模式识别与机器学习

集成学习（2）

12.4 决策树

12.5 随机森林

12.4 决策树　　决策树基本概念

关于分类问题

X　　y

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

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(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）。

本文中“属性”和“特征”可互换

分类（Classification）任务就是通过学习获得目标函数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

决策树

决策树是一种典型的分类方法，首先对数据进行处理，利用归纳算法生成可读的规则和决策树，然后使用决策对新数据进行分析。

本质上决策树是通过一系列规则对数据进行分类的过程。

决策树的构建过程就是选取特征和确定决策规则的过程。

决策树的优点

1、推理过程容易理解，决策推理过程可以表示成If Then形式；

2、推理过程完全依赖于属性变量的取值特点；

3、可自动忽略对目标变量没有贡献的属性变量，也为判断属性变量的重要性、减少变量的数目提供参考。

决策树算法

与决策树相关的重要算法

CLS, ID3，C4.5，CART

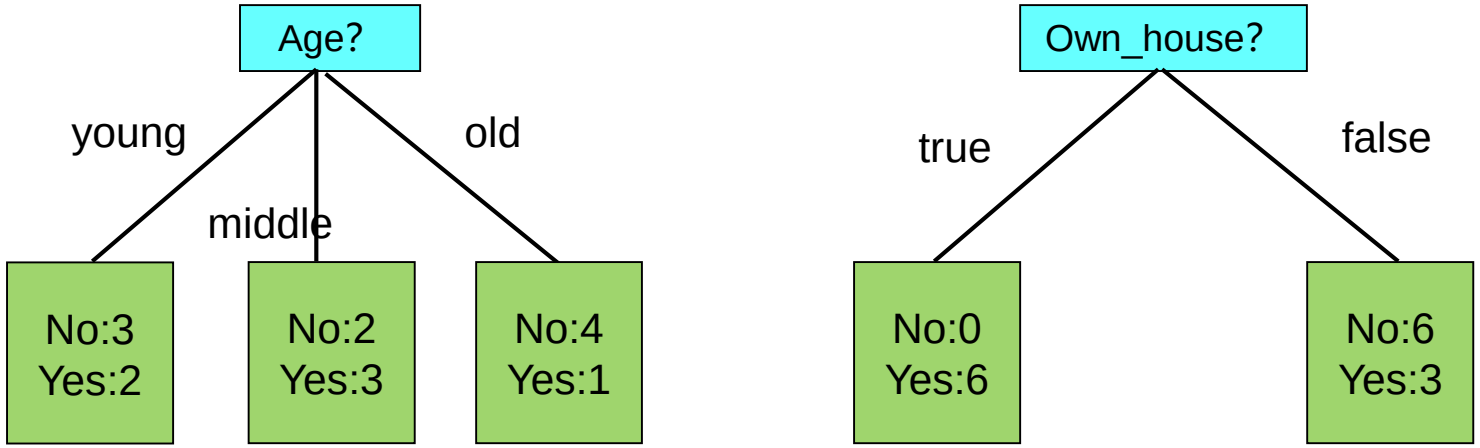
1、CLS学习系统 (Hunt,Marin和Stone ,1966年)
2、ID3算法 (J.R. Quinlan,1986年)
3、ID4算法 (Schlimmer 和Fisher ,1986年)
4、ID5算法 (Utgoff,1988年)
5、C4.5算法 (J.R. Quinlan,1993年)
6、CART算法 (Breiman，1984)

决策树算法

例：申请贷款的数据集合

ID	Age	Has-job	Own_house	Credit_rating	Class
1	Young	False	False	Fair	No
2	Young	False	False	Good	No
3	Young	True	False	Good	Yes
4	Young	True	True	Fair	Yes
5	Young	False	False	Fair	No
6	Middle	False	False	Fair	No
7	Middle	False	False	Good	No
8	Middle	True	True	Good	Yes
9	Middle	False	True	Excellent	Yes
10	Middle	False	True	Excellent	Yes
11	Old	False	True	Excellent	Yes
12	Old	False	True	Good	Yes
13	Old	True	False	Good	Yes
14	Old	True	False	Excellent	Yes
15	Old	False	False	fair	no

决策树算法



（a）　上例可能的两种根节点　（b）

若采用Age或Own_house作为根节点。根节点的可能取值构成了根节点分支。
对每个分支，列出该分支上每个类（Yes或No）的训练数据的数目。

从少数服从多数分类预测的观点来看，(b) 比(a)好，(b)比(a)犯错的可能性要小。
选择(b)时，当Own_house=true时，每个样例都分配到yes类中。在Own_house=false时，如果将它们都归类到No类中，则整个分类会有三个错误。
选择(a)时，如果按照少数服从多数的方法，则会产生5个错误的分类。

说明，节点（特征）的选择对结果是有影响的。

决策树的构建过程就是选取特征和确定决策规则的过程。

决策树算法

决策树的用途

计数	年龄	收入	学生	信誉	归类：买计算机？
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

假定公司收集了左表数据，那么对于任意给定的客人（测试样例），你能帮助公司将这位客人归类吗？

即：你能预测这位客人是属于“买”计算机的那一类，还是属于“不买”计算机的那一类？

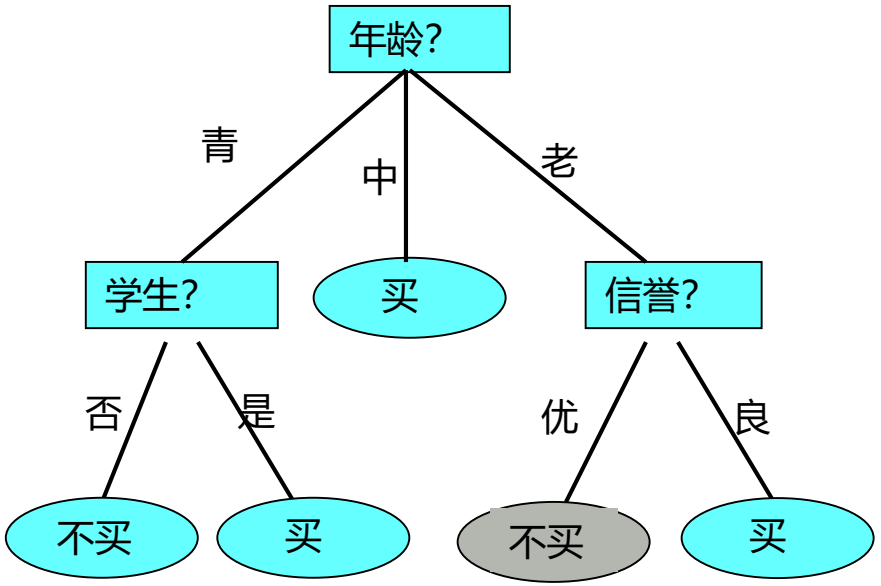
又：你需要多少有关这位客人的信息才能回答这个问题？

决策树算法

决策树的用途

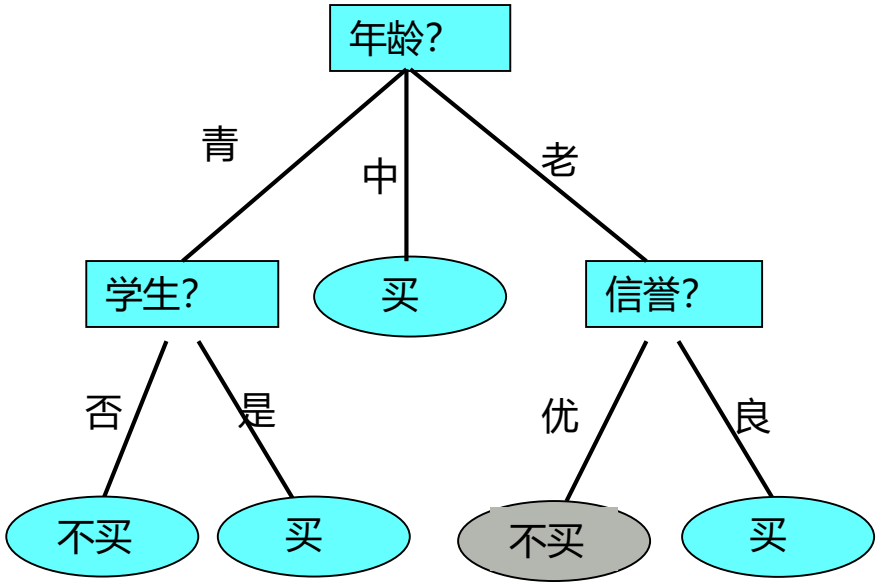
计数	年龄	收入	学生	信誉	归类：买计算机？
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

谁在买计算机?



决策树算法

决策树的表示



决策树的基本组成部分：决策结点、分支和叶子。

- 决策树中最上面的结点称为根结点。是整个决策树的开始。
- 每个分支是一个新的决策结点，或者是树的叶子。
- 每个决策结点代表一个问题或者决策，通常对应待分类对象的属性。
- 每个叶结点代表一种可能的分类结果。

决策树的构建过程就是选取特征和确定决策规则的过程。

利用决策树进行分类的过程，即是利用若干变量来判断属性类别：

沿决策树从上到下的遍历过程中，每个结点上都有一个测试。对每个结点上问题的不同测试输出导致不同的分枝，最后会到达一个叶子结点。

训练集的熵H(S) -Entropy of a training set

ID3算法的基础：香农信息论中的熵的定义

$P(c_i)$是S属于c_i类的概率（“熵”描述了用来预测的信息位数），训练集的熵记作H(S):

$$H(S)=-\sum_i P(c_i)\log_2 P(c_i)$$

熵 / 熵不纯度

“熵”是随机变量不确定性的一种度量方法。熵值越大，随机变量的不确定性越大

- S是训练样本中的实例集
- 对某个节点上的样本，熵是测量节点上的特征对样本分类的“不纯度”的一种方法。
  - 当结点很纯时，希望其度量值应为0
  - 当不纯性最大时（比如所有类都有同样的可能），其度量值应最大
  - 度量应该服从多级特性，这样决策树才能分阶段建立起来

信息增益Information Gain

InfoGain (S | A) = H(S) – H(S | A)
也记作InfoGain (S, A)

S：某个节点的样本；A：这个节点的某个属性

信息增益指前后信息的差值。

在决策树分类中，指决策树在进行属性划分前、后的信息差值

- 信息增益InfoGain(S, A)：表示得知属性A的信息，使得类的信息的不确定性减少(即熵减少）的程度。

简记：Gain(S, A) = $InfoGain(S,A) = H(S) - H(S \mid A)$

$$= H(S) - \sum_{a} p(A=a) H(S \mid A=a)$$

$$= H(S) - \sum_{a \in Values(A)} \frac{|S_a|}{|S|} H(S_a)$$

|S|表示划分前的实例个数，$|S_a|$表示划分后某个分支的实例个数， S_a是S的一个子集（S中属性A取值为a的数据子集）。

- 信息增益大的特征具有更强的分类能力，可以选取信息增益最大的特征属性作为划分的最优特征属性A。

InfoGain(S，A) 越大，说明选择的测试属性对分类提供的信息越多

决策树算法

$$H(S)=-\sum_i P(c_i)\log_2 P(c_i)$$

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

第1步：计算不考虑任何特征时决策属性的熵

共1024个实例：641个正例（买），
383个负例（不买）

当前数据S（原始状态）的信息量（熵）：

决策属性“买计算机？”。
该属性将数据分两类：买/不买

S1(买)=641
S2（不买）= 383
S=S1+S2=1024

P2=383/1024=0.3740

H(S) = I([S1,S2]) = I([641,383])
= E(P1,P2)
= $-P1\log_2 P1-P2\log_2 P2$
=0.9537

H(S=买计算机？)
$= -\sum_i P_2(c_i)\log_2 P(c_i)$
$= -P(\text{买})\log_2 P(\text{买}) - P(\text{不买})\log_2 P(\text{不买})$

决策树算法

$$H(S|A)=\sum_{a\in Values(A)}\frac{|S_a|}{|S|}H(S_a)$$

计数	年龄	收入	学生	信誉	归类：买计算机？
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

计算利用不同属性进行划分时的熵

条件属性共有4个。分别是年龄、收入、学生、信誉。

分别计算不同属性划分时的信息增益。

决策树算法

$$H(S|A)=\sum_{a\in Values(A)}\frac{|S_a|}{|S|}H(S_a)$$

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

计算利用不同属性进行划分时的熵（续）

第2-1步计算利用年龄划分时的熵

年龄共分三个组： 青年、中年、老年

年龄 取值 为“青年“时：

青年买与不买比例为128/256
S1(买)=128
S2（不买）= 256
S=S1+S2=384

P1=128/384,　　P2=256/384

$H(S_a$=青年买计算机?)=
I([S1,S2]) = I([128,256])
= -(P1Log$_2$P1+P2Log$_2$P2)
= 0.9183

$H(S_a$=青年买计算机?)

决策树算法

$$H(S|A)=\sum_{a\in Values(A)}\frac{|S_a|}{|S|}H(S_a)$$

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

计算利用不同属性进行划分时的熵（续）

第2-2步计算利用年龄划分时的熵

年龄共分三个组： 青年、中年、老年

年龄 取值 为“中年“时：

中年买与不买比例为256/0
S1(买)=256
S2（不买）= 0
S=S1+S2=256

P1=256/256
P2=0/256

$H(S_a$=中年买计算机？）=
I([S1,S2]) = I([256，0])
=-(P1Log_2P1+P2Log_2P2)
=0

$H(S_a$=中年买计算机？）

决策树算法

$$H(S|A)=\sum_{a\in Values(A)}\frac{|S_a|}{|S|}H(S_a)$$

计数	年龄	收入	学生	信誉	归类：买计算机？
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

计算利用不同属性进行划分时的熵（续）

第2-3步计算年龄划分时的熵

年龄共分三个组： 青年、中年、老年

年龄 取值 为“老年”时：

老年买与不买比例为257/127
S1(买) = 257
S2（不买）= 127
S = S1 + S2= 384

P1=257/384
P2=127/384

$H(S_a$=老年买计算机？）=
I([S1,S2]) = I([257，127])
= -(P1Log$_2$P1+P2Log$_2$P2)
= 0.9157

$H(S_a$=老年买计算机？）

决策树算法

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

计算利用不同属性进行划分时的熵（续）

第2-4步计算年龄划分时的熵

年龄共分三个组： 青年、中年、老年
各占比例
青年组 384/1024=0.375
中年组 256/1024=0.25
老年组 384/1024=0.375

$$H(S|A)=\sum_{a\in Values(A)}\frac{|S_a|}{|S|}H(S_a)$$

计算年龄划分时的平均信息期望

H（S | 年龄）= E（年龄）
= 0.375*0.9183+ 0.25*0+ 0.375*0.9157
=0.6877

计算年龄划分时的信息增益

G（年龄信息增益）= InfoGain（S,年龄）
= H(S)-H(S|年龄)
=0.9537-0.6877
=0.2660 (1)

决策树算法

计数	年龄	收入	学生	信誉	归类：买计算机？
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

计算利用不同属性进行划分时的熵（续）

第3步计算收入划分时的熵

按收入共分三个组： 高、中、低

E（收入）= 0.9361

收入信息增益 =0.9537-0.9361

=0.0176 (2)

第4步计算学生划分时的熵

按学生共分二个组： 学生、非学生

E（学生）=0.7811

学生信息增益=0.9537-0.7811

=0.1726 (3)

第5步计算信誉的熵

按信誉分二个组： 良好，优秀

E（信誉）= 0.9048

信誉信息增益=0.9537-0.9048

=0.0453 (4)

决策树算法

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买

第6步 计算选择节点

年龄信息增益 =0.9537-0.6877
=0.2660 (1)

收入信息增益 =0.9537-0.9361
=0.0176 (2)

学生信息增益 = 0.9537-0.7811
=0.1726 (3)

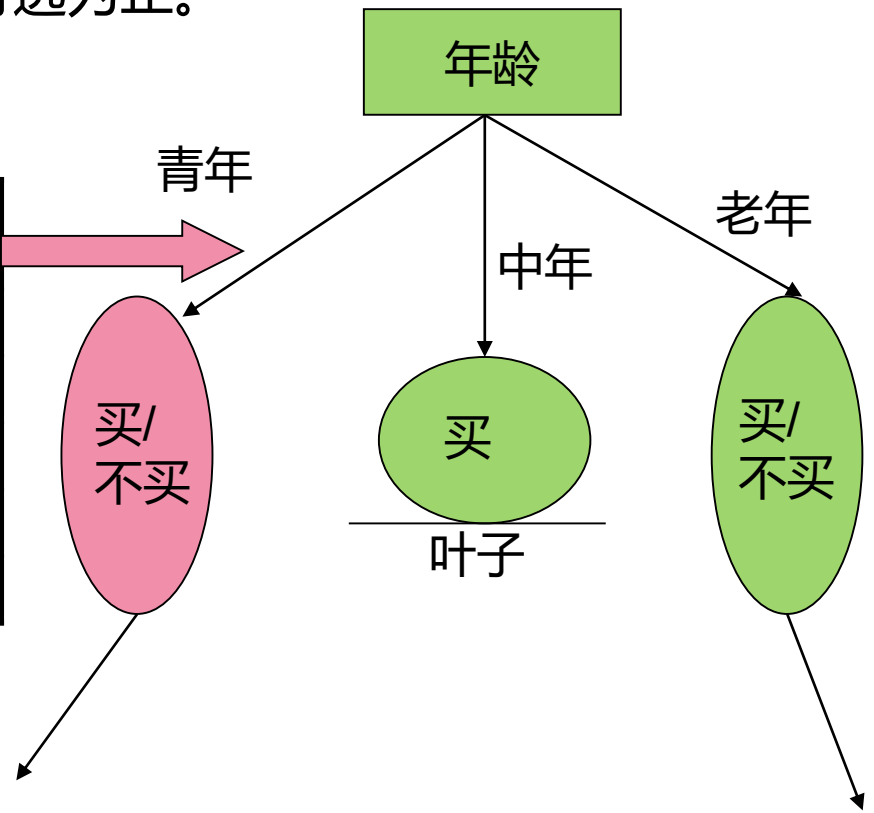
信誉信息增益 = 0.9537-0.9048
=0.0453 (4)

