机器学习和人工智能
@图灵的猫

机器学习简析

什么是机器学习

- 简化问题
  - 假设空间、特征、数据集、算法...
  - 归纳偏好
- 发展历程
- 实际应用

模型评估与选择

- 经验误差与过拟合
  - 实际预测输出与样本的真实输出之间的差异
  - 通过为有别于必须学习的"新样本"训练样本来解决
- 评估方法
  - 留出法
    - 交叉验证法(留一法)
      - 5-7次人相机的五千张
      - k-10折法,得比k相同稳
      - k次10折测试求均值
    - 自助法
      - 调参与最终模型
      - 引入验证集
- 性能度量
  - 错误率与精度
    - 分类错误的样本数占样本总数的比例
    - 精度=1-错误率
  - 查准率/查全率/F1
    - 对二分类问题,可将样例根据其真实类别与学习器预测类别的组合划分为真正例、假正例、真反例、假反例四种情形
    - 查准率P=真正例/(真正例+假正例)
    - 查全率R=真正例/(真正例+假反例)
    - 调和平均F1=2×P×R/(P+R)=样例总数×真正例-真正例×预测正例/样例总数×查准率×查全率
    - 宏查准率macro-P=各个查准率的平均
    - 宏查全率macro-R=各个查全率的平均
    - 宏F1=2×macro-P×macro-R/(macro-P+macro-R)
    - 微查准率micro-P=真正例在预测正例中的比例
    - 微查全率micro-R=真正例在真正例+假反例中的比例
    - 微F1=2×micro-P×micro-R/(micro-P+micro-R)
  - ROC与AUC
    - 对学习器的泛化性能进行评估,不仅需有效可行的实验估计方法,还需有衡量模型泛化能力的评价标准,这就是性能度量
    - 反映的是学习器把样例排序的好坏
    - ROC曲线纵轴是"真正例率"(TPR),横轴是"假正例率"(FPR)
    - 根据学习器的预测结果对样例进行排序,按此顺序逐个把样例作为正例进行预测,每次计算出两个值,分别以它们为横、纵坐标作图,就得到了ROC曲线
    - 若一个学习器的ROC曲线被另一个学习器的曲线完全"包住",则可断言后者的性能优于前者
    - 若两个学习器的ROC曲线发生交叉,则难以一般性地断言两者孰优孰劣,只能在具体的查准率或查全率条件下进行比较
    - 然而在很多情形下,人们往往仍希望把学习器A与B比个高低,这时一个比较合理的判据是比较ROC曲线下的面积,即AUC(Area Under ROC Curve)
    - AUC考虑的是样本预测的排序质量,因此它与排序误差有紧密联系
    - 给定m+个正例和m-个反例,令D+和D-分别表示正、反例集合,则排序"损失"(loss)定义为
    - 上式对应的是ROC曲线之上的面积:若一个正例在ROC曲线上对应标记点的坐标为(x,y),则x恰是排序在其之后的反例所占的比例,即假正例率。因此有
    - AUC=1-ℓrank
  - 代价敏感错误率与代价曲线
    - 为权衡不同类型错误所造成的不同损失,可为错误赋予"非均等代价"(unequal cost)
    - 在非均等代价下,ROC曲线不能直接反映学习器的期望总体代价,而"代价曲线"(cost curve)则可达到该目的
    - 代价曲线图的横轴是取值在[0,1]的正例概率代价,其中p是样例为正例的概率
    - 纵轴是取值为[0,1]的归一化代价
    - 归一代价=(假反例率×正例概率代价+假正例率×(1-正例概率代价))/(正例概率代价+(1-正例概率代价))
    - 绘制代价曲线:ROC曲线上每一点对应了代价平面上一条线段,设ROC曲线上点的坐标为(TPR,FPR),则可相应计算出FNR,然后在代价平面上绘制一条从(0,FPR)到(1,FNR)的线段,线段下面的面积即表示了该条件下的期望总体代价。如此将ROC曲线上的每个点转化为代价平面上的一条线段,所有线段的下界围成的面积,即为在所有条件下学习器的期望总体代价
- 偏差与方差
  - 偏差-方差分解
    - 偏差度量了学习算法的期望预测与真实结果的偏离程度,即刻画了学习算法本身的拟合能力
    - 方差度量了同样大小的训练集的变动所导致的学习性能的变化,即刻画了数据扰动所造成的影响
    - 噪声则表达了在当前任务上任何学习算法所能达到的期望泛化误差的下界,即刻画了学习问题本身的难度
    - 泛化误差可分解为偏差、方差与噪声之和

