Task3 XGB算法梳理

参考来源：

<https://blog.csdn.net/m0_38019841/article/details/85223633>

https://blog.csdn.net/qq\_39756719/article/details/85221193

**1. XGBoost**

XGBoost（eXtreme Gradient Boosting）是一种tree boosting的可扩展机器学习系统。这个系统可以作为开源的软件包使用。XGBoost是一个优化的分布式梯度增强库，旨在实现高效，灵活和便携。 它在Gradient Boosting框架下实现机器学习算法。XGBoost提供了并行树提升（也称为GBDT，GBM），可以快速准确地解决许多数据科学问题。 相同的代码在主要的分布式环境（Hadoop，SGE，MPI）上运行，并且可以解决超过数十亿个样例的问题。

**2. CART树**

分类与回归树（Classification and Regression Trees, CART），既可用于分类也可用于回归。当CART作为分类树时，采用GINI值作为节点分裂的依据。当CART作为回归树时，采用样本的最小方差作为节点分裂的依据。

**2.1 CART算法流程：**

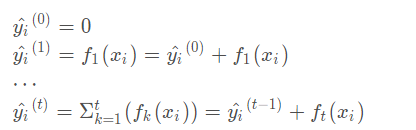
* 若满足停止分裂条件（样本个数小于预定阈值，或Gini指数小于预定阈值（样本基本属于同一类，或没有特征可供分裂），则停止分裂；
* 否则，选择最小Gini指数进行分裂；
* 递归执行1-2步骤，直至停止分裂。

**3. 算法原理**

假设每次迭代生成一棵树，第t次训练的目标函数可以写成：



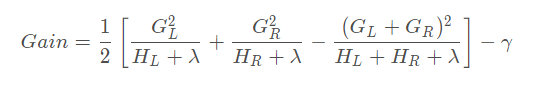
其中前半部分是训练误差，后半分是每棵树的复杂度。为第t步迭代的预测值，且有以下迭代关系：



XGBoost通过不断的生成新树，来优化决策，每一棵树都学习之前生成树的结论，不断优化。其基本思想和GBDT是一样的，都是按照损失函数的负梯度方向提升，只是XGBoost进行了泰勒二次展开，加了一些正则项。

**4. 分裂节点算法**

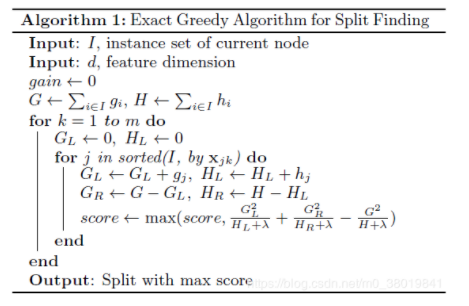
如何来寻找一个最优结构的树，加入到我们的模型中？常用的是贪心法，每一次尝试对已经的叶子加入一个分割，对一个具体的分割方案，我们可以获得分割后的增益为：



第一项为左子节点的得分，第二项为右子节点的得分，第三项为增加一个额外节点复杂度的损失。如果Gain<0,则此节点不应该split成左右两支。

对于每次扩展，我们还是要枚举所有可能的分割方案，实际应用中，先将从小到大排序，然后进行遍历，看每个结点是否需要分裂。

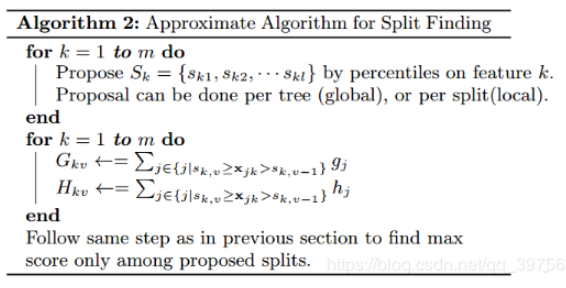
**4.1 贪心算法(exact greedy algorithm)**



当数据量过大，由于要遍历每个分割点，十分消耗内存，贪心算法就显得不再适用，所以提出了额外一种近似算法能加快运行时间。

**4.2 近似算法（approximate algorithm）**

近似算法根据特征的分布情况，先提出proposal，然后分割点就从候选proposal中选择，该算法大大的提高了效率。这里有两种proposal的方式，global、local的。global的是在建树之前就做proposal然后之后每次分割都要更新一下proposal，local的方法是在每次分裂之后更新proposal。

