xgb

1.算法概述

1.1 决策树（DT）是一种基本的分类和回归方法。在分类问题中它可以认为是if-then规则的集合，也可以认为是定义在特征空间与类空间上的条件概率分布，学习思想包括ID3,C4.5,CART（摘自《统计学习方法》）。

1.2 Bagging :基于数据随机重抽样的集成方法（Ensemble methods），也称为自举汇聚法（boostrap aggregating）,整个数据集是通过在原始数据集中随机选择一个样本进行替换得到的。进而得到S个基预测器（ base estimators），选择estimators投票最多的类别作为分类结果，estimators的平均值作为回归结果。（摘自《统计学习方法》和scikit集成方法介绍）

1.3 随机森林（RF）： 基于boostrap重抽样和随机选取特征，基预测器是决策树的集成方法（Ensemble methods）

1.4 Boosting :通过改变样本的权重（误分样本权重扩大）学习多个基预测器，并将这些预测器进行线性加权的集成方法 （摘自《统计学习方法》）

1.5 梯度提升决策树（GBDT）:基于boosting方法，提升方向是梯度方向的决策树的集成方法（Ensemble methods）

1.6 XGBDT:基于GBDT的一种升级版本，对目标函数做了二阶导数，主要改进是使用了正则化和特征分块存储并行处理（参考大杀器xgboost指南）

1.7 回归/分类树树模型函数：

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

，这里数据集被划分为R1,...,Rm个区域，每一个区域对应一个预测值Cm;其中I()是指示函数，当满足条件时返回1，否则为0

1.8 决策树常见的损失函数：

用于分类的损失函数：01损失，LR的对数损失，softmax的mlogloss

用于回归的损失函数：线性回归的MSE

2.算法推导

2.1 决策树生成过程就是一个递归的过程，如果满足某种停止条件（样本都是同一类别，迭代次数或者其他预剪枝参数）则返回多数投票的类作为叶结点标识；否则选择最佳划分特征和特征值生成|T|个子节点，对子节点数据进行划分；所以划分属性的计算方式是DT的精髓，以下总结各种划分属性的计算方法（附一个java实现决策树的demo）：

ID3与C4.5中使用的信息增益和信息增益率：

信息熵（Entropy）是表示随机变量不确定性的度量：

$$H(S) = -\sum_{x \in X} p(x) \log_2 p(x)$$

，其中S是数据集，X是类别集合，p（x）是类别x占数据集的比值。

信息增益（Information gain）表示数据集以特征A划分，数据集S不确定性下降的程度

$$IG(A, S) = H(S) - \sum_{t \in T} p(t) H(t)$$

，其中H(S)是原数据集S的熵；T是S以特征A划分的子集集合，即

$$S = \bigcup_{t \in T} t$$

p(t)是T的某一划分子集t占数据集S的比值，H(t)是划分子集t的熵。

信息增益率（为了克服ID3倾向于特征值多的特征）：

IG_Ratio = IG(A,S) / H(A)，特征A的熵越大，受到的惩罚也越多，从而保证特征之间的信息增益率可比

信息增益/信息增益率越大，样本集合的不确定性越小

CART中使用的Gini指数：

基尼（gini）指数是元素被随机选中的一种度量：

数据集D的gini系数：

$$\text{Gini}(D) = 1 - \sum_{k=1}^{K} \left(\frac{|C_k|}{|D|} \right)^2$$

在数据集D中以特征A划分的gini系数：

$$\text{Gini}(D, A) = \frac{|D_1|}{|D|} \text{Gini}(D_1) + \frac{|D_2|}{|D|} \text{Gini}(D_2)$$

gini指数越小，样本集合的不确定性越小

2.2回归树：CART,GBDT以及XGBoost 都可以用作回归树，所以这里梳理下回归树是如何确定划分特征和划分值的：

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right]$$

,其中C1,C2是划分区域R1,R2的均值，J,S是划分特征和划分的特征值

2.3 GBDT算法（来自这个论文,可以参考《统计学习方法》中例8.2手算一下）

输入：训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$，$x_i\in\mathcal{X}\subseteq\mathbf{R}^n$，$y_i\in\mathcal{Y}\subseteq\mathbf{R}$；损失函数 $L(y,f(x))$；

输出：回归树 $\hat{f}(x)$.

