# 分类

决策树是解决分类问题一种强大的方法。

## 分类的基本原理

假设有一个样本数据集：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| temp | wind | holidays | award\_date | sports | weekday | grade |
| 1 | 4.5 | 3 | 7 | 1 | 7 | 略大 |
| -2.5 | 4.5 | 3 | 1 | 1 | 1 | 小 |
| -0.5 | 4.5 | 0 | 2 | 1 | 2 | 很大 |
| -0.5 | 4.5 | 0 | 1 | 1 | 3 | 小 |
| -0.5 | 4.5 | 0 | 4 | 1 | 4 | 大 |
| -0.5 | 4.5 | 0 | 1 | 1 | 5 | 很小 |
| 0 | 4.5 | 1 | 1 | 1 | 6 | 小 |
| -6 | 6 | 2 | 7 | 1 | 7 | 大 |
| -7 | 6.5 | 0 | 1 | 1 | 1 | 小 |
| -8 | 6 | 0 | 2 | 1 | 2 | 大 |
| -7 | 5 | 0 | 1 | 1 | 3 | 略小 |
| -7.5 | 5 | 0 | 4 | 1 | 4 | 大 |
| -10 | 6.5 | 0 | 1 | 1 | 5 | 很小 |

给定一个数据样本集，包含描述样本属性的特征向量（表中temp、weekday等字段）和样本所属的类别（表中grade字段）。

分类是一个构建分类器的过程，通过该分类器把具有某些特征的一系列样本映射到一个预定义的类中。其中分类器是在样本的其它属性已知时预测样本所属的类的模型。

通过已知属性以及已知属性对应的类别，可以构建分类器，通过构建好的分类器，当给定属性时，我们可以预知该属性对应的类别。

在机器学习中，分类的过程不需要我们了解，只需要提供数据集机器会通过“学习”构建好分类器。

## **数据挖掘分类模型的建模方法：**

1. 与相关的专家或专家组会谈，可以得到分类模型。大多基于知识的系统是通过该方法构造的。
2. 检查大量已记录的分类，归纳概括数据挖掘应用的主要示例，得到分类模型。

## **分类问题的建模类型：**

1. 总结样本集的统计属性。
2. 基于逻辑，这些建模方法的分类结果包含更便于解释的特征，使用者容易理解推理模型。

决策树和决策规则是以逻辑模型的方式给出结果的典型数据挖掘技术

## **应用分类的例子：**

金融市场中走向的分类，大型图形数据库中对象的识别。

# 什么是决策树

决策树是一种基于归纳的学习方法，代表的是一种对象属性与对象值之间的一种映射关系。

决策树是有监督学习的分层模型，它通过待检验函数的决策节点，在一系列递归的分支处识别出局部区间。

决策树也是一个非参数化模型，它没有给类的密度假设任何参数形式。

典型的决策树学习系统采用自上而下的方法，在部分收拾空间中搜索解决方案。它可以确保求出一个简单的决策树，但未必是最简单的。

简单的决策树例子：

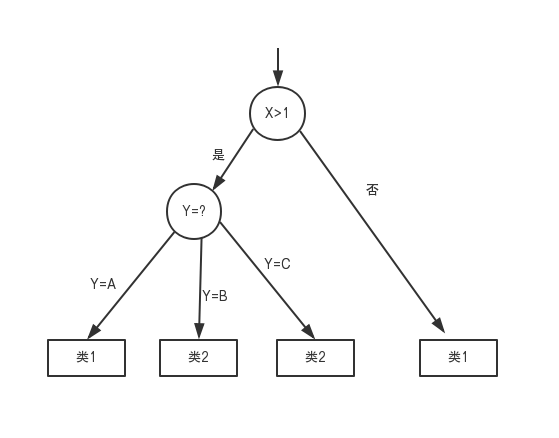


图1： 检验属性X和Y的简单决策树

图1中的简单决策树对有两个输入属性X和Y的样本进行分类。所有属性值X>1和Y=B的样本都属于类2。不论Y的值为多少X≤1的样本都属于类1.