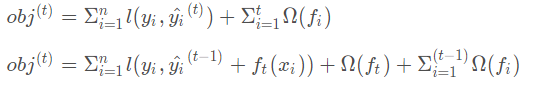


**4.3 分布式加权直方图算法 （Weighted Quantile Sketch）**

近似算法在特征分布然后做proposal的时候，用到了加权分位直方图。**将数据投射在一个小的存储空间内作为整个数据的概要，需要分裂节点时查看数据概要，以此来减少计算分裂节点是的用于储存数据的内存。**

**5 损失函数**

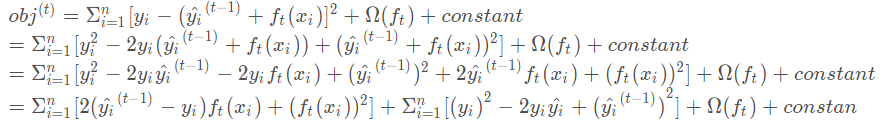
第t次训练的目标函数：



对于第t步来说 是已知的，所以可以得到下一步：



如果使用平方损失函数，则有



其中对于第t步来说，



也是常数，所以目标函数优化为：



其中一般叫做残差。

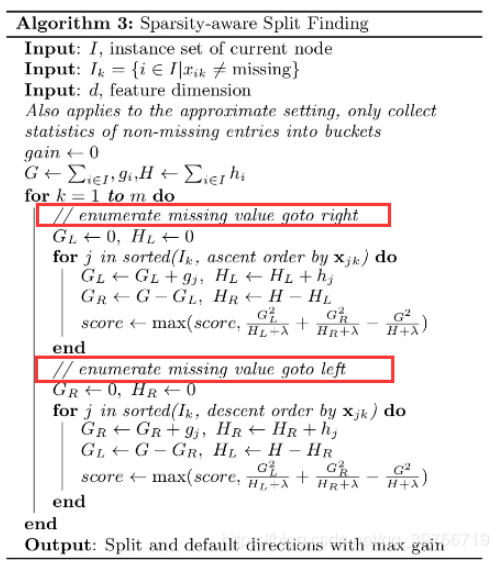
在实际的业务场景中，需要自定义损失函数：<https://github.com/dmlc/xgboost/blob/master/demo/guide-python/custom_objective.py>

**6. 正则化**

xgboost在代价函数里加入了正则项，用于控制模型的复杂度。正则项中包含了树的叶子节点个数、每个叶子节点上输出的score的L2模的平方和。 正则项降低了模型的variance，使学习出来的模型更加简单，防止过拟合。

**7. 对缺失值的处理**

XGBoost处理缺失值的方法如下图所示（截取自陈天奇论文），**XGBoost把缺失值当做稀疏矩阵来对待**，节点分裂时不考虑的缺失值的数值。在训练模型时训练模型时缺失数据会被分到左子树和右子树分别计层损失，选取结果较优的子树。在预测时预测数据中的缺失值，默认分类到右子树。



