机器学习-决策树

目录 Contents

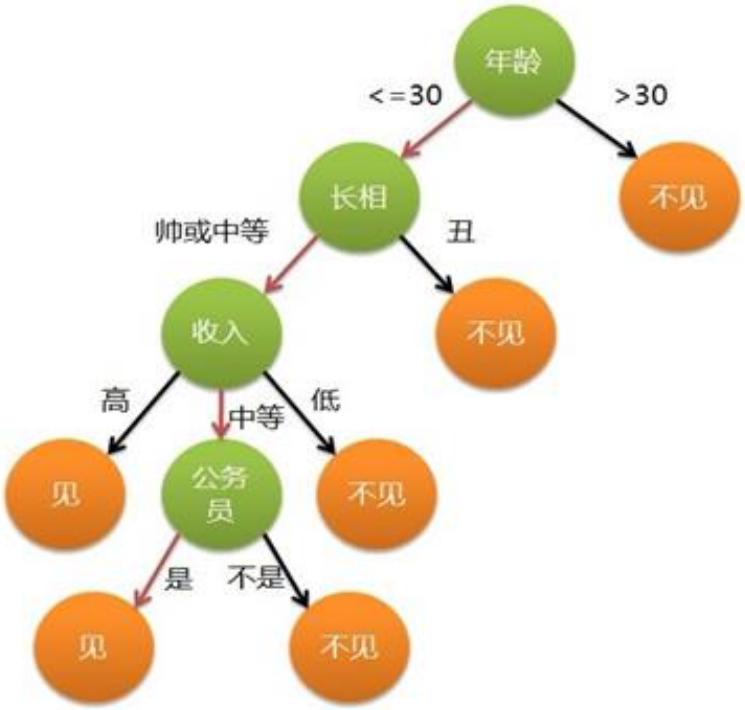
什么是决策树?

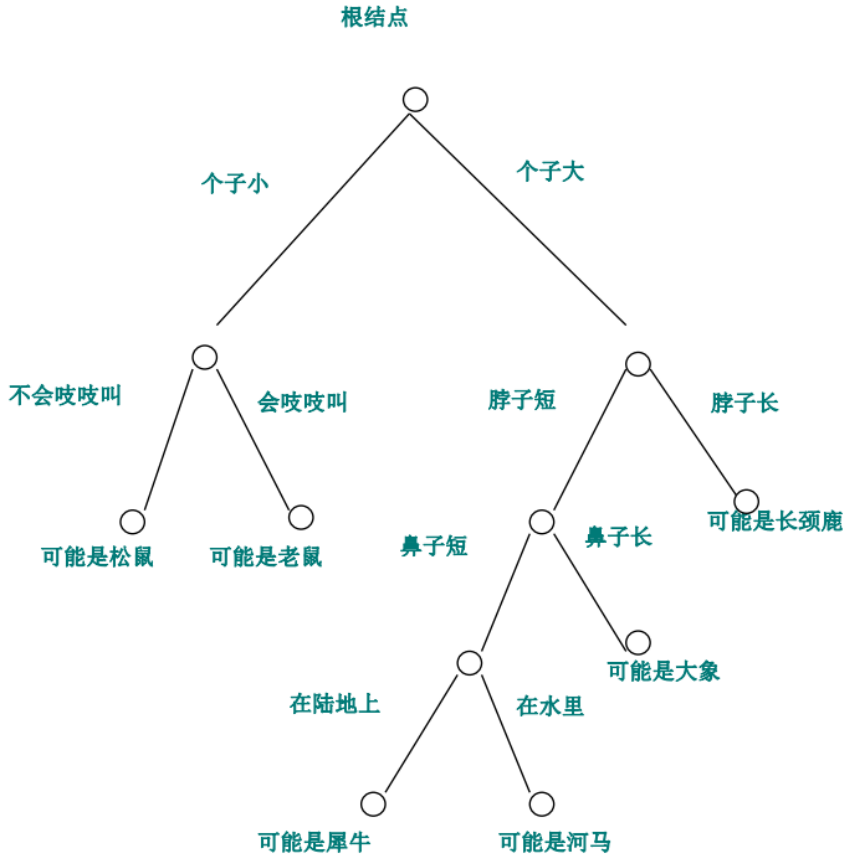
- 一种描述概念空间的有效的归纳推理办法。基于决策树的学习方法可以进行不相关的多概念学习，具有简单快捷的优势，已经在各个领域取得广泛应用。
- 决策树是一种树型结构，其中每个内部结点表示在一个属性上的测试，每个分支代表一个测试输出，每个叶结点代表一种类别。

