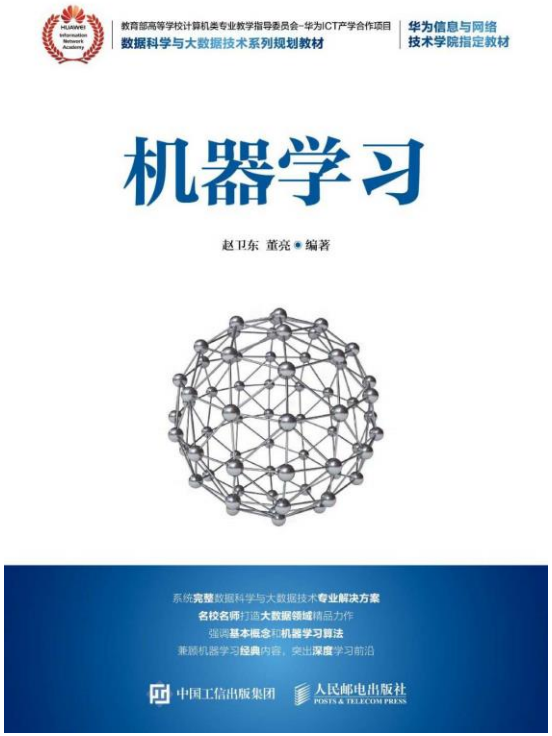


机器学习
决策树与分类算法

复旦大学 **赵卫东** 博士

wdzhao@fudan.edu.cn

章节介绍

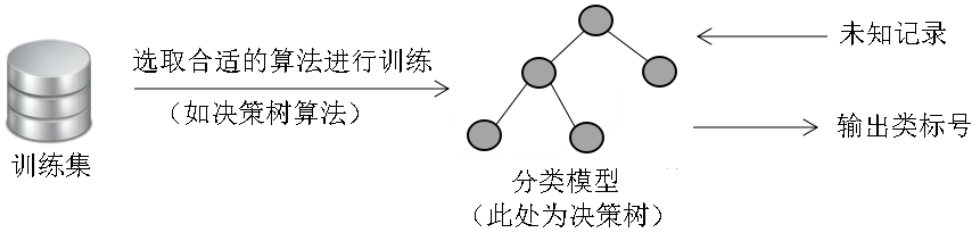
- 分类的任务是将样本(对象)划分到合适的预定义目标类中
- 本章主要介绍决策树算法，它是机器学习中的一个经典的监督式学习算法，被广泛应用F金融分析、生物学、天文学等多个领域
- 本章首先介绍决策树的1D3、C4.5、 C5. 0、CART 等常用算法，然后讨论决策树的集成学习，包括装袋法、提升法、随机森林、GBDT、 AdaBoost 等算法。最后介绍决策树算法的应用案例

章节结构

- 决策树算法
  - 分支处理
  - 连续属性离散化
  - 过拟合问题
  - 分类效果评价
- 集成学习
  - 装袋法
  - 提升法
  - GBGT
  - 随机森林
- 决策树应用

决策树算法

- 分类算法是利用训练样本集获得分类函数即分类模型(分类器)，从而实现将数据集中的样本划分到各个类中。分类模型通过学习训练样本中属性集与类别之间的潜在关系，并以此为依据对新样本属于哪一类进行预测

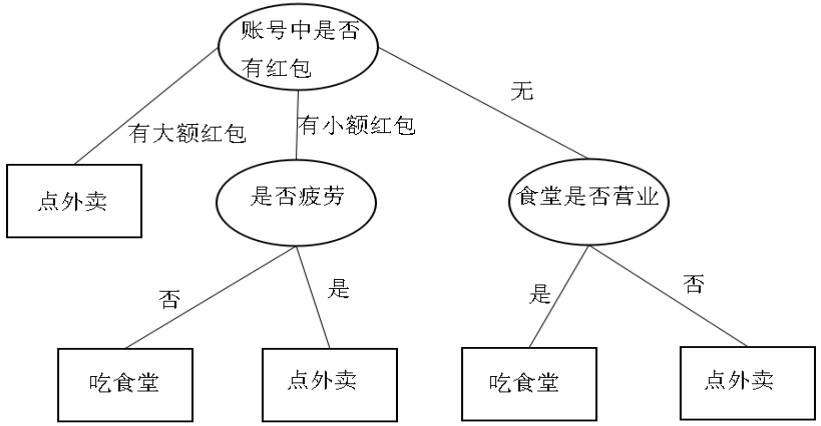


决策树算法

- 决策树通过把数据样本分配到某个叶子结点来确定数据集中样本所属的分类
- 决策树由决策结点、分支和叶子结点组成
  - 决策结点表示在样本的一个属性上进行的划分
  - 分支表示对于决策结点进行划分的输出
  - 叶结点代表经过分支到达的类。
- 从决策树根结点出发，自顶向下移动，在每个决策结点都会进行次划分，通过划分的结果将样本进行分类，导致不同的分支，最后到达个叶子结点，这个过程就是利用决策树进行分类的过程

