人工智能与机器学习

第十讲 决策树学习

倪东 叶琦
工业控制研究所

杭州 • 浙江大学 • 2022

ICDM 2006会议的算法投票结果

共有145人参加了ICDM 2006 Panel (会议的专题讨论),
并对18种候选算法进行投票，选出了机器学习10大算法

排名	主题	算法	得票数	发表时间	作者	陈述人
1	分类	C4.5	61	1993	Quinlan, J.R	*Hiroshi Motoda*
2	聚类	k-Means	60	1967	MacQueen, J.B	*Joydeep Ghosh*
3	统计学习	SVM	58	1995	Vapnik, V.N	*QiangYang*
4	关联分析	Apriori	52	1994	Rakesh Agrawal	*Christos Faloutsos*
5	统计学习	EM	48	2000	McLachlan, G	*Joydeep Ghosh*
6	链接挖掘	PageRank	46	1998	Brin, S.	*Christos Faloutsos*
7	集装与推进	AdaBoost	45	1997	Freund, Y.	*Zhi-Hua Zhou*
8	分类	kNN	45	1996	Hastie, T	*Vipin Kumar*
9	分类	Naïve Bayes	45	2001	Hand, D.J	*Qiang Yang*
10	分类	CART	34	1984	L.Breiman	*Dan Steinberg*

概 论

- 决策树学习是应用最广的归纳推理算法之一
- 是一种逼近离散值函数的方法
- 很好的健壮性
- 能够学习析取表达式
- ID3, Assistant, C4.5
- 搜索一个完整表示的假设空间
- 归纳偏置是优先选择较小的树
- 决策树表示了多个if-then规则

提 纲

- 决策树定义
- 适用问题特征
- 基本ID3算法
- 决策树学习的归纳偏置
- 训练数据的过度拟合
- …

决策树基本概念

关于分类问题

分类（Classification）任务就是通过学习获得一个目标函数（Target Function）f, 将每个属性集x映射到一个预先定义好的类标号y。

分类任务的输入数据是记录的集合，每条记录也称为实例或者样例。用元组(X,y)表示，其中，X 是属性集合，y是一个特殊的属性，指出样例的类标号（也称为分类属性或者目标属性）

决策树基本概念

关于分类问题

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
人类	恒温	毛发	是	否	否	是	否	哺乳动物
海龟	冷血	鳞片	否	半	否	是	否	爬行类
鸽子	恒温	羽毛	否	否	是	是	否	鸟类
鲸	恒温	毛发	是	是	否	否	否	哺乳类

X

y

分类与回归

分类目标属性y是离散的，回归目标属性y是连续的

决策树基本概念

解决分类问题的一般方法

通过以上对分类问题一般方法的描述，可以看出分类问题一般包括两个步骤：

1、模型构建（归纳）
通过对训练集合的归纳，建立分类模型。

2、预测应用（推论）
根据建立的分类模型，对测试集合进行测试。

决策树基本概念

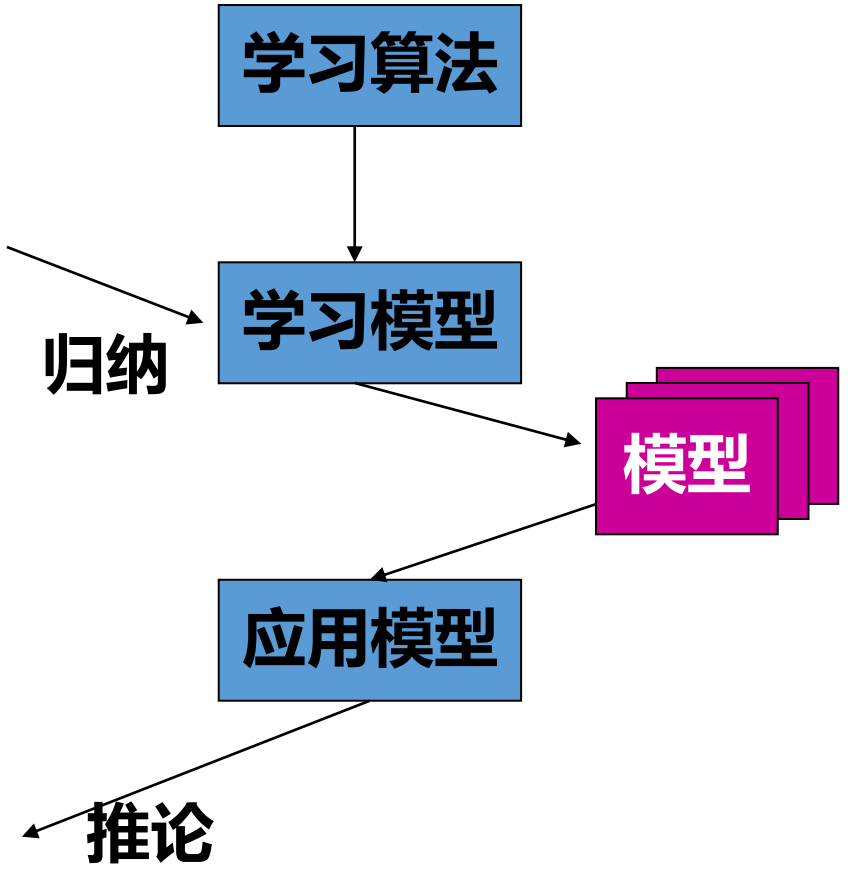
解决分类问题的一般方法

训练集（类标号已知）

TID	A1	A2	A3	类
1	Y	100	L	N
2	N	125	S	N
3	Y	400	L	Y
4	N	415	M	N

检验集（类标号未知）

TID	A1	A2	A3	类
1	Y	100	L	?
2	N	125	S	?
3	Y	400	L	?
4	N	415	M	?



决策树表示法

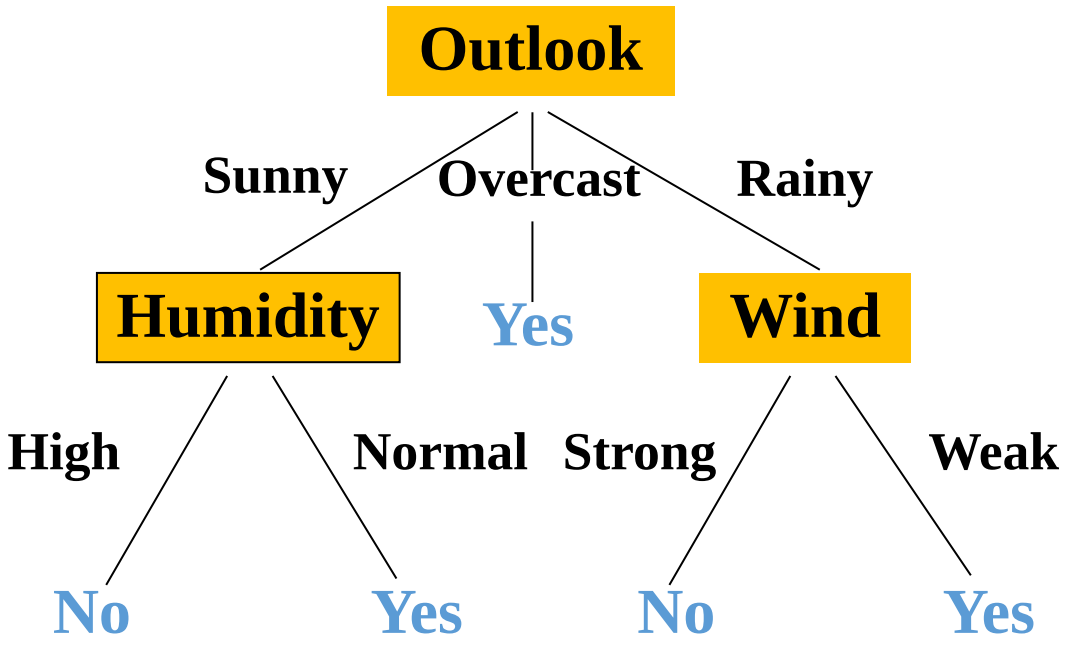


图3-1 概念Play Tennis的决策树

- 内部节点(包括根节点)指定了对实例的某个属性的测试
- 节点的每个后继分支对应于该属性的一个可能值
- 叶子节点即为实例所属的分类
- 决策树代表实例属性值约束的合取的析取式

