机器学习

Machine Learning

主讲人：张敏 清华大学长聘副教授

ML 机器学习

MACHINE LEARNING-MIN ZHANG

Unit.03

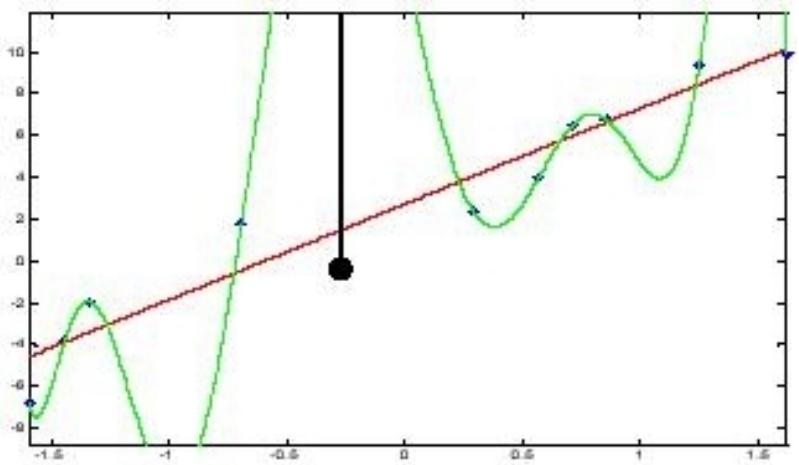
决策树学习

*图片均来自网络或已发表刊物

3.1决策树（回顾）

- 介绍及基本概念
- 以ID3算法为例
  - 算法描述
  - 选择特征
  - 终止条件
  - ID3算法的归纳偏置
- 过拟合问题

什么是过拟合（回顾）？



- 我们说$h \in H$ 对训练集过拟合，如果存在另一个假设 $h' \in H$ 满足:

$$err_{train}(h) < err_{train}(h')$$

AND

$$err_{test}(h) > err_{test}(h')$$

决策树过拟合的一个极端例子：

- 每个叶节点都对应单个训练样本 —— 每个训练样本都被完美地分类
- 整个树相当于仅仅是一个数据查表算法的简单实现

四、如何避免过拟合

如何避免过拟合

- 对决策树来说有两种方法避免过拟合
  I. 当数据的分裂在统计意义上并不显著时，就停止增长：预剪枝
  II. 构建一棵完全树，然后做后剪枝

对选项 I:

类型 I. 预剪枝: 何时停止分裂(1) : 基于样本数

- 通常 一个节点不再继续分裂，当：
  - 到达一个节点的训练样本数小于训练集合的一个特定比例 (例如 5%)
  - 无论混杂度或错误率是多少
  - 原因：基于过少数据样本的决定会带来较大误差和泛化错误

类型 I. 预剪枝: 何时停止分裂(2) : 基于信息增益的阈值

- 设定一个较小的阈值，如果满足下述条件则停止分裂：

$$\Delta i(s) \leq \beta$$

- 优点:
  - 用到了所有训练数据
  - 叶节点可能在树中的任何一层
- 缺点: 很难设定一个好的阈值

避免过拟合: 类型 II

- 对决策树来说有两种方法避免过拟合
  I. 当数据的分裂在统计意义上并不显著时，就停止增长：预剪枝
  II. 构建一棵完全树，然后做后剪枝

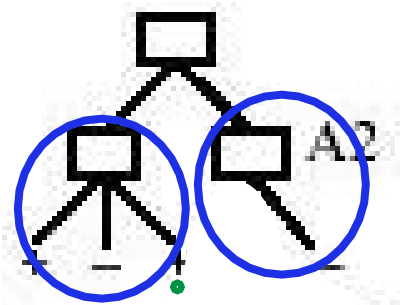
对于选项 II:

- 如何选择 “最佳” 的树？
  - 在另外的验证集合上测试效果
- MDL (Minimize Description Length 最小描述长度):

  minimize (*size*(tree) + *size*(misclassifications(tree)))

类型 II. 后剪枝 (1)：错误降低剪枝

- 把数据集分为训练集和验证集
  - 验证集:
    - 已知标签
    - 测试效果
    - 在该集合上不做模型更新!
- 剪枝直到再剪就会对损害性能:
  - 在验证集上测试剪去每个可能节点(和以其为根的子树)的影响
  - 贪心地去掉某个可以提升验证集准确率的节点

