### 初看Xgboost，翻了多篇博客发现关于xgboost原理的描述实在难以忍受，缺乏逻辑性，写一篇供讨论。

——以下是抛砖引玉。   
观其大略，而后深入细节，一开始扎进公式反正我是觉得效率不高，还容易打消人的积极性。

### 首先说下决策树

**决策树是啥？**   
举个例子，有一堆人，我让你分出男女，你依靠头发长短将人群分为两拨，长发的为“女”，短发为“男”，你是不是依靠一个指标“头发长短”将人群进行了划分，你就形成了一个简单的决策树，官方细节版本自行baidu或google

**划分的依据是啥？**   
这个时候，你肯定问，为什么用“头发长短”划分啊，我可不可以用“穿的鞋子是否是高跟鞋”，“有没有喉结”等等这些来划分啊，Of course！那么肯定就需要判断了，那就是哪一种分类效果好，我就选哪一种啊。

**分类效果如何评价量化呢？**   
怎么判断“头发长短”或者“是否有喉结”…是最好的划分方式，效果怎么量化。直观来说，如果根据某个标准分裂人群后，**纯度越高效果越好**，比如说你分为两群，“女”那一群都是女的，“男”那一群全是男的，这个效果是最好的，但事实不可能那么巧合，所以越接近这种情况，我们认为效果越好。于是量化的方式有很多，信息增益（ID3）、信息增益率（C4.5）、基尼系数（CART）等等，来用来量化纯度

其他细节如剪枝、过拟合、优缺点、并行情况等自己去查吧。决策树的灵魂就已经有了，依靠某种指标进行树的分裂达到分类/回归的目的（上面的例子是分类），总是希望纯度越高越好。

### 说下Xgboost的建树过程

Xgboost是很多CART回归树集成

概念1：回归树与决策树   
事实上，分类与回归是一个型号的东西，只不过分类的结果是离散值，回归是连续的，本质是一样的，都是**特征（feature）到结果/标签（label）之间的映射**。说说决策树和回归树，在上面决策树的讲解中相信决策树分类已经很好理解了。

回归树是个啥呢？

直接摘抄人家的一句话，分类树的样本输出（即响应值）是类的形式，如判断蘑菇是有毒还是无毒，周末去看电影还是不去。而**回归树的样本输出是数值的形式**，比如给某人发放房屋贷款的数额就是具体的数值，可以是0到120万元之间的任意值。

那么，这时候你就没法用上述的信息增益、信息增益率、基尼系数来判定树的节点分裂了，你就会采用新的方式，**预测误差**，常用的有均方误差、对数误差等。而且节点不再是类别，是数值（预测值），那么怎么确定呢，有的是节点内样本均值，有的是最优化算出来的比如Xgboost。   
细节[http://blog.csdn.net/app\_12062011/article/details/52136117](http://blog.csdn.net/app_12062011/article/details/52136117" \t "_blank)博主讲的不错

概念2：boosting集成学习，由多个相关联的决策树联合决策，什么叫相关联，举个例子，有一个样本[数据->标签]是[(2，4，5)-> 4]，第一棵决策树用这个样本训练得预测为3.3，那么第二棵决策树训练时的输入，这个样本就变成了[(2，4，5)-> 0.7]，也就是说，**下一棵决策树输入样本会与前面决策树的训练和预测相关**。

与之对比的是random foreast（随机森林）算法，**各个决策树是独立的**、每个决策树在样本堆里随机选一批样本，随机选一批特征进行独立训练，各个决策树之间没有啥毛线关系。

所以首先Xgboost首先是一个boosting的集成学习，这样应该很通俗了

这个时候大家就能感觉到一个回归树形成的关键点：（1）分裂点依据什么来划分（如前面说的均方误差最小，loss）；（2）分类后的节点预测值是多少（如前面说，有一种是将叶子节点下各样本实际值得均值作为叶子节点预测误差，或者计算所得）