线性模型

- 线性回归
  - 简单线性回归
    - 均方误差
    - 最小二乘法
    - 对多元线性回归,最小二乘法试图找到一个线性模型,使得尽可能地接近真实标记,即
    - 令E(w,b)=∑(yi-wxi-b)²,分别对w和b求导,得到E(w,b)最小化的闭式解
  - 多元线性回归
    - 广义线性模型
    - 对数线性回归
    - 对数几率回归
    - 线性判别分析(LDA)
    - 多分类学习
    - 类别不平衡问题
- 决策树
  - 基本流程
    - 一般一棵决策树包含一个根结点、若干个内部结点和若干个叶结点
    - 叶结点对应于决策结果,其他每个结点则对应于一个属性测试
    - 决策树的生成是一个递归过程
    - 在决策树基本算法中,有三种情形会导致递归返回:(1)当前结点包含的样本全属于同一个类别,无需划分;(2)当前属性集为空,或是所有样本在所有属性上取值相同,无法划分;(3)当前结点包含的样本集合为空,不能划分
  - 划分选择
    - 信息增益(ID3)
      - 信息熵(information entropy)是度量样本集合纯度最常用的一种指标
      - 假定当前样本集合D中第k类样本所占的比例为pk(k=1,2,...,|y|),则D的信息熵定义为
      - Ent(D)的值越小,则D的纯度越高
      - 信息增益越大,则意味着用属性a来进行划分所获得的"纯度提升"越大
      - 一般而言,信息增益准则对可取值数目较多的属性有所偏好
    - 增益率(C4.5)
      - 增益率准则对可取值数目较多的属性有所偏好,为减少这种偏好可能带来的不利影响,C4.5算法并不直接选择增益率最大的候选划分属性,而是使用了一个启发式:先从候选划分属性中找出信息增益高于平均水平的属性,再从中选择增益率最高的
      - 其中,属性a的固有值(intrinsic value)
      - 属性a的可能取值数目越多(即V越大),则IV(a)的值通常就越大
      - 增益率准则对可取值数目较少的属性有所偏好
    - 基尼指数(Gini)
      - CART决策树使用"基尼指数"来选择划分属性
      - 数据集D的纯度可用基尼值来度量
      - 直观来说,Gini(D)反映了从数据集D中随机抽取两个样例,其类别标记不一致的概率。因此,Gini(D)越小,则数据集D的纯度越高
      - 属性a的基尼指数定义为
      - 于是,我们在候选属性集合A中,选择那个使得划分后基尼指数最小的属性作为最优划分属性,即a*=arg min Gini_index(D,a)
- 剪枝处理
  - 为了尽可能正确分类训练样本,有时会造成决策树分支过多,这时就可能因训练样本学得"太好"了,以致于把训练集自身的一些特点当作所有数据都具有的一般性质而导致过拟合。因此,可通过主动去掉一些分支来降低过拟合的风险
  - 预剪枝(pre-pruning)
    - 预剪枝是指在决策树生成过程中,对每个结点在划分前先进行估计,若当前结点的划分不能带来决策树泛化性能提升,则停止划分并将当前结点标记为叶结点
    - 后剪枝是先从训练集生成一棵完整的决策树,然后自底向上地对非叶结点进行考察,若将该结点对应的子树替换为叶结点能带来决策树泛化性能提升,则将该子树替换为叶结点
    - 后剪枝决策树通常比预剪枝决策树保留了更多的分支。一般情形下,后剪枝决策树的欠拟合风险很小,泛化性能往往优于预剪枝决策树,但后剪枝过程是在生成完全决策树之后进行的,并且要自底向上地对树中的所有非叶结点进行逐一考察,因此其训练时间开销比未剪枝决策树和预剪枝决策树都要大得多
  - 决策树的优缺点
    - 优点:
      - 计算简单,易于理解,可解释性强
      - 比较适合处理有缺失属性的样本
      - 能够同时处理不相关的特征
      - 在相对短的时间内能够对大型数据源做出可行且效果良好的结果
      - 对缺失值不敏感
      - 可以处理不相关特征数据
      - 效率高,决策树只需要一次构建,反复使用,每一次预测的最大计算次数不超过决策树的深度
    - 缺点:
      - 多类别时,计算量比较大
      - 决策树容易过拟合,导致泛化能力不强
      - 可能会产生过于复杂的子树,对数据的扰动非常敏感,即数据没有科学的小变动就会绘制出完全不同的决策树
      - 决策树模型是基于"规则"的,有些类型的问题决策树很难学习,因为它很难简洁的表达出来,比如XOR问题
      - 寻找最优的决策树是一个NP完全问题,现实中常采用启发式方法学习简单决策树
      - 决策树模型学习的几何含义是在特征空间中寻找轴平行的决策边界,在处理斜的决策边界时会导致模型很复杂
      - 数值型特征两端样本近邻判别出现分类面急剧变化,预测不稳定,参见PLA的图形表示
      - 通常的决策树学习算法对样本数据的学习是"贪心"的,即关注使当前划分取得最大的信息增益,不能进行回溯调整,可能陷入局部最优而非全局最优,可以通过多变量决策树一定程度上缓解这个问题
      - 决策树难以学习到异或(XOR)、奇偶(parity)等简单关系
      - 若某些类别的样本比例过大,学习出来的决策树将会偏向这些类别。在学习前需要对样本进行平衡处理,常见的做法有样本均衡采样和权重均衡
      - 对连续值特征进行决策时,将其视为单个结点上的连续取值处理,未能充分挖掘这种特征的潜在信息
      - 对训练样本的依赖较大,若训练样本学得树已经完美匹配了训练样本,但对未见样本却可能表现不佳,也就是发生了过拟合