（1）初始化

$$f_0(x)=\arg\min_c\sum_{i=1}^{N}L(y_i,c)$$

（2）对 $m=1,2,\cdots,M$

（a）对 $i=1,2,\cdots,N$，计算

$$r_{mi}=-\left[\frac{\partial L(y_i,f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}$$

（b）对 r_{mi} 拟合一个回归树，得到第 m 棵树的叶结点区域 R_{mj}，$j=1,2,\cdots,J$

（c）对 $j=1,2,\cdots,J$，计算

$$c_{mj} = \arg\min_{c} \sum_{x_i \in R_{mj}} L(y_i, f_{m-1}(x_i) + c)$$

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

（3）得到回归树

$$\hat{f}(x) = f_M(x) = \sum_{m=1}^{M} \sum_{j=1}^{J} c_{mj} I(x \in R_{mj}) \quad \blacksquare$$

算法第 1 步初始化，估计使损失函数极小化的常数值，它是只有一个根结点的树. 第 2 (a) 步计算损失函数的负梯度在当前模型的值，将它作为残差的估计. 对于平方损失函数，它就是通常所说的残差；对于一般损失函数，它就是残差的近似值. 第 2 (b) 步估计回归树叶结点区域，以拟合残差的近似值. 第 2 (c) 步利用线性搜索估计叶结点区域的值，使损失函数极小化. 第 2 (d) 步更新回归树. 第 3 步得到输出的最终模型 $\hat{f}(x)$.

附：

处理	损失函数	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
回归	$1/2[y_i - f(x_i)]^2$	$y_i - f(x_i)$
回归	$\lvert y_i - f(x_i)\rvert$	$\text{sign}[y_i - f(x_i)]$
回归	Huber	$y_i - f(x_i)$，如果 $\lvert y_i - f(x_i)\rvert \leqslant \delta_m$ $\delta_m \text{sign}[y_i - f(x_i)]$，如果 $\lvert y_i - f(x_i)\rvert > \delta_m$ 其中，δ_m = 第 α 个分位数 $\{\lvert y_i - f(x_i)\rvert\}$
分类	散离	第 k 个分量：$I(y_i = \mathcal{G}_k) - p_k(x_i)$

参考自：《统计学习基础 数据挖据、推理与预测》 by Friedman 10.9节

2.4 XGBoost

模型函数：

$$J(f_t) = \sum_{i=1}^{n} [L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \| f_t(x_i) \| + C$$

$$\text{其中} g_i = \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}, h_i = \frac{\partial^2 L(y_i, \hat{y}_i^{(t-1)})}{\partial^2 \hat{y}_i^{(t-1)}}, \| f_t(x_i) \| \text{是关于决策树的范数函数}$$

最终得：

$$G_j = \sum_{i \in I_j} g_i, H_j = \sum_{i \in I_j} h_i, \text{其中} T \text{是叶子节点的集合}$$

$$J(f_t) = -\frac{1}{2}\sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma \cdot \mathrm{T}$$

详细的推导会附在文尾

3.算法特性及优缺点

决策树的优（特性）缺点：

优点：

（1）输出结果易于理解，

（2）对缺失值不敏感，可以处理无关数据，可以处理非线性数据

（3）对于异常点的容错能力好，健壮性高

（4）不需要提前归一化

（5）可以处理多维度输出的分类问题。

缺点：

（1）容易过拟合，需要剪枝或者RF避免

（2）只注重单个特征的局部划分，而不能像LR那样考虑整体的特征

ID3算法比较：

缺点：

（1）按照特征切分后，特征不在出现，切分过于迅速；

（2）只能处理类别类型，决策树不一定是二叉树，导致学习出来的树可能非常宽

（3）不能回归

（4）信息增益的结果偏向于数值多的特征

CART算法比较：

优点：离散或者连续特征可以重复使用；可以处理连续性特征；可以回归

RF的特性

优点：并行处理速度快，泛化能力强,可以很好的避免过拟合；能够得到特征的重要性评分（部分参考这篇总结）

对于不平衡数据集可以平衡误差（参考中文维基百科）

缺点：偏差会增大（方差减小）

GBDT的特性

优点：精度高;可以发现多种有区分性的特征以及特征组合

缺点：串行处理速度慢

XGB算法比较：

使用了二阶导数，对特征分块，从而可以并行加快运算速度

4.注意事项

4.1 树的剪枝（结合sklearn中的参数进行总结）

max_depth： DT的最大深度（默认值是3）

max_features： 最大特征数（默认值是None）

min_samples_split： 触发分割的最小样本个数，如果是2，两个叶结点各自1个样本

min_samples_leaf： 叶子节点的最小样本数，如果是2，两个叶结点至少2个样本

min_impurity_split： 最小分割纯度（与分割标准有关，越大越不容易过拟合）

（附：树模型调参）

4.2 如何计算的属性评分（结合sklearn总结）

The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance [R245].