**是时候看看Xgboost了**

首先明确下我们的目标，希望建立K个回归树，使得树群的预测值尽量接近真实值（准确率）而且有尽量大的泛化能力（更为本质的东西），**从数学角度看这是一个泛函最优化，多目标，**看下目标函数：

L(ϕ)=∑il(y^i−yi)+∑kΩ(fk)L(ϕ)=∑il(y^i−yi)+∑kΩ(fk)

其中ii表示第i个样本，l((y^i−yi)l((y^i−yi)表示第ii个样本的预测误差，误差越小越好，不然你算得上预测么？后面∑kΩ(fk)∑kΩ(fk)表示树的复杂度的函数，越小复杂度越低，泛化能力越强，这意味着啥不用我多说。表达式为

Ω(f)=γT+12λ||w||2Ω(f)=γT+12λ||w||2

TT表示叶子节点的个数，ww表示节点的数值（这是回归树的东西，分类树对应的是类别）

直观上看，**目标要求预测误差尽量小，叶子节点尽量少，节点数值尽量不极端**（这个怎么看，如果某个样本label数值为4，那么第一个回归树预测3，第二个预测为1；另外一组回归树，一个预测2，一个预测2，那么倾向后一种，为什么呢？前一种情况，第一棵树学的太多，太接近4，也就意味着有较大的过拟合的风险）

ok，听起来很美好，可是**怎么实现呢**，上面这个目标函数跟实际的参数怎么联系起来，记得我们说过，回归树的参数:（1）选取哪个feature分裂节点呢；（2）节点的预测值（总不能靠取平均值这么粗暴不讲道理的方式吧，好歹高级一点）。上述形而上的公式并没有“直接”解决这两个，那么是如何间接解决的呢？

**先说答案：贪心策略+最优化（二次最优化，恩你没看错）**

通俗解释贪心策略：就是决策时刻按照当前目标最优化决定，说白了就是眼前利益最大化决定，“目光短浅”策略，他的优缺点细节大家自己去了解，经典背包问题等等。

这里是怎么用贪心策略的呢，刚开始你有一群样本，**放在第一个节点**，这时候T=1T=1，ww多少呢，不知道，是求出来的，这时候所有样本的预测值都是ww（这个地方自己好好理解，决策树的节点表示类别，回归树的节点表示预测值）,带入样本的label数值，此时loss function变为

L(ϕ)=∑il(w−yi)+γ+12λ||w||2L(ϕ)=∑il(w−yi)+γ+12λ||w||2

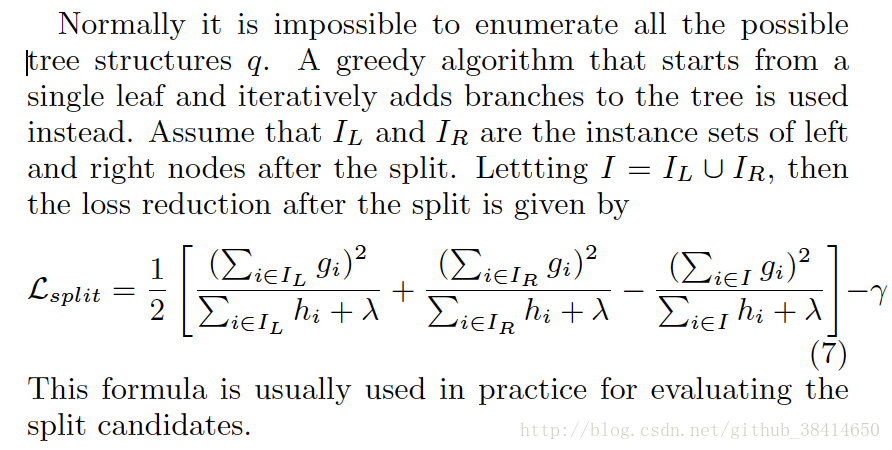
如果这里的l(w−yi)l(w−yi)误差表示用的是平方误差，那么上述函数就是一个关于ww的二次函数求最小值，取最小值的点就是这个节点的预测值，最小的函数值为最小损失函数。

