

3. 结构化机器学习项目

深度学习

1. 机器学习项目策略(1)

1.1 正交化Orthogonalization

正交化是指在机器学习系统的改进过程中，每次调整只会影响模型某一方面的性能，同时保证对其他功能没有影响。具体而言监督学习中的共有四个目标，我们希望他们之间都是正交的，改进一个不会影响另一个：

- 模型在训练集上表现良好
- 模型在验证集上表现良好
- 模型在测试集上表现良好
- 模型在实际应用时表现良好

1.2 评估指标

- **准确率(Accuracy)**

TP(True Positive)——将正类预测为正类数、FN(False Negative)——将正类预测为负类数、FP(False Positive)——将负类预测为正类数、TN(True Negative)——将负类预测为负类数

- **精确率(Precision)** = $\frac{TP}{TP+FP}$

- **召回率(Recall)** = $\frac{TP}{TP+FN}$

- **F1** = $\frac{2}{1/Precision+1/Recall} = \frac{2TP}{2TP+FP+FN}$

进行模型评估时尽量只使用一个指标。如果我们同时使用精确率和召回率评估模型，那么当A模型精确率优于B模型，但召回率低于B模型时，就很难比较模型的优劣。因此应该尽量选择一个综合性的指标，例如F1 score等。

最优化指标和满足指标

实际应用中，项目所要求的评估指标都由最优指标和满足指标构成。例如，项目要求在召回率 > 70% 的条件下，使准确率尽量高。其中准确率为最优化指标，召回率为满足指标。(类似于优化问题中的目标函数和限制条件)

1.3 训练、验证和测试集的划分

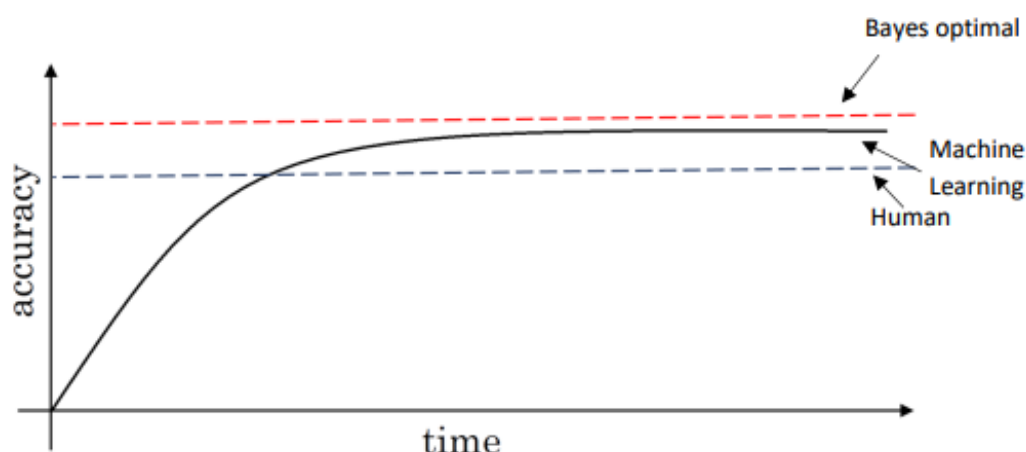
- 各数据集的数据必须来源于同一分布，否则会使模型不准
- 数据划分：常用方法为60%/20%/20%；但当数据量很大时，可以考虑缩小验证集和测试集比例，比如98%/1%/1%。只要满足验证集和测试集规模足够大，可以准确地测试模型的性能就可以

1.4 损失函数的改进

在实际应用中，模型产生的不同类型的错误对业务带来的影响是不一样的。为了使模型不出现我们不希望看到的错误类型(图像识别中漏掉色情图片、信贷中漏掉违约用户等)，可以对损失函数中的每个样本的损失赋予不同权重 $J = \frac{1}{\sum w} \sum_{i=1}^m w^{(i)} L(\hat{y}_{(i)}, y_{(i)})$

1.5 与人类表现水平进行比较

机器学习系统一般在项目初期进步得很快，但当接近或超过人类表现的时候，就几乎要到达理论的模型上限了，之后进展会很慢。理论上的最高水平也称为贝叶斯误差。



建立机器学习系统时，人的表现是一个很重要的参照物。例如，如果当前训练集错误率是8%、验证集是10%，人类的表现是1%，那么可知目前模型的偏差还是很大的，主要精力应放在减小偏差上；而当人类的错误率是7.5%，那么模型已经很接近上限了，应该通过正则化、加入更多数据等方法缩小方差。