决策树算法

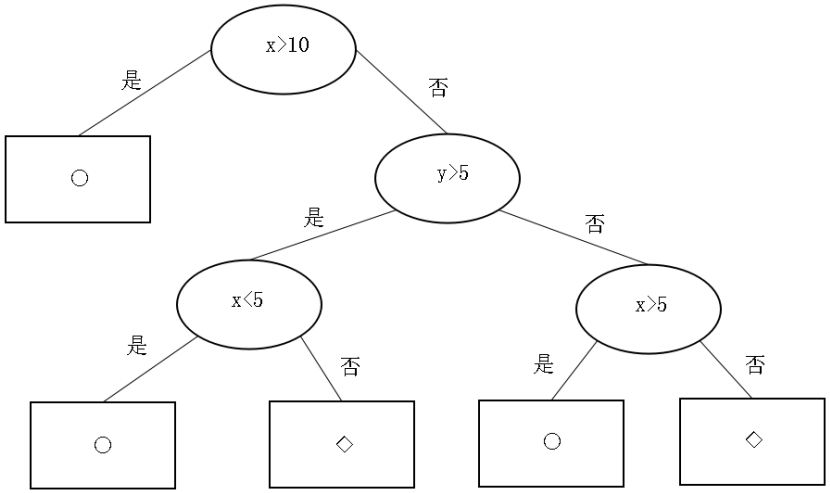
- 外卖订餐决策树



决策树算法

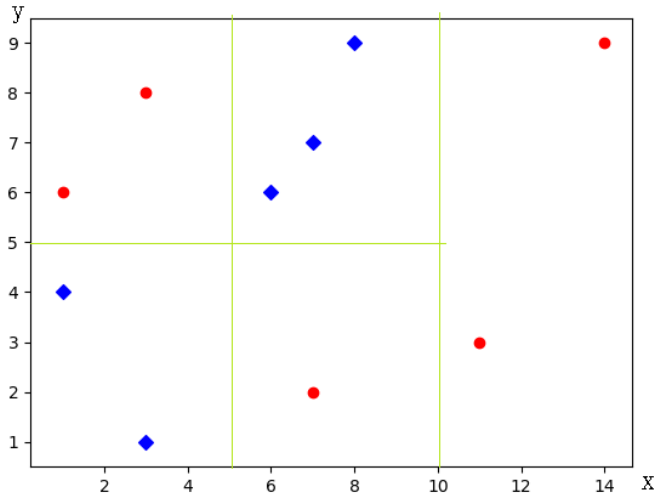
- 连续变量

坐标	(1,4)	(1,6)	(3,1)	(3,8)	(6,6)	(7,2)	(7,7)	(8,9)	(11,3)	(14,9)
分类	◇	○	◇	○	◇	○	◇	◇	○	○



决策树算法

- 决策树对应二维空间的分割结果



分支处理

- 往往使用启发式算法来进行决策树的构造，例如，使用贪婪算法对每个结点构造部分最优决策树
- 对于一个决策树的构建，最重要的部分就在于其分支处理，即确定在每个决策结点处的分支属性
- 分支属性的选取即对决策节点上选择哪一个属性来对数据集进行划分，要求每个分支中样本的类别纯度尽可能高，而且不要产生样本数量太少的分支

ID3 算法

- ID3算法是在每个结点处选取能获得最高信息增益的分支属性进行分裂
- 在每个决策结点处划分分支、选取分支属性的目的是将整个决策树的样本纯度提升
- 衡量样本集合纯度的指标则是熵

$$Entropy(S) = -\sum_{i=1}^{m} p_i log_2\ (p_i)\ , p_i = \frac{|C_i|}{n}$$

- 举例来说，如果有一个大小为10的布尔值样本集S_b，其中有6个真值、4个假值，那么该布尔型样本分类的熵为：

$$Entropy(S_b) = -\left(\frac{6}{10}\right)\log_2\frac{6}{10} - \left(\frac{4}{10}\right)log_2\left(\frac{4}{10}\right) = 0.9710$$

ID3 算法

- 计算分支属性对于样本集分类好坏程度的度量——信息增益
- 由于分裂后样本集的纯度提高，则样本集的熵降低，熵降低的值即为该分裂方法的信息增益

$$Gain(S,A) = Entropy(S) - \sum_{i=1}^{v} \frac{|S_i|}{|S|} Entropy(S_i)$$

ID3 算法

- 脊椎动物分类训练样本集

动物	饮食习性	胎生动物	水生动物	会飞	哺乳动物
人类	杂食动物	是	否	否	是
野猪	杂食动物	是	否	否	是
狮子	肉食动物	是	否	否	是
苍鹰	肉食动物	否	否	是	否
鳄鱼	肉食动物	否	是	否	否
巨蜥	肉食动物	否	否	否	否
蝙蝠	杂食动物	是	否	是	是
野牛	草食动物	是	否	否	是
麻雀	杂食动物	否	否	是	否
鲨鱼	肉食动物	否	是	否	否
海豚	肉食动物	是	是	否	是
鸭嘴兽	肉食动物	否	否	否	是
袋鼠	草食动物	是	否	否	是
蟒蛇	肉食动物	否	否	否	否

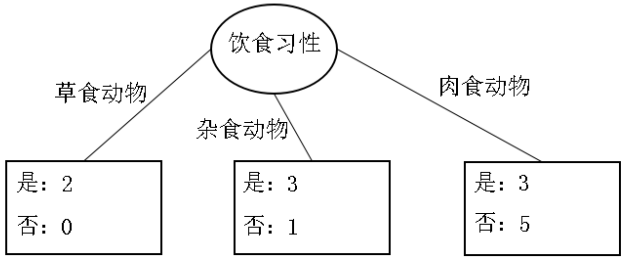
ID3 算法

- 此样本集有“饮食习性”、“胎生动物”、“水生动物”、“会飞”四个属性可作为分支属性，而“哺乳动物”作为样本的分类属性，有“是”与“否”两种分类，也即正例与负例。共有14个样本，其中8个正例，6个反例，设此样本集为 S，则分裂前的熵值为

$$Entropy(S) = -\left(\frac{8}{14}\right)\log_2\left(\frac{8}{14}\right) - \left(\frac{6}{14}\right)\log_2\left(\frac{6}{14}\right) = 0.9852$$

ID3 算法

- 脊椎动物训练样本集以“饮食习性”作为分支属性的分裂情况



- “饮食习性”为“肉食动物”的分支中有3个正例、5个反例，其熵值为：

$$Entropy(\text{肉食动物}) = -\left(\frac{3}{8}\right)\log_2\left(\frac{3}{8}\right) - \left(\frac{5}{8}\right)\log_2\left(\frac{5}{8}\right) = 0.9544$$

ID3 算法

- 同理，计算出“饮食习性”分类为“草食动物”的分支与分类为“杂食动物”的分支中的熵值分别为

$$Entropy(\text{草食动物}) = -\left(\frac{2}{2}\right)\log_2\left(\frac{2}{2}\right) - 0 = 0$$

$$Entropy(\text{杂食动物}) = -\left(\frac{3}{4}\right)\log_2\left(\frac{3}{4}\right) - \left(\frac{1}{4}\right)\log_2\left(\frac{1}{4}\right) = 0.8113$$