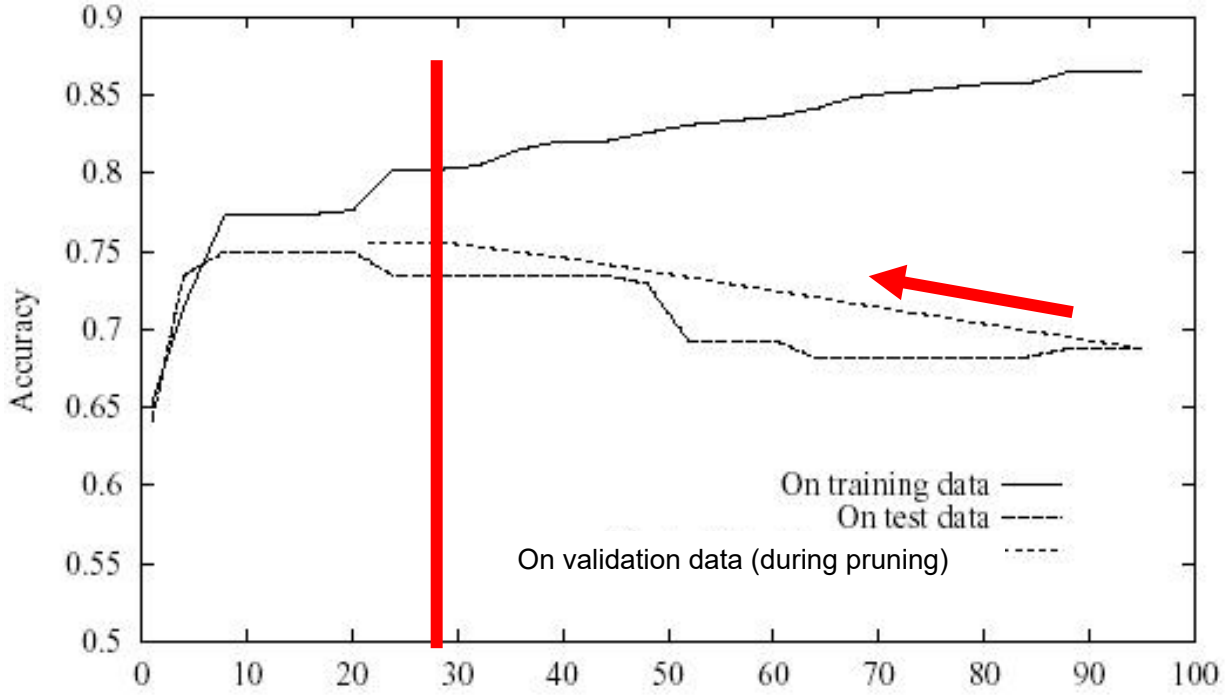


如何定义新的叶节点的标签?

剪枝后新的叶节点的标签赋值策略

- 赋值成最常见的类别
- 给这个节点多类别的标签
  - 每个类别有一个支持度 (根据训练集中每种标签的数目)
  - 测试时: 依据概率选择某个类别或选择多个标签
- 如果是一个回归树 (数值标签)，可以做平均或加权平均
- ……

错误降低剪枝的效果


0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
Accuracy
0
10
20
30
40
50
60
70
80
90
100
On training data
On test data
On validation data (during pruning)

类型 II. 后剪枝 (2)：规则后剪枝

1, 把树转换成等价的由规则构成的集合

- e.g. if (outlook=sunny)∧(humidity=high) then playTennis = no

2, 对每条规则进行剪枝，去除哪些能够提升该规则准确率的规则前件

- i.e. (outlook=sunny), (humidity=high)

3, 将规则排序成一个序列 (根据规则的准确率从高往低排序)

4, 用该序列中的最终规则对样本进行分类（依次查看是否满足规则序列）

(注：在规则被剪枝后，它可能不再能恢复成一棵树)

一种被广泛使用的方法，例如C4.5

为什么在剪枝前将决策树转化为规则?