场景

- 母亲：给你介绍个对象。
- 女儿：年纪多大了？
- 母亲：26。
- 女儿：长的帅不帅？
- 母亲：挺帅的。
- 女儿：收入高不？
- 母亲：不算很高，中等情况。
- 女儿：是公务员不？
- 母亲：是，在税务局上班呢。
- 女儿：那好，我去见见。

决策树示意图


年龄
<=30
>30
长相
不见
帅或中等
丑
收入
不见
高
中等
低
见
公务员
不见
是
不是
见
不见


根结点
个子小
个子大
不会吱吱叫
会吱吱叫
脖子短
脖子长
可能是松鼠
可能是老鼠
鼻子短
鼻子长
可能是长颈鹿
可能是大象
在陆地上
在水里
可能是犀牛
可能是河马

决策树的思想

- 决策树学习是以实例为基础的归纳学习。
- 决策树学习采用的是自顶向下的递归方法，其基本思想是以信息熵为度量构造一棵熵值下降最快的树，到叶子节点处的熵值为零，此时每个叶节点中的实例都属于同一类。
- 决策树学习算法的最大优点是，它可以自学习。在学习的过程中，不需要使用者了解过多背景知识，只需要对训练例子进行较好的标注，就能够进行学习。
  - 显然，属于有监督学习。
  - 从一类无序、无规则的事物(概念)中推理出决策树表示的分类规则。

决策树与if-then规则

- 可以将决策树看成一个if-then规则的集合。即由决策树的根结点到叶节点的每一条路径构建一条规则；路径上内部结点的特征对应着规则的条件，而叶结点的类对应着规则的结论。
- 决策树的路径或其对应的if-then规则集合的重要性质：互斥且完备（每一个实例都被一条路径或一条规则所覆盖，且只被一条路径或一条规则所覆盖，这里的覆盖是指实例的特征与路径上的特征一致或实例满足规则的条件）

决策树与条件概率分布

- 决策树还表示给定特征条件下类的条件概率分布，它定义在特征空间的一个划分。将特征空间划分为互不相交的单元，并在每个单元定义一个类的概率分布就构成了一个条件概率分布。决策树的每一条路径对应于划分中的一个单元。
- 假设X为表示特征的随机变量，Y为表示类的随机变量，那么这个条件概率分布可以表示为P(X|Y)，各叶结点上的条件概率往往偏向于某一个类，即属于某一类的概率越大。决策树分类时将该结点的实例强行分到条件概率大的那一类去。

信息熵

熵在信息论中代表随机变量不确定度的度量。一个离散型随机变量的熵定义为：

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$

直觉上，信息量等于传输该信息所用的代价，这个也是通信中考虑最多的问题。比如说：赌马比赛里，有4匹马 $\{A, B, C, D\}$ ，获胜概率分别为 $\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\}$ 。

接下来，让我们将哪一匹马获胜视为一个随机变量 $X \in \{A, B, C, D\}$ 。假定我们需要用尽可能少的二元问题来确定随机变量 X 的取值。

例如：问题1：A获胜了吗？问题2：B获胜了吗？问题3：C获胜了吗？最后我们可以通过最多3个二元问题，来确定 X 的取值，即哪一匹马赢了比赛。

如果 $X = A$ ，那么需要问1次（问题1：是不是A？），概率为 $\frac{1}{2}$ ；

如果 $X = B$ ，那么需要问2次（问题1：是不是A？问题2：是不是B？），概率为 $\frac{1}{4}$ ；

如果 $X = C$ ，那么需要问3次（问题1，问题2，问题3），概率为 $\frac{1}{8}$ ；

如果 $X = D$ ，那么同样需要问3次（问题1，问题2，问题3），概率为 $\frac{1}{8}$ ；

信息熵

那么很容易计算，在这种问法下，为确定 X 取值的二元问题数量为：

$$E(N) = \frac{1}{2} \cdot 1 + \frac{1}{4} \cdot 2 + \frac{1}{8} \cdot 3 + \frac{1}{8} \cdot 3 = \frac{7}{4}$$

那么我们回到信息熵的定义，会发现通过之前的信息熵公式，神奇地得到了：

$$H(X) = \frac{1}{2}\log(2) + \frac{1}{4}\log(4) + \frac{1}{8}\log(8) + \frac{1}{8}\log(8) = \frac{1}{2} + \frac{1}{2} + \frac{3}{8} + \frac{3}{8} = \frac{7}{4}\text{bits}$$