最大信息增益选择特征属性年龄作为节点（选择划分节点的属性）

决策树算法

选择信息增益最大的特征“年龄”作为节点，该特征的不同取值建立子节点（将训练数据集分为三个子集）；
对各子节点递归的进行上述节点选取的操作，构建决策树；
直到所有特征的信息增益都很小或没有特征可选为止。
最后得到一个决策树。

计数	年龄	收入	学生	信誉	归类：买计算机？
64	青	高	否	良	不买
64	青	高	否	优	不买
128	青	中	否	良	不买
64	青	低	是	良	买
64	青	中	是	优	买



决策树算法（例续）

青年买与不买比例为128/256

S1(买)=128　　S2（不买）= 256 、 S=S1+S2=384

买：P1=128/384　　不买：P2=256/384

H(S) = I([S1,S2])= I([128,256]) = -(P1Log$_2$P1+P2Log$_2$P2)=0.9183

$$Gain(S,A) = H(S) - H(S \mid A)$$

$$= H(S) - \sum_{a \in Values(A)} \frac{|S_a|}{|S|} H(S_a)$$

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	青	中	否	良	不买
64	青	低	是	良	买
64	青	中	是	优	买

S

如果选择收入作为下一节点：分高、中、低

高　　H(S$_a$=收入高 买计算机?) = I([0,128])=0

比例：128/384=0.3333

中　　H(S$_a$=收入中 买计算机?) = I([64,128])=0.9183

比例：192/384=0.5

低　　H(S$_a$=收入低 买计算机?) = I([64,0])=0

比例：64/384=0.1667

计算收入划分时的平均信息期望（加权总和）：

H（S | 收入）= E（收入） = 0.3333 * 0 + 0.5 * 0.9183 + 0.1667 * 0 = 0.4592

InfoGain(收入) = I(128, 256) - E(收入)= 0.9183 – 0.4592 = 0.4591

注意

决策树算法（例续）

InfoGain(收入) = I(128, 256) - E(收入)= 0.9183 – 0.4592 = 0.4591

计数	年龄	收入	学生	信誉	归类：买计算机?
64	青	高	否	良	不买
64	青	高	否	优	不买
128	青	中	否	良	不买
64	青	低	是	良	买
64	青	中	是	优	买

S

如果选择学生作为下一节点，分是、否

是 I(128,0)=0
比例:128/384 = 0.3333

否 I(0,256)=0
比例: 256/384=0.667

平均信息期望（加权总和）：

E(学生) = 0.3333 * 0 + 0.667 * 0 = 0

infoGain(学生) = I(128, 256) - E(学生)=0.9183 – 0 = 0.9183

此值最大，其他可不用计算

如果选择信誉作为下一节点，分优、良

优：I(64,64)= -2×（0.5Log$_2$0.5）= 1， 比例:128/384 = 0.3333

良：I(64,192)= 0.8113， 比例: 256/384= 0.667

其中，$\text{info}([64,192]) = -\frac{64}{256}\log_2\left(\frac{64}{256}\right) - \frac{192}{256}\log_2\left(\frac{192}{256}\right) = 0.8113$

平均信息期望（加权总和）：

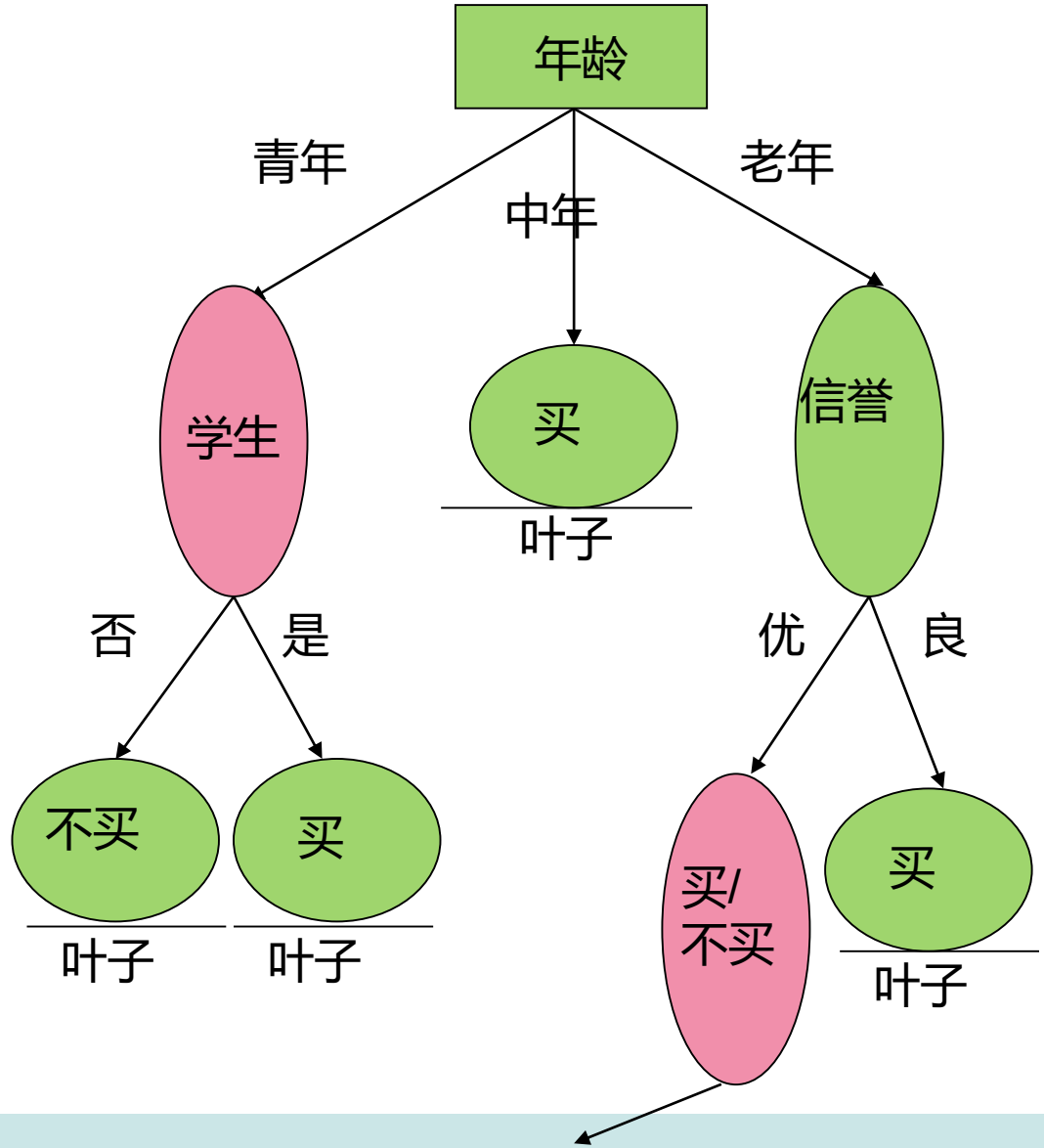
E(信誉) = 0.3333 * 1 + 0.667 * 0.8113 = 0.8744

infoGain(信誉) = I(128, 256) - E(信誉)= 0.9183 - 0.8744 = 0.0439

最大信息增益选择特征学生作为青年下的下一划分节点

决策树算法

计数	年龄	收入	学生	信誉	归类：买计算机？
64	青	高	否	良	不买
64	青	高	否	优	不买
128	中	高	否	良	买
60	老	中	否	良	买
64	老	低	是	良	买
64	老	低	是	优	不买
64	中	低	是	优	买
128	青	中	否	良	不买
64	青	低	是	良	买
132	老	中	是	良	买
64	青	中	是	优	买
32	中	中	否	优	买
32	中	高	是	良	买
63	老	中	否	优	不买
1	老	中	否	优	买



决策树建立步骤（ID3）

1)决定分类属性

2)对目前的数据表，建立一个节点N。

3)如果数据表中的数据都属于同一类，N就是树叶，在树叶上标上所属的那一类。

4)如果数据表中没有其他属性可以考虑，N也是树叶，按照少数服从多数的原则在树叶上标上所属类别。

5)否则，根据最大信息增益Gain值（或最小平均信息期望值E）选出一个最佳属性作为节点N的测试属性。

6)节点属性选定以后，对于该属性的每一个值：

- 从N生成一个分支， 并将数据表中与该分支有关的数据收集形成分支节点的数据表，在表中删除节点属性那一栏。
- 如果分支数据表非空，则运用以上算法从该节点建立子树。

决策树算法

ID3算法小结

ID3算法是一种经典的决策树学习算法，由Quinlan于1979年提出。

基本思想：以信息熵为度量，用于决策树节点的属性选择，每次优先选取信息量最多的属性，亦即能使熵值变为最小的属性，以构造一颗熵值下降最快的决策树，到叶子节点处的熵值为0。此时，每个叶子节点对应的实例集中的实例属于同一类。

ID3采用信息增益做度量时存在的问题:

以信息增益作为划分训练数据集的特征更偏向于选择取值较多的特征属性。

C4.5算法-信息增益比替代信息增益

使用信息增益比可对此问题进行校正（特征属性选取的另一准则）

Common algorithms

- ID3　　(利用信息增益 选取 特征属性)
- C4.5　(利用信息增益比 选取 特征属性)
- CART　(classification and regression tree)

决策树算法

- An extension of ID3:
- C4.5决策树在ID3决策树的基础之上稍作改进，继承了ID3算法的优点，同时在以下方面做了改进：

1)用信息增益率来选择属性，克服了用信息增益选择属性时偏向选择取值多的属性的不足

2)在树构造过程中进行剪枝

3)能够完成对连续属性的离散化处理

4)能够对不完整数据进行处理　　(查资料了解)

决策树算法

- 信息增益比/增益率度量是用信息增益度量Gain(S，A)和分裂信息度量SplitInfo(S,A)来共同定义的，定义如下：

$$GainRatio(S,A)=\frac{Gain(S,A)}{SplitInfo(S,A)}=\frac{Gain(S,A)}{H_A(S)}=\frac{Gain(S,A)}{-\sum_{a\in Values(A)}\frac{|S_a|}{|S|}\log_2\frac{|S_a|}{|S|}}$$

其中，分裂信息度量被定义为(分裂信息用来衡量属性分裂数据的广度和均匀)

$$Gain(S,A)=H(S)-H(S\mid A)=H(S)-\sum_{a\in Values(A)}\frac{|S_a|}{|S|}H(S_a)$$

这里 S_a是S的一个子集，其中属性A取值为a。

属性A做划分时的平均信息期望

$$H(S)=-\sum_i P(c_i)\log_2 P(c_i)$$

决策树算法

C4. 5

优点：产生的分类规则易于理解，准确率较高。

缺点：在构造树的过程中，需要对数据集进行多次的顺序扫描和排序，因而导致算法的低效。此外，C4.5适合于能够驻留于内存的数据集，当训练集大得无法在内存容纳时程序无法运行。

- C4.5并不是一个算法，而是一组算法—C4.5，非剪枝C4.5和C4.5规则。下图中的算法将给出C4.5的基本工作流程：

C4.5算法基本流程

Algorithm 1.1 C4.5(D)

Input: an attribute-valued dataset D

1: Tree = {}
2: **if** D is "pure" OR other stopping criteria met **then**
3: terminate
4: **end if**
5: **for all** attribute $a \in D$ **do**
6: Compute information-theoretic criteria if we split on a
7: **end for**
8: a_{best} = Best attribute according to above computed criteria
9: Tree = Create a decision node that tests a_{best} in the root
10: D_v = Induced sub-datasets from D based on a_{best}
11: **for all** D_v **do**
12: $\text{Tree}_v = \text{C4.5}(D_v)$
13: Attach Tree_v to the corresponding branch of Tree
14: **end for**
15: **return** Tree

如果D是"纯粹"的或满足其他停止条件则终止

如果按照a劈开,计算信息衡量标准

决策树算法

C4.5算法举例

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	85	85	Weak	No
D2	Sunny	80	90	Strong	No
D3	Overcast	83	78	Weak	Yes
D4	Rain	70	96	Weak	Yes
D5	Rain	68	80	Weak	Yes
D6	Rain	65	70	Strong	No
D7	Overcast	64	65	Strong	Yes
D8	Sunny	72	95	Weak	No
D9	Sunny	69	70	Weak	Yes
D10	Rain	75	80	Weak	Yes
D11	Sunny	75	70	Strong	Yes
D12	Overcast	72	90	Strong	Yes
D13	Overcast	81	75	Weak	Yes
D14	Rain	71	80	Strong	No

连续值　　连续值

共14个实例：9个正例（yes），5个负例（no）

当前数据S（原始状态）的信息量用熵来计算：

$$H(S)=\text{info}(play?)=\text{info}([9,5])$$

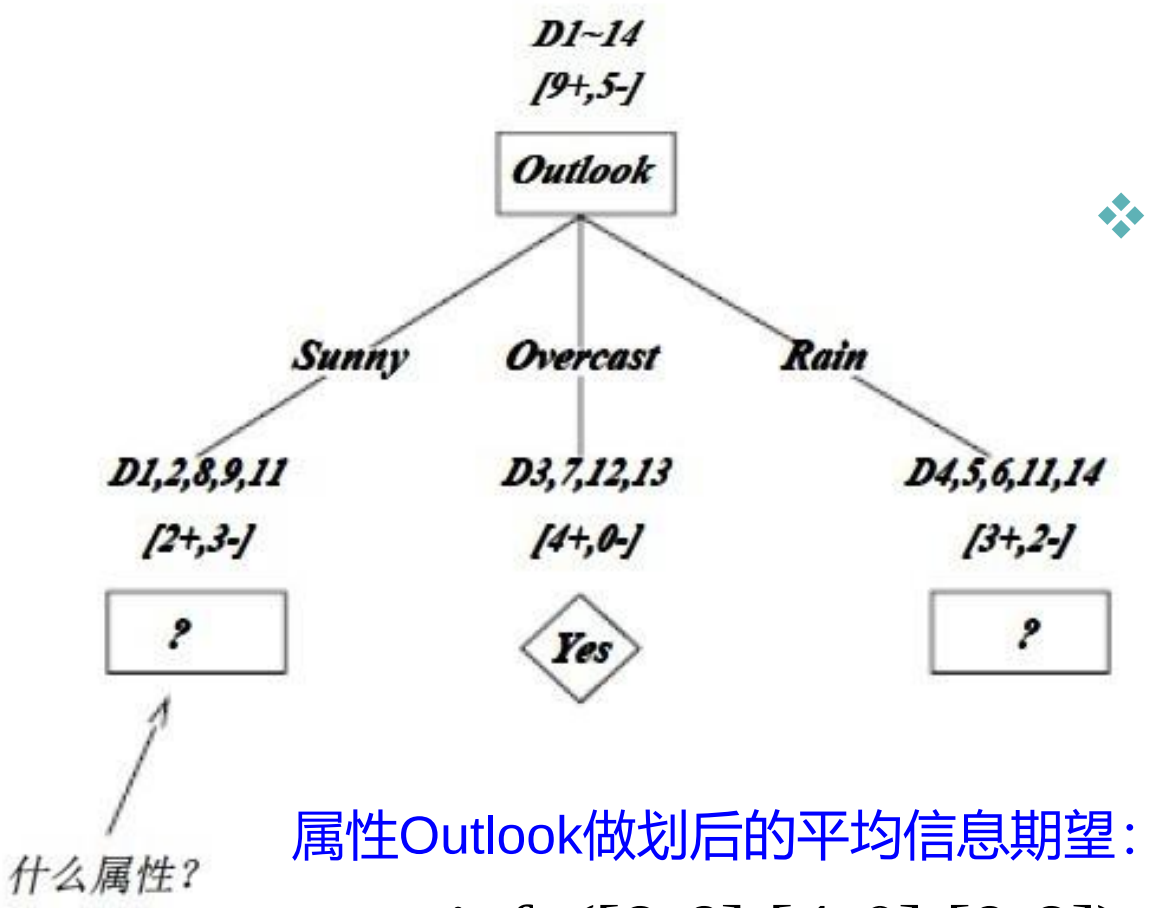
$$=E(\frac{9}{14},\frac{5}{14})=entropy(\frac{9}{14},\frac{5}{14})$$

$$=-\frac{9}{14}\log_2\left(\frac{9}{14}\right)-\frac{5}{14}\log_2\left(\frac{5}{14}\right)=0.940$$

- 如何得到信息增益？

续上：选取Outlook属性来划分，见下图

$$InfoGain(S,A) = H(S) - H(S \mid A)$$
$$= H(S) - \sum_{a} p(A=a) H(S \mid A=a)$$
$$= H(S) - \sum_{a \in Values(A)} \frac{|S_a|}{|S|} H(S_a)$$



- 在用Outlook属性划分后，可以看到数据被分成三份，则各分支的信息（熵）计算如下：

$$\text{info}([2,3]) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) = 0.971 bits$$

$$\text{info}([4,0]) = -\frac{4}{4}\log_2\left(\frac{4}{4}\right) - \frac{0}{4}\log_2\left(\frac{0}{4}\right) = 0 bits$$

$$\text{info}([3,2]) = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) = 0.971 bits$$

属性Outlook做划后的平均信息期望：

$$\text{info}([2,3],[4,0],[3,2]) = \frac{5}{14}\times 0.971 + \frac{4}{14}\times 0 + \frac{5}{14}\times 0.971 = 0.693 bits$$

InfoGain (S, **Outlook**)=0.940-(5/14)*0.971-(4/14)*0-(5/14)*0.971 = 0.940-0.693 =0.247

决策树算法

- 计算*Outlook*的*GainRatio*

  Sunny:5，*Overcast*:4，*Rain*:5

$$SplitInformation(S,Outlook)=-\sum_{i=1}^{3}\frac{|S_i|}{|S|}\log_2\frac{|S_i|}{|S|}$$

$$=-(\frac{5}{14}\log_2\frac{5}{14}+\frac{4}{14}\log_2\frac{4}{14}+\frac{5}{14}\log_2\frac{5}{14})\doteq 1.577$$

属性*Outlook*的熵

$$GainRatio=\text{Info}Gain(S,Outlook)\ /\ SplitInformation(S,Outlook)=0.247/1.577\doteq 0.156$$

0.94-0.693

- 计算*Temperature*的*GainRatio*

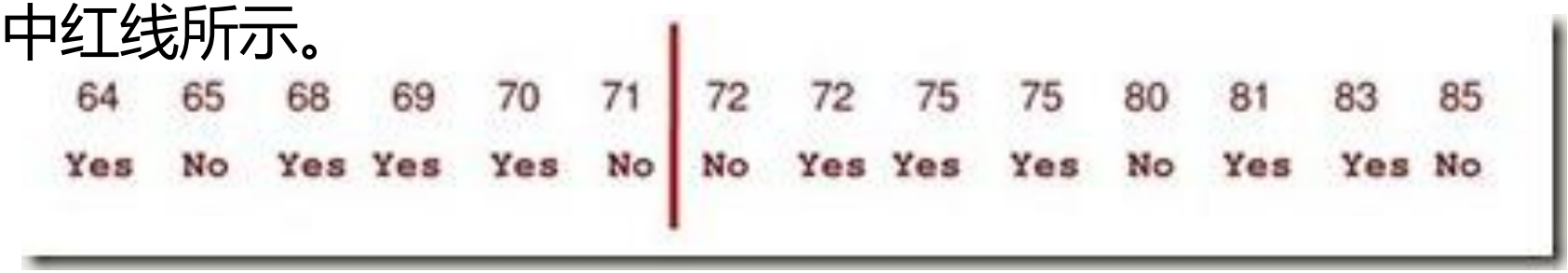
64	65	68	69	70	71	72	72	75	75	80	81	83	85
Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	Yes	No

那么可以看到有13个可能的候选阈值点，比如middle[64,65], middle[65,68]….,middle[83,85]。那么最优的阈值该选多少呢?