决策树学习的适用问题

- 适用问题的特征
  - 实例由“属性-值”对表示
  - 目标函数具有离散的输出值
  - 可能需要析取的描述
  - 训练数据可以包含错误
  - 训练数据可以包含缺少属性值的实例
- 问题举例
  - 医学中的应用（如根据疾病分类患者、疾病分析与预测）
  - 根据起因分类设备故障（故障诊断）
  - 根据拖欠支付的可能性分类贷款申请
- 分类问题
  - 核心任务是把样例分类到各可能的离散值对应的类别

基本的决策树学习算法ID3

- 大多数决策树学习算法是一种核心算法的变体
- 采用自顶向下的贪婪搜索遍历可能的决策树空间
- ID3是这种算法的代表
- 该方法使用信息增益度选择测试属性。

ID3算法通过自顶向下构造决策树来进行学习

构造过程：

- 选择根节点 - 使用统计测试确定每一个实例属性单独分类训练样例的能力，分类能力最好的属性被选作树的根节点
- 为根节点属性的每个可能值产生一个分支，并把训练样例排列到适当的分支
- 重复上面的过程，用每个分支节点关联的训练样例来选取在该点被测试的最佳属性，直到满足以下两个条件中的任一个：

  1） 所有的属性已经被这条路径包括；

  2）与这个节点关联的所有训练样例具有相同的目标属性值

ID3算法的核心问题是选取在树的每个节点要测试的属性。

表3-1 用于学习布尔函数的ID3算法

- ID3(Examples, Target_attribute, Attributes)
- 创建树的root节点
- 如果Examples都为正,返回label=+的单节点树root
- 如果Examples都为反,返回label=-的单节点树root
- 如果Attributes为空，那么返回单节点root，label=Examples中最普遍的Target_attribute值
- 否则开始

  A←Attributes中分类examples能力最好的属性

  root的决策属性←A

  对于A的每个可能值vi

  在root下加一个新的分支对应测试A=vi

  令$Examples_{vi}$为Examples中满足A属性值为vi的子集

  如果$Examples_{vi}$为空

  在这个新分支下加一个叶子节点，节点的label=Examples中最普遍的Target_attribute值

  否则在新分支下加一个子树ID3

  ($Examples_{vi}$,Target_attribute,Attributes-{A})
- 结束
- 返回root

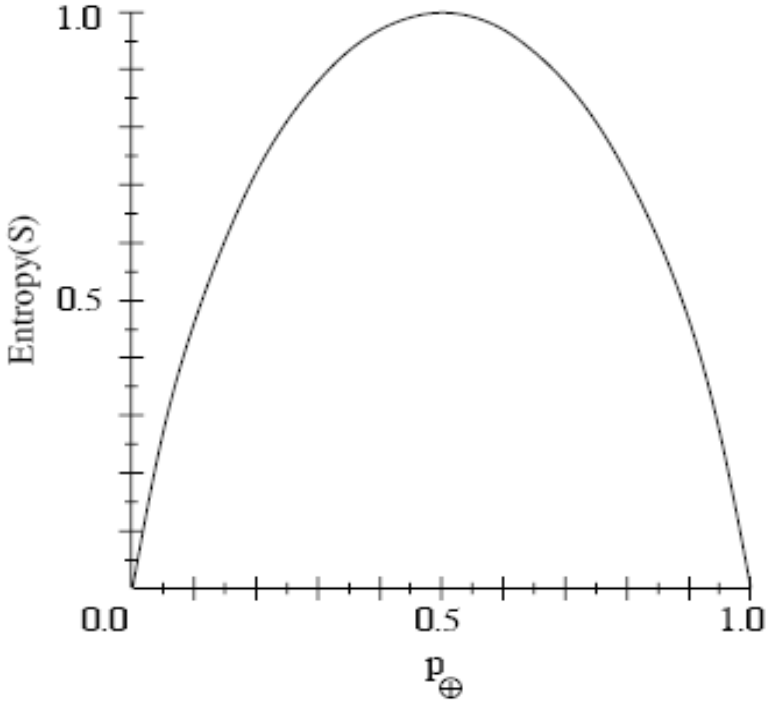
最佳分类属性

- 信息增益(Information Gain)
  - 用来衡量给定的属性区分训练样例的能力
  - ID3算法在增长树的每一步使用信息增益从候选属性中选择属性
- 用熵度量样例的均一性
  - 给定包含关于某个目标概念的正反样例的样例集S，那么S相对这个布尔型分类的熵为

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

  - 信息论中对熵的一种解释，熵确定了要编码集合S中任意成员的分类所需要的最少二进制位数
  - 更一般地，如果目标属性具有c个不同的值，那么S相对于c个状态的分类的熵定义为

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$



- S的所有成员属于同一类，*Entropy(S)*=0；
- S的正反样例数量相等，*Entropy(S)*=1；
- S的正反样例数量不等，熵介于0，1之间

- 抛一枚均匀硬币的信息熵是多少？
  解：出现正面与反面的概率均为0.5，信息熵是

$$E(x) = -\sum_{i=1}^{q} p(x_i) \log p(x_i)$$

$$= -(0.5 \log 0.5 + 0.5 \log 0.5)$$

$$= 1$$

- 用信息增益度量期望的熵降低
  - 属性的信息增益，由于使用这个属性分割样例而导致的期望熵降低
  - 一个属性A相对样例集合S的信息增益Gain（S，A）被定义为：

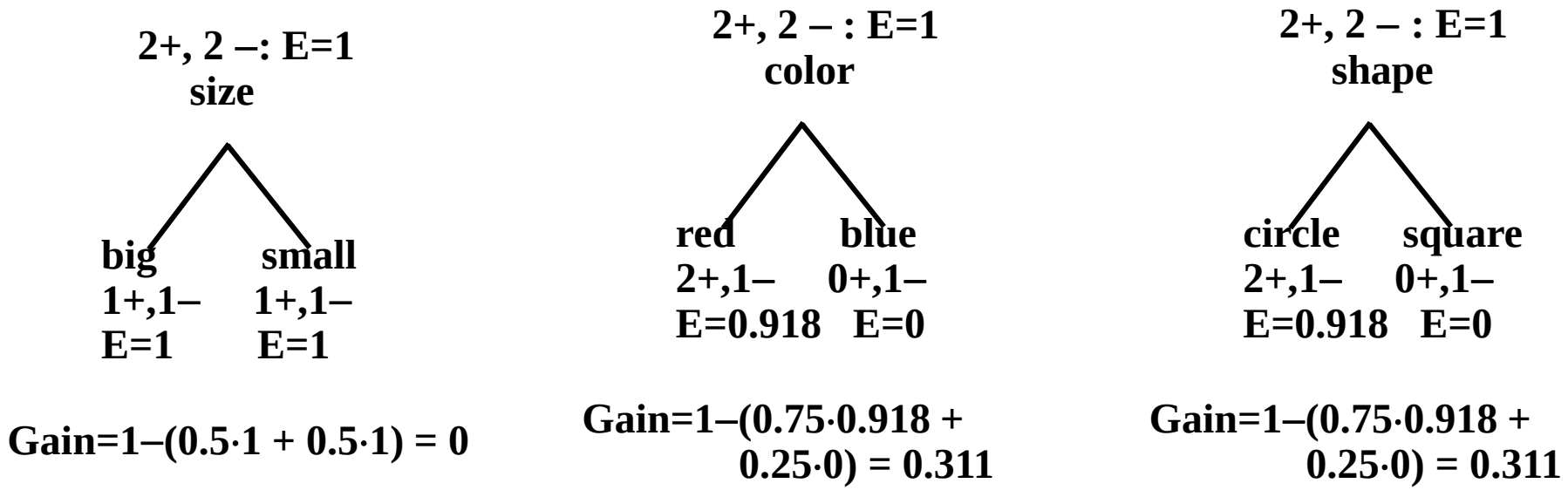
$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Values(*A*)是属性*A*所有可能值的集合，*Sv*是*S*中属性*A*的值为*v*的子集