在二进制计算机中，一个比特为0或1，其实就代表了一个二元问题的回答。也就是说，在计算机中，我们给哪一匹马夺冠这个事件进行编码，所需要的平均码长为1.75个比特。

联合熵和条件熵

- 两个随机变量X，Y的联合分布，可以形成联合熵Joint Entropy，用H(X,Y)表示
- H(X,Y) – H(Y)
  - (X,Y)发生所包含的信息熵，减去Y单独发生包含的信息熵=在Y发生的前提下，X发生的信息熵
  - 该式子定义为Y发生前提下，X的熵：
    - 条件熵H(X|Y) = H(X,Y) – H(Y)

$$H(x|y) = -\sum_{i=1}^{n} p(x_i|y)\log_b p(x_i|y)$$

信息增益

- 概念：当熵和条件熵中的概率由数据估计(特别是极大似然估计)得到时，所对应的熵和条件熵分别称为经验熵和经验条件熵。
- 信息增益表示得知特征A的信息而使得类X的信息的不确定性减少的程度。
- 定义：特征A对训练数据集D的信息增益g(D,A)，定义为集合D的经验熵H(D)与特征A给定条件下D的经验条件熵H(D|A)之差，即：
  - g(D,A)=H(D) – H(D|A)

信息增益比

单纯的信息增益只是个相对值，因为这依赖于H(D)的大小，所以信息增益比更能客观地反映信息增益。特征A对训练数据集D的信息增益比gR(D,A)定义为其信息增益g(D,A)与训练数据集D关于特征A的值的熵HA(D)之比，即

$$g_R(D,A) = \frac{g(D,A)}{H_A(D)}$$

其中，$H_A(D) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \log_2 \frac{|D_i|}{|D|}$，n是特征A取值的个数。

决策树学习的生成算法

- 建立决策树的关键，即在当前状态下选择哪个属性作为分类依据。根据不同的目标函数，建立决策树主要有一下三种算法。
  - ID3
  - C4.5
  - CART

ID3生成算法

算法 5.2（ID3 算法）

输入：训练数据集 D，特征集 A，阈值 ε；

输出：决策树 T.

（1）若 D 中所有实例属于同一类 C_k，则 T 为单结点树，并将类 C_k 作为该结点的类标记，返回 T；

（2）若 $A=\varnothing$，则 T 为单结点树，并将 D 中实例数最大的类 C_k 作为该结点的类标记，返回 T；

（3）否则，按算法 5.1 计算 A 中各特征对 D 的信息增益，选择信息增益最大的特征 A_g；

（4）如果 A_g 的信息增益小于阈值 ε，则置 T 为单结点树，并将 D 中实例数最大的类 C_k 作为该结点的类标记，返回 T；

（5）否则，对 A_g 的每一可能值 a_i，依 $A_g=a_i$ 将 D 分割为若干非空子集 D_i，将 D_i 中实例数最大的类作为标记，构建子结点，由结点及其子结点构成树 T，返回 T；

（6）对第 i 个子结点，以 D_i 为训练集，以 $A-\{A_g\}$ 为特征集，递归地调用步 (1) ～步 (5)，得到子树 T_i，返回 T_i. ■

C4.5算法

- C4.5算法用信息增益比来选择属性，继承了ID3算法的优点。并在以下几方面对ID3算法进行了改进：
  - 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；
  - 在树构造过程中进行剪枝；
  - 能够完成对连续属性的离散化处理；
  - 能够对不完整数据进行处理。
- C4.5算法产生的分类规则易于理解、准确率较高；但效率低，因树构造过程中，需要对数据集进行多次的顺序扫描和排序。也是因为必须多次数据集扫描，C4.5只适合于能够驻留于内存的数据集。在实现过程中，C4.5算法在结构与递归上与ID3完全相同，区别只在于选取决决策特征时的决策依据不同，二者都有贪心性质：即通过局部最优构造全局最优。

C4.5算法

算法 5.3（C4.5 的生成算法）

输入：训练数据集 D，特征集 A，阈值 ε；

输出：决策树 T.

（1）如果 D 中所有实例属于同一类 C_k，则置 T 为单结点树，并将 C_k 作为该结点的类，返回 T；

（2）如果 $A=\varnothing$，则置 T 为单结点树，并将 D 中实例数最大的类 C_k 作为该结点的类，返回 T；

（3）否则，按式(5.10)计算 A 中各特征对 D 的信息增益比，选择信息增益比最大的特征 A_g；

（4）如果 A_g 的信息增益比小于阈值 ε，则置 T 为单结点树，并将 D 中实例数最大的类 C_k 作为该结点的类，返回 T；

（5）否则，对 A_g 的每一可能值 a_i，依 $A_g=a_i$ 将 D 分割为子集若干非空 D_i，将 D_i 中实例数最大的类作为标记，构建子结点，由结点及其子结点构成树 T，返回 T；

（6）对结点 i，以 D_i 为训练集，以 $A-\{A_g\}$ 为特征集，递归地调用步(1)～步(5)，得到子树 T_i，返回 T_i.