对连续属性的离散化处理

决策树算法

以middle[71,72]为例，如下图中红线所示。



E.g.: temperature<71.5: yes/4,no/2 ; temperature>=71.5: yes/5,no/3

Info([4,2],[5,3])=6/14*Info([4,2])+8/14Info([5,3])=0.939 bits，此时平均信息期望为0.939。

计算过程需对每个候选分割阈值进行增益或熵的计算，选择信息增益最大的分裂点，即最优的阈值，需要计算N-1次增益或熵（上例需要13次计算）。

能不能减少计算量?如何减少计算量?

->对连续属性先排序,只有在决策属性发生改变的地方才需要试探性的split.

64 | 65 | 68 69 70 | 71 72 | 72 75 75 | 80 | 81 83 | 85

Yes | No | Yes Yes Yes | No No | Yes Yes Yes | No | Yes Yes | No

本来有13种离散化的情况，现在只需计算7种。

对连续属性的离散化处理

决策树算法

增益率　　增益

$$GainRatio(S,A)=\frac{Gain(S,A)}{SplitInfo(S,A)}=\frac{Gain(S,A)}{H_A(S)}=\frac{Gain(S,A)}{-\sum_{a\in Values(A)}\frac{|S_a|}{|S|}\log_2\frac{|S_a|}{|S|}}=\frac{H(S)-\sum_{a\in Values(A)}\frac{|S_a|}{|S|}H(S_a)}{-\sum_{a\in Values(A)}\frac{|S_a|}{|S|}\log_2\frac{|S_a|}{|S|}}$$

- 如果利用增益率来选择连续值属性的分界点，会导致一些副作用。
- 分界点将样本分成两个部分，这两部分的样本个数之比也会影响增益率。

  根据增益率公式，当分界点把样本分成数量相等的两个子集时（此时的分界点为等分分界点），增益率的抑制会被最大化，因此等分分界点被过分抑制了。
- 子集样本个数能够影响分界点，显然不合理。因此在决定分界点时还是采用增益这个指标，而选择属性的时候才使用增益率这个指标。这个改进能够很好抑制连续值属性的倾向。当然还有其它方法也可以抑制这种倾向，比如MDL（最小描述长度）。

对连续属性的离散化处理

决策树算法

最小描述长度（MDL）基本概念

MDL

自习

最小描述长度准则

解释一组数据的最好理论，应该使得下面两项之和最小：

- 描述理论所需要的比特长度；
- 在理论的协助下，对数据编码所需要的比特长度。

（最小描述长度也称为给定数据的随机复杂性）

决策树中的最小描述长度准则

我们应该寻求这样一种合理且较小的树，使得训练样本的大多数数据符合这棵树，把样本中不符合的数据作为例外编码，使得下面两项最小：

- 编码决策树所需的比特，它代表了猜想；
- 编码例外实例所需要的比特。

最小描述长度（MDL）基本概念（续）

决策树中的最小描述长度（MDL）准则（续）

自习

在决策树学习中，最小化决策树编码对应于简化决策树，而最小化编码例外对应于增加决策树的正确率。

MDL决策树编码示例：

属性	Outlook	Temperature	Humidity	Windy	类
1	Overcast	Hot	High	Not	N
2	Sunny	Mild	Normal	Very	P
⋮	⋮	⋮	⋮	⋮	⋮
8	**Rain**	**Hot**	**High**	**Not**	**P**
⋮	⋮	⋮	⋮	⋮	⋮
24	Rain	Mild	High	Very	N
24条记录	3个属性值	3个属性值	2个属性值	3个属性值	2个类

表1

属性	outlook	temperature	humidity	windy	类
1	overcast	hot	high	not	no
2	overcast	hot	high	very	no
3	overcast	hot	high	medium	no
4	sunny	hot	high	not	yes
5	sunny	hot	high	medium	yes
6	rain	mild	high	not	no
7	rain	mild	high	medium	no
8	rain	hot	normal	not	yes
9	rain	cool	normal	medium	no
10	rain	hot	normal	very	no
11	sunny	cool	normal	very	yes
12	sunny	cool	normal	medium	yes
13	overcast	mild	high	not	no
14	overcast	mild	high	medium	no
15	overcast	cool	normal	not	yes
16	overcast	cool	normal	medium	yes
17	rain	mild	normal	not	no
18	rain	mild	normal	medium	no
19	overcast	mild	normal	medium	yes
20	overcast	mild	normal	very	yes
21	sunny	mild	high	very	yes
22	sunny	mild	high	medium	yes
23	sunny	hot	normal	not	yes
24	rain	mild	high	very	no

自习

由表1生成的决策树

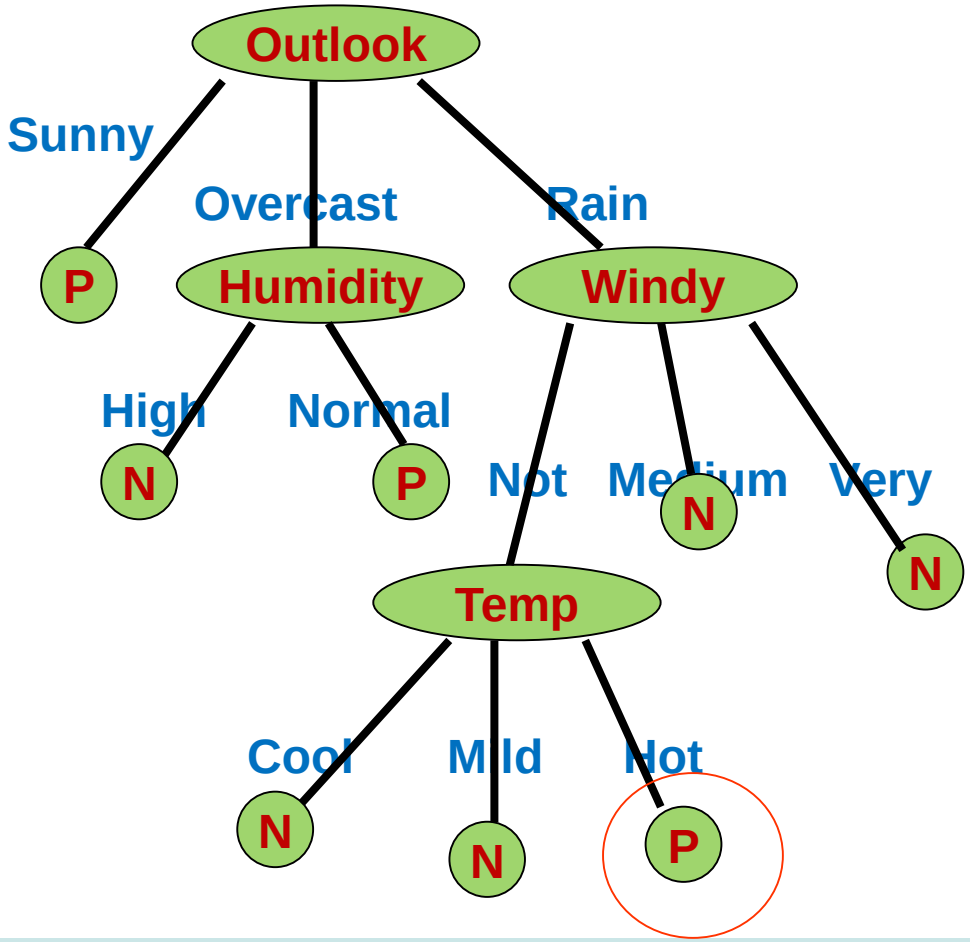


图1

将记录8的类标签改为N的决策树

自习

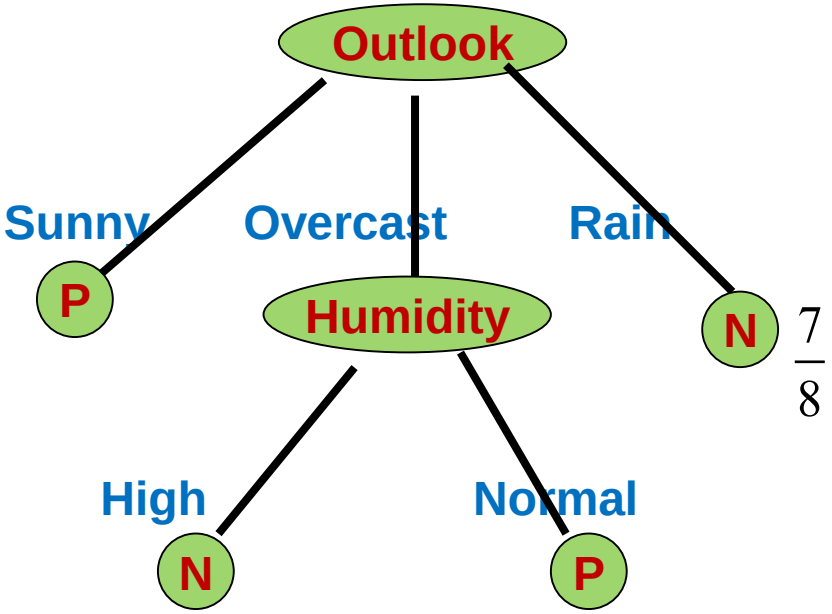


图2

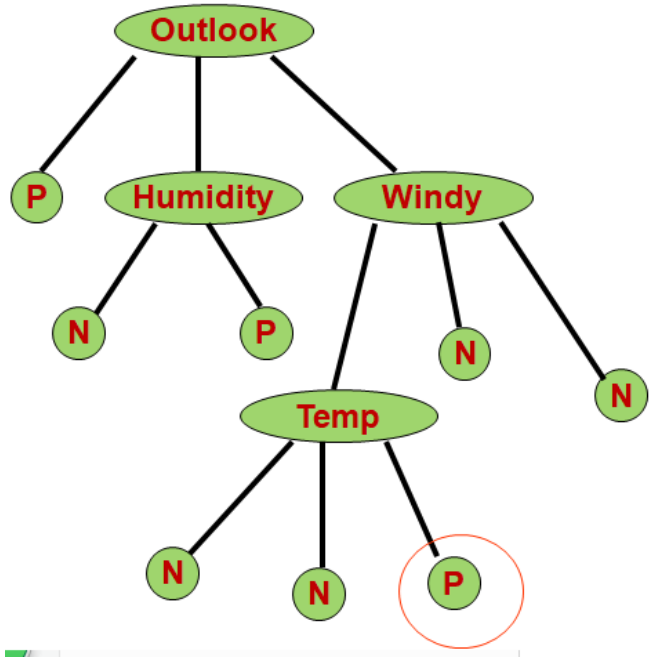
由上图可以看出，仅仅一条噪声数据导致了决策树很大不同。

- 我们对图1 的决策树**直接编码**。如果采用深度优先遍历对其进行存储，可以表示为：

自习

1 Outlook 0 P 1 Humidity 0 N 0 P 1 Windy
0 N 0 N 1 Temperature 0 N 0 N 0 P

用1表示下一个节点是内节点，然后记下节点的相应属性；
用0表示下一个节点是叶节点，并用**N**或**P**记下节点的类属性。



编码时除了四个属性的编码所需的比特数，还需要20个比特数，每个比特数表示1、0，或P、N

表示根节点上的Outlook需要两个比特(有四种可能)；表示下一层的属性Humidity需要$\log_2 3$个比特(Outlook已被选出)；表示属性Windy需要1个比特；表示Temperature需要1个比特(其实一个都不需要)。这样总共需要25.585个比特。

$20+2+\log_2 3+1+1=25.585$

自习

♦假设经过剪枝后的决策树如图2所示，此时最右边的叶节点的 8 条数据中有1条是被错误分类的。按照**“树+例外”编码**时，首先要对图2 编码，需要13.585个比特。$10+2+\log_2 3=13.585$

1 Outlook 0 P 1 Humidity 0 N 0 P 0 N

接下来对仅有的一个例外编码，指定它在54 (3*3*2*3) 种可能性中的位置，这需要$\log_2 54=5.75$个比特。因此总共需要19.335比特。明显可以发现该编码方案优于对整棵树编码。

总结：可选择最小描述长度准则用于树生成时的测试属性（如同Gini系数、信息熵），使得每次树扩展时总的“树+例外”描述长度(total description length, TDL) 增加最小。

待树生成后，进行自下而上的剪枝，直到继续剪枝不能减少TDL为止。

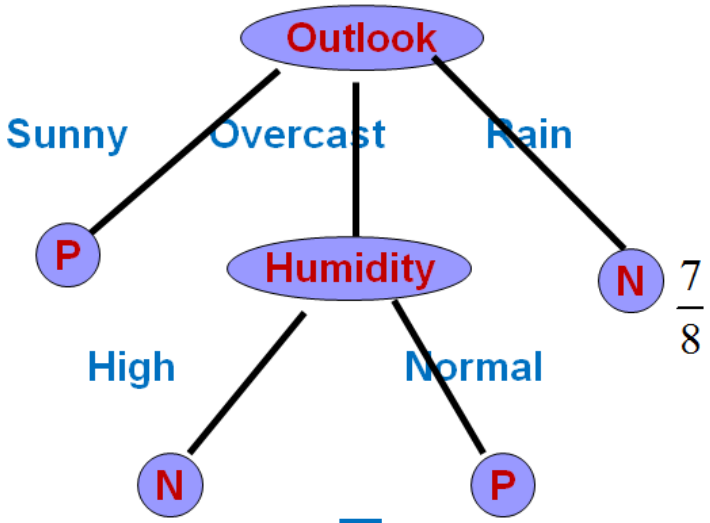


图2

决策树算法过拟合问题及策略

过拟合的原因：学习时过多考虑对训练数据的正确分类，从而构建出过于复杂的决策树。

生成决策树的所有叶节点都是“纯”的（达到最小的不纯度）
决策树本身就是100%完美拟合训练样本的产物，

？极端情况

解决思路：考虑决策树的复杂度，对生成的决策树进行简化（剪枝pruning），即是从已生成的树上裁掉一些子树或叶节点，并将其根节点或父节点作为新的叶节点，简化分类树的模型。

- 如何防止决策树和训练样本集的过度拟合

分支停止准则

①预剪枝：在建树过程中判断当前节点是否需要继续划分的剪枝方法

1）信息增益：

2）χ^2检验：一种假设检验技术，用χ^2统计量定量估计候选分支是否有统计上的“意义”，即是判断该分支对否明显有别于随机分支

②后剪枝：待决策树“充分生长”后，再判断是否将某些分支变成节点

1）错误分类率

2）决策树编码长度

C4.5采用悲观剪枝法，它使用训练集生成决策树又用它来进行剪枝，不需要独立的剪枝集。

PEP后剪枝技术由Quinlan提出，不需要像REP(错误率降低修剪)那样需要用部分样本作为测试数据，而是完全使用训练数据来生成决策树，又用这些训练数据来完成剪枝。

CART(classification and regression tree)

- CART Proposed by Breiman et. al. (1984)
- Constant numerical values in leaves　叶节点的常量值
- Variance as measure of impurity 方差（Gini指数）作为不纯度测量标准

- 是一种产生二叉决策树的技术

- 两个重要的思想:
  （1）递归地划分自变量空间;
  （2）用验证数据进行剪枝.

