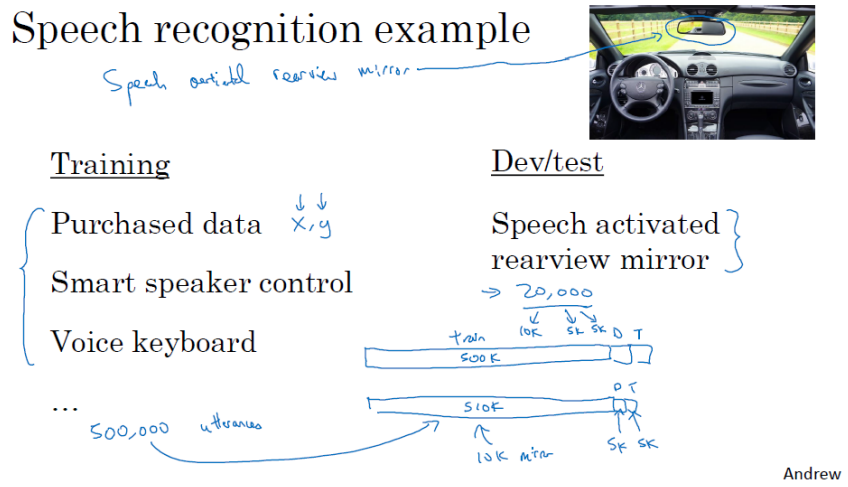
1.5 不匹配的数据集

1.5.1 训练和测试集不同分布

有时，我们很难得到来自同一个分布的训练集和验证/测试集。还是以猫识别作为例子，我们的训练集可能由网络爬取得到，图片比较清晰，而且规模较大（例如 20 万）；而验证/测试集可能来自用户手机拍摄，图片比较模糊，且数量较少（例如 1 万），难以满足作为训练集时的规模需要。

虽然验证/测试集的质量不高，但是机器学习模型最终主要应用于识别这些用户上传的模糊图片。考虑到这一点，在划分数据集时，可以将 20 万张网络爬取的图片和 5000 张用户上传的图片作为训练集，而将剩下的 5000 张图片一半作验证集，一半作测试集。比起混合数据集所有样本再随机划分，这种分配方法虽然使训练集分布和验证/测试集的分布并不一样，但是能保证验证/测试集更接近实际应用场景，在长期能带来更好的系统性能。





1.5.2 偏差和方差的分析

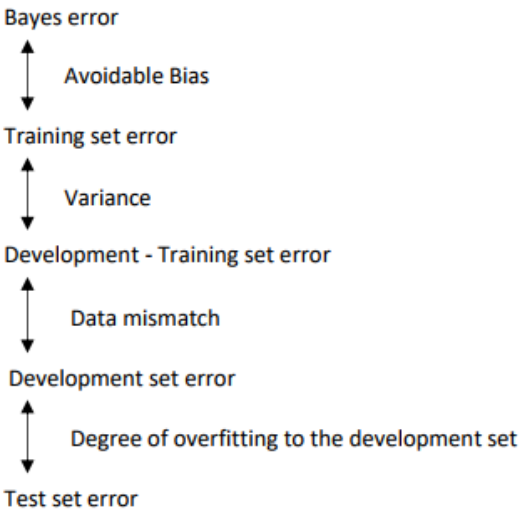
之前的学习中，我们通过比较人类水平误差、训练集错误率、验证集错误率的相对差值来判断进行偏差/方差分析。但在训练集和验证/测试集分布不一致的情况下，无法根据相对差值来进行偏差/方差分析。这是因为训练集错误率和验证集错误率的差值可能来自于算法本身（归为方差），也可能来自于样本分布不同，和模型关系不大。

在可能存在训练集和验证/测试集分布不一致的情况下，为了解决这个问题，我们可以再定义一个训练-验证集（Training-dev Set）。训练-验证集和训练集的分布相同（或者是训练集分割出的子集），但是不参与训练过程。