■

CART

- 分类树与回归树（classification and regression tree，CART）模型（Breiman）由特征选择、树生成及剪枝组成，既可用于分类也可用于回归。CART是在给定输入随机变量X条件下输出变量Y的条件概率分布的学习方法。它假定决策树是二叉树，内部取值为“是”（左分支）和“否”（右分支）。
- 基本步骤为
  - 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，这是用损失函数最小作为剪枝的标准。

分类树

- 具体算法步骤如下：

  1）设结点的训练数据集为D，计算现有特征对该数据集的基尼指数。此时，对每一个特征A，对其可能取的每个值a，根据样本点对A=a的测试为”是”或者“否”将D分割为D1和D2两部分，计算其基尼系数。

  2）在所有可能的特征A以及他们所有可能的切分点a中，选择基尼系数最小的特征及其对应的切分点作为最优特征与最优切分点。依最优特征与最优切分点，从现结点生成两个子结点，将训练数据集依特征分配到两个子结点中去。

  3）对两个子结点递归地调用上述两个步骤，直至满足停止条件。

  4）生成CART决策树

决策树的例子

表 1. 训练样本

Samples	A1	A2	A3	A4	A5	Class
1	1	3	2	2	2	2
2	2	1	2	2	1	2
3	1	3	2	1	1	1
4	2	1	2	2	3	1
5	1	3	1	2	2	2