**暂停下，这里你发现了没，二次函数最优化！**   
要是损失函数不是二次函数咋办，哦，泰勒展开式会否？，不是二次的想办法近似为二次。

接着来，接下来要选个feature分裂成两个节点，变成一棵弱小的树苗，那么需要：（1）确定分裂用的feature，how？最简单的是粗暴的枚举，选择loss function效果最好的那个（关于粗暴枚举，Xgboost的改良并行方式咱们后面看）；（2）如何确立节点的ww以及最小的loss function，大声告诉我怎么做？对，二次函数的求最值（细节的会注意到，计算二次最值是不是有固定套路，导数=0的点，ok）

那么节奏是，选择一个feature分裂，计算loss function最小值，然后再选一个feature分裂，又得到一个loss function最小值…你枚举完，找一个效果最好的，把树给分裂，就得到了小树苗。

在分裂的时候，你可以注意到，**每次节点分裂，loss function被影响的只有这个节点的样本**，因而每次分裂，计算分裂的增益（loss function的降低量）只需要关注打算分裂的那个节点的样本。原论文这里会推导出一个优雅的公式，我不想敲latex公式了，



想研究公式的去这里吧   
[http://matafight.github.io/2017/03/14/XGBoost-%E7%AE%80%E4%BB%8B/](http://matafight.github.io/2017/03/14/XGBoost-%E7%AE%80%E4%BB%8B/" \t "_blank)….

**接下来，继续分裂，按照上述的方式，形成一棵树，再形成一棵树**，**每次在上一次的预测基础上取最优进一步分裂/建树，是不是贪心策略？！**

凡是这种**循环迭代的方式必定有停止条件，什么时候停止**呢：   
（1）当引入的**分裂带来的增益小于一个阀值的时**候，我们可以剪掉这个分裂，所以并不是每一次分裂loss function整体都会增加的，有点预剪枝的意思（其实我这里有点疑问的，一般后剪枝效果比预剪枝要好点吧，只不过复杂麻烦些，这里大神请指教，为啥这里使用的是预剪枝的思想，当然Xgboost支持后剪枝），阈值参数为γγ正则项里叶子节点数T的系数（大神请确认下）；   
（2）当树达到最大深度时则停止建立决策树，设置一个超参数max\_depth，这个好理解吧，树太深很容易出现的情况学习局部样本，**过拟合**；   
（3）当样本权重和小于设定阈值时则停止建树，这个解释一下，涉及到一个超参数-最小的样本权重和min\_child\_weight，和GBM的 min\_child\_leaf 参数类似，但不完全一样，大意就是一个叶子节点样本太少了，也终止**同样是过拟合**；   
（4）貌似看到过有树的最大数量的…这个不确定   
**具体数学推导细节（只要是那个节点分裂增益计算的公式），请参看作者（论文作者哦）介绍，很细致！**[http://www.52cs.org/?p=429](http://www.52cs.org/?p=429" \t "_blank)，上面那个也可以   
问题1：节点分裂的时候是按照哪个顺序来的，比如第一次分裂后有两个叶子节点，先裂哪一个？   
答案：呃，同一层级的（多机）并行，确立如何分裂或者不分裂成为叶子节点，来源   
[https://wenku.baidu.com/view/44778c9c312b3169a551a460.html](https://wenku.baidu.com/view/44778c9c312b3169a551a460.html" \t "_blank)

