**第五章 决策树**

决策树（Decision Tree）是一个有趣的模型，它的建模思路是尽量模拟人做决策的过程。因此，决策树与其它机器学习模型不同，它几乎没有任何数学抽象，完全通过生成决策规则来解决分类和回归问题。

这个模型在学术上被称为白盒模型（White Box Model），因为它的整个运行机制能很直接地被翻译成人类语言，即使对这个领域完全不了解的人也能很好地理解它。

大多数机器学习系统需要能够为利益相关者解释为何做出这样特定的预测。 在选择合适的机器学习模型时，我们通常会在准确性与可解释性之间权衡。

准确与“黑盒”：

诸如神经网络、梯度增强模型或复杂集成学习之类的黑盒模型通常提供很高的准确性。 这些模型的内部工作难以理解，并且它们不能估计每个特征对模型预测的重要性，也不容易理解不同特征之间如何相互作用。

较弱的“白盒”：

另一方面，诸如线性回归和决策树之类的简单模型具备较弱的预测能力。 但是这些却是很容易被理解和解释的。

比如一家保险公司有一个汽车保险的定价项目。数据科学家尝试了两类模型：一类是容易解释的逻辑回归和决策树模型，另一类是较为复杂的随机森林模型。随机森林模型的预测效果更好，如果将其投入生产，每年能产生数千万的利润。但是随机森林模型难以解释，监管部门根本不接受，所以只能退而求其次，使用效果较差但更容易解释的决策树模型。



图1 流程图形式的决策图

图1所示的流程图就是一个决策树。长方形和椭圆形表示树的一个节点。长方形代表判断模块（Decision Block），中间节点（Internal Node）。椭圆形代表终止模块（Terminating Block），叶子节点（Leaf Node），表示已经得出结论，可以终止运行。从判断模块引出的左右箭头称作分支（Branch），它可以达到另一个判断模块或者终止模块。

决策树的一个重要任务是为了理解数据中所蕴含的知识信息，因此决策树可以使用不熟悉的数据集合，并从中提取出一些列规则，这些机器根据数据集创建规则的过程，就是机器学习的过程。

**决策树模型的核心是决策规则**。

划分规则的指标有三种：信息增益、信息增益率和基尼系数（Gini）。基于这三种指标，构造决策树的算法有：ID3 算法（信息增益）、C4.5 算法（信息增益率）以及CART 算法（基尼系数，Classification and Regression Trees）。