CCP (Cost-Complexity Pruning) 代价复杂度剪枝（详见p68）

- CART算法: 两步
  （1）决策树生成：基于训练数据集生成决策树，生成的决策树要尽量大;
  （2）决策树剪枝：用验证数据集对已生成的树进行剪枝并选择最优子树;

另一个度量指标--Gini指标（CART算法）

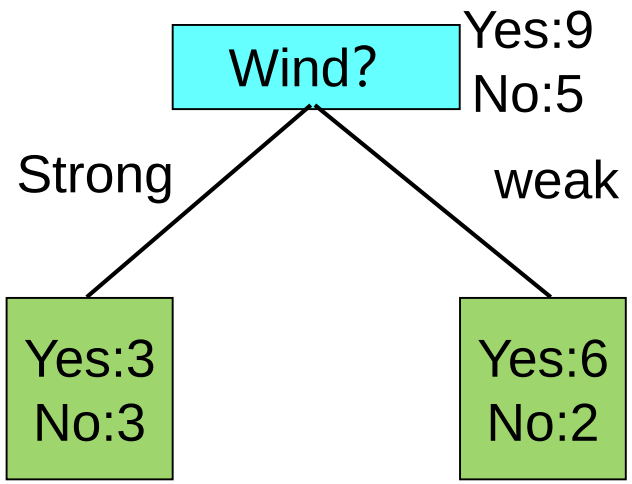
假设某节点v处的数据样本集合S_v包含c个类别的记录，其中，p_i为类别i在节点v处的概率，*Gini*指标定义为：

$$Gini(v_i) = 1 - \sum_{i=1}^{c} p_i^2$$

如果集合分成l个部分，那么关于分割V 的平均*Gini*指标定义为：

$$Gini(V) = \sum_{k=1}^{l} \frac{n_i}{n} Gini(v_i)$$

Gini指数：
总体内包含的类别越杂乱，GINI指数就越大（与熵的概念相似）



计算各特征的基尼指数，选取最优特征及最优切分点

比较数据集在不同的特征Ai及特征取值x的切分下的基尼指数

Gini(Wind/strong)=1-(3/6)²-(3/6)² =0.5

Gini(Wind/weak) =1-(2/8)²-(6/8)² =0.375

分割*V=wind* 时的平均*Gini*指标：

Gini(Wind)=6/14×0.5+8/14×0.375=0.429

基尼指数值越大样本集合的不确定性越大。

具有最低基尼指数的属性是最好的属性。

CART—递归划分自变量空间

- 递归划分

  用Y表示因变量(分类变量)，用X1,X2,…,XP表示自变量，通过递归的方式把关于X的P维空间划分为不重叠的矩形.

  - 划分步骤:

    （1）选择一个自变量，例如Xi和Xi的一个值Si,若选择Si把P维空间分为两部分：一部分包含的点都满足Xi<=Si;另一部分包含的点满足Xi>Si.

    （2）将上步中得到的两部分中的一个部分,通过选择一个变量和该变量的划分值以相似的方式再划分.

    重复上述步骤，直至把整个X空间划分成的每个小矩形都尽可能的是同构的或“纯”的。“纯”的意思是（矩形）所包含的点都属于同一类.

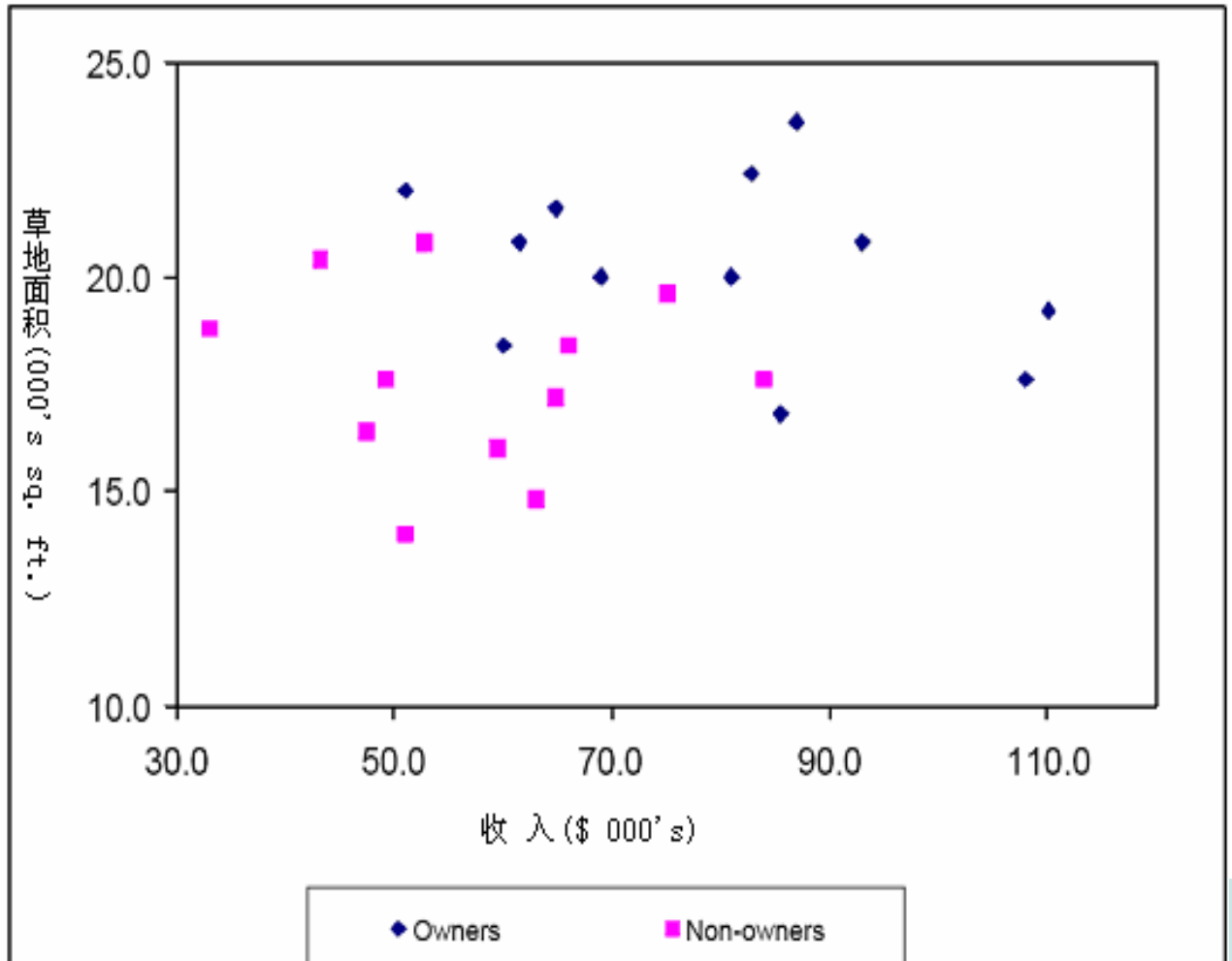
    划分点搜索准则为Gini指数

CART-递归划分的过程

例1　割草机制造商意欲发现一个把城市中的家庭分成那些愿意购买乘式割草机和不愿意购买的两类的方法。在这个城市的家庭中随机抽取12个拥有者和12个非拥有者的家庭作为样本。

这些数据如图所示。这里的自变量是收入（X1）和草地面积（X2）。
类别变量Y有两个类别：拥有者和非拥有者。

图 1



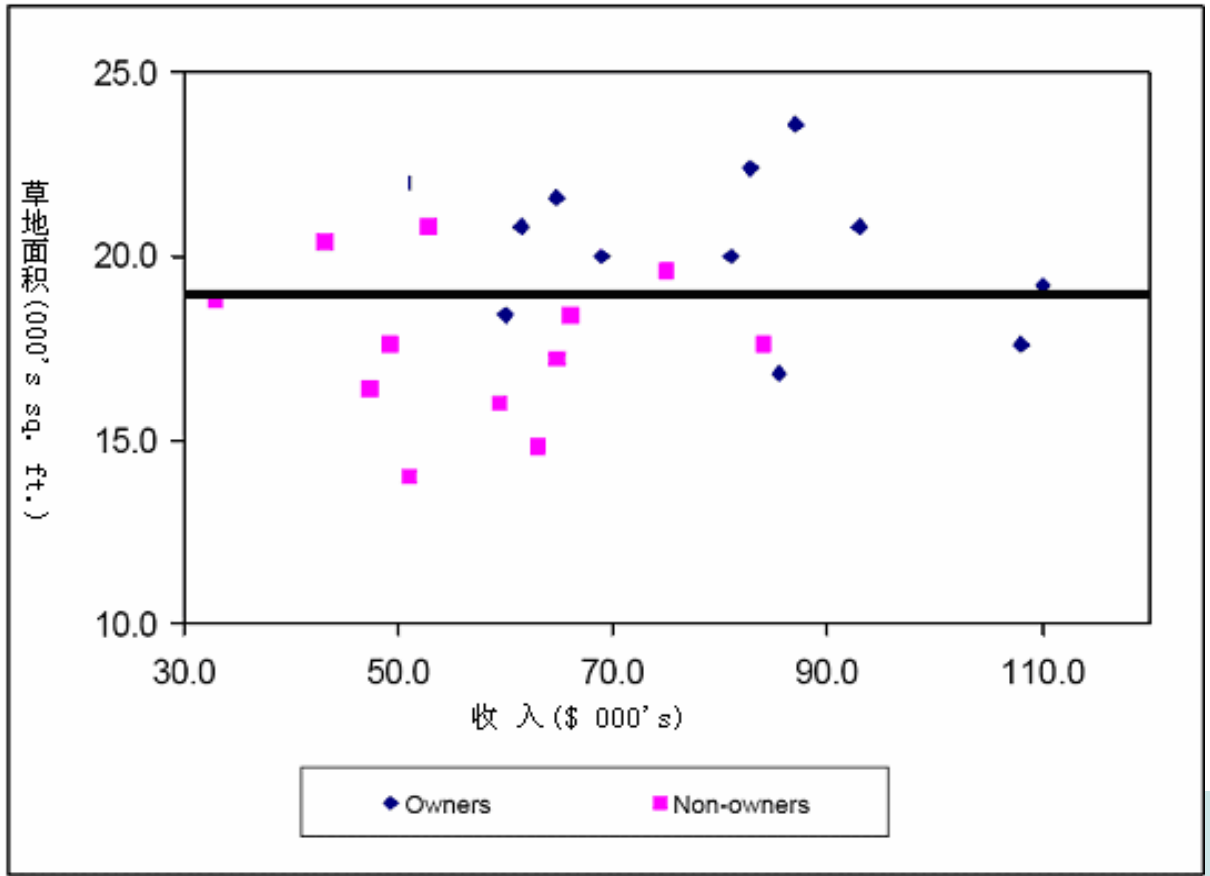
CART如何选择划分点?

- 对于连续值处理引进“分裂点”的思想，假设样本集中某个属性共n个连续值，则有n-1个分裂点，每个“分裂点”为相邻两个连续值的均值 (a[i] + a[i+1]) / 2。

  X1可能划分点是{38.1,45.3,50.1…,109.5};
  X2可能划分点是{16.4,15.4,16.2…23}.

选择草地面积变量X2=19做第一次分割,由(X1,X2)组成的空间被分成X2<=19和X2>19的两个矩形.

图2



CART如何选择划分点?

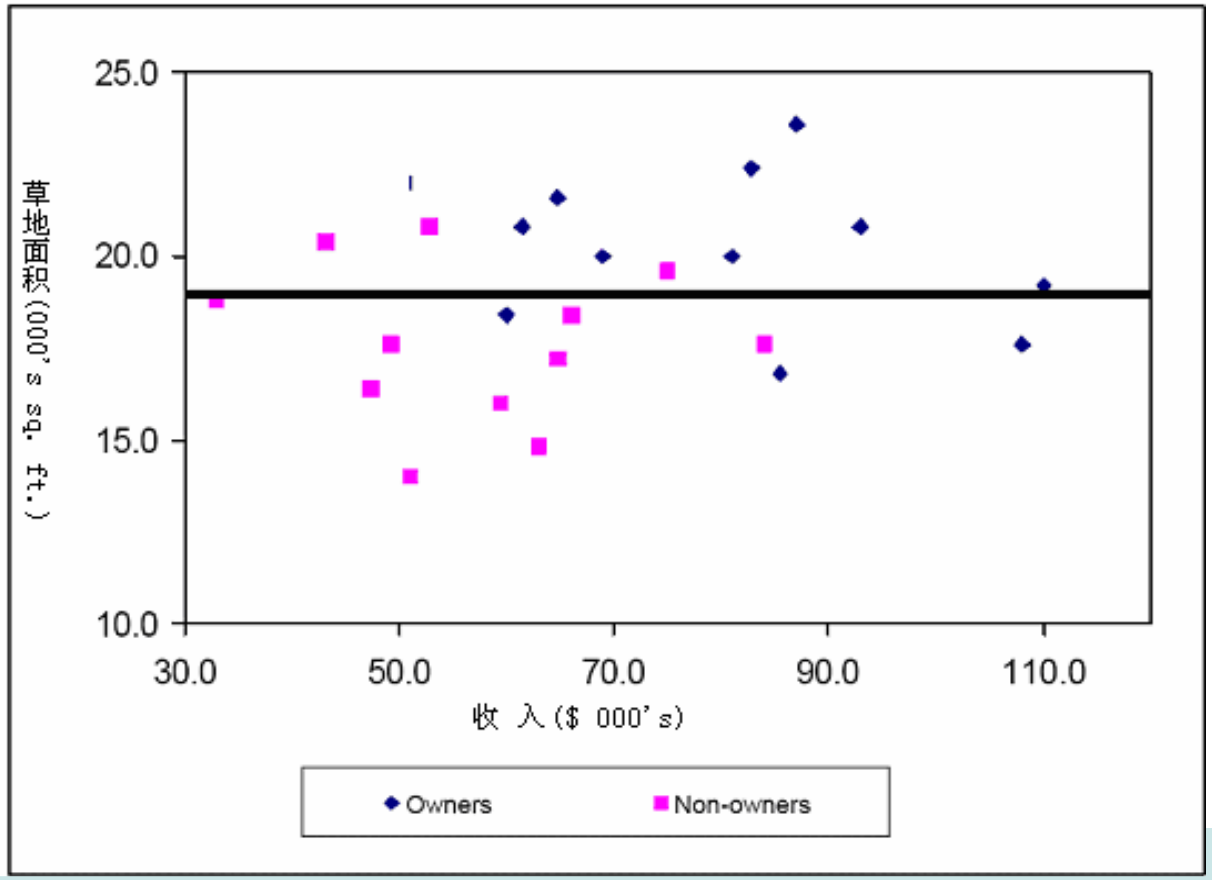
这些划分点按照能减少杂质的多少来分级.

杂质度量方法：Gini指标. $Gini(v_i)=1-\sum_{i=1}^{c} p_i^2$ $Gini(V)=\sum_{k=1}^{l}\frac{n_i}{n}Gini(v_i)$

其中K=1,2,…C,表示类，P_k是观测点中属于类K的比例.

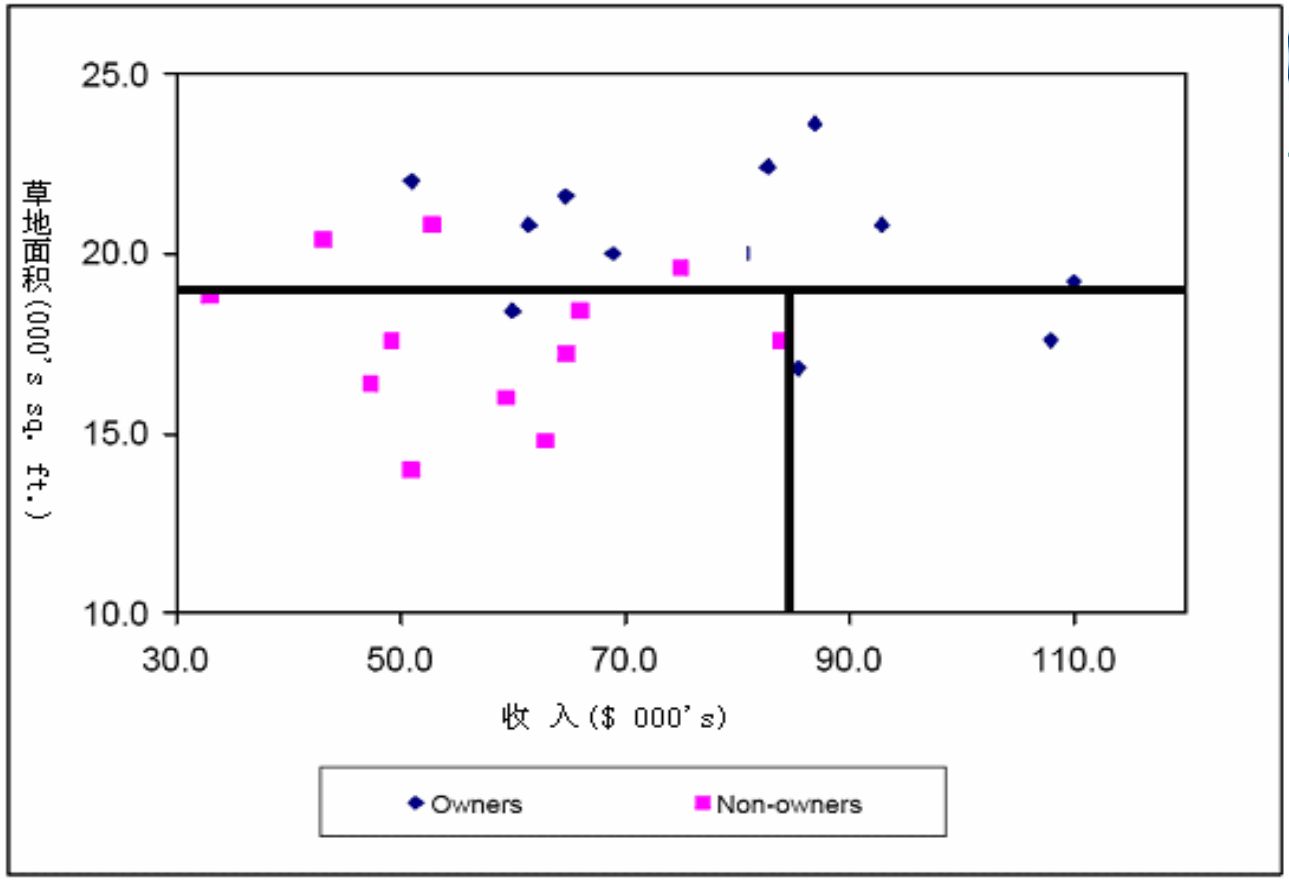
选择草地面积变量X2=19做第一次分割,由(X1,X2)组成的空间被分成X2<=19和X2>19的两个矩形.

