决策树和集成方法学习笔记

Decision Trees and Ensemble Methods

姚炜彤

版本：*1.0.0*

更新：*May 5, 2019*

Decision Trees and Ensemble Methods 介绍了 Regression, Classification 两类决策树和决策树的构建，并详细介绍了 Bagging, Random Forest (RF), Boosting (L2 Boosting and Adaboosting) 三种集成学习的方法。本文主要对 Lecture 的内容做学习回顾和总结，在此基础上补充 pruning 的两种方法，ID3、C4.5、OC1（多变量决策树）三类决策树算法和 Boosting 里的梯度提升算法。Github Page：决策树和集成方法学习笔记

1 学习回顾与总结

Framework
I. Regression Tree and Classification tree
II. Build a tree:
1. 特征选择：（Node purity measure）Gini index, misclassification rate, cross-entropy
2. 剪枝处理：预剪枝和后剪枝
3. 算法：ID3, C4.5, CART, OC1（多变量决策树）
III. Ensemble method: 某种策略结合 individual learner(base learner)
1. Bagging: bootstrap sampling，降低方差
2. Random Forest: 在每个 node 随机选择一个包含 m 个属性的子集，再选择最优属性进行划分。
3. Boosting:
weak→strong，by reweighting training data
Weighted sum of individual classifier
Loss function: (1) Adaboosting: exponential loss (2) L2 boosting: L2 loss (3) Gradient boosing
4. 结合策略：simple average, weighted average, majority voting, plurality voting

表 1: *from Decision Trees and Ensemble Methods, by Jiaming Mao*

2 学习回顾：剪枝 Pruning

决策树的生成过程只考虑了局部的模型，因此容易造成过拟合（想象一下为了精确分类每个 leaf 下只有 1-2 个样本的情况）。Pruning 是在决策树学习中对 internal node 进行简化，使之成为新的 terminal node，达到减少模型的复杂程度、防止过拟合的目的。Pruning 的判断依据是让整体的代价函数最小化，即每次 pruning 都必须重新计算，并比较 before pruning 和 after pruning 的整体代价函数，如果有 $C_\alpha(T_A) \leqslant C_\alpha(T_B)$，则进行剪枝。

预剪枝 Prepruning 是在树生成过程中对 Node 事先进行估计，判断是否进行 prune，判断结果直接决定决策树的生成。后剪枝 Postpruning 则是先生成决策树，再由 internal node 开始自下而上，从后往前进行推断判断每个 Node 是否剪枝，根据判断结果对原先的决策树 internal node 进行替换。区别：postpruning 在模型 generalization 上有比较优势，但是由于需要事先生成决策树，训练时间比 prepruning 更长。

3 学习补充：算法介绍与比较

3.1 ID3 和 C45 算法

ID3 算法的本质是 CLS 的衍生，原理是在决策树的各个 Node 上计算各个特征的信息增益（或者信息增益率），通过最大信息增益选择特征构造下一层的 Node, 如此往复构造，但有两种特例：（1）piecewise constant 决策树下的样本同属一个类别，直接返回 T；（2）特征 A 为空值，视为单节点树，直接返回 T。最大的信息增益本质上等用于 Lecture 里面最小的 cross-entropy(符号相反)，数据 Rm 的 cross entropy 和在特征 A 下的 cross-entropy 分别为：

$$Q_m = -\sum_{j=1}^{j} \widehat{p}_j^m \log \widehat{p}_j^m \qquad Q(m|A) = \sum_{i=1}^{n} \frac{|R_i|}{|D|} Q_{m_i} \tag{1}$$

信息增益和信息增益率为：

$$q(D,A) = Q_m - Q(m|A) \qquad q_R(D,A) = \frac{q(D,A)}{Q_m} \tag{2}$$

ID3 算法通过判断信息增益 $q(D,A)$ 最大值选择特征，C4.5 是 ID3 的改进：通过最大化信息增益率 $q_R(D,A)$。C4.5 的优势体现在：（1）如果同一个特征下的样本比较多，则信息增益会受样本数增大，缺乏公允性。（2）C4.5 可以处理离散值也可以处理连续值。（3）可以在样本的某个特征缺失值时仍对样本进行分类——如果用 ID3 只能得到缺失属性的样本分布，但 C4.5 可以判断最终的类别：

