决策树学习

维基百科，自由的百科全书

统计学，数据挖掘和机器学习中的**决策树训练**，使用决策树作为预测模型来预测样本的类标。这种决策树也称作**分类树**或**回归树**。在这些树的结构里， 叶子节点给出类标而内部节点代表某个属性。

在决策分析中，一棵决策树可以明确地表达决策的过程。在数据挖掘中，一棵决策树表达的是数据而不是决策。本页的决策树是数据挖掘中的决策树。

目录

推广

在数据挖掘中决策树训练是一个常用的方法。目标是创建一个模型来预测样本的目标值。例如右图。每个 内部节点 对应于一个输入属性，子节点代表父节点的属性的可能取值。每个叶子节点代表输入属性得到的可能输出值。

一棵树的训练过程为：根据一个指标，分裂训练集为几个子集。这个过程不断的在产生的子集里重复递归进行，即递归分割。当一个训练子集的类标都相同时 递归停止。这种 *决策树的自顶向下归纳* (TDITD) [1] 是 贪心算法的一种, 也是目前为止最为常用的一种训练方法，但不是唯一的方法。

数据以如下方式表示:

$$(\mathbf{x}, Y) = (x_1, x_2, x_3, \ldots, x_k, Y)$$

其中Y是目标值，向量**x**由这些属性构成, x_1, x_2, x_3 等等，用来得到目标值。

决策树的类型

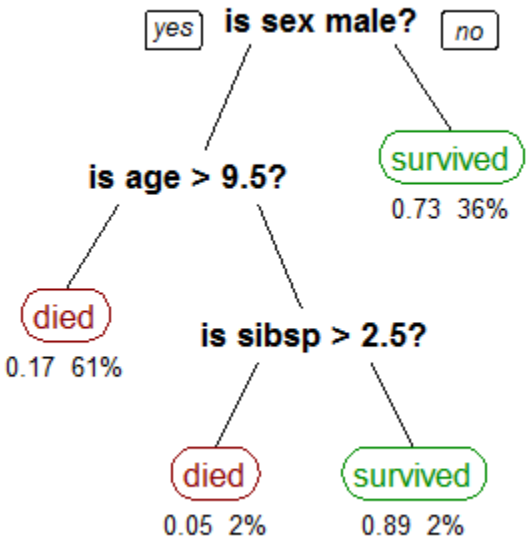
在数据挖掘中，决策树主要有两种类型：

- **分类树** 的输出是样本的类标。
- **回归树** 的输出是一个实数 (例如房子的价格，病人待在医院的时间等)。

术语分类和回归树 **(CART)** 包含了上述两种决策树, 最先由Breiman 等提出.[2] 分类树和回归树有些共同点和不同点—例如处理在何处分裂的问题。[2]

有些*集成*的方法产生多棵树：

- **装袋算法（Bagging）**，是一个早期的集成方法，用有放回抽样法来训练多棵决策树，最终结果用投票法产生。[3]
- **随机森林（Random Forest）** 使用多棵决策树来改进分类性能。
- **提升树（Boosting Tree）** 可以用来做回归分析和分类决策[4][5]
- 旋转森林（Rotation forest） – 每棵树的训练首先使用主元分析法 (PCA)。[6]



一个描述泰坦尼克号上乘客生存的决策树 ("sibsp"指甲板上的兄妹和配偶)。每个决策叶下标识该类乘客的生存几率和观察到的比率

还有其他很多决策树算法，常见的有：

- ID3算法
- C4.5算法
- **CHi-squared Automatic Interaction Detector (CHAID)**. 在生成树的过程中用多层分裂.[7]
- MARS:可以更好的处理数值型数据。

模型表达式

构建决策树时通常采用自上而下的方法，在每一步选择一个最好的属性来分裂。[8] "最好" 的定义是使得子节点中的训练集尽量的纯。不同的算法使用不同的指标来定义"最好"。本部分介绍一些最常见的指标。

基尼不纯度指标

在CART算法中, 基尼不纯度表示一个随机选中的样本在子集中被分错的可能性。基尼不纯度为这个样本被选中的概率乘以它被分错的概率。当一个节点中所有样本都是一个类时，基尼不纯度为零。

假设y的可能取值为J个类别，另$i \in \{1, 2, \ldots, J\}$，$p_i$表示被标定为第$i$类的概率，则基尼不纯度的计算为：

$$\mathrm{I}_G(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i(1-p_i) = \sum_{i=1}^{J} (p_i - {p_i}^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} {p_i}^2 = 1 - \sum_{i=1}^{J} {p_i}^2$$

信息增益

ID3, C4.5 和 C5.0 决策树的生成使用信息增益。信息增益 是基于信息论中信息熵与自信息理论.

