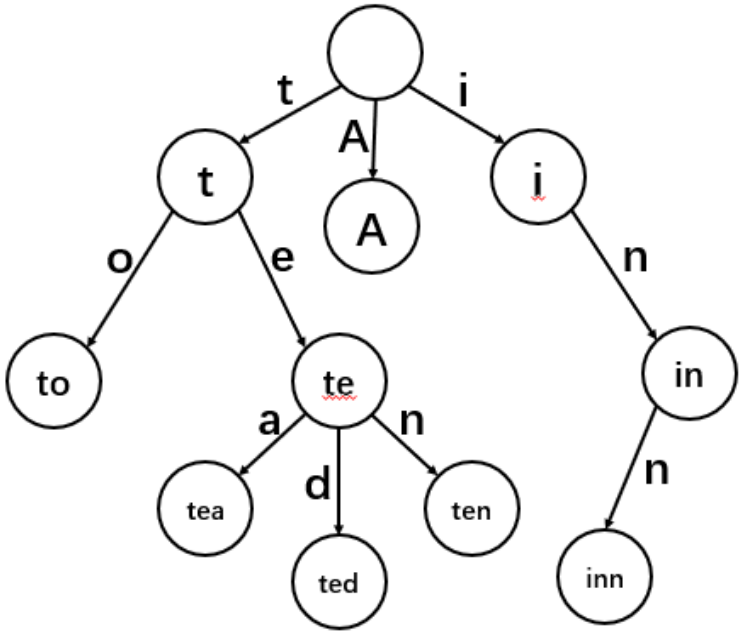
树的扩展应用

拓展延伸：前缀树

前缀树：又名Trie树、字典树、单词查找树，是专门处理**字符串匹配**的树形结构

典型应用：可以存储大量的字符串并从中快速查找指定的字符串，所以经常被搜索引擎系统用于**文本词频统计**



前缀树

- 逻辑结构上，前缀树是一棵k叉树，k通常等于构成字符串的字符集规模
- 结点的每个分支对应字符集中唯一的一个字符
- 从根到各结点的路径，路径经过的分支序代表结点对应的字符串
- 每个结点对应的字符串不同，因此前缀树把所有字符串的共同前缀合并在一条路径上表示，从而最大限度地减少多余的字符串比较

拓展延伸：前缀树

前缀树是一种以空间换时间的算法

前缀树的时间效率

(1) 前缀树的2个基本操作：插入字符串、查找是否存放有指定的字符串

(2) 插入字符串以及查询字符串的时间开销均为O(n), n是字符串的长度，与前缀树本身的规模无关

(3) 前缀树各结点的数据不仅可以存放字符串，还能用于计数，比如统计其对应的字符串是多少个存放的字符串的公共前缀

前缀树的空间开销

(1) 在插入新字符串时，字符串的每个字符都有可能创建一个新的（k叉树）结点，导致前缀树的内存消耗会非常大

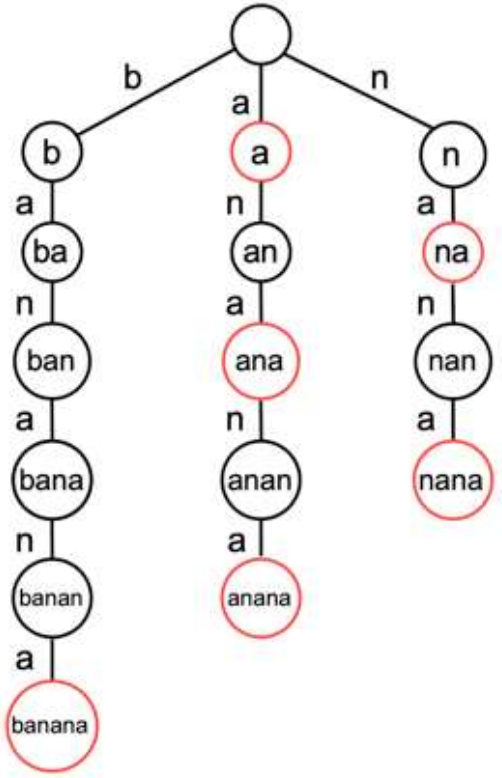
(2) 最坏情况下，前缀树的空间复杂度可达O(KN)，其中N为所有字符串的长度之和，K为字符集规模