图 2



选择收入变量X1=84. 75

图 3



选择收入变量X1=84. 75

图 3

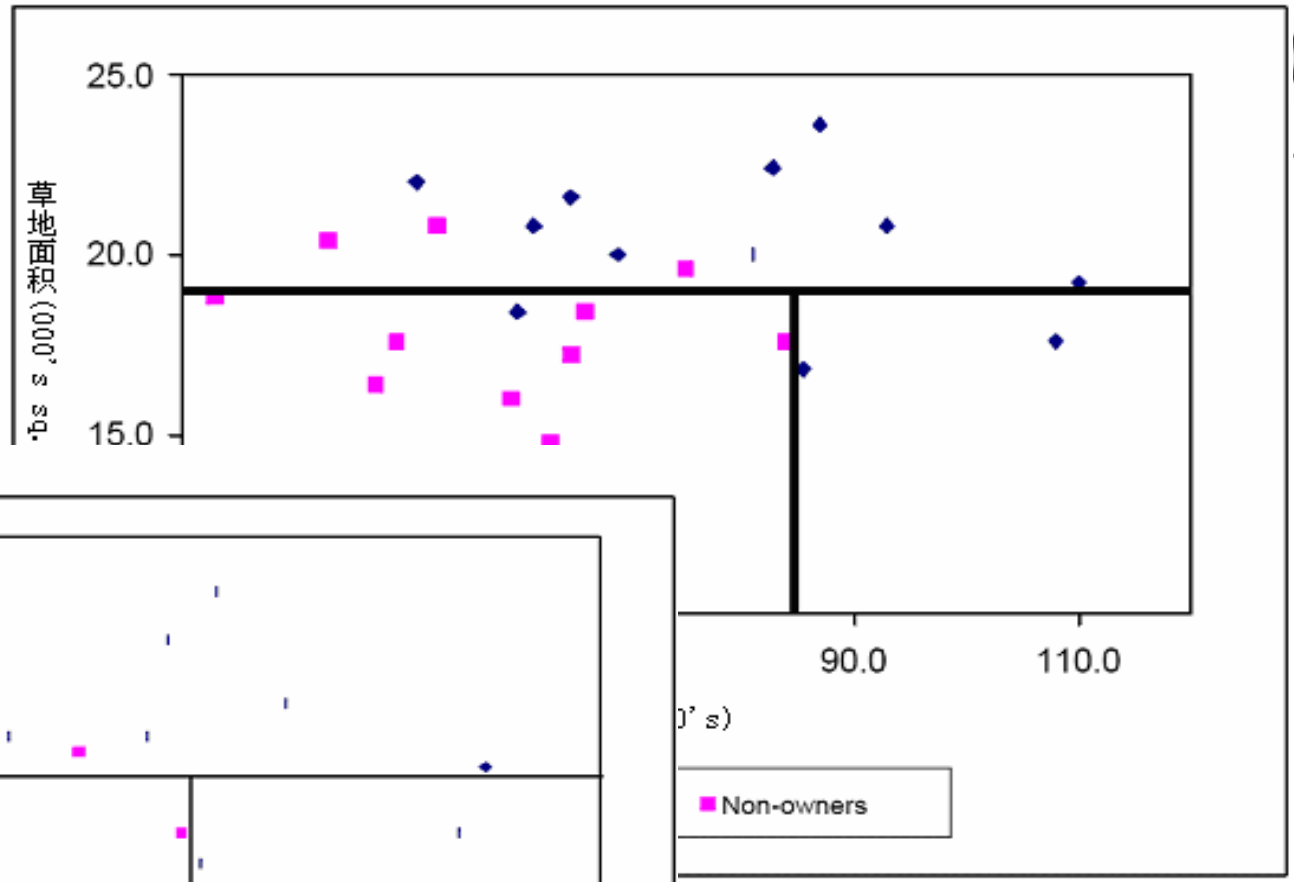
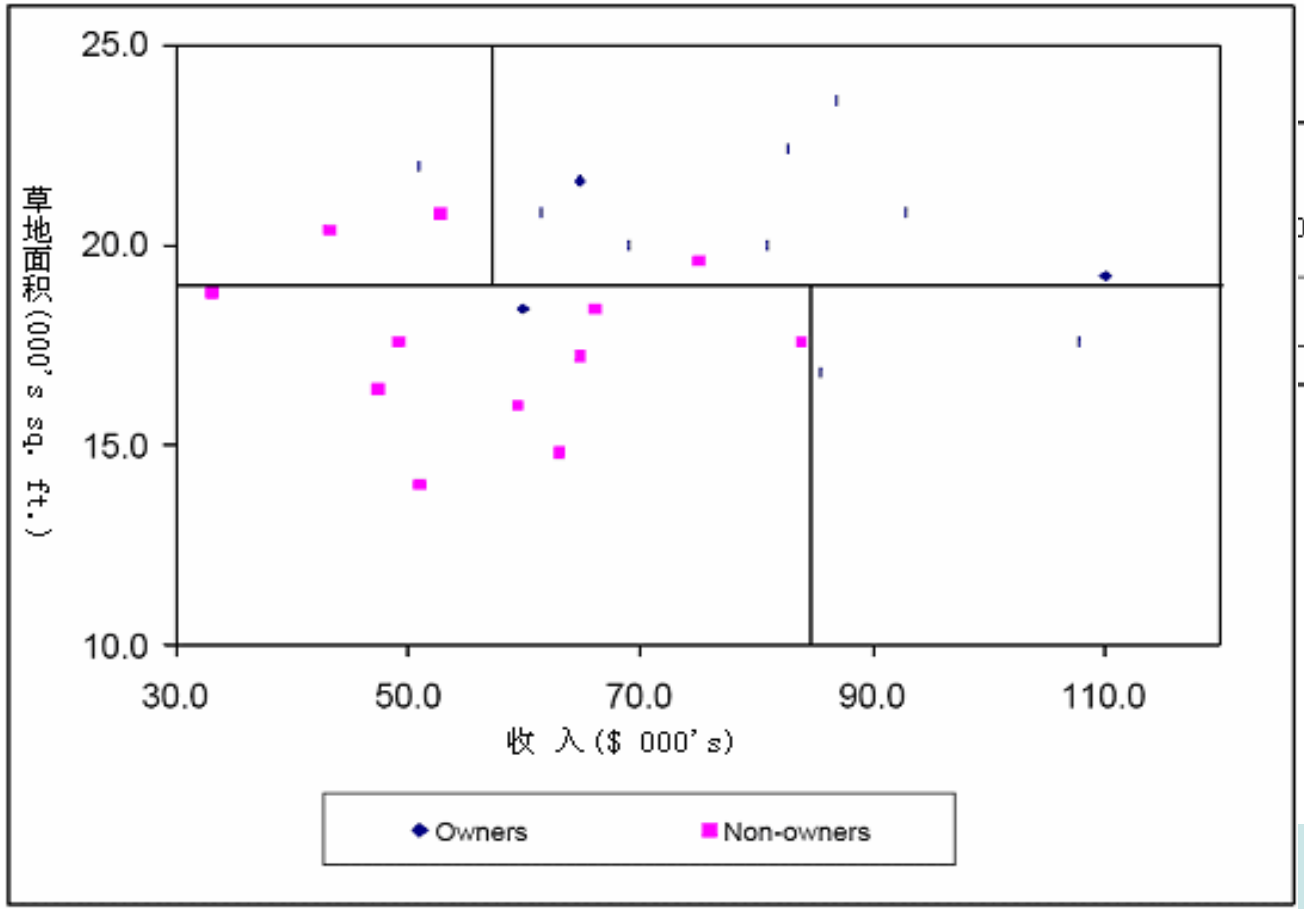
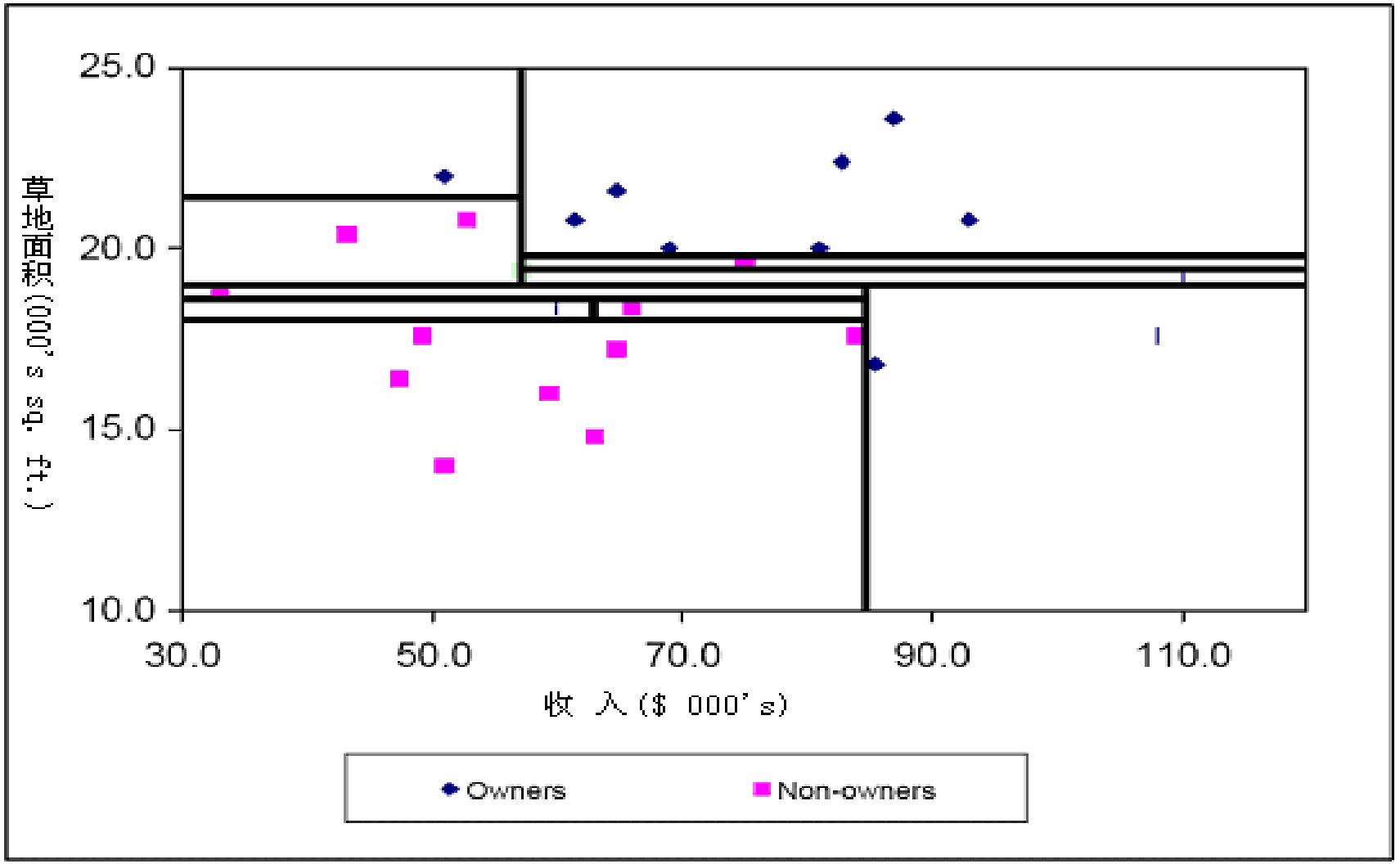


图 4



我们能看到递归划分是如何精炼候选矩形，使之变得更纯的算法过程。最后阶段的递归分析如图5所示

图 5



这个方法被称为分类树的原因是每次划分都可以描述为把一个节点分成两个后续节点. 第一次分裂表示为树的根节点的分支, 如图

图 6

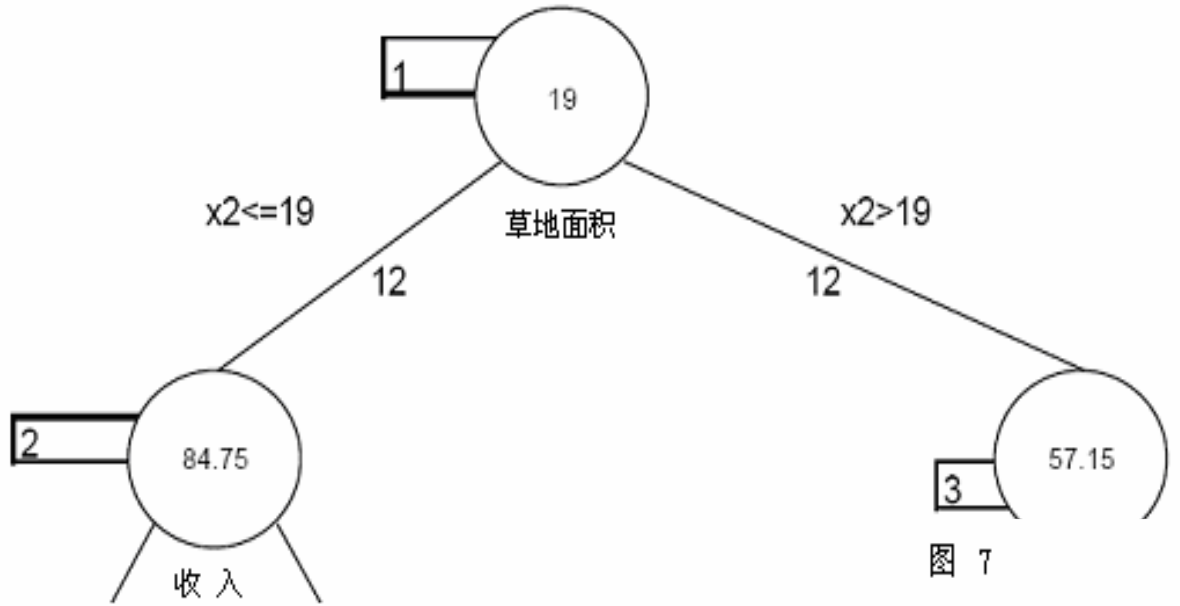
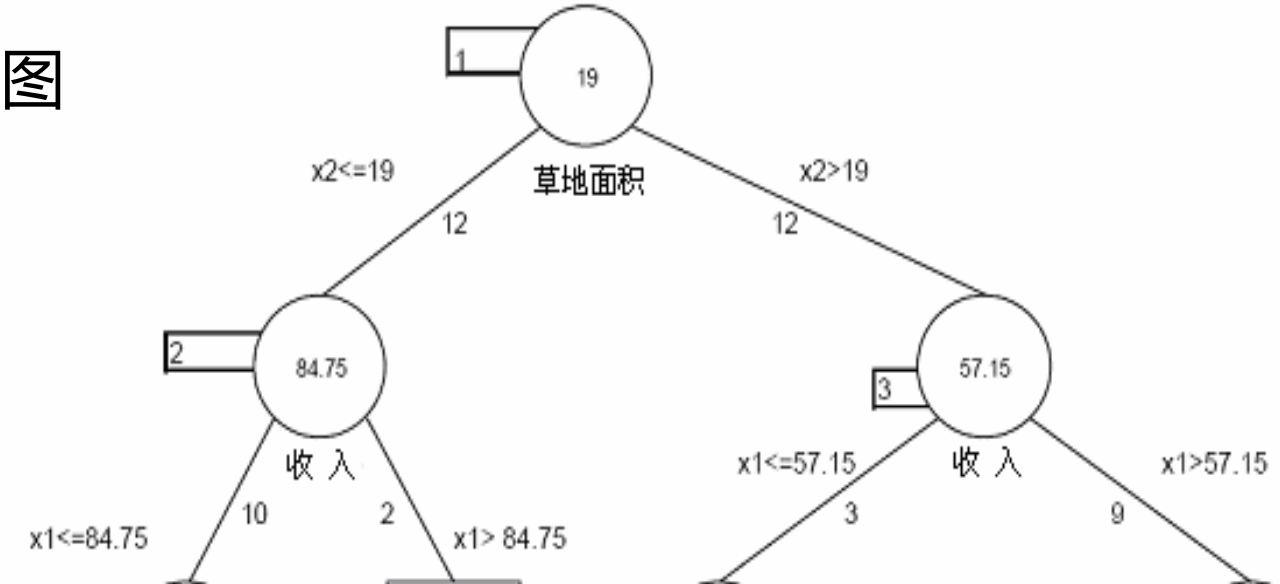
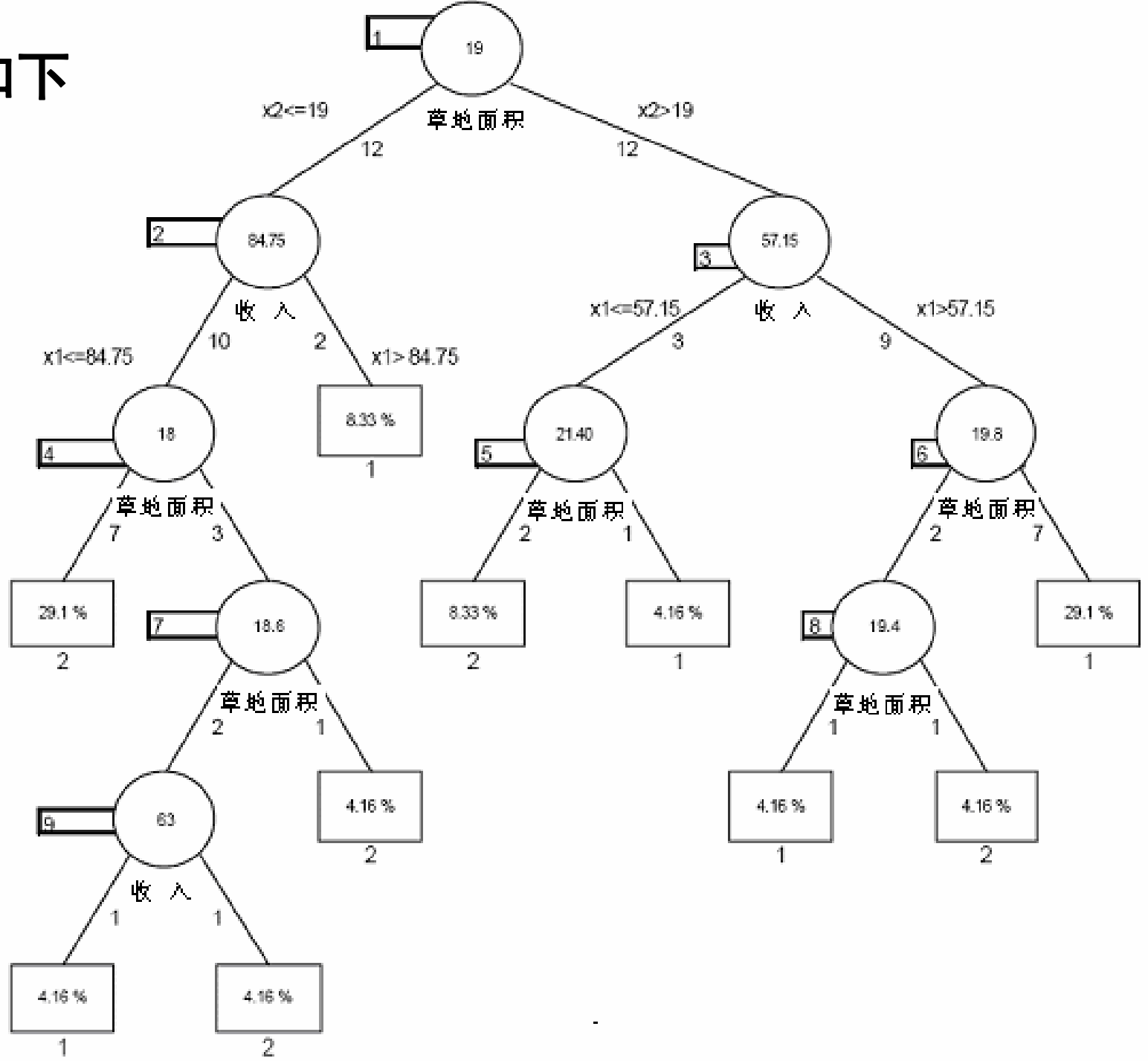