决策树可以用来对新样本分类，这种分类从树的根节点开始移动样本，直至移动到叶节点为止。在每个非叶决策节点处（如Y），都要确定该节点的特征检验结果（使用方法：ID3熵, cart 基尼），然后考虑所选子树的根节点。

决策树每一个节点代表某个对象，树中的每一个分叉路径代表某个可能的属性值，而每一个叶子节点则对应从根节点到该叶子节点所经历的路径所表示的对象的值。

到达决策树的叶节点的每条路径都表示一个分类规则。

决策树仅有单一输出，如果有多个输出，可以分别建立独立的决策树以处理不同的输出。

运用基于归纳的学习方法，必须满足关键要求：

1. 属性描述

描述样本的属相可以离散或者数值型，必须不同，表示对象或样本的所有信息

1. 预定义的类
2. 离散的类
3. 充足的数据
4. 逻辑分类模型

# 决策树算法

## ID3算法

ID3算法是决策树的一种，它是基于奥卡姆剃刀原理的，即用尽量用较少的东西做更多的事。ID3算法，即Iterative Dichotomiser 3，迭代二叉树3代，越是小型的决策树越优于大的决策树。

ID3算法的核心思想就是以信息增益来选择需要检验的属性，它基于信息论的熵。选择分裂后信息增益最大的属性进行分裂。

ID3算法采用自顶向下的贪婪搜索来遍历可能的决策空间。

ID3和CD4.5算法的属性选择基准都是使样本中节点所包含的信息熵最小化。在数据库中对样本进行分类时，所做检验的数量最小。

### ID3算法的自顶向下和贪婪搜索：

1. **自顶向下：**

ID3算法在开始运行时，所有训练样本都位于树的根节点。该算法选取一个属性来对这些样本分区，为每一个属性值创建一个分支，如果某样本自己的属性等于分支指定的值，该样本自己就移到新生成的子节点上。

自上而下的算法，最重要的是对节点属性的选择。

1. **无回溯贪婪搜索:**

这个算法递归地应用于每个子节点，直到一个节点上的所有样本都属于一个类为止。

只要通过试探法选择了某个使进度最大化的检验，且当前的训练样本集已分区好，就不会去探索其他选择。

### 信息熵与信息增益

**1. 信息熵**

熵这个概念最早起源于物理学，在物理学中是用来度量一个热力学系统的无序程度，而在信息学里面，熵是对不确定性的度量。

在1948年，香农引入了信息熵，将其定义为离散随机事件出现的概率，一个系统越是有序，信息熵就越低，反之一个系统越是混乱，它的信息熵就越高。所以信息熵可以被认为是系统有序化程度的一个度量。

**2. 信息增益**

信息增益是针对一个一个特征而言的，就是看对于一个特征，系统有它和没有它时的信息量各是多少，两者的差值就是这个特征给系统带来的信息量，即信息增益。

ID3属性选择

ID3属性选择是根据一个假设：决策树的复杂度和所给属性值表达的信息量是密切相关的。基于信息的试探法会选择信息量最大的属性，即这个属性可以最小化把样本分类到结果子树中所需的信息。

**3. 信息熵与信息增益的计算**

假设选择有n个输出（所有特征的n个值）的检验，把训练样本集T分区成子集T1 ,T2 ,..., Tn。仅有的指导信息是类在T和他的子集Ti中的分布。如果S是任意样本集，设freq(Ci, S）代表S中属于类Ci的样本数量， |S|表示集S中的样本数量。

计算集合T中的熵（单位为比特）：



按照一个属性检验X的n个输出分区T后，现在考虑另一个相似的度量标准。所需信息可通过这些子集的熵的加权和求得：



信息增益为：



表示按照检验X分区T所得到的的信息。该增益标准选择了使Gain(X)最大化的检验X，即此标准选择的是具有最高信息增益的属性。

**4.度量标准的使用创建决策树示例**

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

C4.5算法，C5.0算法， CART算法

C4.5算法

<https://www.cnblogs.com/starfire86/p/5791264.html>

C4.5算法把分类范围从类别属性扩展到数值属性，这个度量标准使用的属性能把数据区分成类熵较低的子集，即该子集中的大部分样本都属于一个类。这个算法选择的属基本上可以使类之间的区别程度为局部最大。

C4.5 是 ID3 的后继者，并且通过动态定义将连续属性值分割成一组离散间隔的离散属性（基于数字变量），消除了特征必须被明确分类的限制。C4.5 将训练的树（即，ID3算法的输出）转换成 if-then 规则的集合。然后评估每个规则的这些准确性，以确定应用它们的顺序。如果规则的准确性没有改变，则需要决策树的树枝来解决。

C5.0 是 Quinlan 根据专有许可证发布的最新版本。它使用更少的内存，并建立比 C4.5 更小的规则集，同时更准确。

CART（Classification and Regression Trees （分类和回归树））与 C4.5 非常相似，但它不同之处在于它支持数值目标变量（回归），并且不计算规则集。CART 使用在每个节点产生最大信息增益的特征和阈值来构造二叉树。