其中信息增益和信息增益率都是基于信息熵，这里面就会涉及大量的对数运算。

**信息熵**

集合信息的度量方式称为香农熵或熵，这个名字来源于信息论之父克劳德·香农。



图2 克劳德·艾尔伍德·香农（Claude Elwood Shannon，1916年4月30日—2001年2月24日），是美国数学家、信息论的创始人。

信息论之父克劳德·香农给出的信息熵的三个性质：

[1] 单调性，发生概率越高的事件，其携带的信息量越低；

[2] 非负性，信息熵可以看作为一种广度量，非负性是一种合理的必然；

[3] 累加性，即多随机事件同时发生存在的总不确定性的量度是可以表示为各事件不确定性的量度的和，这也是广度量的一种体现。

熵定义为信息的期望值。如果待分类的事物可能划分在多个类之中，则符号​的信息定义为：

其中，是选择该分类的概率。

为了计算熵，我们需要计算所有类别所有可能值所包含的信息期望值，通过下式得到：

其中，为分类数目。熵越大，随机变量的不确定性就越大。

对于确定性问题，信息熵为0，

当

信息熵最大，即

每个事件概率相同时，熵最大，这件事越不确定。

也就是平均信息量，或者说对不确定度的度量。

信息熵特点：（以概率和为1为前提）

　　a) 不同类别的概率分布越均匀，信息熵越大；

　　b) 类别个数越多，信息熵越大；

　　c) 信息熵越大，越不容易被预测；（变化个数多，变化之间区分小，则越不容易被预测）。

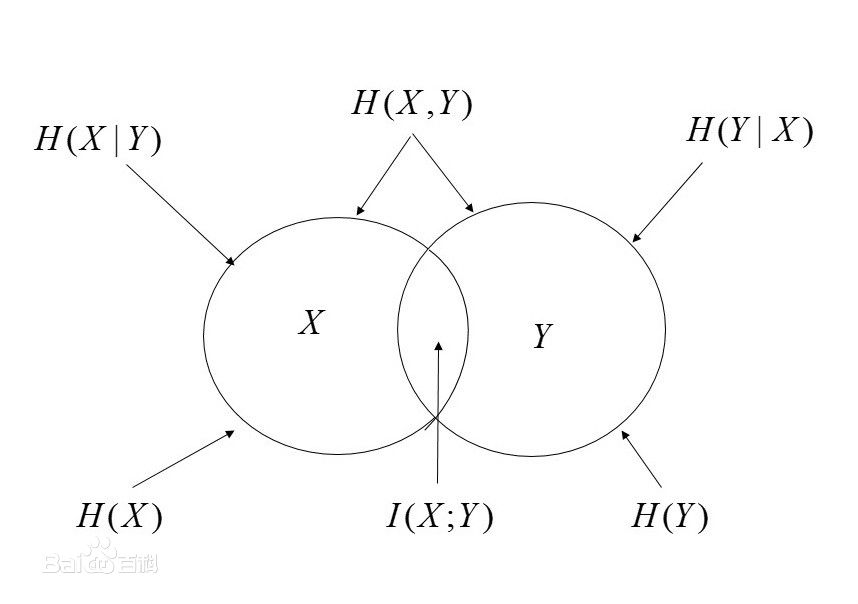
熵，就是一种可能性的信息量，一个事件总的信息量就是每一种可能的情况的信息量乘以它们发生的概率，其实就是信息量的数学期望。

**条件熵（Conditional Entropy）**：

通俗讲，知道情况下，的信息量。

**信息增益（Information Gain），互信息（Mutual Information）**：

互信息就是知道，给的信息量带来多少损失（或者知道，给的信息量带来多少损失）。



熵、条件熵、互信息（百度百科）

**决策树构造方法**

构造决策树的时候，会基于纯度来构建。而经典的 “不纯度”（Impurity）的指标有三种，分别是信息增益（ID3 算法）、信息增益率（C4.5 算法）以及基尼指数（CART 算法）。

信息增益：

信息增益指的就是划分可以带来纯度的提高，信息熵的下降。它的计算公式，是父亲节点的信息熵减去所有子节点的信息熵的加权和。

ID3算法就是用信息增益大小来判断当前节点应该用什么特征来构建决策树，用计算出的信息增益最大的特征来建立决策树的当前节点。

在C4.5算法中，采用了信息增益比来选择特征，以减少信息增益容易选择特征值多的特征的问题。

CART分类树算法使用基尼系数来代替信息增益比，基尼系数代表了模型的不纯度，基尼系数越小，则不纯度越低，特征越好。这和信息增益（比）是相反的。

对于确定性问题，为0，

当

最大，即

每个事件概率相同时，最大，这件事越不确定。

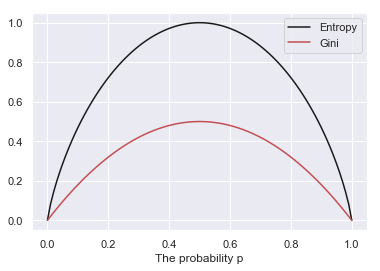


图3 两类问题的熵和基尼系数

表1 分类树示例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Day | Outlook (O) | Temperature (T) | Humidity (H) | Wind (W) | Play Tennis (y) |
| 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 |

预测是否Play Tennis，两类Yes（6） or No（4）。

信息熵为：

对于特征Humidity (H)：

信息熵为：

信息增益（IG）：

Gini：

对于特征Wind (W)：

信息熵为：

信息增益（IG）：

Gini：

信息增益（越大越好）：

Gini（越小越好）:

同样，我们可以计算出

,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Outlook (O) | Temperature (T) | Humidity (H) | Wind (W) |
| 信息增益（IG） | **0.809** | 0.096 | 0.125 | 0.092 |
| Gini | **0.3** | 0.567 | 0.4 | 0.419 |