(3) 在实际应用中，前缀树通常用二维数组（顺序变）来实现

拓展延伸：后缀树与后缀自动机

后缀树：用一个字符串的所有后缀构建的**前缀树**

典型应用：最长重复子串问题、最长公共子串问题、精确字符匹配问题等



字符串“banana”构成的后缀树

- 使用字符串的所有后缀构建前缀树，可以在树上保留所有子串信息，由此大幅提升子串的查询效率
- 查询字符串S中是否包含字符串P，只需在由S构建的后缀树上检索P，并且查询的时间复杂度为O(|P|)，与后缀树规模无关，比KMP等算法效率高
- 建立字符串S的后缀树的时间与空间复杂度均为$O(K|S|^2)$， K是字符集规模
- 合并树中的相似结构，可以大幅压缩后缀树的规模，最终形成一个点和边的规模均为 O(|S|) 的有向无环图，这就是**后缀自动机**

应用场景：决策树

决策树：一种解决分类问题的算法

实例：

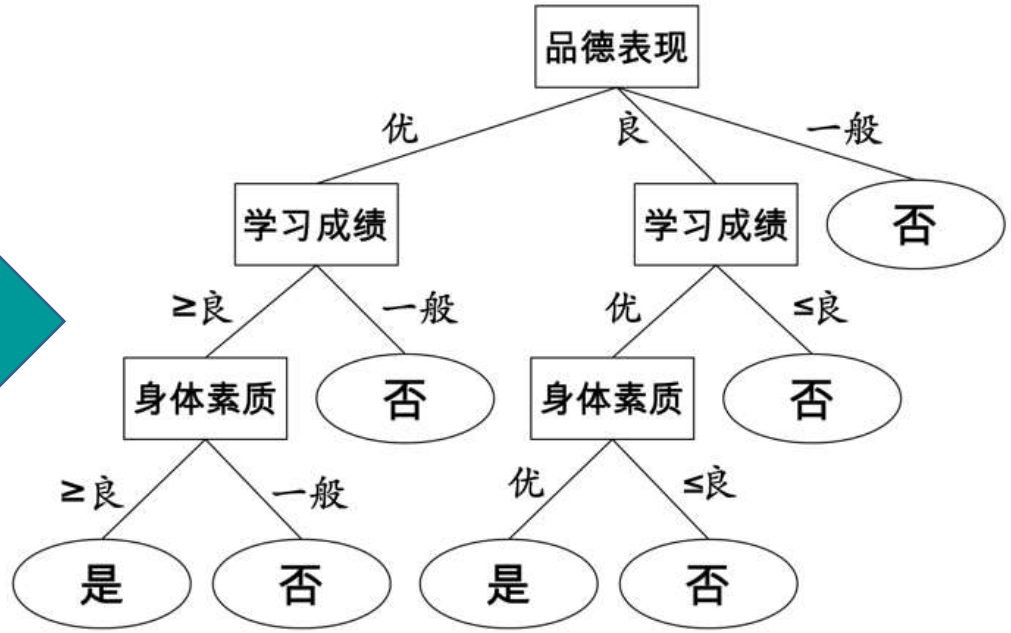
某高校的优秀学生选拔标准

1）学生在德、智、体三个方面都取得良以上的成绩

2）按照“为学须先立志”的原则，品行优异的学生可评为优秀

3）对于品德表现良好的学生，必须“文武全才”，即学习成绩和身体素质皆优

转换

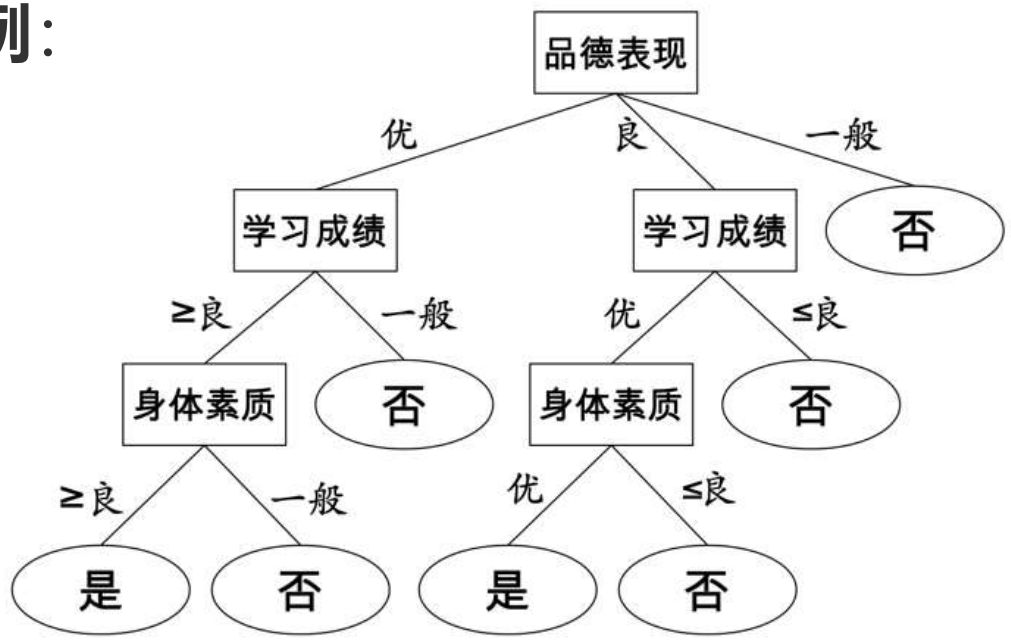


决策树

应用场景：决策树

决策树：一种解决分类问题的算法

实例：



决策树

决策树

- 每个中间结点代表一个特征属性，每个分支代表一个属性值
- 中间结点对属性值进行测试，根据判断结果决定进入下面哪个子结点
- 叶结点代表最终的决策

应用 对学生进行分类

- 根结点包含了所有学生，而每个中间结点包含一个学生集合（子集）
- 根据特征属性的测试结果将集合划分给各个子结点