- 设“饮食习性”属性为Y，由此可以计算得出,作为分支属性进行分裂之后的信息增益为

$$Gain(Y) = Entropy(S) - Entropy(S|Y)$$
$$= 0.9852 - \frac{8}{14}\cdot 0.9554 - \frac{2}{14}\cdot 0 - \frac{4}{14}\cdot 0.8113 = 0.2080$$

ID3 算法

- 同理，可以算出针对其他属性作为分支属性时的信息增益
- 计算可得，以“胎生动物”“水生动物”“会飞”作为分支属性时的信息增益分别为0.6893、0.0454、0.0454
- 由此可知“胎生动物”作为分支属性时能获得最大的信息增益，即具有最强的区分样本的能力，所以在此处选择使用“胎生动物”作为分支属性对根结点进行划分

ID3 算法

- 由根结点通过计算信息增益选取合适的属性进行分裂，若新生成的结点的分类属性不唯一，则对新生成的结点继续进行分裂，不断重复此步骤，直至所有样本属于同 一类，或者达到要求的分类条件为止
- 常用的分类条件包括结点样本数最少于来设定的值、决策树达到预先设定的最大深度等
- 在决策树的构建过程中，会出现使用了所有的属性进行分支之后，类别不同的样本仍存在同一个叶子结点中。当达到了限制条件而被强制停止构建时，也会出现结点中子样本集存在多种分类的情况。对于这种情况，一般取此结点中子样本集占数的分类作为结点的分类
- 分支多的属性并不一定是最优的，就如同将100个样本分到99个分支中并没有什么意义，这种分支属性因为分支太多可能相比之下无法提供太多的可用信息，例如个人信息中的“省份”属性

C4.5算法

- C4.5算法总体思路与ID3类似，都是通过构造决策树进行分类，其区别在于分支的处理，在分支属性的选取上，ID3算法使用信息增益作为度量，而C4.5算法引入了信息增益率作为度量

$$\text{Gain\_ratio(A)} = \frac{Gain(A)}{-\sum_{i=1}^{v}\frac{|S_i|}{|S|}\log_2\frac{|S_i|}{|S|}}$$

- 由信息增益率公式中可见，当v比较大时，信息增益率会明显降低，从而在一定程度上能够解决ID3算法存在的往往选择取值较多的分支属性的问题
- 在前面例子中，假设选择“饮食习性”作为分支属性，其信息增益率为

$$\text{Gain\_ratio(饮食习性)} = \frac{Gain(\text{饮食习性})}{-\sum_{i=1}^{3}\frac{|S_i|}{|S|}\log_2\frac{|S_i|}{|S|}} = \frac{0.2080}{-\left(\frac{8}{14}\log_2\frac{8}{14}+\frac{2}{14}\log_2\frac{2}{14}+\frac{4}{14}\log_2\frac{4}{14}\right)} = 0.1509$$

C5.0算法

- C5.0算法是Quinlan在C4.5算法的基础上提出的商用改进版本，目的是对含有大量数据的数据集进行分析
- C5.0算法与C4.5算法相比有以下优势：
  - 决策树构建时间要比C4.5算法快上数倍，同时生成的决策树规模也更小，拥有更少的叶子结点数
  - 使用了提升法(boosting)，组合多个决策树来做出分类，使准确率大大提高
  - 提供可选项由使用者视情况决定，例如是否考虑样本的权重、样本错误分类成本等

CART算法

- CART算法采用的是一种二分循环分割的方法，每次都把当前样本集划分为两个子样本集，使生成的决策树的结点均有两个分支，显然，这样就构造了一个二叉树。如果分支属性有多于两个取值，在分裂时会对属性值进行组合，选择最佳的两个组合分支。假设某属性存在q个可能取值，那么以该属性作为分支属性，生成两个分支的分裂方法共有$2^{q-1}-1$种
- CART算法在分支处理中分支属性的度量指标是Gini指标

$$Gini(S)=1-\sum_{i=1}^{m}p_i^2, p_i=\frac{|C_i|}{|S|}$$

- 在前面例子中，假设选择“会飞”作为分支属性，其Gini指标为

$$Gini(\text{会飞})=\frac{|S_1|}{|S|}Gini(S_1)+\frac{|S_2|}{|S|}Gini(S_2)=\frac{11}{14}\times\left[1-\left(\frac{7}{11}\right)^2-\left(\frac{4}{11}\right)^2\right]+\frac{3}{14}\times\left[1-\left(\frac{1}{3}\right)^2-\left(\frac{2}{3}\right)^2\right]=0.4589$$