  - *Gain(S,A)*是在知道属性A的值后可以节省的二进制位数；

计算属性的信息增益

<big, red, circle>: +　　<small, red, circle>: +
<small, red, square>: –　　<big, blue, circle>: –



Gain=1–(0.5·1 + 0.5·1) = 0

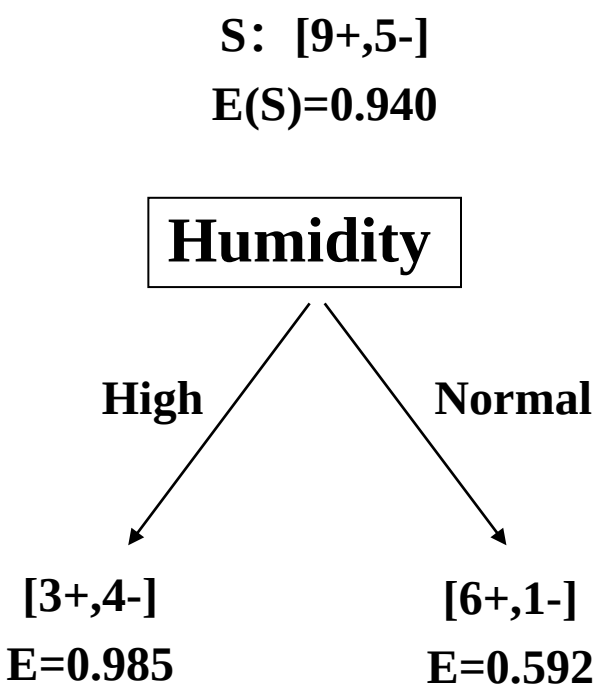
Gain=1–(0.75·0.918 + 0.25·0) = 0.311

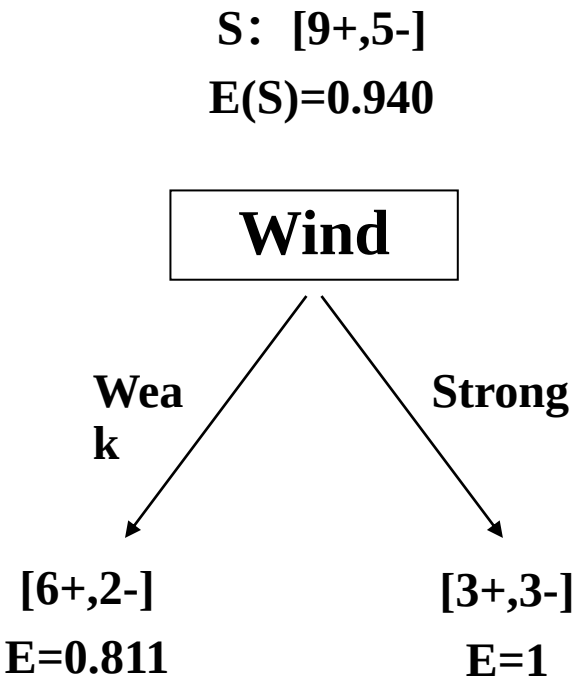
Gain=1–(0.75·0.918 + 0.25·0) = 0.311

表3-2 目标概念PlayTennis的训练样例

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

计算属性的信息增益





Gain (*S, Humidity*)

= 0.940-(7/14)*0.985- (7/14)*0.592

=0.151

Gain (*S, Wind*)

= 0.940-(8/14)*0.811- (6/14)*1

=0.048

ID3算法示例

考虑表3-2的训练数据所代表的学习任务。

1. 创建决策树的根节点。

   计算每一个候选属性的信息增益，然后选择信息增益最高的一个。

   $Gain(S, Outlook) = 0.246$

   $Gain(S, Humidity) = 0.151$

   $Gain(S, Wind) = 0.048$

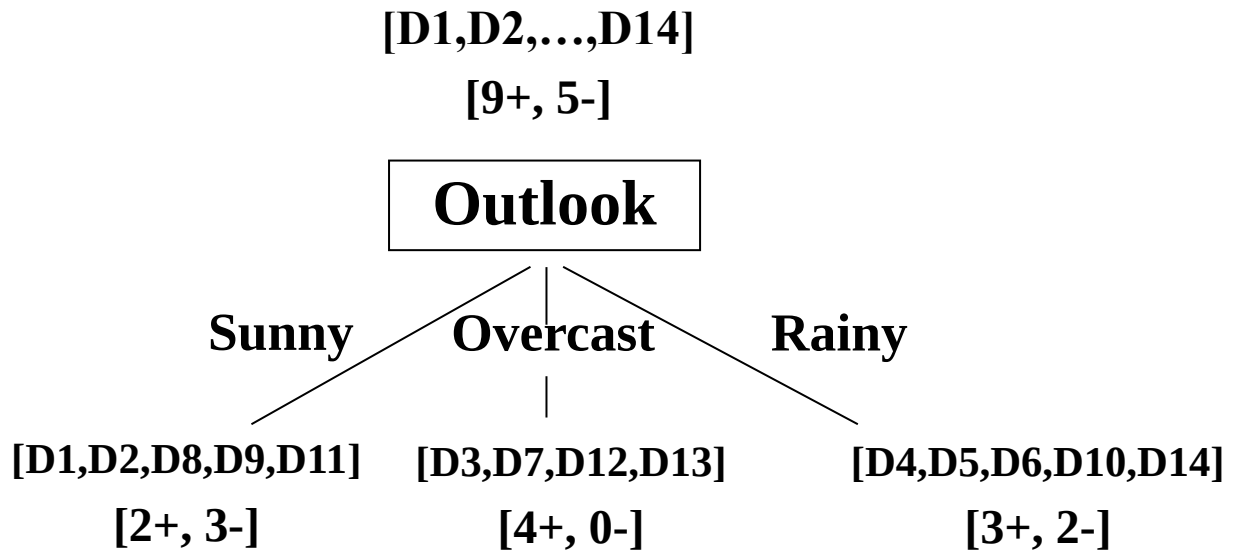
   $Gain(S, Temperature) = 0.029$

   根据信息增益标准，属性Outlook被选作根节点的决策属性，并为它的每一个可能值（Sunny、Overcast和Rainy）在根节点下创建分支，得到部分决策树显示在图3-4中。

2. 对非终端的后继节点再重复前面的过程以选择一个新的属性来分割训练样例，这一次仅使用与这个节点关联的训练样例，直到满足结束条件。

表3-2 目标概念PlayTennis的训练样例

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rainy	Mild	High	Weak	Yes
D5	Rainy	Cool	Normal	Weak	Yes
D6	Rainy	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rainy	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rainy	Mild	High	Strong	No