- 每个叶结点存放一个类别，表示最终的分类结果

应用场景：决策树

决策树：一种解决分类问题的算法

决策树的构建：给定一组训练数据，每个数据包含多个特征属性并且带有表示类别的标记。从训练数据集中归纳出一组分类规则，并由此构造一个决策树，使它能够对训练数据执行正确的分类，也可用于预测新数据的类别。

决策树构建算法：经典的决策树生成算法有ID3、C4.5与CART，其中ID3是最早提出的机器学习算法

ID3算法是一种贪心法，其核心是“信息熵”。假设样本数据集C包含k个独立的类别，每个类别构成C的一个子集C_j $(1 \le j \le k)$，则数据集C的总信息熵为：

$$H(C) = -\sum_{j=1}^{k} \frac{|C_j|}{|C|} \log \frac{|C_j|}{|C|}$$

ID3算法

ID3算法流程：

(1) 依次计算各特征属性的信息增益度，如果没有特征可选或信息增益量小于阈值，结束计算；否则执行(2)的操作

(2) 选择信息增益度最大的特征属性作为决策树结点，对特征值区间进行划分并建立子结点，每个子结点对应不同的特征值

(3) 把数据集按特征值划分给每个子结点

(4) 对各子结点重复执行(1)的操作。

信息增益

设特征属性A的取值为$\{a_1, a_2, \ldots, a_m\}$ $(m \geq 1)$。用D_{a_i}表示数据集C中属性A取值a_i $(1 \leq i \leq m)$的所有数据集合，对D_{a_i}按类别标记进行分类，可得信息熵：

$$H(D_{a_i}) = -\sum_{j=1}^{k} \frac{|D_{a_i} \cap C_j|}{|D_{a_i}|} \log \frac{|D_{a_i} \cap C_j|}{|D_{a_i}|}$$