多分类学习

- 一般来说,对于多分类问题可以分解为多个二分类任务求解,分解的关键是对多分类任务进行拆分,并为每个二分类任务训练一个分类器。拆分策略有三种基本类型:一对一(OvO)、一对其余(OvR)和多对多(MvM)
- 一对一(OvO)
  - 给定数据集D={(x1,y1),(x2,y2),...,(xm,ym)},yi∈{C1,C2,...,CN}
  - OvO将这N个类别两两配对,从而产生N(N-1)/2个二分类任务,例如OvO将为区分类别Ci和Cj训练一个分类器,该分类器把D中的Ci类样例作为正例,Cj类样例作为反例,用该分类器对新样本x进行预测时,得到的结果是Ci或Cj
  - 在测试阶段,新样本将同时提交给所有分类器,于是我们将得到N(N-1)/2个分类结果,最终结果可通过投票产生:即把被预测得最多的类别作为最终分类结果
- 一对其余(OvR)
  - OvR则是每次将一个类的样例作为正例、所有其他类的样例作为反例来训练N个分类器,在测试时若仅有一个分类器预测为正类,则对应的类别标记作为最终分类结果,若有多个分类器预测为正类,则通常考虑各分类器的预测置信度,选择置信度最大的类别标记作为分类结果
- 多对多(MvM)
  - MvM是每次将若干个类作为正类,若干个其他类作为反类。显然,OvO和OvR是MvM的特例。MvM的正、反类构造必须有特殊的设计,不能随意选取,否则就会被惩罚,从而影响性能。
  - 纠错输出码(ECOC)
    - 编码:对N个类别做M次划分,每次划分将一部分类别划为正类,一部分划为反类,从而形成一个二分类训练集;这样一共产生M个训练集,可训练出M个分类器。
    - 解码:M个分类器分别对测试样本进行预测,这些预测标记组成一个编码。将这个预测编码与每个类别各自的编码进行比较,返回其中距离最小的类别作为最终预测结果。
    - 类别划分通过"编码矩阵"(coding matrix)指导,编码矩阵有多种形成方式,阐述二元码和多元码两种常见方法
    - 二元码:将每个类分别指定为正类和反类,也就划分出两个二类训练集,在测试时这两个训练集产生两个预测结果,将这两个预测结果组合起来就得到关于各类别的一个编码。
    - 多元码:若希望某些类别在学习过程中不被考虑,则可以通过三元码来实现。
    - 一般来说,对同一个学习任务,ECOC编码越长,纠错能力越强。然而,编码越长,意味着所需训练M个分类器的计算、存储开销都越大,测试时间开销也越大。
    - 对于等长编码,理论上来说,在编码长度相同的情况下,我们希望任意两个类别之间的编码距离尽可能远些,这样才能使纠错能力最强。然而,在有限长度内寻找理论最优的编码是一个NP难问题,难以有效求解。不过,通常情况下,对较小规模的类别数目还是有可能求得最优解的,或者可以借助优化算法进行求解,即使得到的解不是最优的,但往往也能产生较好的纠错效果。
    - 对同等长度的编码而言,理论上来说,类别数越多则纠错能力越强,因为这时可以用更多的分类器来进行纠错。然而,实际上我们主要关注的是分类器错误相关联的情形,即多个分类器同时犯错的情形,这样才不能有效地纠错。显然,对于有限类别数,可能的组合数目是有限的,甚至随着类别数的增加,好的组合反而减少了,因此,在实际应用中并非类别数越多越好。
    - 对同等长度的编码而言,理论上来说,非均等划分类别产生的分类器所产生的错误有时会相关联,即多个分类器同时犯错的情形,这样才不能有效地纠错。显然,为了减少这种情形的出现,最好是使每次划分时正、反类别所包含的原始类别数目相同,即"均等划分"(equal bi-partition)。
    - 对同等长度的编码而言,理论上来说,非均等划分类别产生的分类器所产生的错误有时会相关联,即多个分类器同时犯错的情形,这样才不能有效地纠错。显然,为了减少这种情形的出现,最好是使每次划分时正、反类别所包含的原始类别数目相同,即"均等划分"(equal bi-partition)。