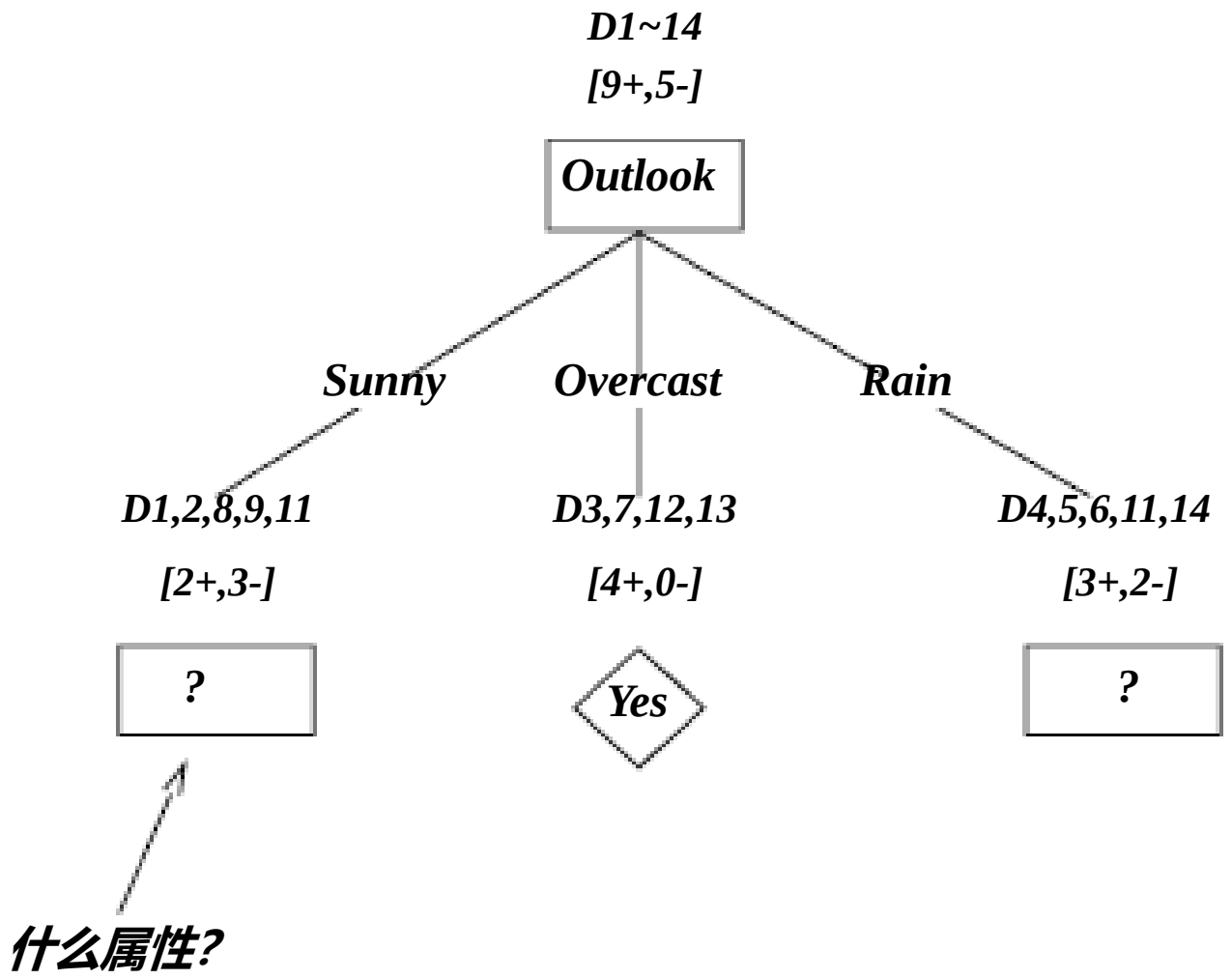


***Gain (S, Outlook)*=0.246**

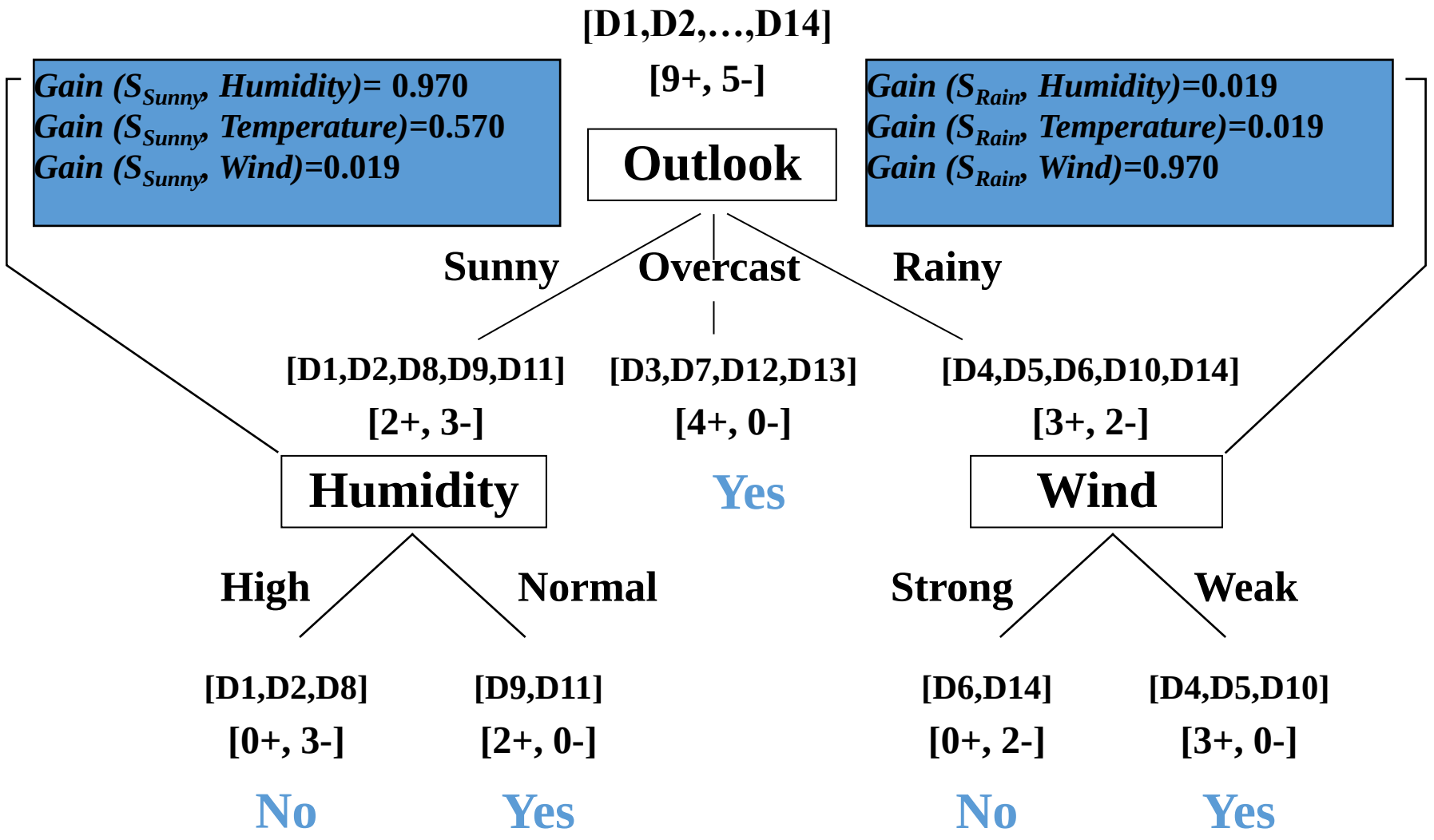
***Gain (S, Humidity)*=0.151**

***Gain (S, Wind)*=0.048**

***Gain (S, Temperature)*=0.029**



ID3算法第一步后形成的部分决策树


[D1,D2,…,D14]
[9+, 5-]
Gain (S_Sunny, Humidity)= 0.970
Gain (S_Sunny, Temperature)=0.570
Gain (S_Sunny, Wind)=0.019
Outlook
Gain (S_Rain, Humidity)=0.019
Gain (S_Rain, Temperature)=0.019
Gain (S_Rain, Wind)=0.970
Sunny
Overcast
Rainy
[D1,D2,D8,D9,D11]
[2+, 3-]
[D3,D7,D12,D13]
[4+, 0-]
[D4,D5,D6,D10,D14]
[3+, 2-]
Humidity
Yes
Wind
High
Normal
Strong
Weak
[D1,D2,D8]
[0+, 3-]
[D9,D11]
[2+, 0-]
[D6,D14]
[0+, 2-]
[D4,D5,D10]
[3+, 0-]
No
Yes
No
Yes