图 7

树的前三次划分如图



整个树如下


1
19
x2<=19
12
草地面积
x2>19
12
2
84.75
3
57.15
收 入
x1<=84.75
10
2
x1>84.75
x1<=57.15
3
收 入
9
x1>57.15
4
18
8.33 %
1
5
21.40
6
19.8
草地面积
7
3
草地面积
2
1
草地面积
2
7
29.1 %
2
7
18.6
8.33 %
2
4.16 %
1
8
19.4
29.1 %
1
草地面积
2
1
草地面积
1
1
9
83
4.16 %
2
4.16 %
1
4.16 %
2
收 入
1
1
4.16 %
1
4.16 %
2

决策树算法

CART算法对离散值Gini指数的计算（二叉树）

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

在属性取值为多值的情况下，需计算属性在不同取值进行切分情况下的基尼指数，取基尼指数最小的值对应的属性取值为该属性 的*最优切分点（最优切分）*

对离散值Gini指数的计算

- 以*Temperature*为例，内含三种属性{Cool，Mild，Hot}，共有2^3个子集，去除子集{Cool，Mild，Hot}和{∅}（未分裂）

计算属性*Temperature*在不同取值进行切分情况下的基尼指数,确定*最优切分点*

CART分类举例

Temperature	Play Tennis
Cool	Yes
Cool	No
Cool	Yes
Cool	Yes
Mild	Yes
Mild	No
Mild	Yes
Mild	Yes
Mild	Yes
Mild	No
Hot	No
Hot	No
Hot	Yes
Hot	Yes

$$Gini_{Temperature\{Cool,Mild\}}(V)$$

$$=\frac{10}{14}Gini(V_1)+\frac{4}{14}Gini(V_2)$$

$$=\frac{10}{14}[1-(\frac{7}{10})^2-(\frac{3}{10})^2]+\frac{4}{14}[1-(\frac{2}{4})^2-(\frac{2}{4})^2]$$