	Classification error (%)					
	Scenario A	Scenario B	Scenario C	Scenario D	Scenario E	Scenario F
Human (proxy for Bayes error)	0	0	0	0	0	4
Training error	1	1	1	10	10	7
Training-development error	-	9	1.5	11	11	10
Development error	10	10	10	12	20	6
Test error	-	-	-	-	-	6

现在，我们有了训练集错误率、训练-验证集错误率，以及验证集错误率。其中，训练集错误率和训练-验证集错误率的差值反映了方差；而训练-验证集错误率和验证集错误率的差值反映了样本分布不一致的问题，从而说明模型擅长处理的数据和我们关心的数据来自不同的分布，我们称之为数据不匹配（Data Mismatch）问题。

人类水平误差、训练集错误率、训练-验证集错误率、验证集错误率、测试集错误率之间的差值所反映的问题如下图所示：



1.5.3 处理办法

这里有两条关于如何解决数据不匹配问题的建议：

- 做错误分析，尝试了解训练集和验证/测试集的具体差异（主要是人工查看训练集和验证集的样本）；
- 尝试将训练数据调整得更像验证集，或者收集更多类似于验证/测试集的数据。

如果你打算将训练数据调整得更像验证集，可以使用的一种技术是人工合成数据。我们以语音识别问题为例，实际应用场合（验证/测试集）是包含背景噪声的，而作为训练样本的音频很可能是清晰而没有背景噪声的。为了让训练集与验证/测试集分布一致，我们可以给训练集人工添加背景噪声，合成类似实际场景的声音。

人工合成数据能够使数据集匹配，从而提升模型的效果。但需要注意的是，不能给每段语音都增加同一段背景噪声，因为这样模型会对这段背景噪音出现过拟合现象，使得效果不佳。

1.6 多任务和端对端学习

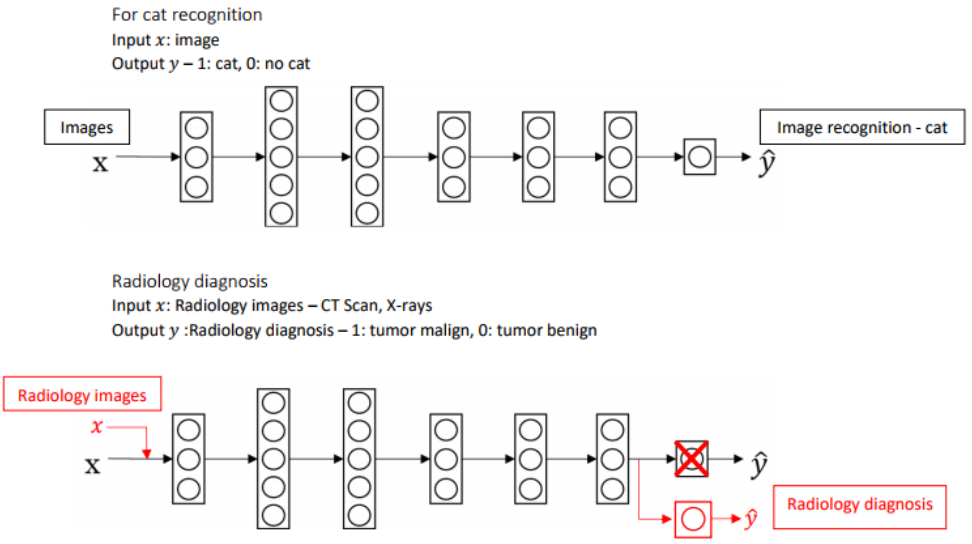
1.6.1 迁移学习

迁移学习（Tranfer Learning）是通过将已训练好的神经网络模型的一部分网络结构应用到另一模型，将一个神经网络从某个任务中学到的知识和经验运用到另一个任务中，以显著提高学习任务的性能。

例如，我们将为猫识别器构建的神经网络迁移应用到放射科诊断中。因为猫识别器的神经网络已经学习到了有关图像的结构和性质等方面的知识，所以只要先删除神经网络中原有的输出层，加入新的输出层并随机初始化权重系数（$W^{[L]}$，$b^{[L]}$），随后用新的训练集进行训练，就完成了以上的迁移学习。