ID3算法步骤2

决策树学习中的假设空间搜索

- ID3算法搜索的假设空间就是可能的决策树的集合。ID3以爬山算法遍历这个假设空间，引导这种爬山搜索的评估函数是信息增益度量。
- 观察ID3的搜索空间和搜索策略，认识到这个算法的优势和不足：

✓ 假设空间包含所有的决策树，它是关于现有属性的有限离散值函数的一个完整空间

✓ 维护单一的当前假设（不同于上章变型空间候选消除算法），不能判断有多少个其他的决策树也与现有的训练数据一致

✓ 不进行回溯，可能收敛到局部最优

✓ 每一步都使用当前所有的训练样例，不同于基于单独的训练样例递增作出决定，容错性增强

决策树学习的归纳偏置

- ID3的搜索策略
  - 优先选择较短的树
  - 选择那些信息增益高的属性离根节点较近的树
  - 很难准确刻画ID3的归纳偏置
- 近似的ID3的归纳偏置
  - 较短的树比较长的树优先
  - 近似在于ID3得到局部最优，而不一定是全局最优
  - 一个精确具有这个归纳偏置的算法，BFS-ID3
- 更贴切近似的归纳偏置
  - 较短的树比较长的树优先，信息增益高的属性更靠近根节点的树优先

限定偏置和优选偏置

- ID3和候选消除算法的比较
  - ID3的搜索范围是一个完整的假设空间，但不彻底地搜索这个空间
  - 候选消除算法的搜索范围是不完整的假设空间，但彻底地搜索这个空间
  - ID3的归纳偏置完全是搜索策略排序假设的结果，来自搜索策略
  - 候选消除算法完全是假设表示的表达能力的结果，来自对搜索空间的定义

限定偏置和优选偏置

- 优选偏置（搜索偏置）
  - ID3的归纳偏置是对某种假设胜过其他假设的一种优选，对最终可列举的假设没有硬性限制
- 限定偏置（语言偏置）
  - 候选消除算法的偏置是对待考虑假设的一种限定
- 通常优选偏置比限定偏置更符合归纳学习的需要
- 优选偏置和限定偏置的结合
  - 考虑第1章下棋的例子
  - 限定偏置：线性函数表示评估函数
  - 优选偏置：LMS作为特定的参数调整方法

为什么短的假设优先?

- ID3的归纳偏置的哲学基础
- 奥坎姆剃刀
  - 优先选择拟合数据的最简单的假设
- 科学上的例子
  - 物理学家优先选择行星运动的简单假设;
  - 简单假设的数量远比复杂假设的数量少;
  - 简单假设对训练样例的针对性更小，更像是泛化的规律，而不是训练样例的另一种描述。

奥坎姆剃刀

- 设想你是在一条积雪的街上行走。在你前面有一个人带着一顶黑色的高筒礼帽。
- 街对面站着一群男孩，觉得这顶礼帽是个很好的目标，其中一个扔雪球一下击中了帽子。让我们举出两种解释来说明这顶帽子的随后遭遇。
- 第一，在帽子受击的一刹那，一队天使疾飞而下，出其不意地把帽子从那人头上揭走了。
- 第二，雪球把帽子击落了。
- 我们将选择？？种解释。
- 这就是科学上普遍适用的所谓“节俭律”的简单说明。这条定律的意义，就在于说明，最可能的解释就是最好的解释，有时这条定律又被称为奥坎姆剃刀

为什么短的假设优先

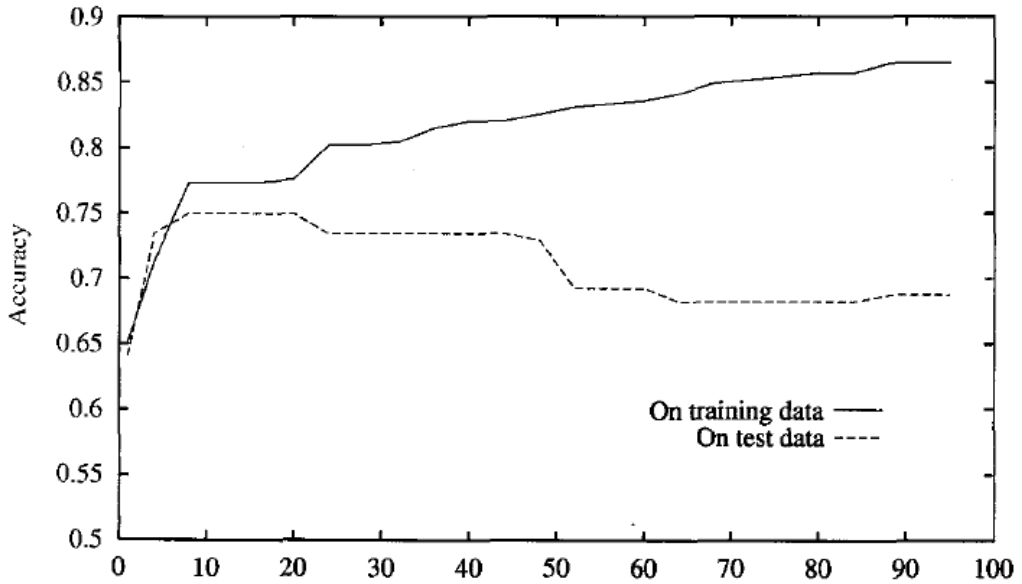
- 奥坎姆剃刀的困难
  - 可以定义很多小的假设集合，根据什么相信有短描述的决策树组成的小假设集合比其他可定义的小假设集合更适当?
  - 假设的规模由学习器内部使用的特定表示决定
- 从生物进化的观点看内部表示和奥坎姆剃刀原则

决策树学习的常见问题

- 决策树学习的实际问题
  - 确定决策树增长的深度
  - 处理连续值的属性
  - 选择一个适当的属性筛选度量标准
  - 处理属性值不完整的训练数据
  - 处理不同代价的属性
  - 提高计算效率
- 针对这些问题，ID3被扩展成C4.5