6	2	1	2	1	1	1
7	1	2	2	2	2	2
8	2	1	1	1	1	1
9	1	2	2	2	3	2
10	1	3	2	1	3	2
11	1	1	2	2	3	1
12	2	2	2	1	1	1
13	2	3	2	1	1	1
14	1	2	1	2	2	2
15	2	3	2	1	3	1

表 2. 测试样本

Smaples	A1	A2	A3	A4	A5	Class
1	2	2	1	1	1	?
2	1	1	1	2	1	?

决策树的例子

极小熵生成决策树,设表1给的数据集为D，根据最大信息增益选择最优特征生成极小熵决策树,计算各特征A1、A2、A3、A4、A5对数据D的信息增益，统计结果为：

上表中的D1和D2，D3分别表示在各个特征中取值为1、2和3的样本子集，根据计算后统计在表格中的数据可得：

H(D)=-8/15*log2(8/15)—7/15*log2(7/15)=0.9968

g(D,A1)=H(D)-[8/15*H(D1)+7/15*H(D2)]=0.2880

g(D,A2)=H(D)-[5/15*H(D1)+4/15*H(D2)+6/15*H(D3)]=0.1398

g(D,A3)=H(D)-[3/15*H(D1)+12/15*H(D2)]=0.0292

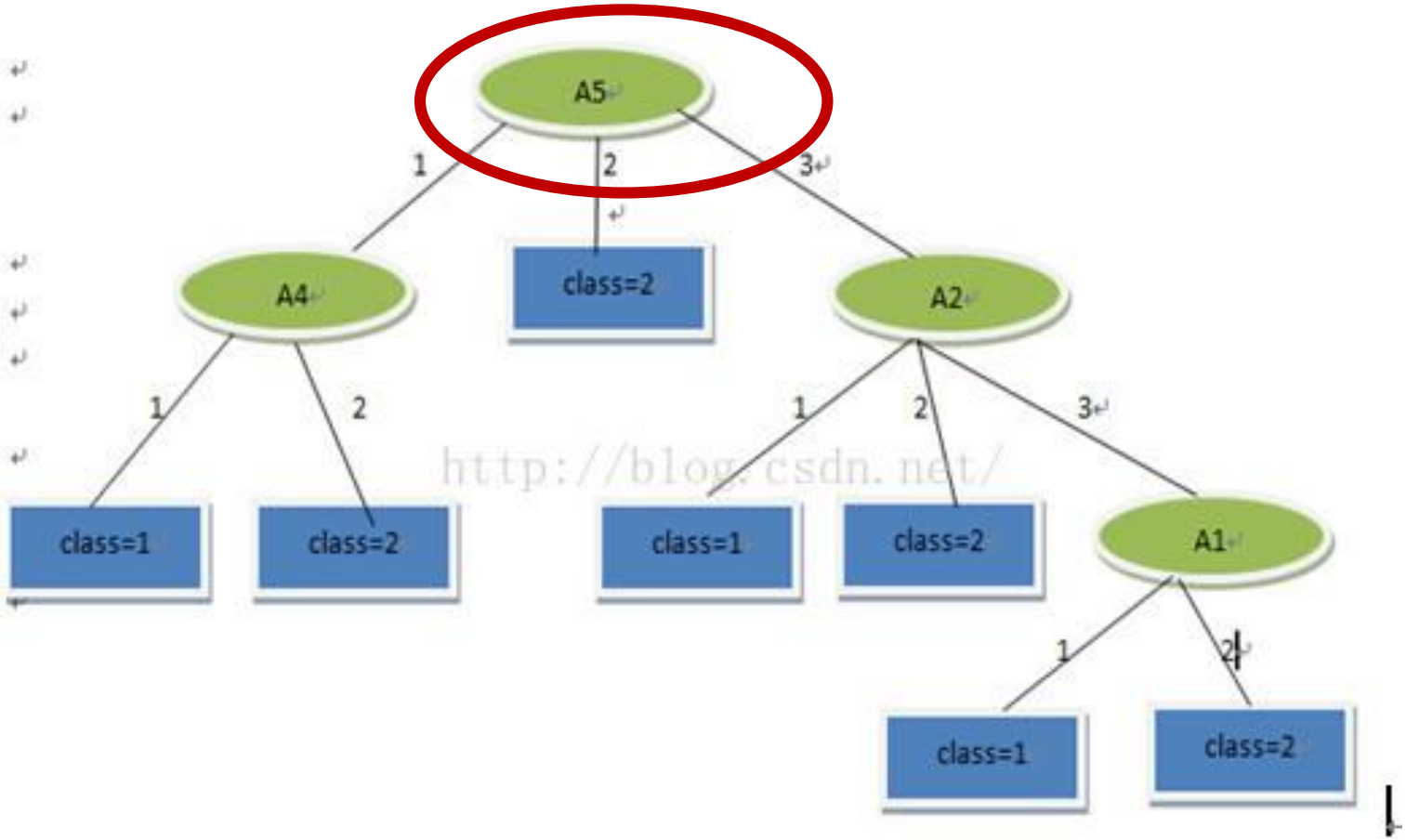
g(D,A4)=H(D)-[7/15*H(D1)+8/15*H(D2)]=0.2880