**8. 优缺点**

与GBDT相比，xgBoosting有以下进步：

* GBDT以传统CART作为基分类器，而xgBoosting**支持线性分类器**，相当于引入L1和L2正则化项的逻辑回归（分类问题）和线性回归（回归问题）；
* GBDT在优化时只用到一阶导数，xgBoosting对代价函数做了二阶Talor展开，**引入了一阶导数和二阶导数**；
* 当样本**存在缺失值时，xgBoosting能自动学习分裂方向；**
* xgBoosting借鉴RF的做法，**支持列抽样**，这样不仅能防止过拟合，还能降低计算；
* xgBoosting的代价函数**引入正则化项**，控制了模型的复杂度，正则化项包含全部叶子节点的个数，每个叶子节点输出的score的L2模的平方和。从贝叶斯方差角度考虑，正则项降低了模型的方差，防止模型过拟合；
* xgBoosting在每次迭代之后，为叶子结点分配学习速率，降低每棵树的权重，减少每棵树的影响，为后面提供更好的学习空间；
* xgBoosting工具**支持并行**,但并不是tree粒度上的，而是**特征粒度**，决策树最耗时的步骤是对特征的值排序，xgBoosting在迭代之前，先进行**预排序**，存为block结构，每次迭代，重复使用该结构，降低了模型的计算；block结构也为模型提供了并行可能，在进行结点的分裂时，计算每个特征的增益，选增益最大的特征进行下一步分裂，那么**各个特征的增益可以开多线程进行**；
* **可并行的近似直方图算法**，树结点在进行分裂时，需要计算每个节点的增益，若数据量较大，对所有节点的特征进行排序，遍历的得到最优分割点，这种贪心法异常耗时，这时引进近似直方图算法，用于生成高效的分割点，即用分裂后的某种值减去分裂前的某种值，获得增益，为了限制树的增长，引入阈值，当增益大于阈值时，进行分裂；

与LightGBM相比，又表现出了明显的不足：

* xgBoosting采用**预排序**，在迭代之前，对结点的特征做预排序，遍历选择最优分割点，**数据量大时，贪心法耗时**，LightGBM方法采用**histogram算法，占用的内存低，数据分割的复杂度更低**；
* xgBoosting采用level-wise生成决策树，同时分裂同一层的叶子，从而进行多线程优化，不容易过拟合，但**很多叶子节点的分裂增益较低，没必要进行跟进一步的分裂，这就带来了不必要的开销**；LightGBM采用**深度优化**，leaf-wise生长策略，每次从当前叶子中选择增益最大的结点进行分裂，循环迭代，但会生长出更深的决策树，产生过拟合，因此引入了一个阈值进行限制，防止过拟合。

**9. sklearn参数**

参考来源：

https://blog.csdn.net/m0\_38019841/article/details/85223633

https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

objective [ default=reg:linear ] 定义学习任务及相应的学习目标，可选的目标函数如下：

“reg:linear” –线性回归。

“reg:logistic” –逻辑回归。

“binary:logistic” –二分类的逻辑回归问题，输出为概率。

“binary:logitraw” –二分类的逻辑回归问题，输出的结果为wTx。

“count:poisson” –计数问题的poisson回归，输出结果为poisson分布。 在poisson回归中，max\_delta\_step的缺省值为0.7。(used to safeguard optimization)

“multi:softmax” –让XGBoost采用softmax目标函数处理多分类问题，同时需要设置参数num\_class（类别个数）

“multi:softprob” –和softmax一样，但是输出的是ndata \* nclass的向量，可以将该向量reshape成ndata行nclass列的矩阵。没行数据表示样本所属于每个类别的概率。

’eval\_metric’ 评估指标:对于回归问题，默认值是rmse，对于分类问题，默认值是error。

“rmse”: 均方根误差

“logloss”: 负对数似然函数值

“error”: 二分类错误率(阈值为0.5)

“merror”: 多分类错误率

“mlogloss”: 多分类logloss损失函数

“auc”: ROC曲线下面积

“mae”：平均绝对误差

lambda [default=0] L2 正则的惩罚系数

alpha [default=0] L1 正则的惩罚系数

lambda\_bias 在偏置上的L2正则。缺省值为0

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]

max\_depth [default=6] 树的最大深度。缺省值为6 ，取值范围为：[1,∞]

min\_child\_weight [default=1] 孩子节点中最小的样本权重和。如果一个叶子节点的样本权重和小于min\_child\_weight则拆分过程结束。在现行回归模型中，这个参数是指建立每个模型所需要的最小样本数。该成熟越大算法越conservative 。取值范围为: [0,∞]

tree\_method[default=’auto’] 可选 {‘auto’, ‘exact’, ‘approx’} 贪心算法(小数据集)/近似算法(大数据集)。

**10. 应用场景**

XGBoost由于其优越性被广泛运用与分类和回归预测中，可以用于二分类也可以用于多分类。同时由于其可以并行话实现也可以用于工业规模的应用中。