$$\doteq 0.443=Gini_{Temperature\{Hot\}}(V)$$

V1：{cool (3Y，1N)，mild（4Y，2N）}

V2：{Hot（2Y，2N）}

同理

$$Gini_{Temperature\{Cool,Hot\}}(V)=Gini_{Temperature\{Mild\}}(V)$$

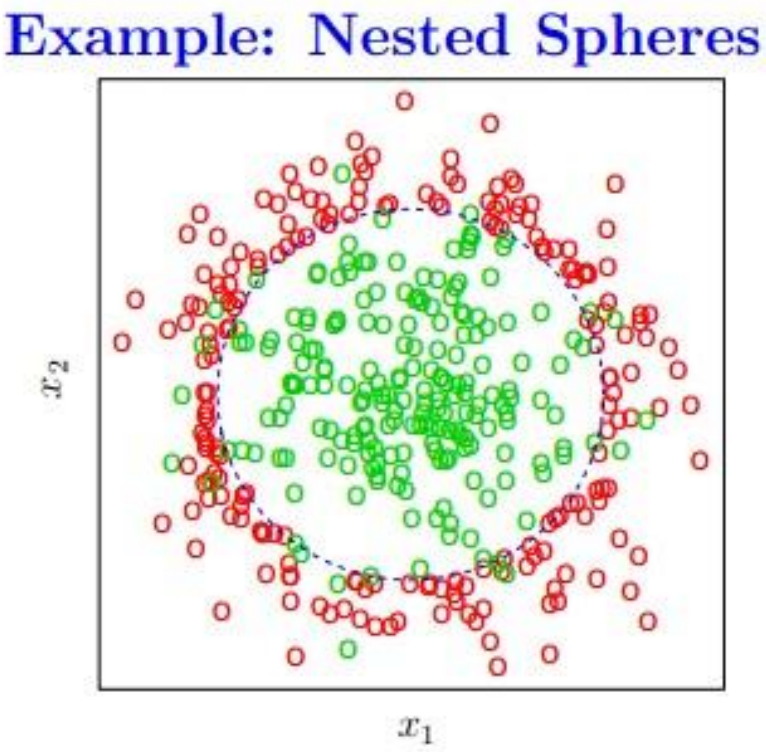
$$\doteq 0.458$$

$$Gini_{Temperature\{Mild,Hot\}}(V)=Gini_{Temperature\{Cool\}}(V)$$

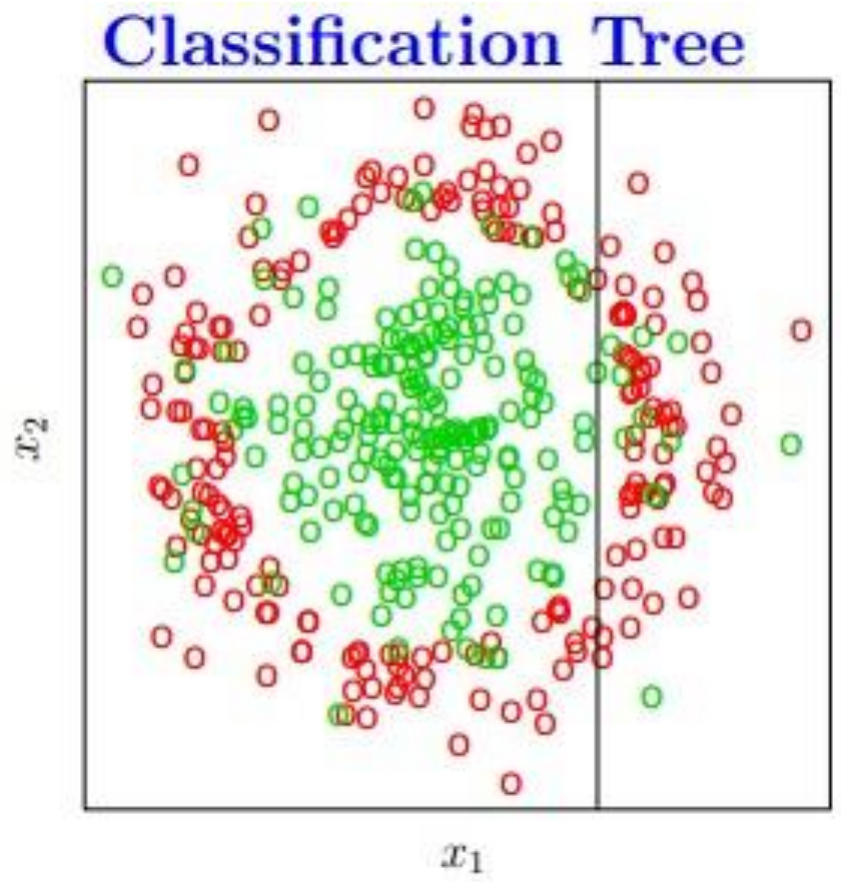
$$\doteq 0.450$$

决策树的例子

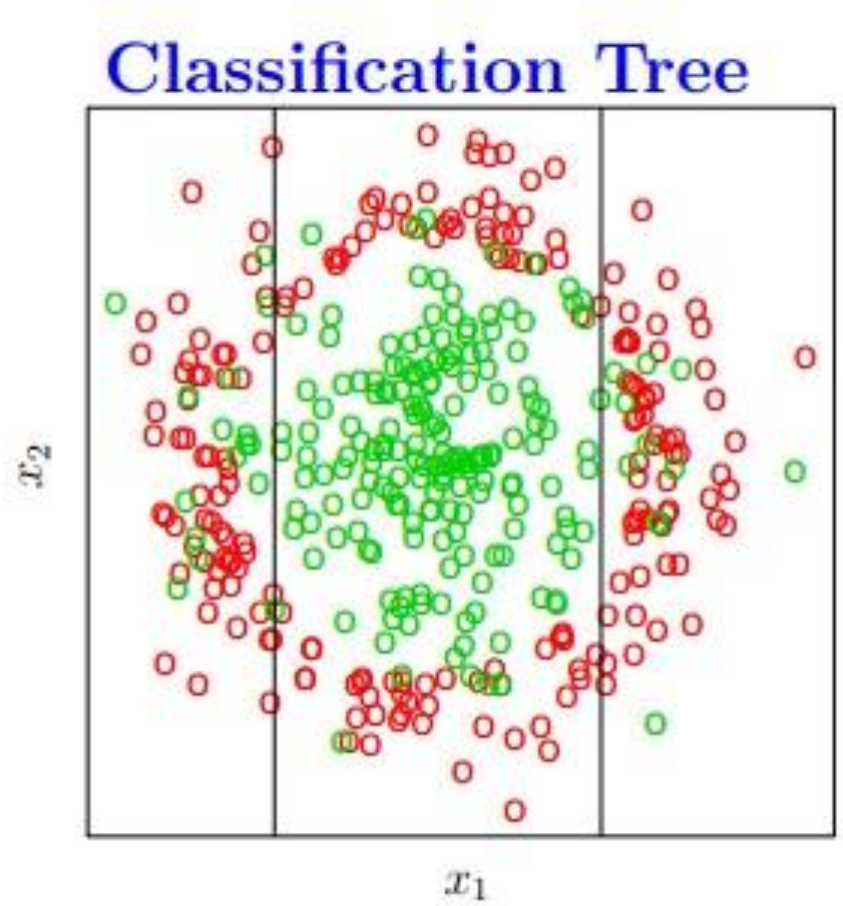
- 对于下面的数据，希望分割成红色和绿色两个类



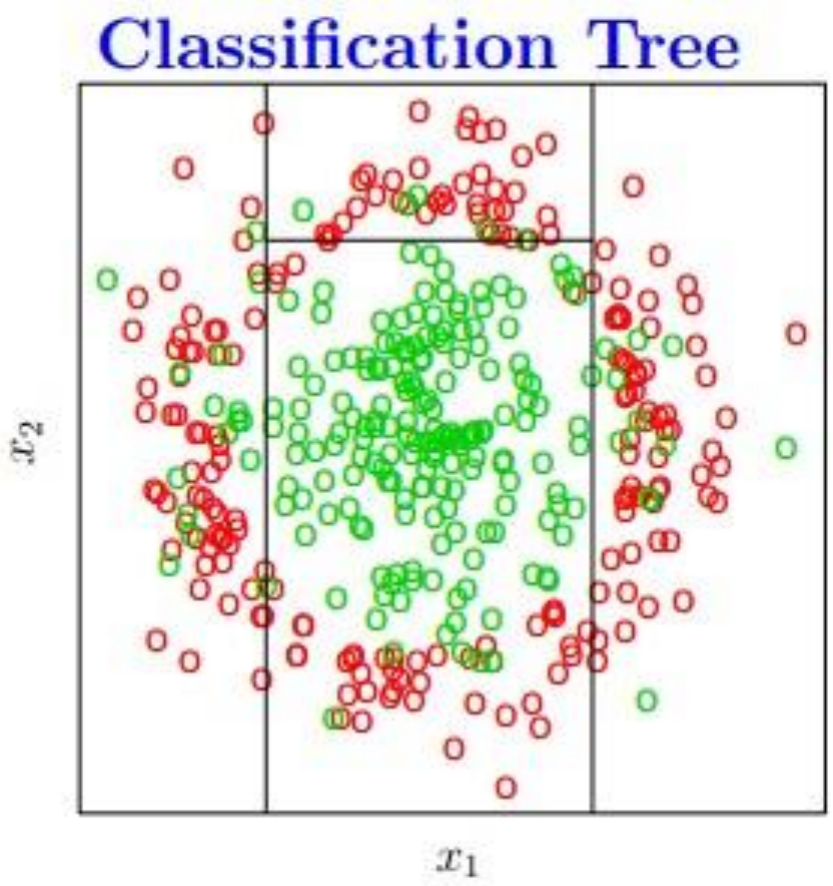
决策树的生成过程



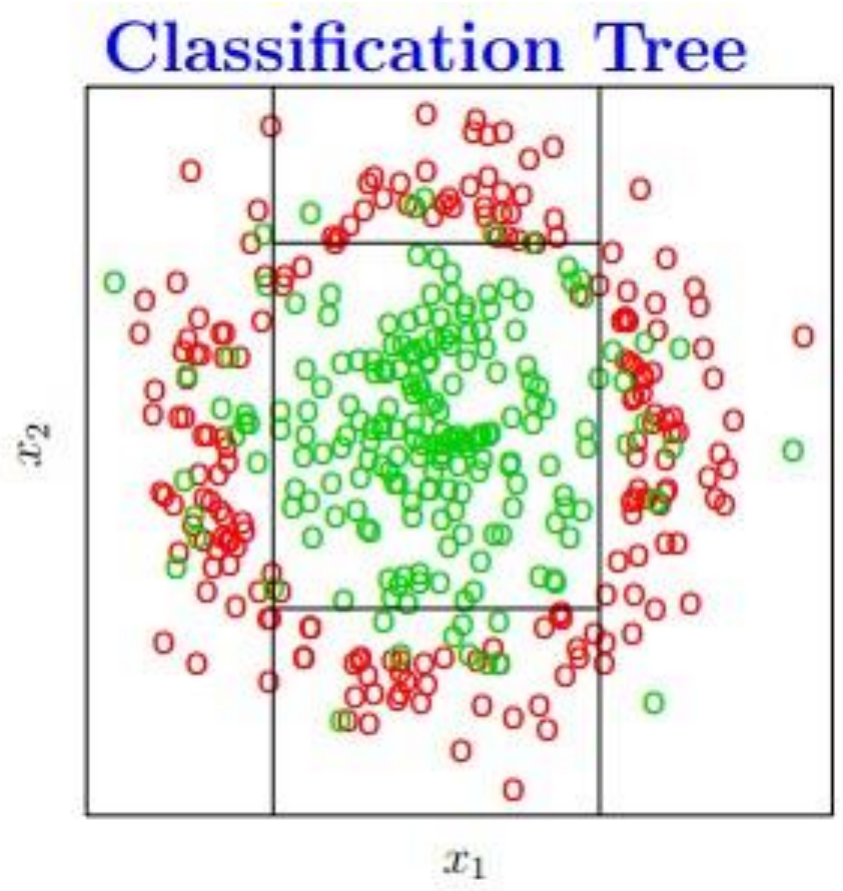
决策树的生成过程



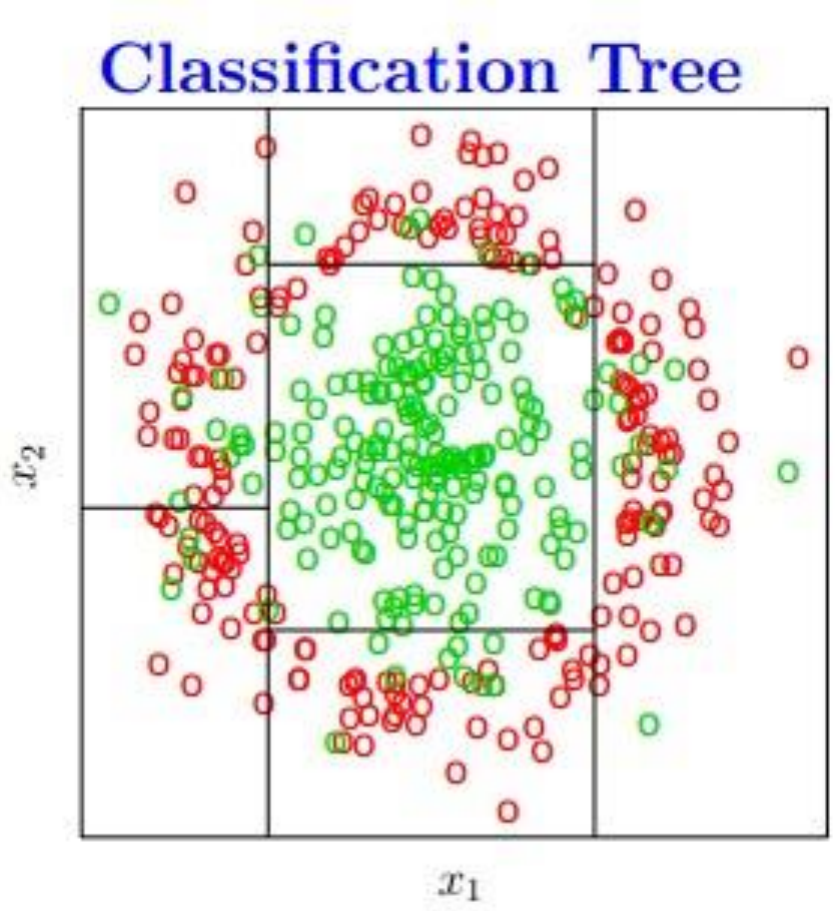
决策树的生成过程



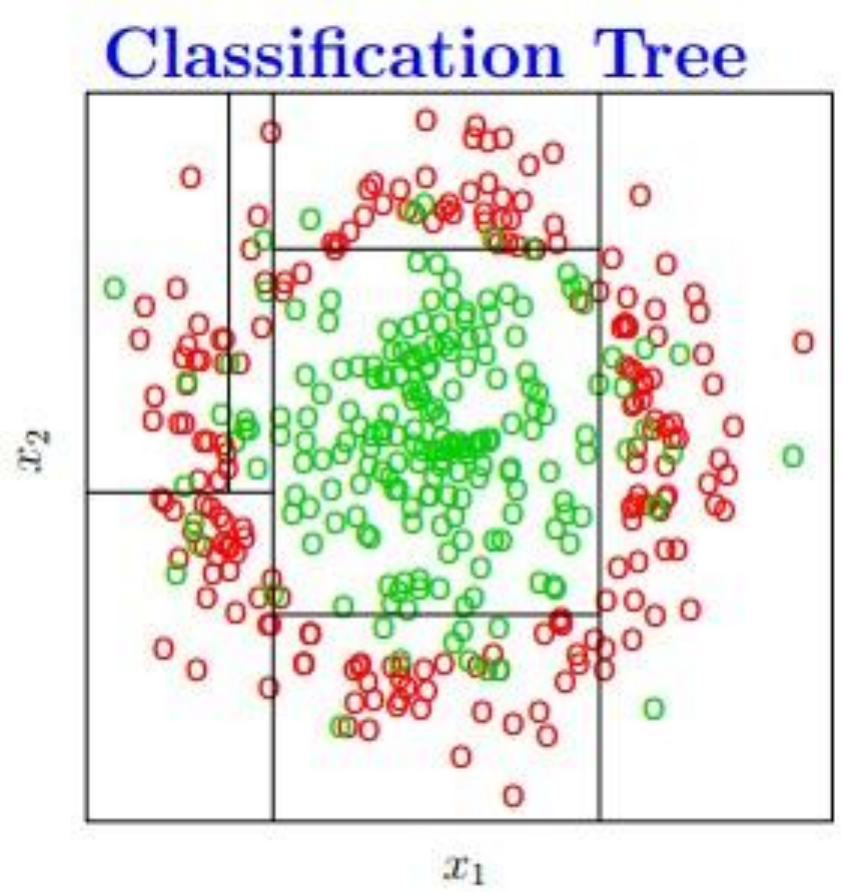
决策树的生成过程



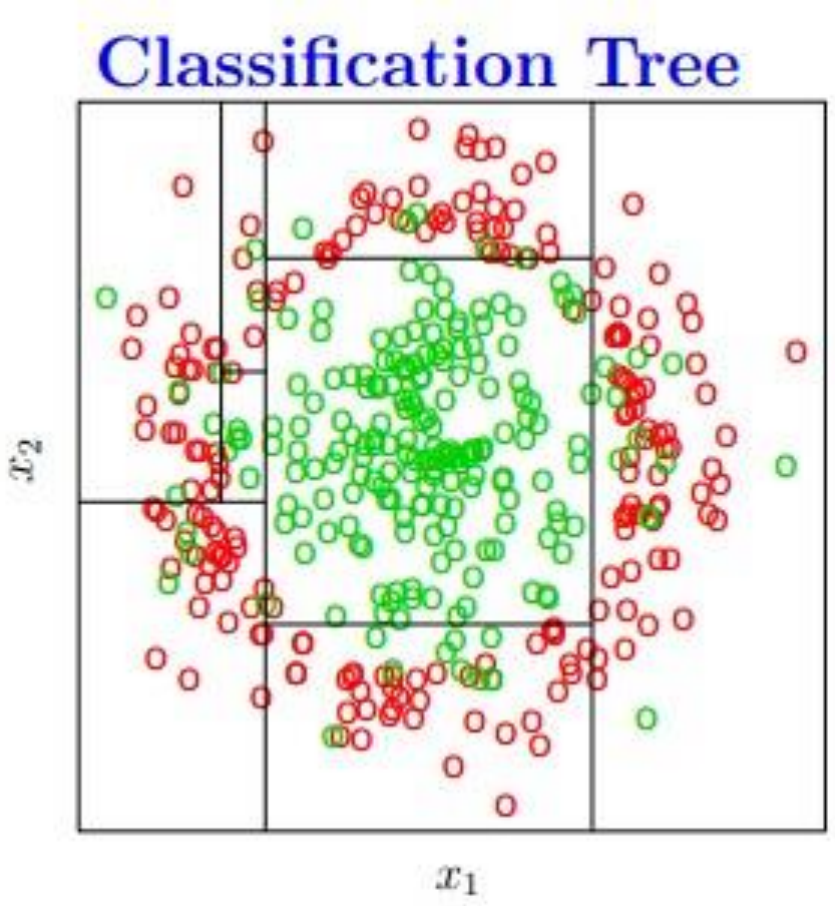
决策树的生成过程



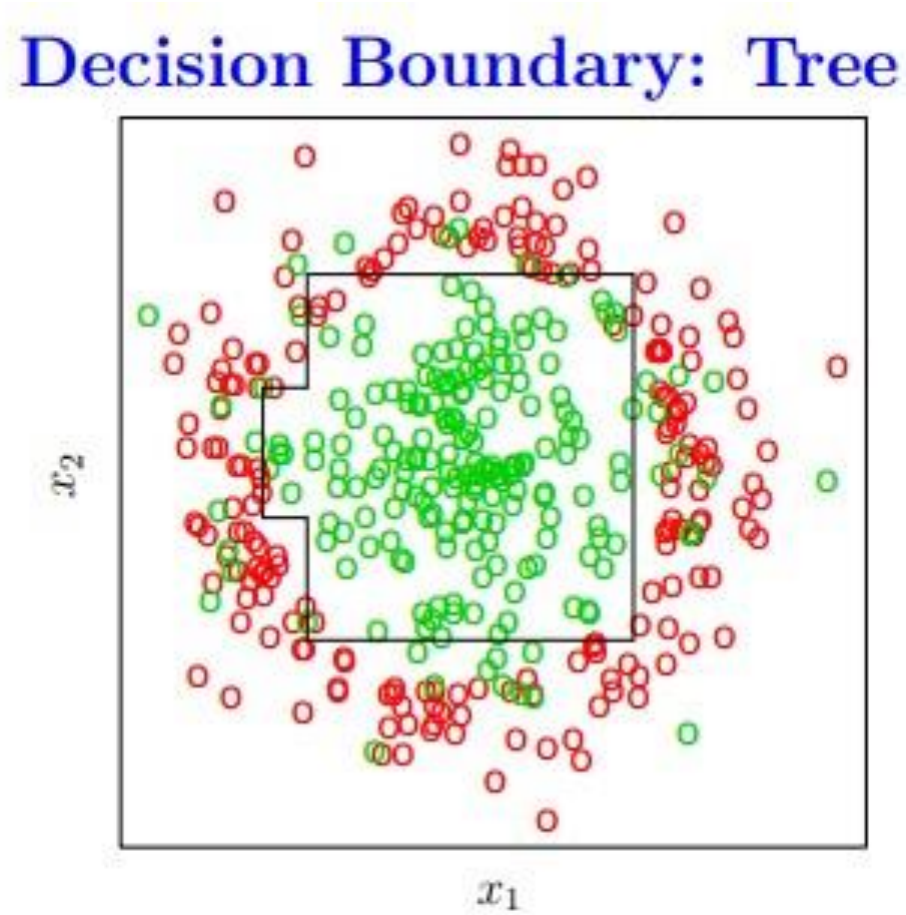
决策树的生成过程



决策树的生成过程



决策树的生成过程



决策树算法- CART（剪枝）

用验证数据进行剪枝

- CART过程中第二个关键的思想是用独立的验证数据集对根据训练集生成的树进行剪枝.

- CART剪枝目的：生成一个具有最小错误率的树.
- CART 剪枝方法

  用“成本复杂性”标准来剪枝，成本复杂性标准是分类树的简单误分(基于验证数据的) 加上一个对树的大小的惩罚因素.

  即成本复杂性标准为Err(T)+α|L(T)|，其中:

  Err(T)是验证数据被树误分部分;

  |L(T)|是树T的叶节点数;

  α是每个节点惩罚成本, α的值大于等于0

决策树算法- CART（剪枝）

- CCP (Cost-Complexity Pruning) 计算代价复杂度剪枝

自习

- 对于分类回归树中的每一个非叶子节点计算它的表面误差率增益值α。

$$\alpha = \frac{R(t) - R(T_t)}{|N_{T_t}| - 1}$$

α则表示剪枝后树的复杂度降低程度与代价间的关系

- $|N_{T_t}|$是子树中包含的叶子节点个数;
- R(t)是节点t的误差代价(节点的预测误差)，如果该节点被剪枝, R(t)=r(t)*p(t)； 其中， r(t)是节点t的误差率; p(t)是节点t上的数据占所有数据的比例。
- R(Tt)是子树T_t的误差代价(子树的预测误差)，如果该节点不被剪枝。它等于子树T_t上所有叶子节点的误差代价之和

比如有个非叶子节点t4如图所示：
已知所有的数据总共有60条，则节点t4的节点误差代价为：

$$R(t)=r(t)*p(t)=\frac{7}{16}*\frac{16}{60}=\frac{7}{60}$$

自习

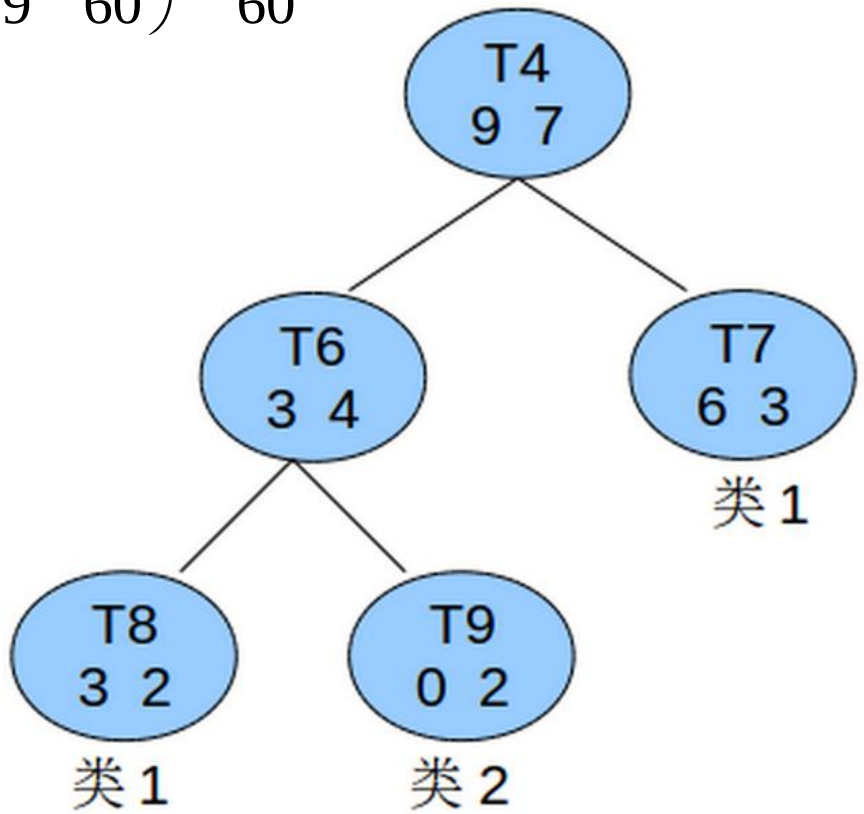
子树误差代价为：

$$R(T_t)=\Sigma R(i)=\left(\frac{2}{5}*\frac{5}{60}\right)+\left(\frac{0}{2}*\frac{2}{60}\right)+\left(\frac{3}{9}*\frac{9}{60}\right)=\frac{5}{60}$$

以t4为根节点的子树上叶子节点有3个，最终：

$$\alpha=\frac{R(t)-R(T_t)}{|N_{T_t}|-1}=\frac{7/60-5/60}{3-1}=\frac{1}{6}$$

找到α值最小的非叶子节点，令其左右孩子为NULL。当多个非叶子节点的α值同时达到最小时，取 $|N_{T_t}|$ 最大的进行剪枝。



三种决策树学习算法

- ID3：利用信息增益来进行特征选择的决策树学习过程

$$Gain(S,A)=H(S)-H(S\mid A)=H(S)-\sum_{a\in Values(A)}\frac{|S_a|}{|S|}H(S_a)$$

熵：$H(S)=-\sum_i P(c_i)\log_2 P(c_i)$

- C4.5：信息增益率

$$GainRatio(S,A)=\frac{Gain(S,A)}{SplitInfo(S,A)}=\frac{Gain(S,A)}{H_A(S)}=\frac{Gain(S,A)}{-\sum_{a\in Values(A)}\frac{|S_a|}{|S|}\log_2\frac{|S_a|}{|S|}}$$

- CART：基尼指数（Gini指数）or 方差不纯度

$$Gini(v_i)=1-\sum_{i=1}^{c}p_i^2$$

$$Gini(V)=\sum_{k=1}^{l}\frac{n_i}{n}Gini(v_i)$$

总结：属性的信息增益越大，表明属性对样本的熵减少的能力更强，该属性使数据由不确定性变成确定性的能力越强。

三种决策树学习算法-度量方法的对比

❖ 两分类问题不纯度的度量对比：

节点N处的不纯度度量。其中属于一类的概率为p，另一类的概率为(1-p)，则概率分布p的几种不纯度如下：

熵不纯度：

$$H(p) = -p\log_2 p - (1-p)\log_2(1-p)$$

$$Gini(v_i) = \sum_{i=1}^{c} p_i(1-p_i) = 1 - \sum_{i=1}^{c} p_i^2$$

Gini不纯度:

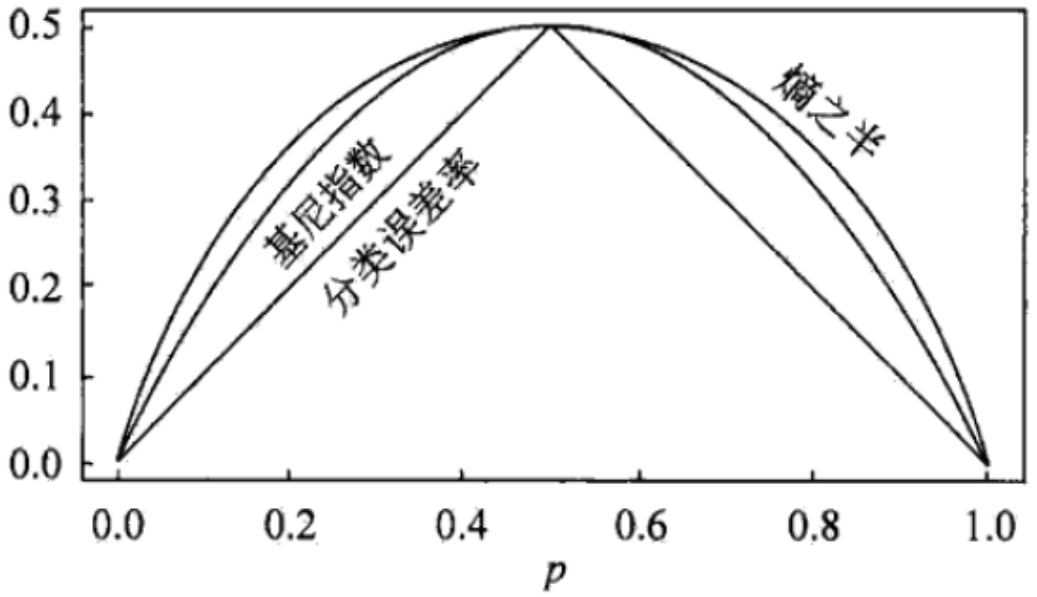
$$Gini(p) = 2p(1-p)$$

也称方差不纯度。在样本为双正态分布的假设下，该值正比于两类分布的总体分布方差

误分类不纯度:

$$\text{分类误差率} = 1 - \max_j p(\omega_j)$$

衡量节点N处训练样本分类误差的最小概率



二类分类中基尼指数、熵之半和分类误差率的关系

Gini指数和熵之半很接近，都可近似代表分类误差率

12.5 随机森林

随机森林的思想来源

发展线

Probably Approximately Correct

Bootstrap aggregation

PAC → Bootstrap → Bagging → Random forest ← CART

一种学习模型

有放回的重采样：
1、样本的多样性
2、有些样本可能永远不会被采样到。这部分样本可以作为验证集

0.6325自助法

集合分类器
并行式集成学习方法

在决策树的训练过程中引入随机属性选择

用随机抽取的方法训练出一群决策树来完成分类任务

输入：训练集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$；
基学习算法 $\mathfrak{L}$；
训练轮数 T.