编号	Outlook	Temp(℉)	Humidity(%)	Windy	Class
1	sunny	75	70	true	Play
2	sunny	80	90	true	Don’t Play
3	sunny	85	85	false	Don’t Play
4	sunny	72	95	false	Don’t Play
5	sunny	69	70	false	Play
6	-	72	90	true	Play
7	overcast	83	78	false	Play

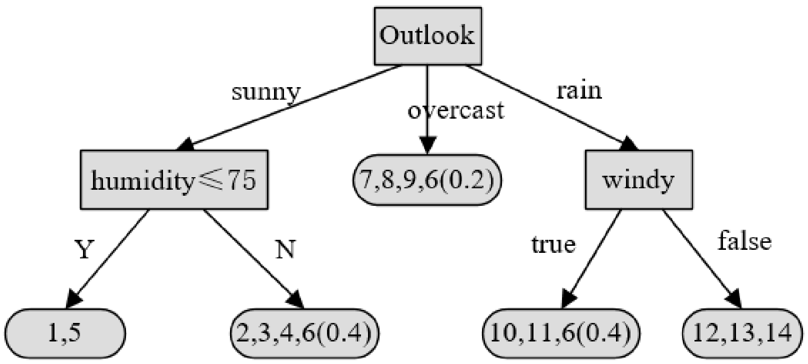
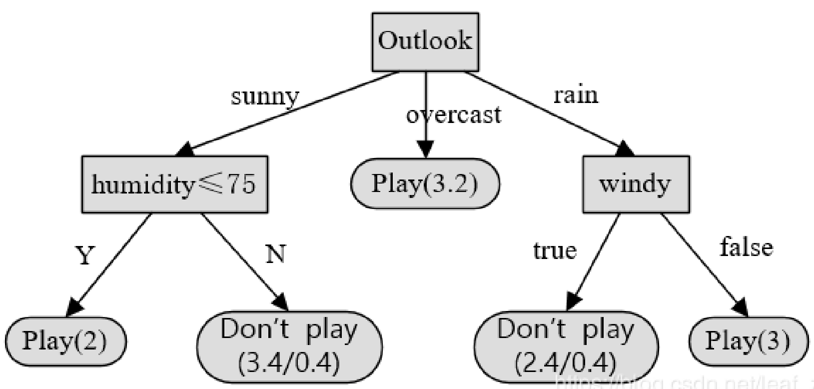


图 1: 括号内为 *ID3* 得到的权重



给每个 terminal node 加个权重（该类别样本数/同特征下其他类别总样本数），对 6 号缺失 Outlook 属性的样本进行判断：已知如果是 sunny，humidity=90>75，play 概率有 0.4；overcast 下 play 概率为 1；rain 下 play 的概率为 0.4，play 的总概率为：3.4+2.4+3.2=9 $\frac{0.4}{3.4}\times\frac{3.4}{9}+1\times\frac{3.2}{9}+\frac{0.4}{24}\times\frac{2.4}{9}=0.44<0.5$，所以 6 号样本分类错误。

3.2 ID3，C4.5，CART 算法比较

ID3 和 C4.5 的比较在上小节已经提到，两者的不足体现在:

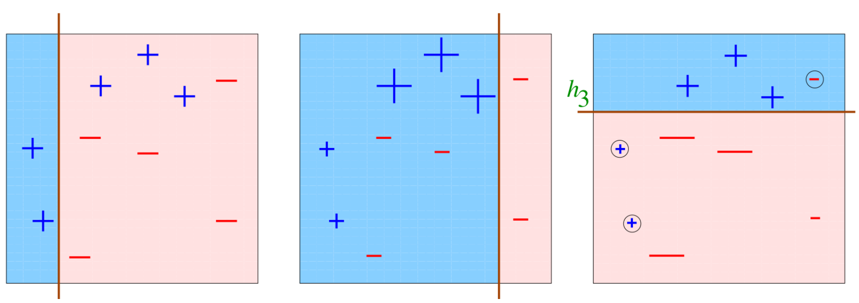
（1）局部进行最优化，容易导致全局过拟合。

（2）C4.5 采用 Pessimistic Error Pruning (PEP) 的剪枝策略：误判率 $p = \frac{\sum_{i=1}^{L} E_i + 0.5L}{\sum_{i=1}^{L} N_i}$，其中 N 为 leaf 的样本数，其中有 E 个判断错误，0.5 为惩罚因子。如果 Prune 前子树样本服从 Binomial(N,P)，则 $E(T_B) = N * p \quad std(T_B) = \sqrt{N * p * (1-p)}$；Prune 后的 terminal node 服从 Bernulli，$E(T_A) = N * ewheree = \frac{E+o.5}{N}$，如果 $E(T_A) < E(T_B) + std(T_B)$ 成立则进行 prune。该策略是自上而下的，容易造成“过剪枝”导致模型不能泛化（类似 prepruning 的缺点）。
（3）C4.5 采用信息增益率，惩罚参数为以特征 A 为条件的信息熵的倒数 $\frac{1}{Q(m|A)}$，因此会倾向于选择惩罚参数较小的特征。