类别不平衡问题

- 欠采样
  - 直接对训练集里的反例进行"采样",即去除一些反例使得正、反例数目接近,然后再进行学习
  - 欠采样法的时间开销通常小于过采样法,因为欠采样后的训练集比原始训练集小,甚至小于原始训练集;但欠采样法若随机丢弃反例,则可能丢失一些重要信息。
  - 欠采样法的代表性算法EasyEnsemble则利用集成学习机制,将反例划分为若干个集合供不同学习器使用,这样对每个学习器来看都进行了欠采样,但在全局来看却不会丢失重要信息。
- 过采样
  - 对训练集里的正例进行"过采样",即增加一些正例使得正、反例数目接近,然后再进行学习
  - 过采样法不能简单地对初始正例样本进行重复采样,否则会招致严重的过拟合;过采样法的代表性算法SMOTE是通过对训练集里的正例进行插值来产生额外的正例。
  - 对于每个正例样本,SMOTE从其近邻中随机选取一个样本,然后在两个样本之间的连线上随机选取一个点作为新合成的正例。
- 阈值移动
  - 阈值移动(threshold-moving)则是直接基于原始训练集进行学习,但在用训练好的分类器进行预测时,将式(3.2)嵌入到其决策过程中。
  - 再缩放(rescaling)也是代价敏感学习的基础。