g(D,A5)=H(D)-[6/15*H(D1)+4/15*H(D2)+5/15*H(D3)]=0.4131

			class=1	class=2	
D			8	7	H(D)= 0.9968
A1	D1	8	2	6	g(D,A1)=0.2880
	D2	7	6	1	
A2	D1	5	4	1	g(D,A2)=0.1398
	D2	4	1	3	
	D3	6	3	3	
A3	D1	3	1	2	g(D,A3)=0.0292
	D2	12	7	5	
A4	D1	7	6	1	g(D,A4)=0.2880
	D2	8	2	6	
A5	D1	6	5	1	g(D,A5)=0.4131
	D2	4	0	4	
	D3	5	3	2	

决策树的例子

所以选择A1作为集合S33的根节点。根据A1的取值划分后的集合也都为叶子节点，至此极小熵决策树就建立起来了，如下图所示。



决策树的例子

根据上面的计算结果，特征A5的信息增益最大，所以选择A5为根节点。根据A5的取值将样本分成3个集合，S1={2,3,6,8,12,13}，S2={1,5,7,14}，S3={4,9,10,11,15}其中集合S2已全部属于同一个类，不需要再分，已成为叶子节点。对于集合S1，计算统计结果为：

H(D)=0.6500

g(D,A1)= 0.0484

g(D,A2)= 0.1909

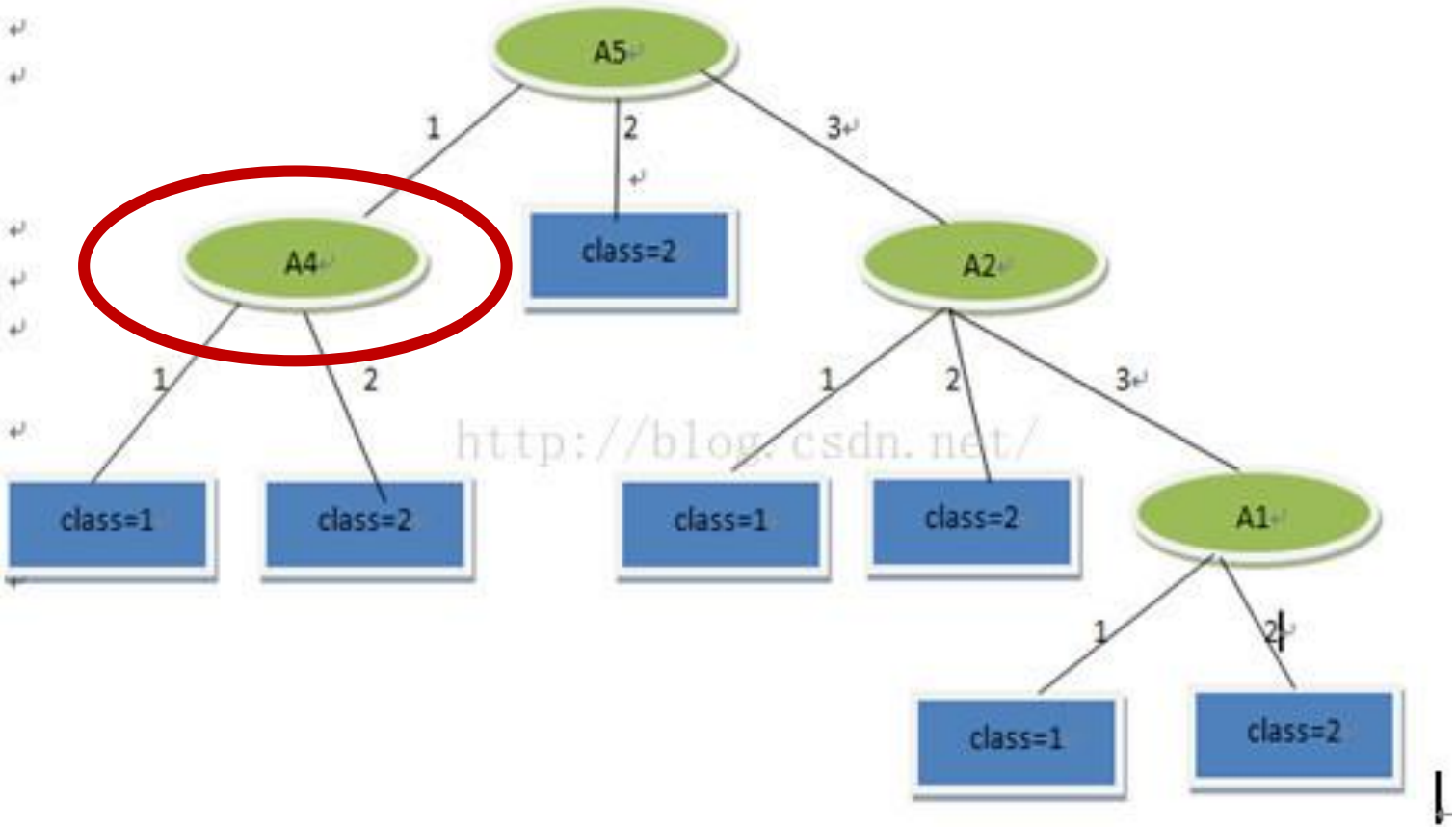
g(D,A3)= 0.0484

g(D,A4)=0.6500

			class=1	class=2	
D			5	1	H(D)=0.6500
A1	D1	1	1	0	G(D,A1)=0.0484
	D2	5	4	1	
A2	D1	3	2	1	G(D,A2)=0.1909
	D2	1	1	0	
	D3	2	2	0	
A3	D1	1	1	0	G(D,A3)=0.0484
	D2	5	4	1	
A4	D1	5	5	0	G(D,A4)=0.6500
	D2	1	0	1	

决策树的例子

所以选择A1作为集合S33的根节点。根据A1的取值划分后的集合也都为叶子节点，至此极小熵决策树就建立起来了，如下图所示。



决策树的例子

根据计算结果，集合S1选择A4为根结点。根据A4的取值，将S1集合划分为S11={3,6,8,12,13} S12={2}，集合S11和集合S12已成为叶节点。对于集合S3，计算统计结果为：