因此，我们选择特征Outlook做为决策规则。应用基尼系数作为指标，我们最终得到的决策树模型为图4，图中“+”表示打网球，“-”表示不打网球。



图4 决策树

**决策树预测算法：**

对于一个给定的数据，根据决策树里各个节点的决策规则，将数据分配到某个叶子节点上，并由叶子节点给出最终预测值。

在决策树中，叶子节点提供预测结果的算法十分简单。

* 对于分类问题：叶子节点给出的其实是数据属于每个类别的概率，而这个概率值等于各个类别的数据数据占比。假设数据分为类，分别记为，叶子节点上一共有个数据，则。基于预测到的概率值，就可以得到最终的结果为出现概率最大的类别。
* 对于回归问题：最终的预测结果等于节点内标签变量的平均值。

**剪枝**

在实际应用中，决策树常常会遇到过拟合的问题。

过拟合：训练误差很小，但测试误差很大（模型太过复杂）。

欠拟合：模型没有很好地捕捉到数据特征，不能够很好地拟合数据，无论是训练误差还是测试误差都很高（模型太过简单）。

我们通常选用80％的数据集来训练模型，剩下的20％来测试模型。在训练阶段，模型对数据的拟合误差，称其为训练误差。同样，在测试阶段产生的偏差被称为测试误差。简单来讲，机器学习模型的性能可以根据以下两个规则来判断：

1. 降低训练误差

2. 缩小训练误差和测试误差的差距

如果模型无法遵循规则1，不能将训练集的误差降到较低水平，就会发生欠拟合。

如果模型无法遵循规则2，以至于训练误差和和测试误差之间的差距太大，就会发生过拟合。这两个简单的规则可以帮助我们理解机器学习算法中的过拟合和欠拟合。

从直观上理解，决策树可以无限制地划分树中的节点，使得每个叶子节点里只剩下极少量的、类别较为单一地数据。为了解决这个问题，需要用一些技术手段防止决策树过细地划分节点，这被称为剪枝（Pruning）。

具体方法分为两类：

前剪枝（Pre-pruning）：这类方法作用于决策树的生成过程中，通过一些阈值来限制决策树的生长。这种方法在实际中使用较多。

后剪枝（Post-pruning）：这类方法作用于决策树生成之后，主要的思路是对于一棵已生成的决策树，将其中不必要的子树剪掉。

**树的集成**

决策树模型虽然简单明了，但是单独使用时效果并不理想。这是因为决策树模型的最后一步算法过于简单：对于分类问题，只考虑叶子节点里哪个类别占比最大；对于回归问题，则计算叶子节点内数据的平均值。这导致它在单独使用时，效果不理想。

通过某种方式将多个决策树组合起来，使用它们的“集体智慧”来解决问题，这被称为集成方法（Ensemble Method）。

集成方法指的是将一些列弱学习组合成一个预测效果更好的复杂模型。弱学习（Weak Learner）指的是机器学习中比较简单的模型。

集成方法并不仅限于决策树。

针对决策树的集成方法通常可以被分为两类：平均方法（Averaging Methods）和提升方法（Boosting Methods）。它们的代表模型分别是随机森林（Random Forest）和梯度提升决策树（Gradient—Boosting Decision Trees, GBDT）。

随机森林由个决策树组成，模型的预测结果等于各决策树结果的某种“加权平均”。

对于分类问题：最终结果等于在决策树预测结果中出现次数最多的类别。可以将随机森林想象为一场投票，通过少数服从多数的原则得到最终的结果。

依据是：一棵树犯错的概率比较大，很多树同时犯错的概率就很小了。

假设对某个分类问题，有3棵相互独立的决策树，它们各自预测错误的概率为20%。如果将它们按少数服从多数的原则组合起来，形成一个随机森林。那么，预测犯错误的概率将下降到