相比之下，如 Lecture 所示 CART 算法的优点体现在：
（1）CART 采用代价复杂剪枝法 $\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \overline{y}_m)^2 + \alpha|T|$，其中 α 权衡了拟合程度和模型的复杂程度从下往上判断剪枝，避免了过拟合。
（2）采用的 Gini Index 作为特征判断，尤其 CART 为二叉树时 Gini(P) = 2p(1 − p)，可以优化信息增益率的误差。
（3）CART 采用二叉树：特征 $\{A1, A2, A3\}$ 会被先分类为 $\{A1, A2\}$ 和 $\{A3\}$，在 $\{A1, A2\}$ 子树上再分类 $\{A1\}$，$\{A2\}$，与 C4.5 的多类别相比，CART 的同一个特征可以参与多次 node 的建立。

3.3 多变量特征决策树与 OC1 算法

三个算法下决策树的分类边界具有轴平行的特点（如下图），即每次分类只能用一个特征值，无法多个特征值共同使用



对于多变量特征决策树是对特征进行选择后，进行线性组合。在每个 node 上，我们希望其特征的个数最小化但是效果最优化，根据 Carla E. Brodley 和 Paul E. Utgoff 的Multivariate Decision Trees（1995），有以下 3 种方法可以进行特征选择：

（1）Sequential Backward Elimination

SBE 是一个自上而下的特征搜索，它从所有特征开始，搜索迭代地删除对测试质量贡献最小的特性。SBE 涉及局部最优和停止准则，局部价值准则和 cross-entropy，Gini Index 相关，停止准则决定何时停止从线性组合测试中消除特征：如果基于 $i-1$ 特性的测试的部分价值标准小于基于 i 特性的测试的部分价值标准，则删除第 i 个特征，直到只保留一个特性，或者停止搜索。

（2）**Sequential Forward Selection**

SFS 一种自下而上的搜索方法，它从零特征开始，并试图添加一些特征，这些特征将导致部分价值准则的最大幅度增长。与 SBE 算法一样，SFS 算法也需要一个局部最优准则和一个停止准则。

（3）**Heuristic Sequential Search** HSS 是前两个的结合：给出一组训练实例，HSS 首先找到一个基于所有特征的线性组合检验和一个基于一个特征的最佳线性检验。在基于局部最优准则下，如果一个特征的结果更好，执行 SFS；反之，执行 SBE。

（4）**CART's linear combination**

CART 的线性组合算法被称为"Trading quality for simplicity"，它的简化包括：只使用数值特征，只使用完整实例，可以选择较少特征的线性组合测试（但是牺牲了精确度）。CART 先执行一个 SBE 得到一个最小化误判的函数，再对每个特征消除的结果进行计算：如果每删除一个特征，（导致 node 的误判增加）每次都必须计算阈值 C（maximum increase in error）$error_{elimination} < \beta * C$ 成立则确认删除该特征，继续搜索。每次搜索时都保持之前的线性组合系数不变，直到最后停止搜索才重新计算系数。

(在 Carla E. Brodley 和 Paul E. Utgoff 的研究里发现 multivariate DT 比 univariate DT 的错误更少，而且 HSS 是一个较优的特征选择策略)

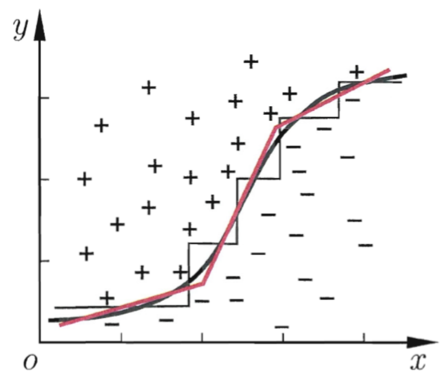


图 2: 特征的线性组合结果是形成不平行于坐标轴的边界

多变量决策树的算法主要是 Murthy 等人提出的OC1 算法：OC1 算法先寻找每个属性的最优权重，在局部优化后，在分类边界加入随机干扰来寻找更好的边界。该篇学习笔记简要介绍一下 Murthy 用的扰动算法（Perturbation Algorithm）：