H(D)=0.9710

g(D,A1)=0.4200

g(D,A2)=0.5710

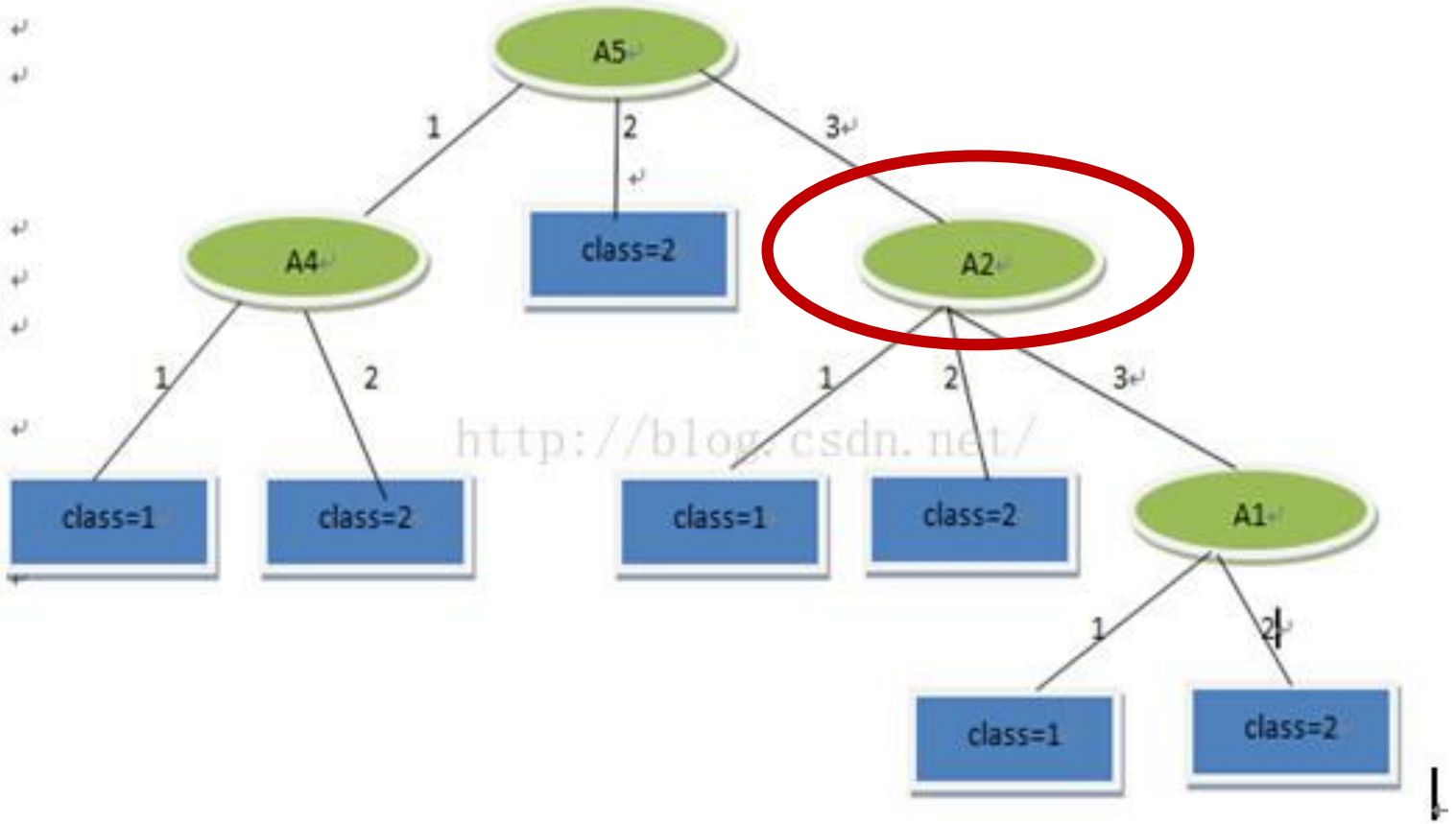
g(D,A3)=0

g(D,A4)=0.0200

			class 1	class 2	
D			3	2	H(D)=0.9710
A1	D1	3	1	2	G(D,A1)=0.4200
	D2	2	2	0	
A2	D1	2	2	0	G(D,A2)=0.5710
	D2	1	1	0	
	D3	2	1	1	
A3	D1	0	0	0	G(D,A3)=0
	D2	5	3	2	
A4	D1	2	1	1	G(D,A4)=0.0200
	D2	3	2	1	

决策树的例子

所以选择A1作为集合S33的根节点。根据A1的取值划分后的集合也都为叶子节点，至此极小熵决策树就建立起来了，如下图所示。



决策树的例子

根据计算结果，所以集合S3选择A2作为根结点，根据A2的取值将S3分成集合S31={4,11}集合S32={9}集合S33={10,15} 集合S32和集合S32已为叶子节点。对于集合S33，计算统计结果为

