树模型

`ID3` `C4.5` `CART` `随机森林` `GBDT` `信息增益` `信息增益率` `基尼指数`

前置

GBM: Gradient Boosting Machine。
Categorical: 无序类别
GBRT: Gradient Boosted Regression Tree
GBDT: Gradient Boosted Decision Tree
MART: Multi Additive Regression Tree

决策树算法是一种比较直观和容易理解的算法，决策树模型类似我们平常在编写代码时写的判断逻辑，根据判断条件产生不同的分支。

ID3算法

节点分裂依据: 信息增益，特征只能是离散特征

ID3在进行节点分裂时选择的是信息增益做为分裂依据。在介绍信息增益前，我们首先需要引入熵的概念，对一个数据集D，假设该数据集中有K个不同的分类，则熵可以表示为

$$Entropy = -1 * \sum_{i=1}^{K} p_i * logp_i$$

其中$p_i = \frac{|D_i|}{|D|}$，即第i个类目的数据个数除以所有数据的个数。
熵表示的是一种混乱程度，简单理解一些，如果相同数目的数据，只有一个类别，整个数据就是高度一致的，混乱度很小，如果存在多个类别，混乱度则会很大。我们可以想象现实生活中的一堆气球，如果只有一个颜色就会很一致并且单调，如果是有多个颜色，就显得稍微混乱并且色彩鲜艳。

ID3中使用的信息增益正是基于熵，在树进行分裂时我们会选取特征进行分裂，ID3分裂的依据是根据某个维度的特征分裂后是否是使得熵减少最多，其实理解起来也比较简单，因为ID3算法本身是做分类问题的，所以我们要尽量将同一类数据分到不同的叶子节点上，因此整个过程会使得熵越来越小，所以我们只要每一次分类选取使得熵减少最大的特征做分类，就可以用比较少的分裂次数达到比较好的分类效果。所以在一个节点出根据每个特征进行分类的信息增益公式为

$$Gain(D_i, k) = Entropy(D_i) - Entropy(D_i, k)$$

其中D_i表示当前节点的数据样本集，$Entropy(D_i)$表示当前节点的熵，$Entropy(D_i, k)$则表示根据第k个特征分裂后产生的节点计算出的熵之和。

通过每一步选取信息增益最大的特征进行分裂，不断分裂下去就可以得到最终的决策树。

C4.5算法

节点分类依据: 信息增益率，特征同样只能是离散特征

C4.5算法和ID3算法出自同一人之手，而C4.5算法也正是对ID3存在的问题进行修正，通过上面我们知道ID3算法每一步分裂是根据信息增益，但是我们由熵的定义可以看出，对于取值比较多的特征，其天然占有优势，举个极端的例子，比如我们要判断用户是否对经济类文章感兴趣，我们选取的特征里面包含性别和用户ID的特征，性别只有男和女两个取值，而用户ID则对应着N个取值，在按照信息增益选取时，很显然会选取用户ID作为分裂特征，但是我们知道每个用户的ID都是唯一的，所以这个模型就不具备泛化能力。所以为了避免在节点分裂时倾向于选取取值较多的特征，C4.5算法对ID3做了改进，将每个节点的分裂依据由信息增益变为信息增益率，公式如下:

$$Gain_R(D_i, k) = \frac{Gain(D_i, k)}{Entropy(k)}$$

其中$Gain(D_i, k)$为我们上面计算出的信息增益，$Entropy(k) = -\sum_{j=1}^{m} \frac{|D_{ij}|}{|D_i|} log \frac{|D_{ij}|}{|D_i|}$，所以可以看出虽然特征取值多会使得增益大，但是同时也会使得分母中的熵变大，因此可以在一定程度上改善每次节点分类只选取取值多的特征的问题。

CART树

节点分类依据: 分类问题使用基尼指数、回归问题则使用最小平方误差

随机森林

树模型的Ensemble方法: Bagging

GBDT

树模型的Ensemble方法: Boosting

GBDT部分，可以参考腾讯Yafei Zhang发布的GBDT相关的两个PDF文档，写的非常简洁易懂了，所以这里不再展开描述。下面贴文档内的一个图看一下GBDT和LR的比较:

	GBDT	LR
linearity	nonlinear	linear
fitting capability	good	less good
feature selection	naturally	only l1-reg
feature combination	naturally	no
regression	yes	no
classification	yes	yes
multi-class classification	support	support
output probability	support	yes
parallelization	hard and less beneficial	easy

LR与GBDT对比的复杂度

	GBDT	LR
speed	slow	fast
memory(training)	$O(N) + O(K\bar{J})$	$O(N) + O(M)$
cpu(training)	$O(K\bar{J}MN \log N)$	$O(KN)$
memory(predicting)	$O(K\bar{J})$	$O(M)$
cpu(predicting)	$O(K\bar{J})$	$O(M)$
N	large	huge
M	medium	huge

K在GBDT中代表K棵决策树，在LR中代表K轮迭代

N代表训练样本个数

M代表特征个数

J代表一棵树中叶子的个数

从我们的一些实际经验，GBDT所需的样本数其实并不是太多，基本百万级别就够了。

Yafei Zhang的文档: 重点推荐，可以说写的非常简洁易懂了。

gbdt.pdf

1.1 MB

gbdt-in-detail1.pdf

371.1 KB