其他常见神经网络

- 神经元模型
  - 阈值
  - 激活函数
    - 理想的阶跃函数:将输入值映射为输出值"0"或"1",显然"1"对应于神经元兴奋,"0"对应于神经元抑制。然而阶跃函数具有不连续、不光滑等不好的性质,因此实际常用Sigmoid函数作为激活函数。典型的Sigmoid函数如对率函数(logistic function)
    - 对率函数可以将较大范围内变化的输入值挤压到(0,1)输出值范围内,因此有时也称为"挤压函数"(squashing function)。
- 感知机和多层网络
  - 感知机由两层神经元组成,输入层接收外界输入信号后传递给输出层,输出层是M-P神经元,亦称"阈值逻辑单元"(threshold logic unit)。
  - 多层前馈神经网络(multi-layer feed-forward neural networks)
    - 每层神经元与下一层神经元全互连,神经元之间不存在同层连接,也不存在跨层连接。这样的神经网络结构通常称为"多层网络"。
    - 隐层和输出层神经元都是具有激活函数的功能神经元。
    - 只需包含隐层,即可称为多层网络。
    - 前馈指的是网络拓扑结构上不存在环或回路;而下面将要讨论的递归神经网络则允许环或回路存在。
    - 神经网络的学习过程,就是根据训练数据来调整神经元之间的"连接权"(connection weight)以及每个功能神经元的"阈值";换言之,神经网络"学"到的东西,蕴涵在连接权与阈值中。
- 误差逆传播算法(BP)
  - 给定训练集D={(x1,y1),(x2,y2),...,(xm,ym)},xi∈Rd,yi∈Rl,即输入示例由d个属性描述,输出l维实值向量。为便于讨论,假定隐层和输出层都使用Sigmoid函数。
  - 对训练例(xk,yk),假定神经网络的输出为ŷk=(ŷ1k,ŷ2k,...,ŷlk),即
  - 则网络在(xk,yk)上的均方误差为
  - 网络中有(d+l+1)q+l个参数需确定:输入层到隐层的d×q个权值、隐层到输出层的q×l个权值、q个隐层神经元的阈值、l个输出层神经元的阈值。BP是一个迭代学习算法,在迭代的每一轮中采用广义的感知机学习规则对参数进行更新估计,任意参数v的更新估计式为
  - 学习率η∈(0,1)控制着算法每一轮迭代中的更新步长,若太大则容易振荡,太小则收敛速度又会过慢。有时为了做精细调节,可令式(5.12)使用一个学习率,式(5.13)使用另一个学习率。
  - 标准BP算法每次仅针对一个训练样例更新连接权和阈值,也就是说,算法的更新规则是基于单个的Ek推导而得。如果类似地推导出基于累积误差最小化的更新规则,就得到了累积误差逆传播(accumulated error backpropagation)算法。
  - 累积BP算法与标准BP算法都很常用。一般来说,标准BP算法每次更新只针对单个样例,参数更新得非常频繁,而且对不同样例进行更新的效果可能出现"抵消"现象。因此,为了达到同样的累积误差极小点,标准BP算法往往需进行更多次数的迭代。累积BP算法直接针对累积误差最小化,它在读取整个训练集D一遍后才对参数进行更新,其参数更新的频率低得多。但在很多任务中,累积误差下降到一定程度后,进一步下降会非常缓慢,这时标准BP往往会更快获得较好的解,尤其当训练集D非常大时更明显。
  - 多隐层网络能够更好地刻画现实世界中一些具有层次结构的状况。每个隐层将上一层隐层的输出信号变换后输出,最终在输出层获得预测结果。
  - 由于多隐层网络往往具有比单隐层网络更好的表现和更大的潜力,于是,研究者们多年来一直致力于多隐层网络的学习算法研究,并形成了深度学习(deep learning)这个热门领域。
  - 从增加模型复杂度的角度来看,增加隐层的数目显然比增加隐层神经元的数目更有效,不仅因为隐层数增多后相当于增加了神经元的数量,还因为增加了函数的复合层数。
  - 从模型容量的角度来看,单隐层多层感知机模型已具有强大的学习能力,例如,它能以任意精度近似任意复杂度的连续函数。
  - 但是,多隐层网络能够更容易地刻画现实世界,这是因为多隐层网络具有强大的特征学习(feature learning)能力,经过逐层特征变换,将样本在原空间的表示变换到一个新特征空间,从而使分类或预测更容易。
  - 与人工设计的"特征工程"相比,特征学习减少了人类专家的工作量,好的特征甚至连专家也很难设计出来。
  - 无论是增加隐层数还是增加隐层神经元数,神经网络的表示能力都会得到显著提升,从而使得模型的训练误差持续降低。
  - 然而,这也给网络的训练带来了很大的困难,过拟合是其中最为棘手的问题。
  - 缓解深度神经网络过拟合的常用方法有两大类:一类是正则化(regularization)策略,另一类是通过增大训练集来降低过拟合风险。
  - 正则化策略:
    - 早停(early stopping)
    - 权重衰减(weight decay)
    - 丢弃法(dropout)
    - 数据增强(data augmentation)
    - 标签平滑(label smoothing)
  - 增大训练集:
    - 数据增强(data augmentation)
    - 对抗训练(adversarial training)
    - 预训练(pre-training)
    - 迁移学习(transfer learning)
    - 多任务学习(multi-task learning)
    - 半监督学习(semi-supervised learning)
    - 自监督学习(self-supervised learning)
    - 无监督学习(unsupervised learning)
    - 弱监督学习(weakly supervised learning)
    - 主动学习(active learning)
    - 人机协同(human-in-the-loop)
    - 强化学习(reinforcement learning)
    - 元学习(meta learning)
    - 终身学习(lifelong learning)
    - 增量学习(incremental learning)
    - 在线学习(online learning