（1）决策树中每个节点的初始超平面（多个特征的线性组合平面）由 OCl 随机选择，而且如果两个超平面拥有同样的点集，OC1 是无法区分的，由此提供扰动的思路。优化的思路是：由将当前超平面旋转到新位置，而且每次只能改变该超平面的一个维度（即一个特征值，此时又回到 axis-parallel 的局部最优化问题）。

（2）假设样本空间 P 包含 n 个例子，每个例子都有 d 个属性。当前的超平面 H 可以被定义为：$\sum_{i=1}^{d}(a_i X_i) + a_{d+1} = 0$。在样本空间 P 里取第 j 个样本 $P_j = (x_{j1}, x_{j2}, \ldots, x_{jd})$，代入初始超平面可

得：$\sum_{i=1}^{d}\left(a_i x_{ji}\right)+a_{d+1}=V_j$，其中 V_j 的符号代表了点 P_j 在超平面的上方还是下方，如果该超平面是没有误判的，则该平面同一侧的所有样本的符合 V_k 应当是相同的。

（3）定义 $U_j=\frac{a_m x_{jm}-V_j}{x_{jm}}$,　$j=1,2,\dots,n$，如果 $a_m>U_j$ 则有 P_j 在超平面的上方，反之在下方。保持其他 d 个特征值 $a_1,\dots,a_{d+1}$ 不变，重复 n 个样本可以得到关于 a_m 的 n 个约束条件，由此超平面的拟合简化成为找到一个 a_m 让其尽可能满足多个约束条件的一维问题。

```
Perturb(H,m)
{
  for j = 1 to n
      Compute U_j (Eq. 1)
  Sort U_1 ... U_n in nondecreasing order.
  a_m1 = best univariate split of the sorted U_js.
  H_1 = result of substituting a_m1 for a_m in H.
  If (impurity(H) < impurity(H_1))
      { a_m = a_m1 ; stagnant = 0 }
  Else if (impurity(H) = impurity(H_1))
      { a_m = a_m1 with probability
                stag_prob = e^(-stagnant)
      stagnant = stagnant + 1 }
}
```

图 3: *扰动算法.from “OC1: A Randomized Induction of Oblique Decision Trees”*

4　梯度提升 Gradient Boosting

根据代价函数的特殊性，可以区分出 L_2Boosting(Loss function 为 L_2) 和 Adaboostings(Loss function 为指数函数 $L(y,f(x))=\exp[-y\sum_{m=1}^{M}\alpha_m G_m(x)]$)。但对于一般的代价函数，采用梯度提升回归算法：把代价函数的负梯度 $-\left[\frac{\partial L(y,f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$ 近似看成残差进行拟合，

再根据新的 R_{mj} 计算 $c_{mj}=\arg\min_c\sum_{x_i\in R_m}L\left(y_i,f_{m-1}\left(x_i\right)+c\right)$，

更新 $f_m(x)=f_{m-1}(x)+\sum_{j=1}^{J}c_{mj}I\left(x\in R_{mj}\right)$，

最后得到 strong learner $f(x)=f_T(x)=f_0(x)+\sum_{t=1}^{T}\sum_{j=1}^{J}c_{tj},I\left(x\in R_{tj}\right)$。

梯度提升分类算法则把代价函数写成对数似然函数 $L(y,f(x))=\log(1+\exp(-yf(x)))$，再重复以上步骤。

5　参考文献

[1]《统计学习方法》, 李航著.

[2]《机器学习》, 周志华著.

[3] *Quinlan, J. R. (1986). Induction of decision trees. Machine Learning*, 1(1):81-106

[4] Grzymala-Busse, Jerzy W. (February 1993). *"Selected Algorithms of Machine Learning from Examples"* (PDF). Fundamenta Informaticae. 18(2): 193–207 –via ResearchGate.

[5] *C4.5: Programs for Machine Learning*, by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993

[6] *《决策树（Decision Tree）-ID3、C4.5、CART 比较》*

[7] *Multivariate Decision Trees,*CARLA E. BRODLEY, PAUL E. UTGOFF.Machine Learning, 19, 45-77 (1995)

[8] *OC1:Randomized Induction of Oblique Decision Trees*, Sreerama Murthy, Simon Kasif, Steven Salzberg, Richard Beigel