在模型表现超过人类之前，模型的改进方向还是比较好判断的，因为我们可以将人类表现视为上限。但当模型表现超过人类之后，进一步改进就很难了，因为我们不知道实际的贝叶斯误差是多少。

在许多应用功能，机器学习的表现已经超越人类了，比如：在线广告、推荐系统、物流、信贷预测等。

1.6 提升模型表现

- 避免偏差：训练结构更复杂的神经网络、使用更先进的优化算法
- 减小方差：加入更多数据、正则化、简化神经网络结构

2. 机器学习项目策略(2)

2.1 误差分析

误差分析是指人工检查算法所犯的的错误，来找到进一步修正算法的方向。例如，我们发现识别猫的分类器将一部分狗识别为猫，一种错误分析的办法就是从开发集中随机抽出100个样本，查看将狗识别为猫的比例有多大。如果我们发现该类错误的概率在50%，显然这种错误是值得系统性地修正的；如果仅有1%，则未必需要对此付出很多精力。

2.2 标注错误

神经网络对随机出现的标注错误有鲁棒性，但无法解决系统性的标注错误。对于标注错误问题，也可以使用误差分析方法，通过抽样查看标注错误的比例是多少、错误是否系统等。

Error analysis

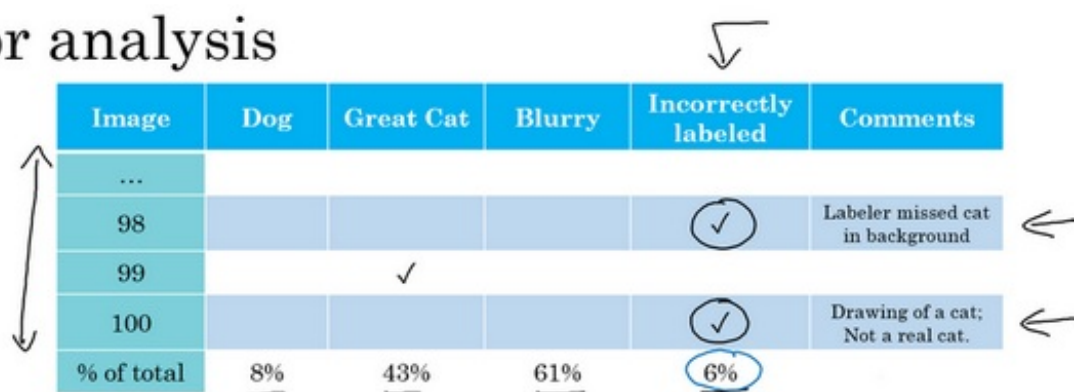


Image	Dog	Great Cat	Blurry	Incorrectly labeled	Comments
...					
98				✓	Labeler missed cat in background
99		✓			
100				✓	Drawing of a cat; Not a real cat.
% of total	8%	43%	61%	6%	

2.3 搭建机器学习系统的基本策略

- 设置好训练、验证、测试集，确定评估指标
- 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

2.4 训练与测试数据分布不同

数据分布不同问题

一般来说，可以接受训练和开发/测试集数据分布略有区别，但开发和测试集分布必须一样。实际应用中，训练数据和开发/测试数据很可能是不一样的。例如，我们的训练数据都来自网络，图片清晰；测试数据来自用户拍摄并上传，因此质量相对较低。假设我们有大量的网络图片和少量用户图片(与实际应用场景的输入数据一致)，合理的方案是由所有网络图片和部分用户图片构成训练集，剩余用户图片构成开发集和测试集。

偏差-方差分析

训练和测试分布不够会造成一个问题：假设开发集和训练集结果差距很大，我们很难知道这种差距来源于过拟合还是数据分布差别。解决办法是从训练集中随机取出一部分样本作为训练-开发集，它的实际作用也是开发集，用来检查模型表现。