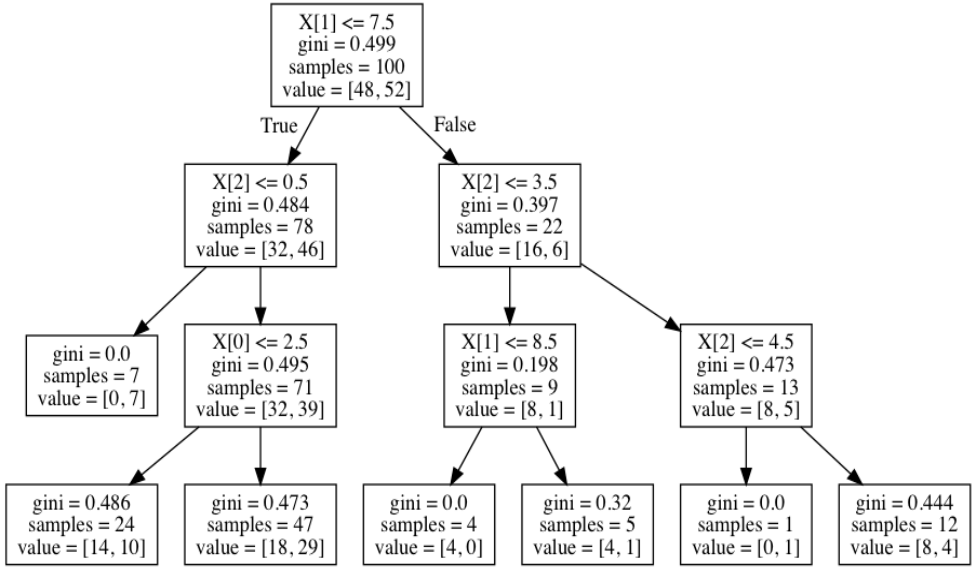
CART算法示例

- 以下是基于sklearn库的CART算法示例代码。通过构建决策树（采用Gini指标）对随机生成（通过np.random.randint方法）的数字进行分类，自变量X为100x4的矩阵，随机生成的数字大于10，因变量Y为大于2的100x1矩阵。树的最大深度限制为3层，训练完成之后将树可视化显示。

```
import numpy as np
import random
from sklearn import tree
from graphviz import Source
np.random.seed(42)
X=np.random.randint(10, size=(100, 4))
Y=np.random.randint(2, size=100)
a=np.column_stack((Y,X))
clf = tree.DecisionTreeClassifier(criterion='gini',max_depth=3)
clf = clf.fit(X, Y)
graph = Source(tree.export_graphviz(clf, out_file=None))
graph.format = 'png'
graph.render('cart_tree',view=True)
```

CART算法示例

- 生成的图片如下所示



连续属性离散化

- 分类数据有二元属性、标称属性等几种不同类型的离散属性
- 二元属性只有两个可能值，如“是”或“否”“对“或“错”，在分裂时，可以产生两个分支。对于二元属性，无须对其数据进行特别的处理
- 标称属性存在多个可能值，针对所使用的决策树算法的不同，标称属性的分裂存在两种方式：多路划分和二元划分
  - 对于ID3、C4.5等算法，均采取多分支划分的方法，标称属性有多少种可能的取值，就设计多少个分支
  - CART算法采用二分递归分割的方法，因此该算法生成的决策树均为二叉树
- 标称属性中有类特别的属性为序数属性， 其属性的取值是有先后顺序的。对于序数属性的分类，往往要结合实际情况来考虑

连续属性离散化

- 非监督离散化不需要使用分类属性值，相对简单，有等宽离散化、等频离散化、聚类等方法
  - 等宽离散化将属性划分为宽度一致的若干个区间
  - 等频离散化将属性划分为若干个区间，每个区间的数量相等
  - 聚类将属性间根据特性划分为不同的簇，以此形式将连续属性离散化
- 非监督离散化的方法能够完成对连续数据进行离散化的要求，但是相比之下，对连续属性监督离散化很多时候能够产生更好的结果。常用的方法是通过选取极大化区间纯度的临界值来进行划分
  - C4.5与CART算法中的连续属性离散化方法均属于监督离散化方法
  - CART 算法使用Gini系数作为区间纯度的度量标准
  - C4. 5算法使用熵作为区间纯度的度量标准

过拟合问题

- 训练误差代表分类方法对于现有训练样本集的拟合程度
- 泛化误差代表此方法的泛化能力，即对于新的样本数据的分类能力如何
- 模型的训练误差比较高，则称此分类模型欠拟合
- 模型的训练误差低但是泛化误差比较高，则称此分类模型过拟合
- 对于欠拟合问题，可以通过增加分类属性的数量、选取合适的分类属性等方法，提高模型对于训练样本的拟合程度

过拟合问题

- 对口罩销售定价进行分类
- 样本集

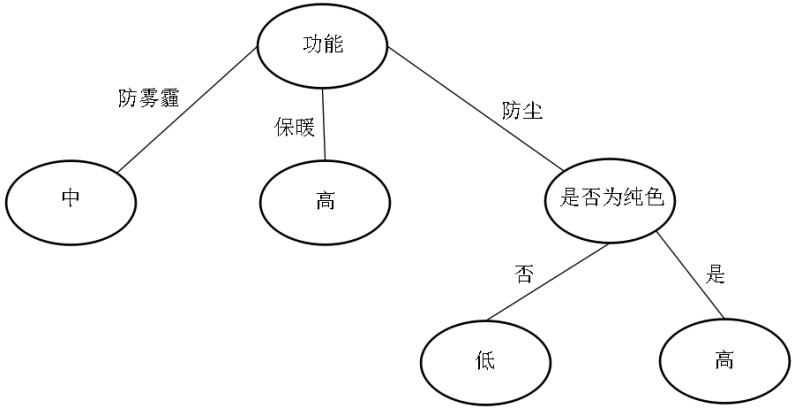
产品名	功能	是否为纯色	销售价位
加厚口罩	防尘	否	低
保暖口罩	保暖	否	高
护耳口罩	保暖	是	高
活性炭口罩	防雾霾	是	中
三层防尘口罩	防尘	否	低
艺人同款口罩	防尘	是	高
呼吸阀口罩	防雾霾	是	中

- 测试集

产品名	功能	是否为纯色	销售价位
儿童口罩	防尘	是	低
情侣口罩	保暖	否	高
一次性口罩	防尘	否	低
无纺布口罩	防尘	是	低
颗粒物防护口罩	防雾霾	否	中