决策树停止的条件，ID3算法的收敛

1. 待分割数据都为同一类

2. 没有新的属性进行节点分割

3. 没有任何未处理的数据

理想的决策树有三种：

1.叶子节点数最少

2.叶子节点深度最小

3.叶子节点数最少且叶子节点深度最小。

# 决策树算法的问题

过度拟合问题

造成多度拟合的潜在原因主要以下两个方面

1.噪声导致的过度拟合

比如错误的分类，或者属性值。

2.缺乏代表性样本所导致的过度拟合

## 解决办法

https://www.cnblogs.com/starfire86/p/5749334.html 剪枝

### 1. 预剪枝

通过提前停止树的构建而对树剪枝，一旦停止，节点就是树叶，该树叶持有子集元祖最频繁的类。

停止决策树生长最简单的方法有：

1.定义一个高度，当决策树达到该高度时就停止决策树的生长

2.达到某个节点的实例具有相同的特征向量，这些实例不属于同一类，也可以停止决策树的生长。这个方法对于处理数据的数据冲突问题比较有效。

3.定义一个阈值，当达到某个节点的实例个数小于阈值时就可以停止决策树的生长

4.定义一个阈值，通过计算每次扩张对系统性能的增益，并比较增益值与该阈值大小来决定是否停止决策树的生长。

### 2. 后剪枝

后剪枝（postpruning）：它首先构造完整的决策树，允许树过度拟合训练数据，然后对那些置信度不够的结点子树用叶子结点来代替，该叶子的类标号用该结点子树中最频繁的类标记。相比于先剪枝，这种方法更常用，正是因为在先剪枝方法中精确地估计何时停止树增长很困难。

后剪枝方法主要有以下几个方法：

1. Reduced-Error Pruning(REP,错误率降低剪枝）

REP方法是一种比较简单的后剪枝的方法，在该方法中，可用的数据被分成两个样例集合：一个训练集用来形成学习到的决策树，一个分离的验证集用来评估这个决策树在后续数据上的精度，确切地说是用来评估修剪这个决策树的影响。

2. Pesimistic-Error Pruning(PEP,悲观错误剪枝）

悲观错误剪枝法是根据剪枝前后的错误率来判定子树的修剪，悲观剪枝的准确度比较高

3. Cost-Complexity Pruning（CCP，代价复杂度剪枝)

该算法为子树Tt定义了代价（cost）和复杂度（complexity），以及一个可由用户设置的衡量代价与复杂度之间关系的参数α

4. EBP(Error-Based Pruning)（基于错误的剪枝）

# 决策树规则的局限性

# 决策树实现

## sklearn 决策树实

- 对于拥有大量特征的数据决策树会出现过拟合的现象。获得一个合适的样本比例和特征数量十分重要，因为在高维空间中只有少量的样本的树是十分容易过拟合的。

- 考虑事先进行降维( PCA , ICA ，使您的树更好地找到具有分辨性的特征

- 通过 export 功能可以可视化您的决策树。使用 max\_depth=3 作为初始树深度，让决策树知道如何适应您的数据，然后再增加树的深度。

- 充树的样本数量会增加树的每个附加级别。使用 max\_depth 来控制输的大小防止过拟合。

- 通过使用 min\_samples\_split 和 min\_samples\_leaf 来控制叶节点上的样本数量。当这个值很小时意味着生成的决策树将会过拟合，然而当这个值很大时将会不利于决策树的对样本的学习。所以尝试 min\_samples\_leaf=5 作为初始值。如果样本的变化量很大，可以使用浮点数作为这两个参数中的百分比。两者之间的主要区别在于 min\_samples\_leaf 保证叶结点中最少的采样数，而 min\_samples\_split 可以创建任意小的叶子

- 在训练之前平衡您的数据集，以防止决策树偏向于主导类.可以通过从每个类中抽取相等数量的样本来进行类平衡，或者优选地通过将每个类的样本权重 (sample\_weight) 的和归一化为相同的值。基于权重的预修剪标准 (min\_weight\_fraction\_leaf) 对于显性类别的偏倚偏小，而不是不了解样本权重的标准，如 min\_samples\_leaf 。

- 如果样本被加权，则使用基于权重的预修剪标准 min\_weight\_fraction\_leaf 来优化树结构将更容易，这确保叶节点包含样本权重的总和的至少一部分。

- 所有的决策树内部使用 np.float32 数组 ，如果训练数据不是这种格式，将会复制数据集。

- 如果输入的矩阵X为稀疏矩阵，建议您在调用fit之前将矩阵X转换为稀疏的csc\_matrix ,在调用predict之前将 csr\_matrix 稀疏。当特征在大多数样本中具有零值时，与密集矩阵相比，稀疏矩阵输入的训练时间可以快几个数量级。

# 决策树应用场景

统计文献

参考资料：

<https://www.cnblogs.com/starfire86/p/5749328.html>

《数据挖掘：概念、模型、方法和算法》