避免过度拟合数据

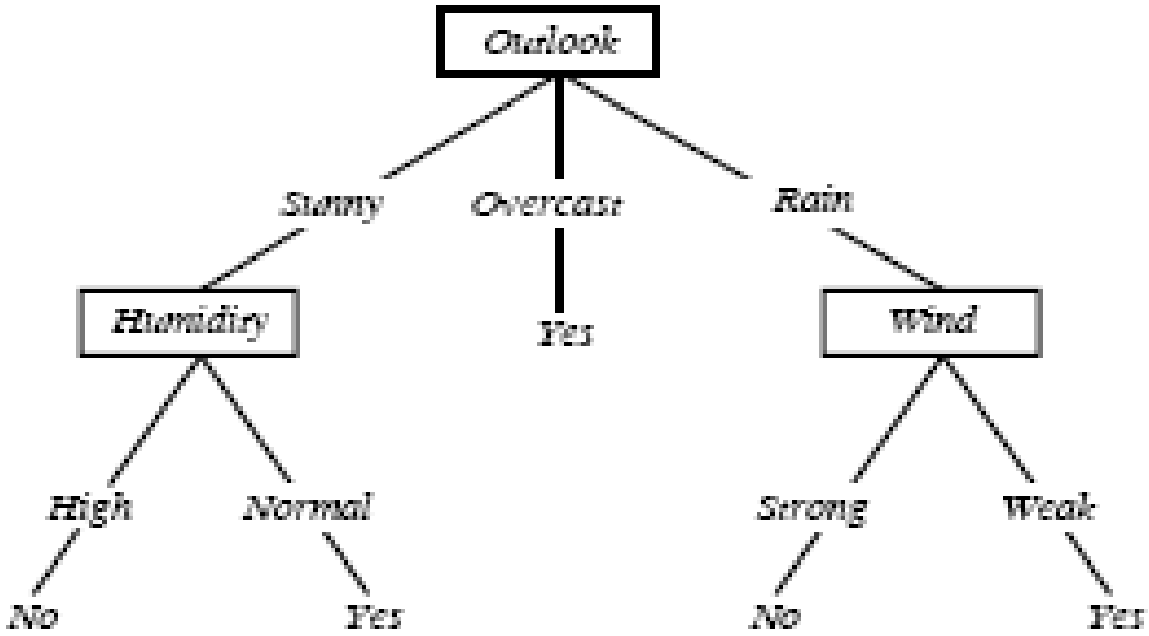
- 过度拟合
  - 对于一个假设，当存在其它的假设对训练样例的拟合比它差，但事实上在实例的整个分布上表现得却更好时，我们说这个假设过度拟合训练样例。
  - 定义：给定一个假设空间H，一个假设h∈H，如果存在其它的假设h’ ∈H，使得在训练样例上h的错误率比h’小，但在整个实例分布上h’的错误率比h小，那么就说假设h过度拟合训练数据。

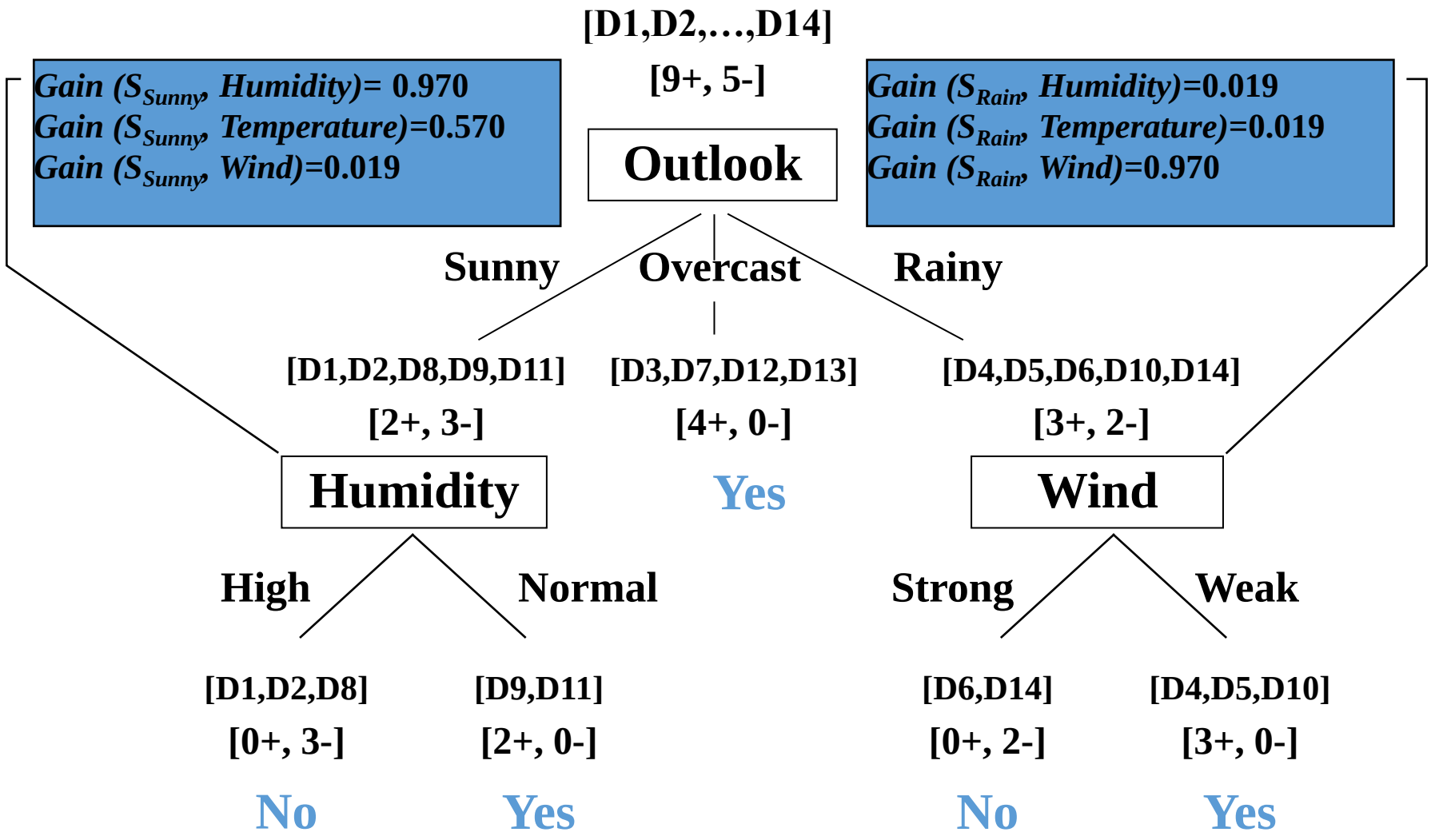


避免过度拟合数据（2）

- 导致过度拟合的原因（1）
  - 一种可能原因是训练样例含有随机错误或噪声

Sunny Hot Normal Strong PlayTennis = No





ID3算法步骤2

Sunny Hot Normal Strong PlayTennis =No

避免过度拟合数据（3）

- 导致过度拟合的原因（2）
  - 当训练数据没有噪声时，过度拟合也有可能发生，特别是当少量的样例被关联到叶子节点时，很可能出现巧合的规律性，使得一些属性恰巧可以很好地分割样例，但却与实际的目标函数并无关系。
  - 过度拟合使决策树的精度降低（10～25）%

避免过度拟合数据（4）

- 避免过度拟合的方法
  - 及早停止树增长
  - 后修剪法
- 两种方法的特点
  - 第一种方法更直观
  - 第一种方法中，精确地估计何时停止树增长很困难
  - 第二种方法被证明在实践中更成功