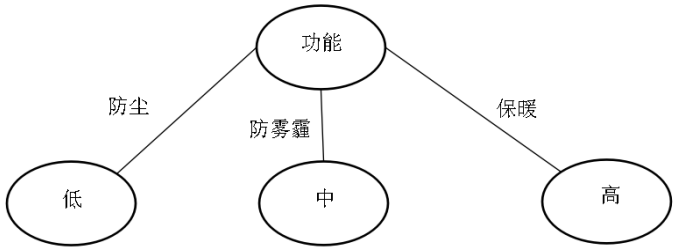
过拟合问题

- 三层决策树
- 训练误差为0，测试误差高达2/5



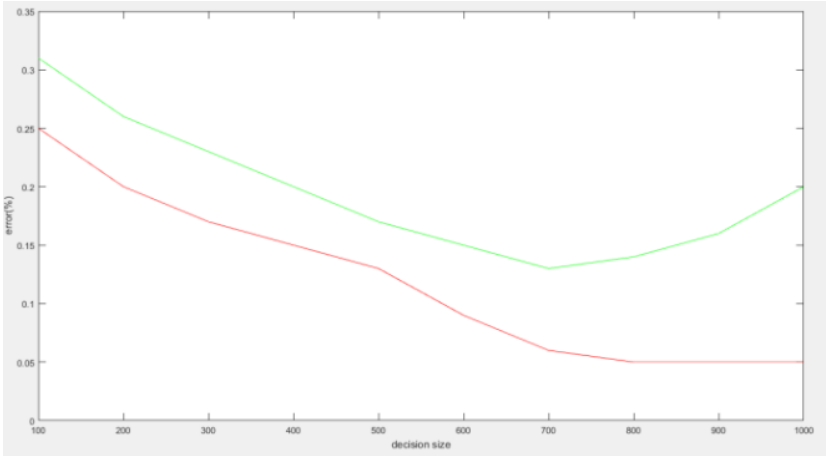
过拟合问题

- 两层决策树
- 训练集拟合程度相比较低，但测试集表现更好



过拟合问题

- 过拟合现象会导致随着决策树的继续增长，尽管训练误差仍在下降，但是泛化误差停止下降，甚至还会提升
- 决策树误差曲线



过拟合问题

- 解决过拟合问题，一方面要注意数据训练集的质量，选取具有代表性样本的训练样本集。另一方面要避免决策树过度增长，通过限制树的深度来减少数据中的噪声对于决策树构建的影响，一般可以采取剪枝的方法
- 剪枝是用来缩小决策树的规模，从而降低最终算法的复杂度并提高预测准确度，包括预剪枝和后剪枝两类
- 预剪枝的思路是提前终止决策树的增长，在形成完全拟合训练样本集的决策树之前就停止树的增长，避免决策树规模过大而产生过拟合
- 后剪枝策略先让决策树完全生长，之后针对子树进行判断，用叶子结点或者子树中最常用的分支替换子树，以此方式不断改进决策树，直至无法改进为止

错误率降低剪枝

- 错误率降低剪枝（ REP ）是后剪枝策略中最简单的算法之一， 该算法从叶子结点向上，依次将决策树的所有子树用其样本中最多的类替换，使用一个测试集进行测试， 记录下对于决策树的每棵子树剪枝前后的误差数之差，选取误差数减少最少的子树进行剪枝，将其用子样本集中最多的类替换。按此步骤自底向上，遍历决策树的所有子树，当发现没有可替换的子树时，即每棵子树剪枝后的误差数都会增多，则剪枝结束
- REP剪枝方法简单、快速，在数据集较大时效果不错，但由于需要比对模型子树替换前后的预测错误率，因此需要从数据集中划分出单独的测试集，故而当数据集较小时，REP剪枝策略的效果会有所下降

悲观剪枝

- 悲观剪枝（ PEP ）与REP相比，PEP不再需要构建一个单独的测试集。其假设某叶子结点t中有$N(t)$个样本，其中有$e(t)$个被错误分类的样本，则此叶子结点误分类率定义

$$r(t) = \frac{e(t) + 0.5}{N(t)}$$

- 其中0.5为修正因子。对于一棵有着N个叶子结点的子树T，其误分类率计算公式如下

$$r(T) = \frac{\sum[e(i) + 0.5]}{\sum N(i)} = \frac{\sum e(i) + \frac{N}{2}}{\sum N(i)}$$

- 由于修正因子的存在，有时即便子树的误差数要小于剪枝后的误差，仍有可能进行剪枝操作，因为误分类率的计算公式中考虑到了叶子结点树大小（N）的影响

代价复杂度剪枝策略

- 代价复杂度剪枝策略(CCP) 定义了代价与复杂度的概念，代价是指在剪枝过程中因为子树被替换而增加的错分样本，复杂度表示剪枝后减少的叶结点数
- CCP算法使用α作为衡量代价与复杂度之间关系的值，其计算公式如下

$$\alpha = \frac{R(t) - R(\mathrm{T}_t)}{|N_1| - 1}$$

- CCP的具体方法为，计算决策树T的每个非叶子结点的α值，每次计算之后剪掉具有最小α值的子树，循环此过程直至只剩下根结点，进行n次剪枝，生成n个决策树，从这n个决策树中根据真实误差估计选择最佳决策树

分类效果评价

- 对于一般分类问题，有训练误差、泛化误差、准确率、错误率等指标
- 对于常见的二分类问题，样本只有两种分类结果，将其定义为正例与反例。那么在进行分类时，对于一个样本，可能出现的分类情况共有四种:
  - 样本为正例，被分类为正例，称为真正类(TP)
  - 样本为正例，被分类为反例，称为假反类(FN)
  - 样本为反例，被分类为正例，称为假正类(FP)
  - 样本为反例，被分类为反例，称为真反类(TN)

分类效果评价

- 准确率：分类模型正确分类的样本数（包括正例与反例）与样本总数的比值

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

- 精确率(precision)：模型正确分类的正例样本数与总的正例样本总数（即正确分类的正例样本数目与错误分类的正确样本数目之和）的比值

$$precision = \frac{TP}{TP + FP}$$

- 召回率(recall，也称为查全率)：模型分类正确的正例样本数与分类正确的样本总数（分类正确的正例和分类正确的反例之和）的比值

$$recall = \frac{TP}{TP + FN}$$

分类效果评价

- F值为精确率和召回率的调和平均

$$F = \frac{(\alpha^2 + 1) \times accuracy \times recall}{\alpha^2(accuracy + recall)}$$

- 其中α为调和参数值，当α取值为1时，F值就是最常见的F1值

$$F_1 = \frac{2 \times accuracy \times recall}{accuracy + recall}$$

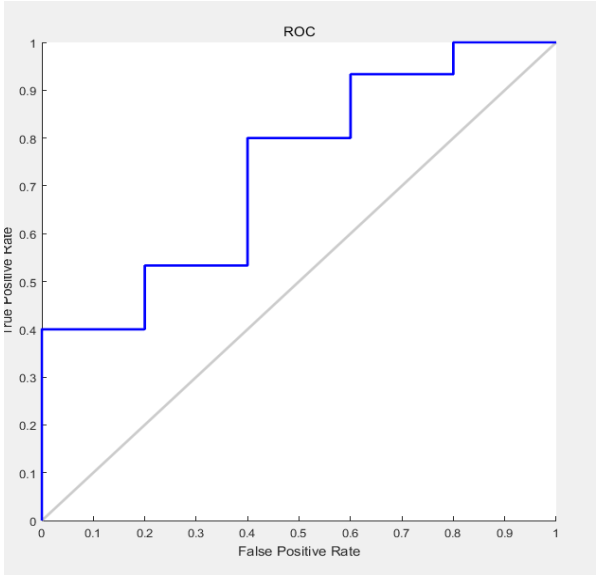
分类效果评价

- 受试者工作特征曲线 (ROC)曲线也是一种常用的综合评价指标。假设检验集中共有20个样本，每个样本为正类或反类，根据分类算法模型可以得出每个样本属于正类的概率，将样本按照此概率由高到低排列