H(D)=1

g(D,A1)=1

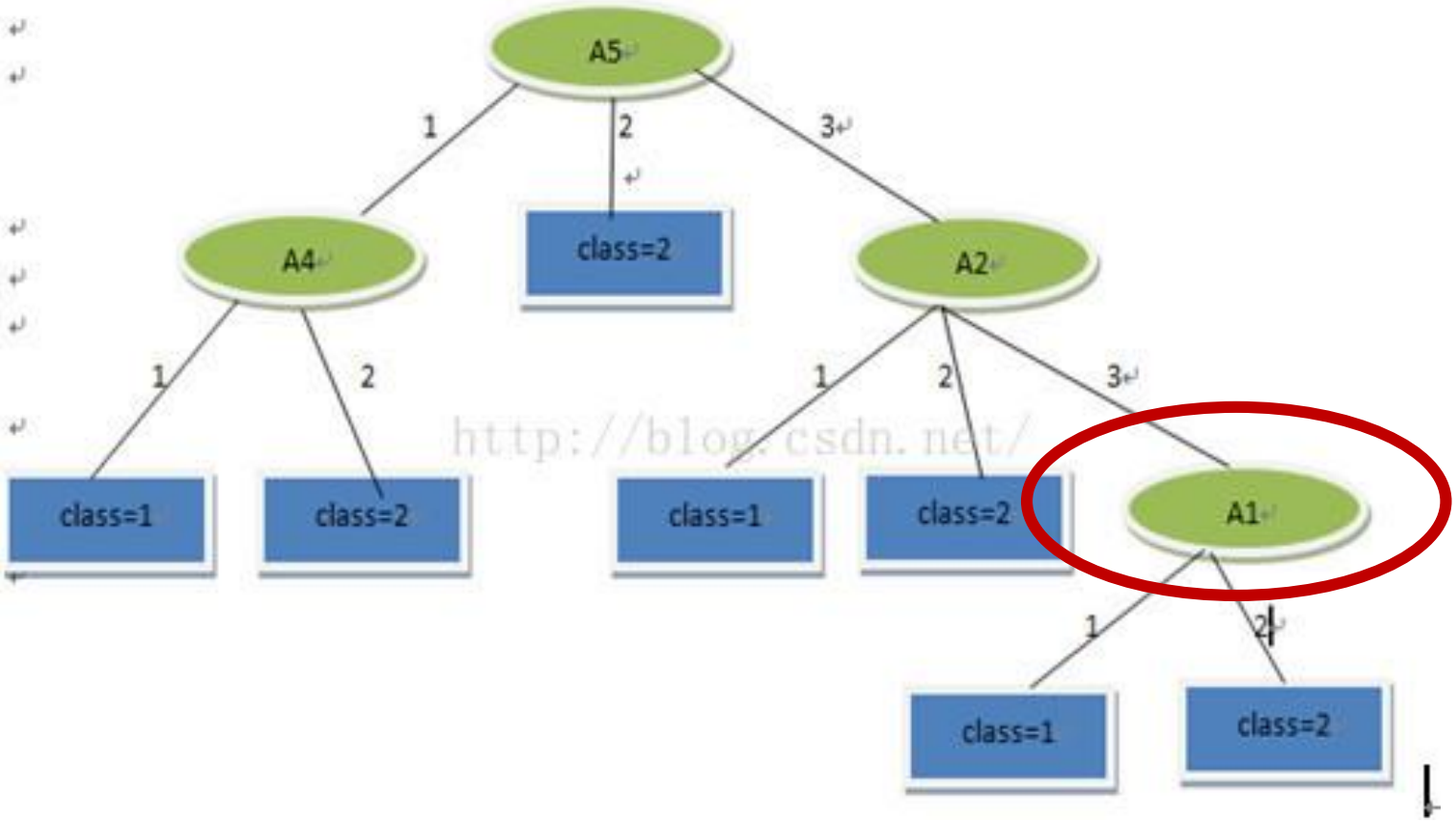
g(D,A3)=0

g(D,A4)=0

			class 1	class 2	
D			1	1	H(D)=1
A1	D1	1	0	1	G(D,A1)=1
	D2	1	1	0	
A3	D1	0	0	0	G(D,A3)=0
	D2	2	1	1	
A4	D1	2	1	1	G(D,A4)=0
	D2	0	0	0	

决策树的例子

所以选择A1作为集合S33的根节点。根据A1的取值划分后的集合也都为叶子节点，至此极小熵决策树就建立起来了，如下图所示。



决策树的剪枝

决策树生成算法对于训练集是很准确的，也就是生成的树枝很详细，但这样会过拟合，需要通过剪枝操作来提高泛化能力，思路很简单，就是在决策树对训练集数据的预测误差和树复杂度之间找一个平衡。

预测误差就是所有叶子节点的经验熵的和，其中 N_t 表示该叶节点的样本点个数，而 $H_t(T)$ 表示该叶子节点的经验熵：

$$C(T)=\sum_{t=1}^{|T|}N_tH_t(T)=-\sum_{t=1}^{|T|}\sum_{k=1}^{K}N_{tk}\log\frac{N_{tk}}{N_t}$$

树的复杂度由叶子节点的个数来表示：T。所以，剪枝的标准就是极小化损失函数：

$$C_a(T)=C(T)+a|T|$$

其中 a 是调节参数。其越大表示选择越简单的树，而越小表示选择越复杂的树，对训练集拟合度高的树。

树的剪枝算法就是从叶子节点往上回溯，比较剪掉该叶子节点前后的损失函数的值。如果剪掉该叶子节点后，损失函数更小就剪掉，

决策树的例子

如果选择极大熵、最小增益则生成的决策树为：