因此，要保证随机森林预测效果的最重要因素就是：森林中的决策树是相互独立的。

可以从以下三个层面来引入决策树的随机性：

[1] 对于每棵决策树，从原始训练集中随机选取训练该决策树的数据；

[2] 在划分节点时，并不遍历全部自变量，而是随机挑选其中的一部分作为候选自变量；

[3] 在选择自变量的划分阈值时，并不求得最优的解，而是随机构成一个候选阈值集合，并从中选取效果最优的。

针对随机森林，第三方库scikit-learn提供了两种实现：random forests和extremely randomized trees。这两者的差异在于前者实现了随机性的前两点，即训练数据随机和候选自变量随机，而后者实现了随机性中的三点。

from sklearn.ensemble import RandomForestRegressor（用于回归问题）

from sklearn.ensemble import RandomForestClassifier（用于分类问题）

from sklearn.ensemble import ExtraTreesRegressor（用于回归问题）

from sklearn.ensemble import ExtraTreesClassifier（用于分类问题）

针对GBDT，第三方库scikit-learn提供了

from sklearn.ensemble import GradientBoostingRegressor（用于回归问题）

from sklearn.ensemble import GradientBoostingClassifier（用于分类问题）

**Python示例**

决策树可以由第三方库scikit-learn提供的DecisionTreeClassifier来实现。

from sklearn.tree import DecisionTreeClassifier

DTC = DecisionTreeClassifier(criterion = "gini", max\_depth = 5,

min\_samples\_split = 2, min\_samples\_leaf = 1, random\_state = 42)

参数“criterion”确定不同的不纯度指标，可选择Gini和基于Entropy的信息增益；

“max\_depth”表示决策树的最大高度；

“min\_samples\_split”表示当叶子的节点的数据量大于这个参数时，决策树才会继续划分节点；

“min\_samples\_leaf”表示当叶子节点的数据量大于这个参数时，相应的节点才会被保留，否则会被剪枝。

数据集：IBM Watson的HR员工流失

<https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset>

<https://www.ibm.com/communities/analytics/watson-analytics-blog/hr-employee-attrition/>

根据IBM声明，“这是由IBM数据科学家创建的虚构数据集”。

数据集包括1470名员工（行）和35个特征（列），一部分已经离职（Attrition = “Yes”）。

离职：237

没离职：1233

其中EmployeeCount、EmployeeNumber、Over18、StandardHours四个特征与建模为无关，故删除这四个特征。

员工流失（减员）是一个组织机构的主要成本所在。组织机构面临着因员工流失所带来的巨大成本。有些成本是有形的，例如培训费用和员工能够开始产出效益所需的时间成本。然而，最重要的成本是无形的。

利用机器学习算法，不仅可以预测员工的流失，而且可以了解影响员工流失的关键因素。

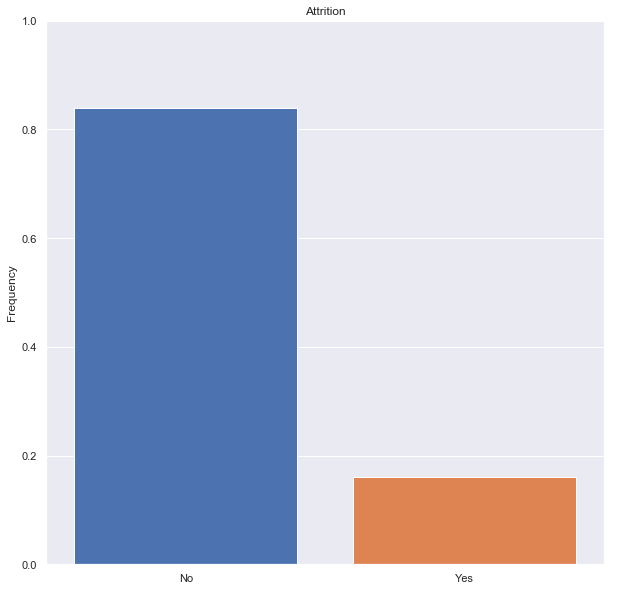


图5

我们将数据集分为训练集（80%）和测试集（20%）。

1. 应用决策树来分类；

2. 应用随机森林来分类；

3. 应用梯度提升决策树来分类。

结果如下表所示。

表2 分类准确率（Accuracy）

|  |  |  |
| --- | --- | --- |
|  | 训练集 | 测试集 |
| 决策树 | 89.4% | 82.3% |
| 随机森林 | 91.2% | 87.1% |
| 梯度提升决策树 | 95.8% | 87.8% |