4.3 正则化

衰减（Shrinkage）因子（learning_rate）：

proposed a simple regularization strategy that scales the contribution of each weak learner by a factor

:

The parameter

is also called the **learning rate** because it scales the step length the gradient descent procedure; it can be set via the learning_rate parameter. ——摘自scikit官网

eta [default=0.3]

- 为了防止过拟合，更新过程中用到的收缩步长。在每次提升计算之后，算法会直接获得新特征的权重。 eta通过缩减特征的权重使提升计算过程更加保守。缺省值为0.3
- 取值范围为：[0,1]
- 通常最后设置eta为0.01~0.2 ——摘自XGBOOST调优

正则项系数：

- lambda [default=0]
- L2 正则的惩罚系数
- 用于处理XGBoost的正则化部分。通常不使用，但可以用来降低过拟合
- alpha [default=0]
- L1 正则的惩罚系数

- 当数据维度极高时可以使用，使得算法运行更快。
- lambda_bias
- 在偏置上的L2正则。缺省值为0（在L1上没有偏置项的正则，因为L1时偏置不重要） ——摘自XGBOOST调优

4.4 其他参数：

class_weight：为了克服类别不均衡问题，引入代价敏感方法，class_weight是各类别的权重 ——摘自scikit官网：

- gamma [default=0]
- minimum loss reduction required to make a further partition on a leaf node of the tree. the larger, the more conservative the algorithm will be.
- range: [0,∞]
- 模型在默认情况下，对于一个节点的划分只有在其loss function 得到结果大于0的情况下才进行，而gamma 给定了所需的最低loss function的值
- gamma值使得算法更conservation，且其值依赖于loss function ，在模型中应该进行调参。 ——摘自XGBOOST调优

注：以上只是列出个人觉得比较重要的超参，具体的调参还请参见这两篇博文：RF和GBDT参数详解：《使用sklearn进行集成学习——实践》；XGBoost参数调优：XGBoost-Python完全调参指南-参数解释篇

5.实现和具体例子

微额借款人品预测竞赛

风控违约预测竞赛

CTR预测（GBDT+LR），sklearn官网RF+LR的进行特征选择的例子

Spark ml GradientBoostedTrees 核心实现部分

预测多变量的例子——(kaggle)预测沃尔玛超市38个购物区购物量

附：XGB推导目标函数推导：

对于：$J(f_t)=\sum_{j=1}^{T}\left[\left(\sum_{i\in I_j}g_i\right)w_j+\frac{1}{2}\left(\sum_{i\in I_j}h_i+\lambda\right)w_j^2\right]+\gamma\cdot T+C$

定义 $G_j\overset{def}{==}\sum_{i\in I_j}g_i,\ H_j\overset{def}{==}\sum_{i\in I_j}h_i$

从而，$J(f_t)=\sum_{j=1}^{T}\left[G_jw_j+\frac{1}{2}(H_j+\lambda)w_j^2\right]+\gamma\cdot T+C$

对W求偏导，得

$$\frac{\partial J(f_t)}{\partial w_j}=G_j+(H_j+\lambda)w_j\overset{令}{==}0\Rightarrow w_j=-\frac{G_j}{H_j+\lambda}$$

回代入目标函数，得 $J(f_t)=-\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda}+\gamma\cdot T$

对于：$J(f_t)=\sum_{j=1}^{T}\left[\left(\sum_{i\in I_j}g_i\right)w_j+\frac{1}{2}\left(\sum_{i\in I_j}h_i+\lambda\right)w_j^2\right]+\gamma\cdot T+C$

定义 $G_j\overset{def}{==}\sum_{i\in I_j}g_i,\ H_j\overset{def}{==}\sum_{i\in I_j}h_i$

从而，$J(f_t)=\sum_{j=1}^{T}\left[G_jw_j+\frac{1}{2}(H_j+\lambda)w_j^2\right]+\gamma\cdot T+C$

对W求偏导，得

$$\frac{\partial J(f_t)}{\partial w_j}=G_j+(H_j+\lambda)w_j\overset{令}{==}0\Rightarrow w_j=-\frac{G_j}{H_j+\lambda}$$

回代入目标函数，得 $J(f_t)=-\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda}+\gamma\cdot T$