由此计算属性A对数据集C的条件熵：

$$H(C, A) = \sum_{i=1}^{m} \frac{|D_{a_i}|}{|C|} H(D_{a_i})$$

信息增益定义为$H(C)$与$H(C,A)$的差值，即

$$Gain(A) = H(C) - H(C, A)$$

$Gain(A)$表示对数据集先按特征属性A进行划分后，对判断任意数据属于哪个类别所需信息量的减少程度。

ID3算法

ID3算法实例

学生的成绩单及分类

学生ID	品德表现	学习成绩	身体素质	是否优秀
1	优	一般	良	否
2	优	良	良	是
3	良	优	一般	否
4	良	优	优	是
5	良	良	优	否
6	优	优	良	是
7	一般	一般	一般	否
8	一般	优	优	否
9	一般	良	一般	否
10	优	优	优	是

全体学生S中有4名优异生、6名非优异生，总信息熵

$$H(S) = -\frac{4}{10}\times\log\left(\frac{4}{10}\right) - \frac{6}{10}\times\log\left(\frac{6}{10}\right) = 0.971$$

按品德表现划分学生，把学生分成三组，即品行优异组、品行良好组以及品行一般组，各组的信息熵为：

$$H(T_{优}) = -\frac{3}{4}\times\log\left(\frac{3}{4}\right) - \frac{1}{4}\times\log\left(\frac{1}{4}\right) = 0.811$$

$$H(T_{良}) = -\frac{1}{3}\times\log\left(\frac{1}{3}\right) - \frac{2}{3}\times\log\left(\frac{2}{3}\right) = 0.918$$

$$H(T_{一般}) = -\frac{0}{3}\times\log\left(\frac{0}{3}\right) - \frac{3}{3}\times\log\left(\frac{3}{3}\right) = 0.0$$

根据上面三个信息熵，可以算出按照品德表现进行划分后，S的条件熵

$$H(S,T) = \frac{4}{10}\times H(T_{优}) + \frac{3}{10}\times H(T_{良}) + \frac{3}{10}\times H(T_{一般}) = 0.6$$

同理，按学习成绩和身体素质划分后，S的条件熵分别是0.761和 0.675。由此可见，品德表现的信息增益度最大

因此，ID3算法选择品德表现这一特征属性作为决策树的根结点，其三个子结点分别对应$T_{优}$、$T_{良}$以及$T_{一般}$这三个子集；然后对各子结点继续进行拆分。

应用场景：决策树

决策树：一种解决分类问题的算法

决策树构建算法：经典的决策树生成算法有ID3、C4.5与CART，其中ID3是最早提出的机器学习算法

- ID3算法只能处理离散型特征，同时信息增益倾向于选择取值较多的属性。
- 针对ID3算法的缺陷，C4.5算法引入信息增益率来作为分类标准，能够处理连续数值型特征属性，同时在决策树构造过程中进行剪枝
- 与C4.5算法相比，CART算法采用了简化的二叉树模型，同时特征选择采用了近似的基尼系数来简化计算，该算法还可用于回归

总结：决策树算法是机器学习中常用的模型，易于理解，可解释性强，可以同时处理数值型和非数值型数据，能够处理关联度低的特征属性，符合人类的直观思维。但容易发生过拟合的现象，容易忽略特征属性之间的关联，并且预测精度易受异常数据的影响。