过程：
1: **for** $t=1,2,\ldots,T$ **do**
2: $h_t=\mathfrak{L}(D,\mathcal{D}_{bs})$
3: **end for**

输出：$H(\boldsymbol{x})=\arg\max_{y\in\mathcal{Y}}\sum_{t=1}^{T}\mathbb{I}(h_t(\boldsymbol{x})=y)$

$\mathcal{D}_{bs}$ 是自助采样产生的样本分布.

Bagging 算法

12.5 随机森林

- 随机森林在以决策树为基学习器bagging集成的基础上做了修改。在决策树的训练过程中引入随机属性选择。

  两次随机抽取（核心思想）

  - 样本随机有放回采样：从样本集中用自助采样法Bootstrap Sampling（随机有放回采样）选出n个样本；
  - 随机属性选择：从所有属性（d个）中随机选择k个属性（特征），再选择最佳分割属性作为节点建立CART决策树；
  - 重复以上两步m次，即建立了m棵CART决策树
  - 这m个CART决策树形成随机森林，通过投票表决结果，决定数据属于哪一类。

  随机森林有两个重要参数：

  （1）k：树节点预选的属性（变量）个数

  （2）m：随机森林中树的个数

12.5 随机森林

- 随机森林，指的是利用多棵树对样本进行训练并预测的一种分类器。该分类器最早由Leo Breiman和Adele Cutler提出，并被注册成了商标。

- 简单来说，随机森林就是由多棵CART构成的。
- 对于每棵树，它们使用的训练集是从总的训练集中有放回采样出来的，这意味着，总的训练集中的有些样本可能多次出现在一棵树的训练集中，也可能从未出现在一棵树的训练集中。
- 在训练每棵树的节点时，使用的属性是从该节点的属性集合（d个）中随机无放回的抽取一个包含k个属性的子集。Leo Breiman [2001a] 建议一般情况下$k=\log_2 d$

Breiman, L. (2001a). “Random forests”. Machine Learning 45(1): 5–32.

12.5 随机森林

随机森林的训练过程：

- (1)给定训练集S，测试集T，特征维数F。

  确定参数：使用到的CART的数量t，每棵树的深度d，每个节点使用到的特征数量f；

  终止条件：节点上最少样本数s，节点上最少的信息增益m。

对于第i棵树，i=1,…,t：

- (2)从S中有放回的抽取大小和S一样的训练集S(i)，作为根节点的样本，从根节点开始训练
- (3)如果当前节点上达到终止条件，则设置当前节点为叶子节点。

  如果是分类问题，该叶子节点的预测输出为当前节点样本集合中数量最多的那一类c(j)，概率p为c(j)占当前样本集的比例；
  如果是回归问题，预测输出为当前节点样本集各个样本值的平均值。

  然后继续训练其他节点。若当前节点没有达到终止条件，则从F维特征中无放回的随机选取f维特征。利用这f维特征，寻找分类效果最好的一维特征k及其阈值th，当前节点上样本第k维特征小于th的样本被划分到左节点，其余的被划分到右节点。继续训练其他节点。

- (4)重复(2)(3)直到所有节点都训练过了或者被标记为叶子节点。
- (5)重复(2),(3),(4)直到所有CART都被训练过。

12.5 随机森林

利用随机森林的预测过程如下：

- 对于第1-t棵树，i=1-t：

(1) 从当前树的根节点开始，根据当前节点的阈值th，判断是进入左节点(<th)还是进入右节点(>=th)，直到到达某个叶子节点，并输出预测值。

(2) 重复执行(1)直到所有t棵树都输出了预测值。

如果是分类问题，则输出为所有树中预测概率总和最大的那一个类，即对每个c(j)的p进行累计；

如果是回归问题，则输出为所有树的输出的平均值。

12.5 随机森林

- 注：有关分类效果的评判标准，因为使用的是CART，因此使用的也是CART的评判标准，和C3.0,C4.5都不相同。
- 对于分类问题（将某个样本划分到某一类），也就是离散变量问题，CART使用Gini值作为评判标准。定义为Gini=1-∑(P(i)*P(i)), P(i)为当前节点上数据集中第i类样本的比例。例如：分为2类，当前节点上有100个样本，属于第一类的样本有70个，属于第二类的样本有30个，则Gini=1-0.7×07-0.3×03=0.42，可以看出，类别分布越平均，Gini值越大；类分布越不均匀，Gini值越小。在寻找最佳的分类特征和阈值时，评判标准为：argmax（Gini-GiniLeft-GiniRight），即寻找最佳的特征f和阈值th，使得当前节点的Gini值减去左子节点的Gini和右子节点的Gini值最大。
- 对于回归问题，相对更加简单，直接使用argmax(Var-VarLeft-VarRight)作为评判标准，即当前节点训练集的方差Var减去左子节点的方差VarLeft和右子节点的方差VarRight值最大。

12.5 随机森林

几个优点：
第一，是随机性，样本的复用，特征属性的随机选择，避免了过拟合。

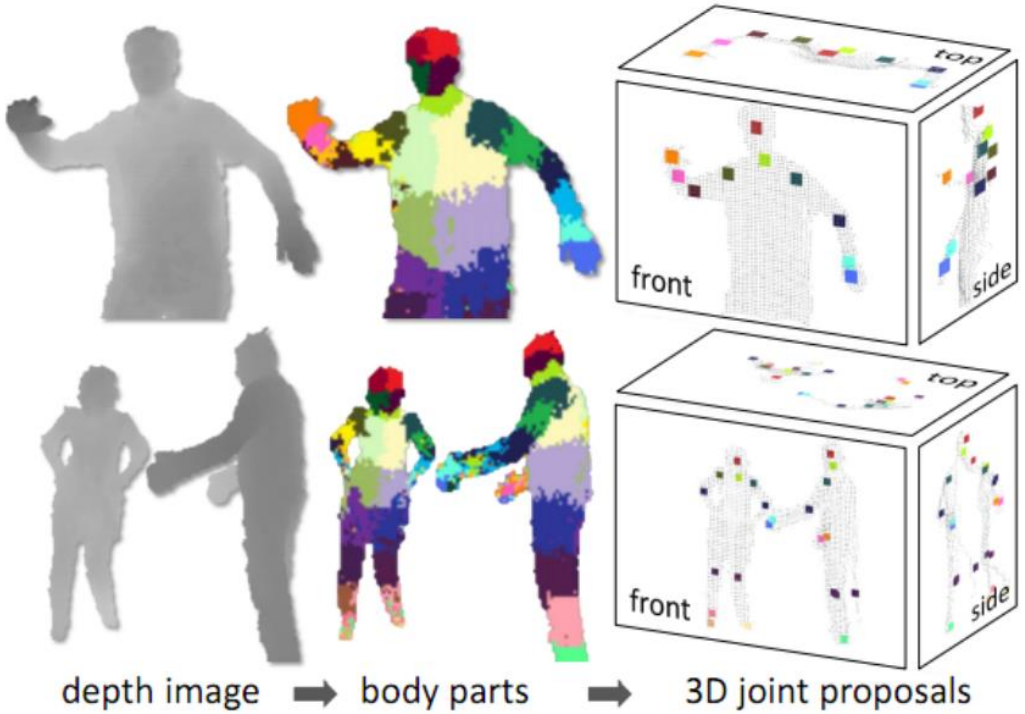
第二，一棵树的训练和构建比决策树快，而不用去实现复杂的剪枝，非常有特色。

第三，就是投票机制，根据多个决策树的结果，来确定回归或分类的结果。

将决策树与不同算法集成框架进行结合得到新的算法：

- 1）Bagging + 决策树 = 随机森林
- 2）AdaBoost + 决策树 = 提升树
- 3）Gradient Boosting + 决策树 = GBDT梯度提升树
- ...

12.6 实例分析：Kinect



Jamie Shotton etc,

Real-Time Human Pose Recognition in Parts from Single Depth Images[C] , CVPR 2011. IEEE, 2011.

3.3. Randomized decision forests

Decision forests are considered effective multi-class classifiers. Forest is a group of T decision trees, where each split node is represented by a feature and a threshold τ. The procedure starts at the root node of a tree, evaluating equation 1 at each split node and branching left or right according to the comparison to threshold τ. The leaf node consists of a learned distribution $P_t(c|I,x)$ over body part label c, in the tree t. Figure IV illustrates the forest approach.

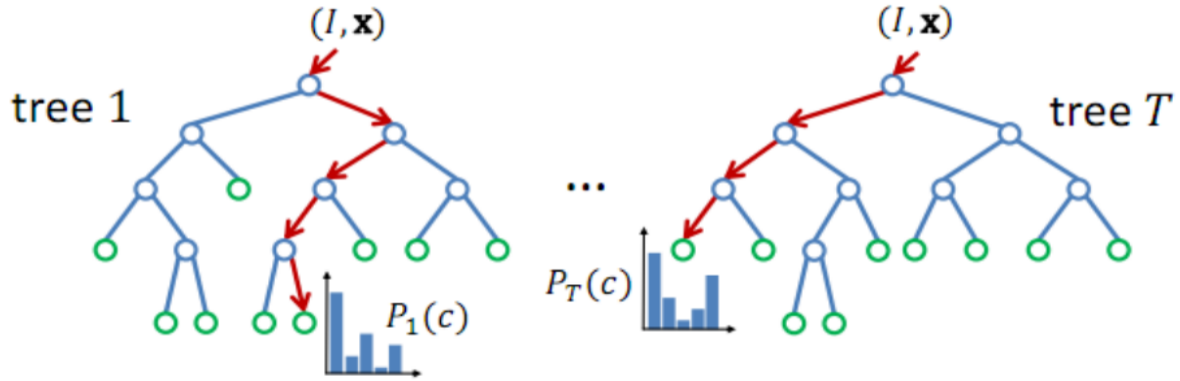


Figure IV. Decision forest.

The final classification is given by averaging all the distributions together in the forest. Equation 2 represents this classification.

$$P(c|I,x) = \frac{1}{T}\sum_{t=1}^{T} P_t(c|I,x) \quad (2)$$

TO LEARN MORE:

Leo BREIMAN, Jerome H. FRIEDMAN, Richard A. OLSHEN and Charles J. STONE. Classification and regression trees. The Wadsworth Statistics/Probability Series, Wadsworth, Belmont, 1984, x + 358 pages

Leo BREIMAN, Jerome H. FRIEDMAN, Richard A. OLSHEN and Charles J. STONE. Classification and regression trees. Chapman & Hall, New York,1993

Shotton J , Fitzgibbon A , Cook M , et al. Real-Time Human Pose Recognition in Parts from a Single Depth Image[C]// CVPR 2011. IEEE, 2011.

Breiman, L. Random forests. Machine Learning 45(1), 5–32 (2001).

附录（参考自学）

ENDING

- 后面三页为练习用数据

例：如下表所述的天气数据，学习目标是 预测Play or not play?

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rain	mild	high	false	yes
rain	cool	normal	false	yes
rain	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rain	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rain	mild	high	true	no

共14个实例：9个正例（yes），5个负例（no）

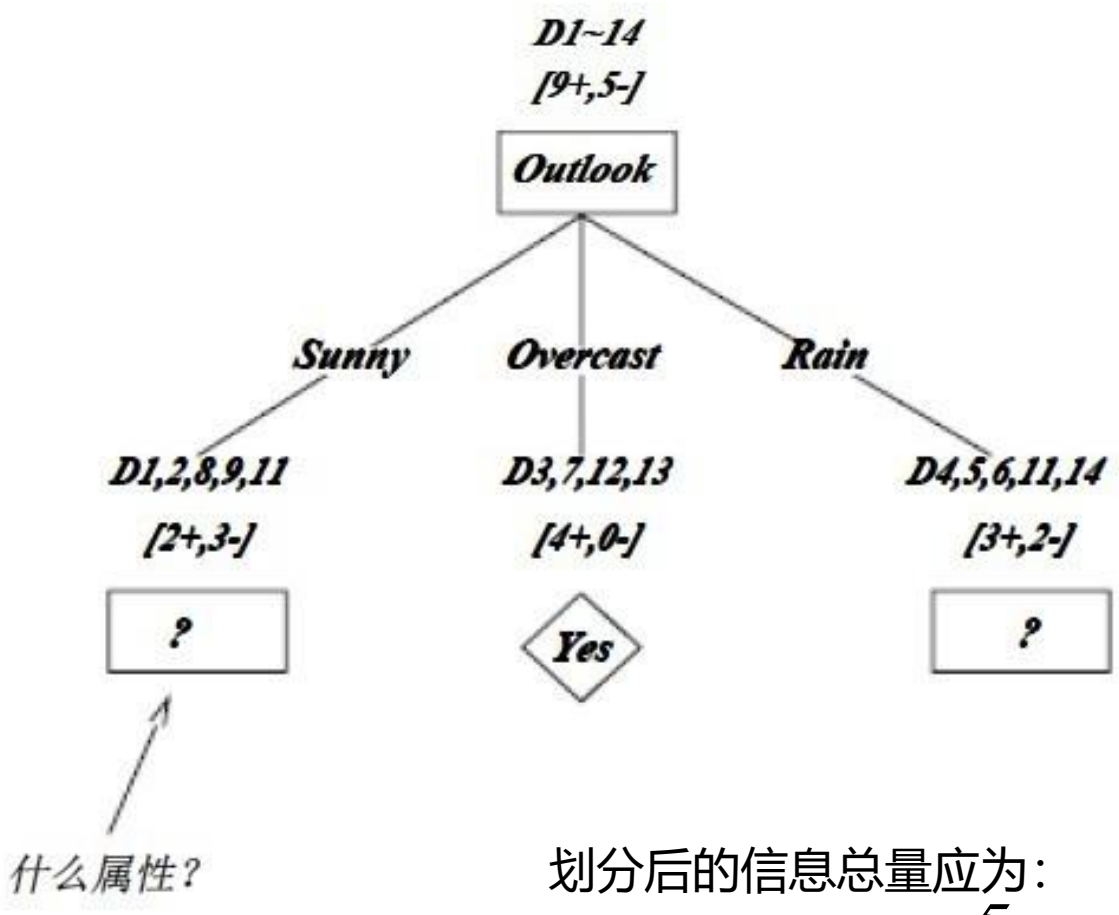
当前数据D（原始状态）的信息量用熵来计算：

$$H(D)=\text{info}(play?)=\text{info}([9,5])$$

$$=E(\frac{9}{14},\frac{5}{14})=entropy(\frac{9}{14},\frac{5}{14})$$

$$=-\frac{9}{14}\log_2\left(\frac{9}{14}\right)-\frac{5}{14}\log_2\left(\frac{5}{14}\right)=0.940$$

续上：选取Outlook属性来划分，见下图



- 在用Outlook属性划分后，可以看到数据被分成三份，则各分支的信息（熵）计算如下：

$$\text{info}([2,3]) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right) = 0.971 bits$$

$$\text{info}([4,0]) = -\frac{4}{4}\log_2\left(\frac{4}{4}\right) - \frac{0}{4}\log_2\left(\frac{0}{4}\right) = 0 bits$$

$$\text{info}([3,2]) = -\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right) = 0.971 bits$$

划分后的信息总量应为：

$$\text{info}([2,3],[4,0],[3,2]) = \frac{5}{14}\times 0.971 + \frac{4}{14}\times 0 + \frac{5}{14}\times 0.971 = 0.693 bits$$

续：

Outlook	Temperature	Humidity	Windy	Play?
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rain	mild	high	false	yes
rain	cool	normal	false	yes
rain	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rain	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rain	mild	high	true	no

$$InfoGain(S,A) = H(S) - H(S \mid A)$$
$$= H(S) - \sum_{a} p(A=a) H(S \mid A=a)$$
$$= H(S) - \sum_{a \in Values(A)} \frac{|S_a|}{|S|} H(S_a)$$

S=[9+,5-]
E=0.940

Humidity

High　　Low

[3+,4-]　　[6+,1-]

E=0.985　　E=0.592

InfoGain (S, **Humidity**)
=0.940-(7/14)*0.985-(7/14)*0.592
=0.151

S=[9+,5-]
E=0.940

Windy

Yes　　No

[6+,2-]　　[3+,3-]

E=0.811　　E=1.00

InfoGain(S, **Windy**)
=0.940-(8/14)*0.811-(6/14)*1
=0.048

InfoGain (S, **Outlook**)=0.940-(5/14)*0.971-(4/14)*0- (5/14)*0.971 = 0.940-0.693 =0.247

InfoGain (S, **Temperature**)=？？？