避免过度拟合数据（5）

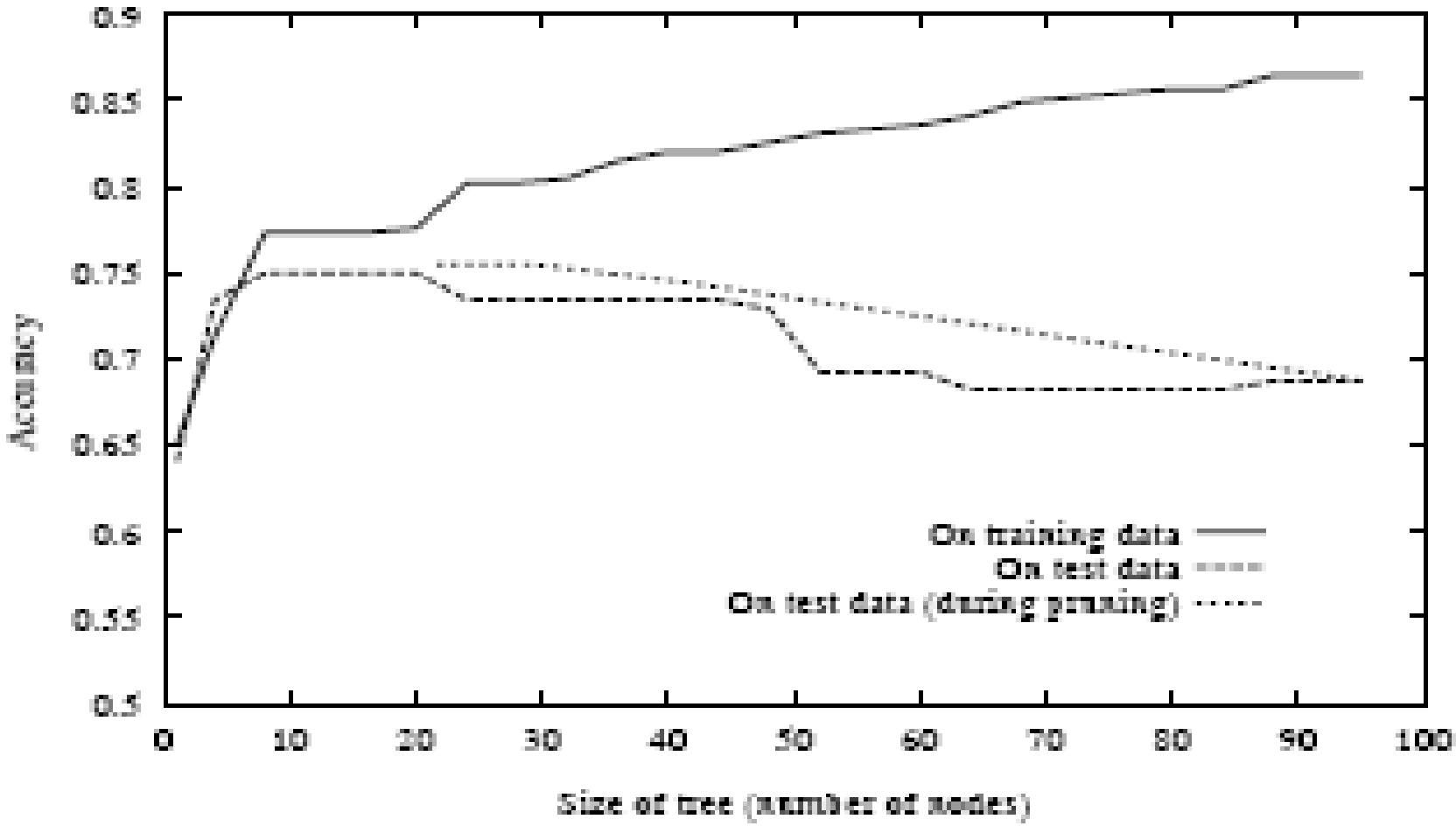
- 避免过度拟合的关键
  - 使用什么样的准则来确定最终正确树的规模
- 解决方法
  - 使用与训练样例截然不同的一套分离的样例，来评估通过后修剪方法从树上修剪节点的效用。
  - 使用所有可用数据进行训练，但进行统计测试来估计扩展（或修剪）一个特定的节点是否有可能改善在训练集合外的实例上的性能。
  - 使用一个明确的标准来衡量训练样例和决策树的复杂度，当这个编码的长度最小时停止树增长。

避免过度拟合数据（6）

- 方法评述
  - 第一种方法是最普通的，常被称为训练和验证集法。
  - 可用数据分成两个样例集合：
    - **训练集合，形成学习到的假设**
    - **验证集合，评估这个假设在后续数据上的精度**
  - 方法的动机：即使学习器可能会被训练集合误导，但验证集合不大可能表现出同样的随机波动
  - 验证集合应该足够大，以便它本身可提供具有统计意义的实例样本。
  - 常见的做法是，样例的三分之二作训练集合，三分之一作验证集合。

错误率降低修剪

- 将树上的每一个节点作为修剪的候选对象
- 修剪步骤
  - 删除以此节点为根的子树，使它成为叶结点
  - 把和该节点关联的训练样例的最常见分类赋给它
  - 反复修剪节点，每次总是选取那些删除后可以最大提高决策树在验证集合上的精度的节点
- 继续修剪，直到进一步的修剪是有害的为止
- 数据分成3个子集
  - 训练样例，形成决策树
  - 验证样例，修剪决策树
  - 测试样例，精度的无偏估计
- 如果有大量的数据可供使用，那么使用分离的数据集合来引导修剪

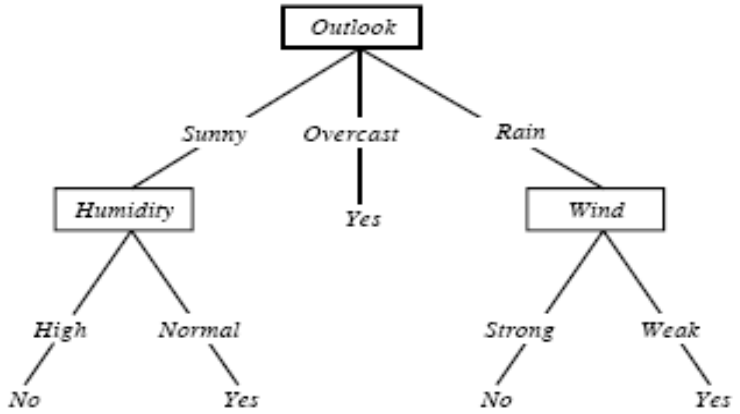


决策树学习中错误率降低的修剪效果

规则后修剪

- 从训练集合推导出决策树，增长决策树直到尽可能好地拟合训练数据，允许过度拟合发生
- 将决策树转化为等价的规则集合，方法是为从根节点到叶节点的每一条路径创建一条规则
- 通过删除不会导致估计精度降低的前件来修剪每一条规则
- 按照修剪过的规则的估计精度对它们进行排序，并按这样的顺序应用这些规则来分类后来的实例

规则后修剪（2）



- 例子
  - *if (outlook=sunny)∧(Humidity=High) then PlayTennis=No*
  - *if (outlook=sunny)∧(Humidity=Normal) then PlayTennis=Yes*
  - …
  - 考虑删除先行词(*outlook=sunny*)或(*Humidity=High*)
  - 选择使估计精度有最大提升的步骤
  - 考虑修剪第二个前件作为进一步的修剪步骤

规则后修剪（3）

- 规则精度估计方法
  - 使用与训练集不相交的验证集
  - 基于训练集合本身
    - **被C4.5使用，使用一种保守估计来弥补训练数据有利于当前规则的估计偏置**
    - **过程**
      - **先计算规则在它应用的训练样例上的精度**
      - **然后假定此估计精度为二项式分布，并计算它的标准差**
      - **对于一个给定的置信区间，采用下界估计作为规则性能的度量**
    - **评论**
      - **对于大的数据集，保守预测非常接近观察精度，随着数据集合的减小，离观察精度越来越远**
      - **不是统计有效（此概念第5章介绍），但是实践中发现有效**