样本编号	分类	预测为正类的概率
1	正类	0.98
2	正类	0.96
3	正类	0.92
4	正类	0.88
5	正类	0.85
6	正类	0.83
7	反类	0.82
8	正类	0.8
9	正类	0.78
10	反类	0.71
11	正类	0.68
12	正类	0.64
13	正类	0.59
14	正类	0.55
15	反类	0.52
16	正类	0.51
17	正类	0.5
18	反类	0.48
19	正类	0.42
20	反类	0.2

分类效果评价

- ROC曲线下的面积称为AUC(Area under Curve)，AUC值越大，表示分类模型的预测准确性越高，ROC曲线越光滑，一般代表过拟合现象越轻



分类效果评价方法

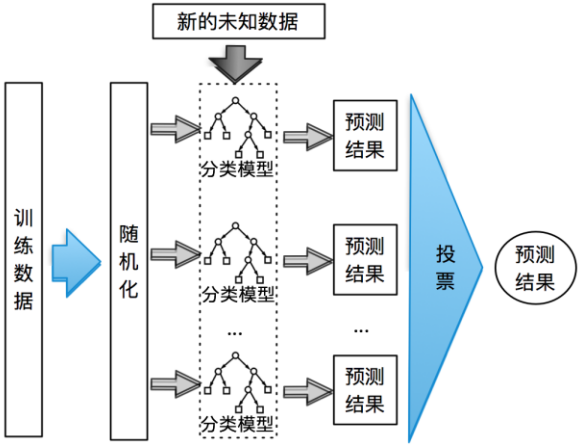
- 保留法将样本集按照定比例划分为训练集与检验集两个集合，两个集合中样本随机分配且不重叠。对于比例的确定，一般情况下， 训练集会大于检验集，例如训练集占70%，检验集占30%，具体比例可结合实际情况进行判定
- 蒙特卡洛交叉验证，也称重复随机二次采样验证，这种验证方法随机将数据集划分为训练集与检验集，使用检验集检验训练集训练的模型效果，多次重复此过程取平均值作为模型好坏的评价标准。蒙特卡洛交叉验证法也可看作是多次进行保留法
- k折交叉验证法将样本集随机地划分为k个大小相等的子集，在每一轮交叉验证中， 选择一个子集作为检验集，其余子集作为训练集，重复k轮，保证每一个子集都作为检验集出现，用K轮检验结果取平均值作为模型好坏的评价标准。最常用的k折交叉验证法为十折交叉验证

分类效果评价方法

- 留一法指每次检验集中只包含一个样本的交叉验证方法
- 留p法是每次使用p个样本作为检验集的交叉验证方法
- 自助法是统计学中的一种有放回均匀抽样方法，即从一个大小为n的样本数据集S中构建一个大小为n'的训练样本集St需要进行n'次抽取，每次均可能抽取到n个样本中的任何一个。n'次抽取之后，剩余的未被抽取到的样本成为检验集

集成学习

- 集成学习(Ensemble learning)是机器学习中近年来的一大热门领域。其中的集成方法是用多种学习方法的组合来获取比原方法更优的结果
- 使用于组合的算法是弱学习算法，即分类正确率仅比随机猜测略高的学习算法，但是组合之后的效果仍可能高于强学习算法，即集成之后的算法准确率和效率都很高



装袋法

- 装袋法(Bagging)又称为Bootstrap Aggregating,其原理是通过组合多个训练集的分类结果来提升分类效果
- 装袋法由于多次采样，每个样本被选中的概率相同，因此噪声数据的影响下降，所以装袋法太容易受到过拟合的影响
- 使用sklearn库实现的决策树装袋法提升分类效果。其中X和Y分别是鸢尾花（iris）数据集中的自变量（花的特征）和因变量（花的类别）

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
#加载iris数据集
iris = datasets.load_iris()
X = iris.data
Y = iris.target
```

装袋法

```
#分类器及交叉验证
seed = 42
kfold = KFold(n_splits=10, random_state=seed)
cart = DecisionTreeClassifier(criterion='gini',max_depth=2)
cart = cart.fit(X, Y)
result = cross_val_score(cart, X, Y, cv=kfold)
print("CART树结果：",result.mean())
model=BaggingClassifier(base_estimator=cart,n_estimators=100, random_state=seed)
result = cross_val_score(model, X, Y, cv=kfold)
print("装袋法提升后结果：",result.mean())
```

- 运行之后的结果如下
  - CART树结果： 0.933333333333
  - 装袋法提升后结果： 0.946666666667
- 可以看到装袋法对模型结果有一定提升。当然，提升程度与原模型的结构和数据质量有关。如果分类回归树的树高度设置为3或5，原算法本身的效果就会比较好，装袋法就没有提升空间

提升法

- 提升法(Boosting)与装袋法相比每次的训练样本均为同一组，并且引入了权重的概念，给每个单独的训练样本都会分配个相同的初始权重。然后进行T轮训练，每-轮中使用一个分类方法训练出一个分类模型，使用此分类模型对所有样本进行分类并更新所有样本的权重:分类正确的样本权重降低，分类错误的样本权重增加，从而达到更改样本分布的目的。由此可知，每一轮训练后，都会生成一个分类模型，而每次生成的这个分类模型都会更加注意在之前分类错误的样本，从而提高样本分类的准确率。对于新的样本，将T轮训练出的T个分类模型得出的预测结果加权平均，即可得出最终的预测结果。

提升法

- 基于sklearn库中的提升法分类器对决策树进行优化，提高分类准确率。Python代码如下，其中load_breast_cancer()方法加载乳腺癌数据集，自变量（细胞核的特征）和因变量（良性、恶性）分别赋给X和Y变量

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
dataset_all = datasets.load_breast_cancer()
X = dataset_all.data
Y = dataset_all.target
seed = 42
```