信息熵定义为：

$$\mathrm{H}(T) = \mathrm{I}_E(p_1, p_2, \ldots, p_J) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

其中$p_1, p_2, \ldots$加和为1，表示当前节点中各个类别的百分比。[9]

$$\overbrace{IG(T,a)}^{\text{Information Gain}} = \overbrace{\mathrm{H}(T)}^{\text{Entropy (parent)}} - \overbrace{\mathrm{H}(T|a)}^{\text{Weighted Sum of Entropy (Children)}}$$

$$= -\sum_{i=1}^{J} p_i \log_2 p_i - \sum_{a} p(a) \sum_{i=1}^{J} -\Pr(i|a) \log_2 \Pr(i|a)$$

例如，数据集有4个属性：*outlook* (sunny, overcast, rainy), *temperature* (hot, mild, cool), *humidity* (high, normal), and *windy* (true, false), 目标值*play*（yes, no）, 总共14个数据点。为建造决策树，需要比较4棵决策树的信息增益，每棵决策树用一种属性做划分。信息增益最高的划分作为第一次划分，并在每个子节点继续此过程，直至其信息增益为0。

使用属性*windy*做划分时，产生2个子节点：*windy*值为真与为假。当前数据集，6个数据点的*windy*值为真，其中3个点的*play*值为真，3个点的*play*值为假；其余8个数据点的*windy*为假，其中6个点的*play*值为真，2个点的*play*值为假。 *windy*=true的子节点的信息熵计算为：

$$I_E([3,3]) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

windy=false的子节点的信息熵计算为：

$$I_E([6,2]) = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4} = 0.8112781$$

这个划分（使用属性*windy*）的信息熵是两个子节点信息熵的加权和：

$$I_E([3,3],[6,2]) = I_E(\text{windy or not}) = \frac{6}{14}\cdot 1 + \frac{8}{14}\cdot 0.8112781 = 0.8921589$$

为计算使用属性*windy*的信息增益，必须先计算出最初（未划分）的数据集的信息熵，数据集的*play*有9个yes与5个no：

$$I_E([9,5]) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940286$$

使用属性*windy*的信息增益是：

$$IG(\text{windy}) = I_E([9,5]) - I_E([3,3],[6,2]) = 0.940286 - 0.8921589 = 0.0481271$$

决策树的优点

与其他的数据挖掘算法相比，决策树有许多优点:

- **易于理解和解释** 人们很容易理解决策树的意义。
- **只需很少的数据准备** 其他技术往往需要数据归一化。

- **即可以处理数值型数据也可以处理类别型 数据**。其他技术往往只能处理一种数据类型。例如关联规则只能处理类别型的而神经网络只能处理数值型的数据。
- **使用白箱 模型.** 输出结果容易通过模型的结构来解释。而神经网络是黑箱模型，很难解释输出的结果。
- **可以通过测试集来验证模型的性能** 。可以考虑模型的稳定性。
- **强健控制.** 对噪声处理有好的强健性。
- **可以很好的处理大规模数据** 。

缺点

- 训练一棵最优的决策树是一个完全NP问题。[10][11] 因此, 实际应用时决策树的训练采用启发式搜索算法例如 贪心算法 来达到局部最优。这样的算法没办法得到最优的决策树。
- 决策树创建的过度复杂会导致无法很好的预测训练集之外的数据。这称作过拟合.[12] 剪枝机制可以避免这种问题。
- 有些问题决策树没办法很好的解决,例如 异或问题。解决这种问题的时候，决策树会变得过大。 要解决这种问题，只能改变问题的领域[13] 或者使用其他更为耗时的学习算法 (例如统计关系学习 或者 归纳逻辑编程).
- 对那些有类别型属性的数据, 信息增益 会有一定的偏置。[14]