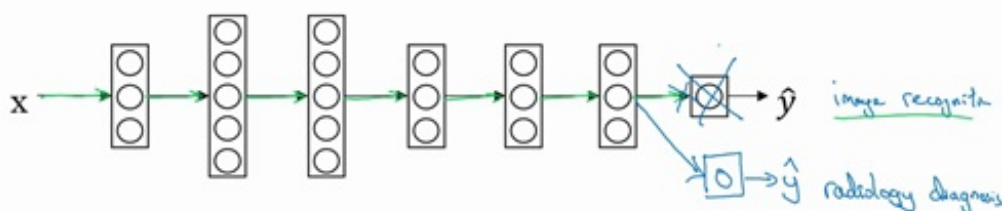
我们实际评估了训练集、训练-开发集、开发集、测试集的表现。训练集和训练-开发集的差距代表了模型的方差；训练-开发集和开发集的差距代表数据的分布差别；开发集和测试集的表现理论上来说应该是相同的。

2.5 迁移学习

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，也就是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。

例如，下面的网络原本是用于训练猫的识别器，我们希望将它迁移到放射性治疗的图像识别的任务中。我们保留所有其他节点的权重，仅将最后一个节点替换并初始化，利用新的放射性治疗数据来训练新模型。

Transfer learning



当数据量很少时，我们可以仅重新训练新替换的节点单元；当数据量足够时，也可以同时更新之前网络的权重。如果利用新数据更新所有节点的权重，那么之前用其他数据集训练的过程可以称为预训练，而用新数据更新权重的过程叫作微调。

迁移学习之所以能够奏效，是因为很多相似任务的低级特征都是有共性的。例如猫的认识和放射性诊断问题，前面几层网络可能都主要用于识别边缘等，因此可以互通。

应用所需条件：

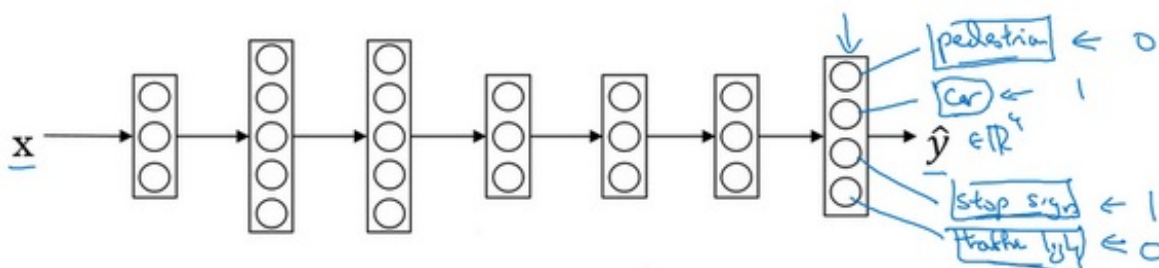
- 两个任务有同样的输入和输出结构(例如，猫的认识和放射性诊断都是图像二分类问题)
- 拥有更多数据的任务A所训练出的模型迁移到数据较少的任务B
- 任务A的低层次特征对另一个任务的学习有帮助

2.6 多任务学习

多任务学习是利用一个网络结构同时学习多个目标。例如，下图网络是应用在自动驾驶中的案例，输出层每个节点分别代表行人、车辆、停止标志、红绿灯，使用同一个网络可以同时学习如下四个内容。

损失函数为 $\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^C L(\hat{y}_j^{(i)}, y_j^{(i)})$

Neural network architecture



注意多任务学习的输出结果和softmax是有区别的，softmax只允许一个节点为1，其他必须为0；而多任务学习没有这个限制，每个节点是0或是1都是独立的。

应用场景：

- 多个任务有同样的输入和输出(所有目标都是二分类识别问题)

- 多个任务共享同样的低级特征
- 各个任务数据量十分相似(可以有部分图片缺少某个类别的标签，但每个任务的有标签样本个数差距不要过大)

2.7 端到端学习

传统的学习过程需要将输入的数据形式进行多次转换再进行训练，也可能需要分为多个阶段，建立多个模型。其中每一步转化过程都加入了许多人为因素，如特征提取等。端到端学习是希望数据能够直接由输入到输出。

例如，在人脸门禁识别任务中，一般将其分为几个步骤：1) 利用目标检测找到人脸；2) 将识别出的人脸图片处理成统一形式(提高接下来的匹配任务的成功率)；3) 进行匹配，判断目标是否在允许通过的群体中。

传统方法会严格执行这些步骤，同时每个阶段都可能要进行数据处理、特征提取等工作；而端到端学习是希望能够只训练一个模型，输入人在摄像头前的照片，然后直接得出结果。

端到端学习的优点是减少了人工干预，因此也可以避开人类一些固化的思维框架；缺点则是只有在数据量特别大时才能发挥作用，鉴于目前能收集到的数据和计算能力有限，在许多任务中很难应用。