- 独立于上下文
  - 否则，如果子树被剪枝，有两个选择:
    - 完全删除该节点
    - 保留它
- 不区分根节点和叶节点
- 提升可读性

五、扩展: 现实场景中的决策树学习

- 问题 & 改进

1. 连续属性值

$$x_l < x_s < x_u$$

温度	40	48	60	72	80	90
决定	No	No	Yes	Yes	Yes	No

- 建立一些离散属性值
- 可选的策略:
  - I.选择相邻但有不同决策的值的中间值 $x_s = (x_l + x_u)/2$

    (Fayyad 在1991年证明了满足这样条件的阈值可以信息增益IG最大化)
  - II. 考虑概率 $x_s = (1-P)x_l + Px_u$

2. 具有过多取值的属性

问题:

- 偏差: 如果属性有很多值，根据信息增益IG，会优先被选择
  - e.g. 享受运动的例子中，将一年里的每一天作为属性
- 一个可能的解决方法: 用 *GainRatio* （*增益比*）来替代

$$GainRatio(S,A) \equiv \frac{Gain(S,A)}{SplitInformation(S,A)}$$

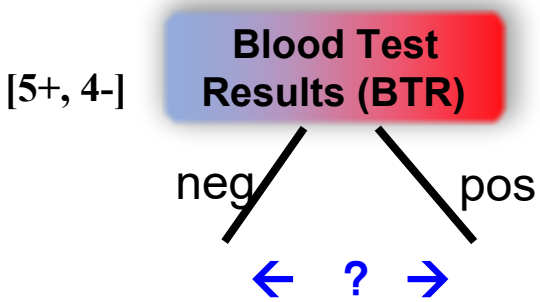
$$SplitInformation(S,A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

惩罚项，集合 S 在 A 属性的熵

3. 未知属性值

BTR	Temp	…	label
neg	normal	…	-
neg	normal	…	-
neg	normal	…	-
neg	normal	…	-
neg	high	…	+
pos	normal	…	+
pos	high	…	+
pos	high	…	+
?	normal	…	+

有缺失数据



训练中最常见: neg

[2+, 4-] [3+, 0-]

相同标签中最常见: pos

[1+, 4-] [4+, 0-]

根据概率赋值: neg 5/8, pos 3/8

[(1+5/8)+, 4-] [(3+3/8)+, 0-]

4. 有代价的属性

- 有时有的属性不容易获得（收集该属性值的代价太大）
- Tan & Schlimmer (1990)

$$\frac{Gain^2(S,A)}{Cost(A)}$$

- Nunez (1988)

$$\frac{2^{Gain(S,A)}-1}{(Cost(A)+1)^w}$$

  - w:[0,1] 代价的重要性

其他信息

- 可能是最简单和频繁使用的算法
  - 易于理解
  - 易于实现
  - 易于使用
  - 计算开销小
- 决策森林:
  - 由C4.5产生的许多决策树

- 更新的算法：C5.0 http://www.rulequest.com/see5-info.html
- Ross Quinlan的主页: http://www.rulequest.com/Personal/

决策树学习总结(基础部分)

- 介绍及基本概念
- 以ID3算法为例
  - 算法描述
  - 选择特征
  - 终止条件
  - ID3算法的归纳偏置
- 过拟合问题
- 剪枝
  - 预剪枝：基于样本数；基于信息增益阈值
  - 后剪枝: 错误降低剪枝；规则后剪枝

在实际应用中，一般预剪枝更快，而后剪枝得到的树准确率更高

决策树学习总结(扩展部分)

- 实际场景中的决策树学习
  - 连续属性值的离散化
  - 具有过多取值的属性处理
  - 未知（缺失）属性值的处理
  - 有代价的属性
- 基本想法来源于人类做决策的过程
- 简单、容易理解：“如果…就…”
- 对噪声数据有鲁棒性

决策树学习总结(实际应用)

- 在研究和应用中广泛使用
  - 医疗诊断(临床症状 → 疾病)
  - 信用分析 (个人信息 → 有价值客户?)
  - 日程规划
  - ……
- 通常在部署更复杂的算法之前，常把决策树作为一个基准方法（baseline）
- 决策树方法常被用作复杂学习框架中的基础部分之一

归纳学习假设

归纳学习假设

- 大部分的学习是从已知的样本中获得一般化的概念.

- 归纳学习算法能够在最大程度上保证输出的假设能够在训练数据上拟合目标概念
  - **注意：过拟合问题**

归纳学习假设

- 归纳学习假设:

  Any hypothesis found to approximate the target function well over **a sufficiently large set** of training examples will also approximate the target function well over unobserved examples.

  (任一假设若在足够大的训练样例集中很好地逼近目标函数，它也能在未见实例中很好地逼近目标函数)