提升法

```
kfold = KFold(n_splits=10, random_state=seed)
dtree = DecisionTreeClassifier(criterion='gini',max_depth=3)
dtree = dtree.fit(X, Y)
result = cross_val_score(dtree, X, Y, cv=kfold)
print("决策树结果：",result.mean())
model = AdaBoostClassifier(base_estimator=dtree,
n_estimators=100,random_state=seed)
result = cross_val_score(model, X, Y, cv=kfold)
print("提升法改进结果：",result.mean())
```

- 运行之后的结果如下。
  - 决策树结果： 0.92969924812
  - 提升法改进结果： 0.970112781955
- 可以看到提升法对当前决策树分类器的分类效果改进较大

GBDT

- 梯度提升决策树算法是利用梯度下降的思想，使用损失函数的负梯度在当前模型的值，作为提升树中残差的近似值，以此来拟合回归决策树。梯度提升决策树的算法过程如下：
- 初始化决策树，估计一个使损失函数最小化的常数构建一个只有根节点的树。
- 不断提升迭代：
  - 计算当前模型中损失函数的负梯度值，作为残差的估计值；
  - 估计回归树中叶子节点的区域，拟合残差的近似值；
  - 利用线性搜索估计叶子节点区域的值，使损失函数极小化；
  - 更新决策树。
- 经过若干轮的提升法迭代过程之后，输出最终的模型

GBDT

- 对于GBDT算法的具体实现，最为出色的是XGBoost树提升系统
- 下面是在Python环境下使用XGBoost模块进行回归的调用示例，首先用pandas构造一个最简单的数据集df，其中x的值为[1,2,3]，y的值为[10,20,30]，并构建训练集矩阵T_train_xbg。代码如下

```
import pandas as pd
import xgboost as xgb
df = pd.DataFrame({'x':[1,2,3], 'y':[10,20,30]})
X_train = df.drop('y',axis=1)
Y_train = df['y']
T_train_xgb = xgb.DMatrix(X_train, Y_train)
params = {"objective": "reg:linear", "booster":"gblinear"}
gbm = xgb.train(dtrain=T_train_xgb,params=params)
Y_pred = gbm.predict(xgb.DMatrix(pd.DataFrame({'x':[4,5]})))
print(Y_pred)
```

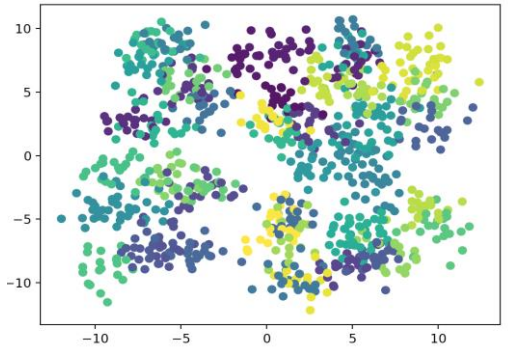
随机森林

- 随机森林是专为决策树分类器设计的集成方式，是装袋法的一种拓展。随机森林与装袋法采取相同的样本抽取方式。装袋法中的决策树每次从所有属性中选取一个最优的属性作为其分支属性，而随机森林算法每次从所有属性中随机抽取t个属性，然后从这t个属性中选取一个最优的属性作为其分支属性，这样就使得整个模型的随机性更强，从而使模型的泛化能力更强。而对于参数t的选取，决定了模型的随机性，若样本属性共有M个，t=1意味着随机选择一个属性来作为分支属性，t=属性总数时就变成了装袋法集成方式，通常t的取值为小于$\log_2(\mathrm{M}+1)$的最大整数。而随机森林算法使用的弱分类决策树通常为CART算法

随机森林

- 使用sklearn库中的随机森林算法和决策树算法进行效果对比，数据集由生成器随机生成，示例代码如下

```
from sklearn.model_selection import cross_val_score
from sklearn.datasets import make_blobs
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.tree import DecisionTreeClassifier
X, y = make_blobs(n_samples=1000, n_features=6, centers=50,
    random_state=0)
pyplot.scatter(X[:, 0], X[:, 1], c=y)
pyplot.show()
```



决策树应用

- VMware公司使用定制的决策树进行定价优化
- 长期以来， VMware公司的产品价格罕有变动。当接收到大订单的时候，销售代表会通过销售人员特别折扣(sales person specific discount, SPF) 标识来给出特定折扣。而VMW的定价部门则希望找到一种方法来优化其产品定价。在各个商业领域，市场上都有着对应的解决定价问题的数据挖掘解决方案。解决方案的目标不仅是优化定价，同时还要让客户的利益最大化。
- VMW公司的分析和数据科学小组通过分析历史价格和相应的销量变化，来理解折扣百分比与SPF标识的关联性，并以此为依据归纳出所有产品的推荐定价。传统的定价优化会根据数量变化来进行价格变动，但是VMW公司并不会改变价格，而是使用折扣作为替代方法，额外的步骤来确定是否需要给出折扣以及折扣率