规则后修剪（4）

- 把决策树转化成规则集的好处
  - 可以区分决策节点使用的不同上下文
  - 消除了根节点附近的属性测试和叶节点附近的属性测试的区别
  - 提高了可读性

合并连续值属性

- ID3被限制为取离散值的属性
  - 学习到的决策树要预测的目标属性必须是离散的
  - 树的决策节点的属性也必须是离散的
- 简单删除上面第2个限制的方法
  - 通过动态地定义新的离散值属性来实现，即先把连续值属性的值域分割为离散的区间集合

合并连续值属性（2）

- 例子，

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

- Temperature应该定义什么样的基于阈值的布尔属性

  选择产生最大信息增益的阈值

  - 按照连续属性排列样例，确定目标分类不同的相邻实例

    产生一组候选阈值，它们的值是相应的A值之间的中间值

    可以证明产生最大信息增益的c值位于这样的边界中（Fayyad1991）

  - 通过计算与每个候选阈值关联的信息增益评估这些候选值

- 方法的扩展

  - 连续的属性分割成多个区间，而不是单一阈值的两个空间

属性选择的其它度量标准

- 信息增益度量存在一个内在偏置，偏向具有较多值的属性
- 避免方法，其它度量，比如增益比率
- 增益比率通过加入一个被称作分裂信息的项来惩罚多值属性，分裂信息用来衡量属性分裂数据的广度和均匀性

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

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

- 分裂信息项阻碍选择值为均匀分布的属性
- 问题，当某个$S_i \approx S$。解决方法：采用一些启发式规则， 比如仅对增益高过平均值的属性应用增益比率测试

属性选择的其它度量标准（2）

- 基于距离的度量
  - 定义了数据划分间的一种距离尺度
  - 计算每个属性产生的划分与理想划分间的距离
  - 选择最接近完美划分的属性
  - Lopez de Mantaras定义了这个距离度量，证明了它不偏向有大量值的属性

此外

- Mingers实验，不同的属性选择度量对最终精度的影响小于后修剪的程度和方法的影响

缺少属性值的训练样例

- 例子，医学领域
- 经常需要根据此属性值已知的实例来估计这个缺少的属性值
- 为了评估属性A是否是决策节点n的最佳测试属性，要计算决策树在该节点的信息增益Gain(S,A)。假定<x,c(x)>是S中的一个训练样例，并且其属性A的值A(x)未知

缺少属性值的训练样例（2）

- 处理缺少属性值的策略
  - 一种策略是赋给它节点n的训练样例中该属性的最常见值
  - 另一种策略是赋给它节点n的被分类为c(x)的训练样例中该属性的最常见值
  - 更复杂的策略，为A的每个可能值赋予一个概率，而不是简单地将最常见的值赋给A(x)

处理不同代价的属性

- 实例的属性可能与代价相关（患者检查的项目）
- 优先选择尽可能使用低代价属性的决策树，仅当需要产生可靠的分类时才依赖高代价属性
- 通过引入一个代价项到属性选择度量中，可以使ID3算法考虑属性代价
- Tan和Schlimmer的通过机械爪抓取分辨物体：
  - 属性代价 通过定位或者操作声纳来获取属性值所需的秒数

C4.5改进的具体方面

- 用信息增益率来选择属性

  克服了用信息增益来选择属性时偏向选择值多的属性的不足。
- 可以处理连续数值型属性
- 采用了一种后剪枝方法
- 对于缺失值的处理

	选择测试属性的技术	连续属性的处理技术	剪枝方法	是否必须独立测试样本	可伸缩性	并行性	决策树的结构
ID3	信息增益	离散化	分类错误	是	差	差	多叉树
C4.5	信息增益率	预排序	分类错误	否	差	差	多叉树
CART	GINI 系数	预排序	分类错误	否	差	差	二叉树
SLIQ	GINI 系数	预排序	MDL	否	良好	良好	二叉树
SPRINT	GINI 系数	预排序	MDL	否	好	好	二叉树

小结和补充读物

- 决策树学习为概念学习和学习其他离散值的函数提供了一个实用的方法
- ID3算法
  - 贪婪算法
  - 从根向下推断决策树
  - 搜索完整的假设空间
  - 归纳偏置，较小的树
- 过度拟合问题
- ID3算法的扩展

参考

C4.5 Tutorial

http://www2.cs.uregina.ca/~dbd/cs831/notes/ml/dtrees/c4.5/tutorial.html

Building Classification Models ID3 and C4.5

http://www.cis.temple.edu/~ingargio/cis587/readings/id3-c45.html

History of Decision-Tree Research

- Hunt and colleagues use exhaustive search decision-tree methods (CLS) to model human concept learning in the 1960’s.
- In the late 70’s, Quinlan developed ID3 with the information gain heuristic to learn expert systems from examples.
- Simulataneously, Breiman and Friedman and colleagues develop CART (Classification and Regression Trees), similar to ID3.
- In the 1980’s a variety of improvements are introduced to handle noise, continuous features, missing features, and improved splitting criteria. Various expert-system development tools results.
- Quinlan’s updated decision-tree package (C4.5) released in 1993.
- Weka includes Java version of C4.5 called J48.

Weka J48 Trace 1

```
data> java weka.classifiers.trees.J48 -t figure.arff -T figure.arff -U -M 1
Options: -U -M 1
J48 unpruned tree
------------------
color = blue: negative (1.0)
color = red
|   shape = circle: positive (2.0)
|   shape = square: negative (1.0)
|   shape = triangle: positive (0.0)
color = green: positive (0.0)

Number of Leaves  :     5
Size of the tree :      7

Time taken to build model: 0.03 seconds
Time taken to test model on training data: 0 seconds
```

Weka J48 Trace 2

```
data> java weka.classifiers.trees.J48 -t figure3.arff -T figure3.arff -U -M 1
Options: -U -M 1
J48 unpruned tree
------------------
shape = circle
|   color = blue: negative (1.0)
|   color = red: positive (2.0)
|   color = green: positive (1.0)
shape = square: positive (0.0)
shape = triangle: negative (1.0)

Number of Leaves  :     5
Size of the tree :      7

Time taken to build model: 0.02 seconds
Time taken to test model on training data: 0 seconds
```

Weka J48 Trace 3

```
data> java weka.classifiers.trees.J48 -t contact-lenses.arff
J48 pruned tree
------------------
tear-prod-rate = reduced: none (12.0)
tear-prod-rate = normal
|   astigmatism = no: soft (6.0/1.0)
|   astigmatism = yes
|   |   spectacle-prescrip = myope: hard (3.0)
|   |   spectacle-prescrip = hypermetrope: none (3.0/1.0)

Number of Leaves  :     4
Size of the tree :      7

Time taken to build model: 0.03 seconds
Time taken to test model on training data: 0 seconds

=== Error on training data ===

Correctly Classified Instances          22               91.6667 %
Incorrectly Classified Instances        2                8.3333 %
Kappa statistic                          0.8447
Mean absolute error                      0.0833
Root mean squared error                  0.2041
Relative absolute error                 22.6257 %
Root relative squared error             48.1223 %
Total Number of Instances               24

=== Confusion Matrix ===

 a  b  c   <-- classified as
 5  0  0 |  a = soft
 0  3  1 |  b = hard
 1  0 14 |  c = none


=== Stratified cross-validation ===

Correctly Classified Instances          20               83.3333 %
Incorrectly Classified Instances        4                16.6667 %
Kappa statistic                          0.71
Mean absolute error                      0.15
Root mean squared error                  0.3249
Relative absolute error                 39.7059 %
Root relative squared error             74.3898 %
Total Number of Instances               24

=== Confusion Matrix ===

 a  b  c   <-- classified as
 5  0  0 |  a = soft
 0  3  1 |  b = hard
 1  2 12 |  c = none
```