延伸

决策图

在决策树中, 从根节点到叶节点的路径采用汇合或 与。 而在决策图中, 可以采用 最小消息长度 (MML) 来汇合两条或多条路径。[15]

用演化算法来搜索

演化算法可以用来避免局部最优的问题[16][17]

参见

- 决策树剪枝
- 二元决策图
- CART
- ID3算法
- C4.5算法

参考资料

1. Quinlan, J. R., (1986). Induction of Decision Trees. Machine Learning 1: 81-106, Kluwer Academic Publishers
2. Breiman, Leo; Friedman, J. H., Olshen, R. A., & Stone, C. J. Classification and regression trees. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software. 1984.

ISBN 978-0-412-04841-8.
3. Breiman, L. (1996). Bagging Predictors. "Machine Learning, 24": pp. 123-140.
4. Friedman, J. H. (1999). *Stochastic gradient boosting.* Stanford University.
5. Hastie, T., Tibshirani, R., Friedman, J. H. (2001). *The elements of statistical learning : Data mining, inference, and prediction.* New York: Springer Verlag.
6. Rodriguez, J.J. and Kuncheva, L.I. and Alonso, C.J. (2006), Rotation forest: A new classifier ensemble method, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(10):1619-1630.
7. Kass, G. V. An exploratory technique for investigating large quantities of categorical data. Applied Statistics. 1980, **29** (2): 119–127. JSTOR 2986296. doi:10.2307/2986296.
8. Rokach, L.; Maimon, O. Top-down induction of decision trees classifiers-a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C. 2005, **35** (4): 476–487. doi:10.1109/TSMCC.2004.843247.
9. Witten, Ian; Frank, Eibe; Hall, Mark. Data Mining. Burlington, MA: Morgan Kaufmann. 2011: 102–103. ISBN 978-0-12-374856-0.
0. Hyafil, Laurent; Rivest, RL. Constructing Optimal Binary Decision Trees is NP-complete. Information Processing Letters. 1976, **5** (1): 15–17. doi:10.1016/0020-0190(76)90095-8.
1. Murthy S. (1998). Automatic construction of decision trees from data: A multidisciplinary survey. *Data Mining and Knowledge Discovery*
2. Principles of Data Mining. SpringerLink. [2018-04-02]. doi:10.1007/978-1-84628-766-4 **（英国英语）**.
3. Inductive Logic Programming. SpringerLink. [2018-04-02]. doi:10.1007/b13700 **（英国英语）**.
4. Deng,H.; Runger, G.; Tuv, E. Bias of importance measures for multi-valued attributes and solutions (PDF). Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN): 293–300. 2011. （原始内容 (PDF)存档于2018-05-10）.
5. http://citeseer.ist.psu.edu/oliver93decision.html
6. Papagelis A., Kalles D.(2001). Breeding Decision Trees Using Evolutionary Techniques, Proceedings of the Eighteenth International Conference on Machine Learning, p.393-400, June 28-July 01, 2001
7. Barros, Rodrigo C., Basgalupp, M. P., Carvalho, A. C. P. L. F., Freitas, Alex A. (2011). A Survey of Evolutionary Algorithms for Decision-Tree Induction (http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5928432). IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, vol. 42, n. 3, p. 291-312, May 2012.

外部链接

- Building Decision Trees in Python (https://web.archive.org/web/20110807035706/http://onlamp.com/lpt/a/6464) From O'Reilly.
- An Addendum to "Building Decision Trees in Python" (http://www.oreillynet.com/mac/blog/2007/06/an_addendum_to_building_decisi.html) From O'Reilly.
- Decision Trees page at aaai.org (https://web.archive.org/web/20110728045409/http://www.aaai.org/aitopics/html/trees.html), a page with commented links.
- Decision tree implementation in Ruby (AI4R) (http://ai4r.rubyforge.org/index.html)
- Building Decision Tree In Bash (http://liuzhiqiangruc.iteye.com/blog/1601922)