如果新的数据集很小，可能只需要重新训练输出层前的最后一层的权重，即 $W^{[L]}$，$b^{[L]}$，并保持其他参数不变；而如果有足够多的数据，可以只保留网络结构，重新训练神经网络中所有层的系数。这时初始权重由之前的模型训练得到，这个过程称为**预训练（Pre-Training）**，之后的权重更新过程称为**微调（Fine-Tuning）**。

你也可以不止加入一个新的输出层，而是多向神经网络加几个新层。



在下述场合进行迁移学习是有意义的：

- 两个任务有同样的输入（比如都是图像或者都是音频）；
- **拥有更多数据的任务迁移到数据较少的任务**；
- 某一任务的低层次特征（底层神经网络的某些功能）对另一个任务的学习有帮助。

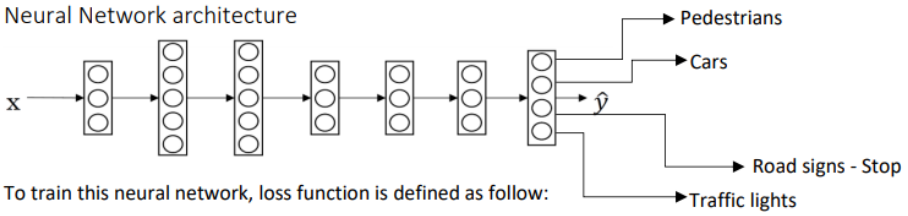
1.6.2 多任务学习

多任务学习（Multi-Task Learning）是采用一个神经网络来同时执行多个任务，且这些任务的执行可以相互促进。

迁移学习中的步骤是串行的；而多任务学习（Multi-Task Learning）使用单个神经网络模型，利用共享表示采用并行训练同时学习多个任务。多任务学习的基本假设是多个任务之间具有相关性，并且任务之间可以利用相关性相互促进。例如，属性分类中，抹口红和戴耳环有一定的相关性，单独训练的时候是无法利用这些信息，多任务学习则可以利用任务相关性联合提高多个属性分类的精度。

以汽车自动驾驶为例，需要实现的多任务是识别行人、车辆、交通标志和信号灯。如果在输入的图像中检测出车辆和交通标志，则输出的 y 为：

$$y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{3}$$



$$-\frac{1}{m}\sum_{i=1}^{m}\boxed{\sum_{j=1}^{4}}\left(y_j^{(i)}\log\left(\hat{y}_j^{(i)}\right)+\left(1-y_j^{(i)}\right)\log\left(1-\hat{y}_j^{(i)}\right)\right)$$

多任务学习模型的成本函数为：

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{c}L(\hat{y}_j^{(i)},y_j^{(i)}) \tag{4}$$

其中，j 代表任务下标，总有 c 个任务。对应的损失函数为：

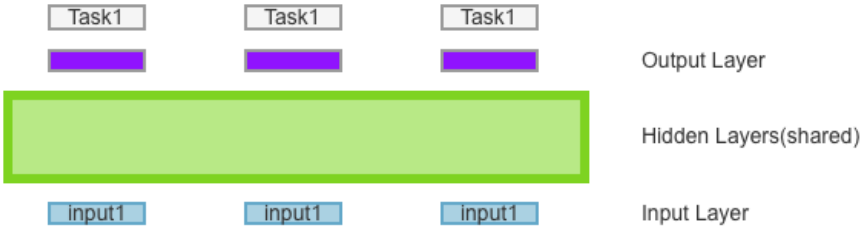
$$L(\hat{y}_j^{(i)},y_j^{(i)})=-y_j^{(i)}log\hat{y}_j^{(i)}-(1-y_j^{(i)})log(1-\hat{y}_j^{(i)}) \tag{5}$$