决策树应用

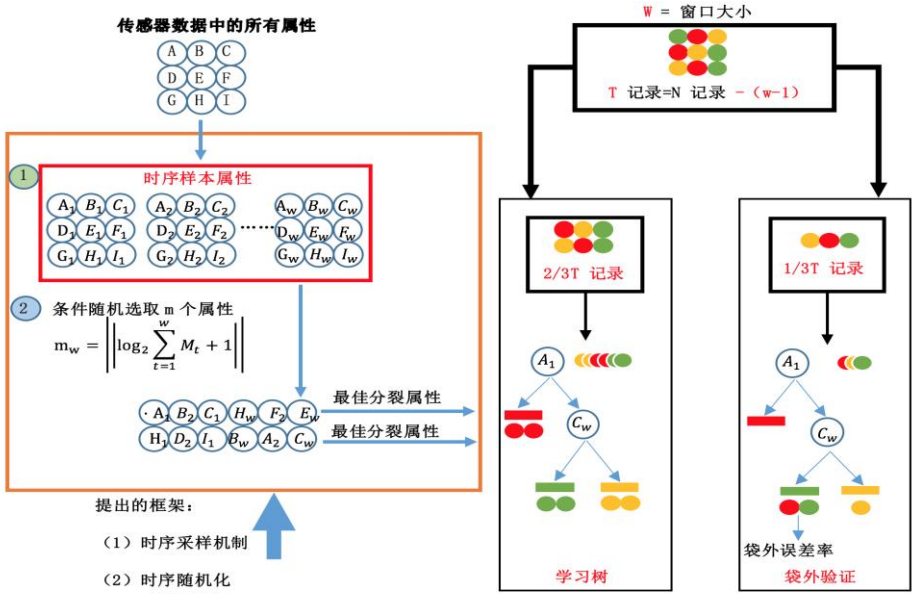
- 小组使用决策树的方法，基于折扣率和SPF标识的使用情况来对所有订单进行分类。根据业务的需求，以一种特定顺序构建出多分支(非二元)决策树。各种可能订单定价的属性都在决策树构建中被考虑进来。决策树构建过程中，基于均方误差、熵、log-loss. Gini 指标等确定了决策树构建过程中分支属性的顺序。构建的定制决策树使用了R语言的开发包从叶子结点自底向上构建决策树。相比于使用标准分裂标准，使用的是根据业务需求定制的分裂标准。分裂停止标准是最小化观测对象数目。针对完成的分块，基于分块的属性特性观察其折扣百分比与SPF使用情况的关联性，并完成产品平台级的定价推荐

决策树应用

- 建模过程包括数据集的创建、针对折扣率定制回归树、针对SPF标识定制分类树、确定相关性。数据集使用Greenplum与Hadoop进行创建，包含企业数据仓库中近年来的大量数据。定制回归树的构建过程中，根据测量标准将订单划分为248个分块，树的不纯度度量为均方误差，每次选取均方误差最小值，用于识别分块的属性有平均折扣百分比、有SPF标识的订单所占比率、交易数、平均订购数量和平均定价。定制分类树的构建过程中，根据测量标准将订单分为188个分块，树的不纯度度量为熵，用于识别分块的属性与回归树相同。其中分类决策树与回归决策树的输出结果被存储回原始数据库，以便进一步使用。为确定系数的相对重要性，根据两个决策树运行了定制的回归，对折扣率决策树使用了线性回归，对SPF使用情况决策树进行了逻辑回归

决策树应用

- 马来西亚多媒体大学使用随机森林的时序拓展对人类活动进行分类
- 随机森林的时序拓展单棵树构建过程

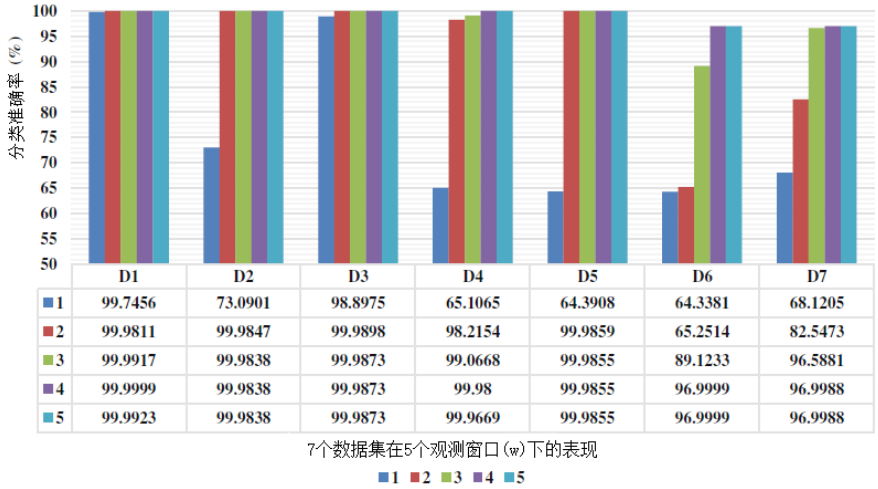


决策树应用

- 对随机森林算法的拓展主要在两方面：
  - 将传感器数据按照发生的时序重排：分类器在对活动进行分类时需要考虑之前时间戳的传感器数据，一组连续数据可以基于设定的时间窗口数量进行合并
  - 时序随机化
- 总的来说，马来西亚多媒体大学的团队采用了随机森林算法的时序拓展来分类人类活动。多个数据集的结果均证明了时序化的随机森林算法在人类活动识别任务中的可用性

决策树应用

- 随机森林的时序拓展准确率结果



	D1	D2	D3	D4	D5	D6	D7
1	99.7456	73.0901	98.8975	65.1065	64.3908	64.3381	68.1205
2	99.9811	99.9847	99.9898	98.2154	99.9859	65.2514	82.5473
3	99.9917	99.9838	99.9873	99.0668	99.9855	89.1233	96.5881
4	99.9999	99.9838	99.9873	99.98	99.9855	96.9999	96.9988
5	99.9923	99.9838	99.9873	99.9669	99.9855	96.9999	96.9988

决策树应用

- 随机森林与实验中使用的随机森林的时序拓展的分类性能比较

数据集	随机森林的时序拓展			Weka随机森林的时序拓展		
	分类准确率(%)	Kappa系数	OOB误差	分类准确率(%)	Kappa系数	OOB误差
D1	99.9999	0.9999	0.0065	99.7457	0.9967	0.0097
D2	99.9847	0.9999	0.0077	73.0897	0.6583	0.2788
D3	99.9898	0.9999	0.0086	98.8975	0.9884	0.0242
D4	99.9800	0.9998	0.0084	65.1067	0.6123	0.4419
D5	99.9859	0.9998	0.0052	64.3908	0.6134	0.3947
D6	96.9999	0.9842	0.0113	64.3645	0.8255	0.5421
D7	96.9988	0.9851	0.0043	69.5897	0.7222	0.4671


THANK
YOU