多任务学习是使用单个神经网络模型来实现多个任务。实际上，也可以分别构建多个神经网络来实现。多任务学习中可能存在训练样本 Y 某些标签空白的情况，这不会影响多任务学习模型的训练。

多任务学习和 Softmax 回归看上去有些类似，容易混淆。它们的区别是，Softmax 回归的输出向量 y 中只有一个元素为 1；而多任务学习的输出向量 y 中可以有多个元素为 1。

在下述场合进行多任务学习是有意义的：

- 训练的一组任务可以共用低层次特征；
- 通常，每个任务的数据量接近；
- 能够训练一个足够大的神经网络，以同时做好所有的工作。多任务学习会降低性能的唯一情况（即和为每个任务训练单个神经网络相比性能更低的情况）是神经网络还不够大。



在多任务深度网络中，低层次信息的共享有助于减少计算量，同时共享表示层可以使得几个有共性的任务更好的结合相关性信息，任务特定层则可以单独建模任务特定的信息，实现共享信息和任务特定信息的统一。

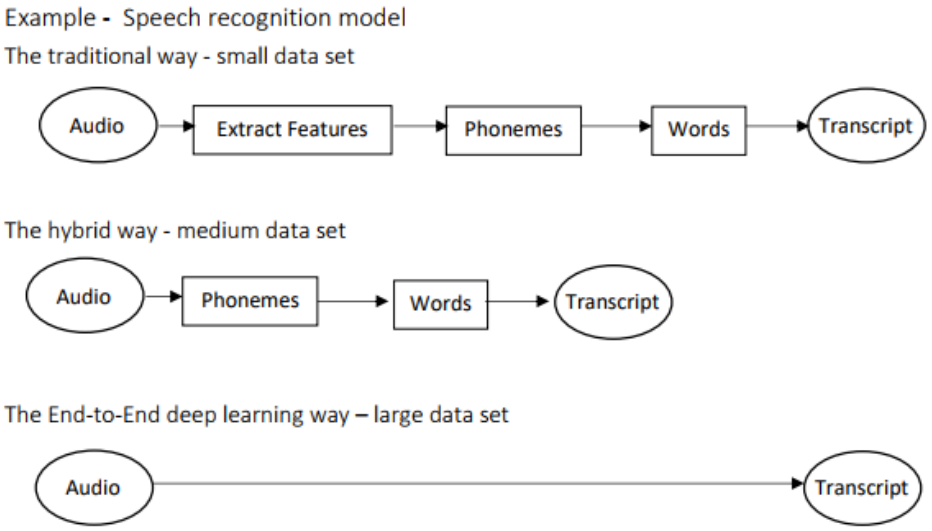
在实践中，多任务学习的使用频率要远低于迁移学习。计算机视觉领域中的物体识别是一个多任务学习的例子。

1.6.3 端对端学习

在传统的机器学习分块模型中，每一个模块处理一种输入，然后其输出作为下一个模块的输入，构成一条流水线。而**端到端深度学习（End-to-end Deep Learning）**只用一个单一的神经网络模型来实现所有的功能。它将所有模块混合在一起，只关心输入和输出。

如果数据量较少，传统机器学习分块模型所构成的流水线效果会很不错。但如果训练样本足够大，并且训练出的神经网络模型足够复杂，那么端到端深度学习模型的性能会比传统机器学习分块模型更好。

而如果数据集规模适中，还是可以使用流水线方法，但是可以混合端到端深度学习，通过神经网络绕过某些模块，直接输出某些特征。



优点与缺点

优点：

- 只要有足够多的数据，剩下的全部交给一个足够大的神经网络。比起传统的机器学习分块模型，可能更能捕获数据中的任何统计信息，而不需要用人类固有的认知（或者说，成见）来进行分析；
- 所需手工设计的组件更少，简化设计工作流程；

缺点：

- 需要大量的数据；
- 排除了可能有用的人工设计组件；

根据以上分析，决定一个问题是否应用端到端学习的关键点是：是否有**足够的数据**，支持能够直接学习从 x 映射到 y 并且足够复杂的函数？