### 看下Xgboost的一些重点

ww是最优化**求出来的**，不是啥平均值或规则指定的，这个算是一个思路上的新颖吧；

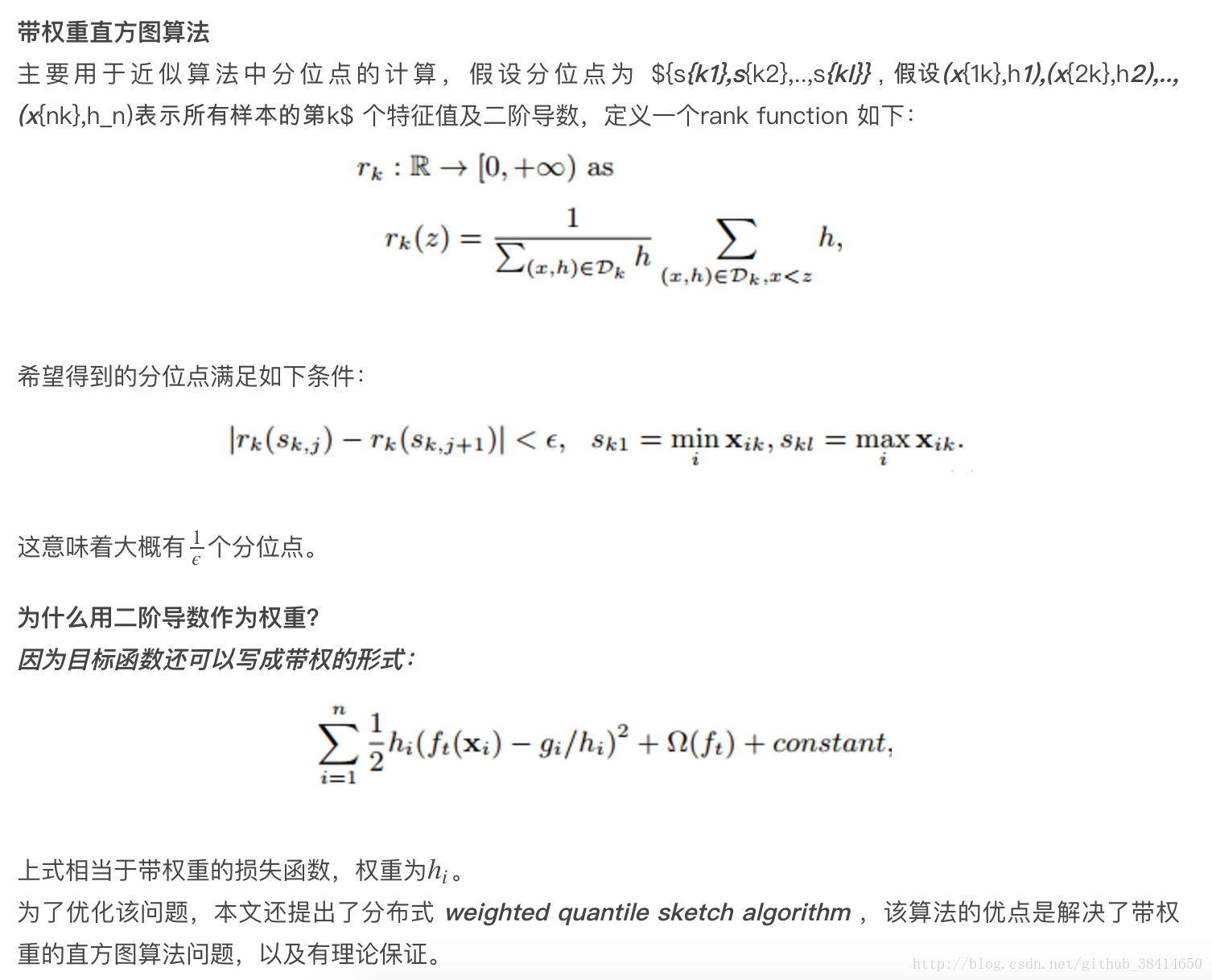
**正则化**防止过拟合的技术，上述看到了，直接loss function里面就有；

支持自定义loss function，哈哈，不用我多说，只要能泰勒展开（能求一阶导和二阶导）就行，你开心就好；

**支持并行化**，这个地方有必要说明下，因为这是xgboost的闪光点，**直接的效果是训练速度快**，boosting技术中下一棵树依赖上述树的训练和预测，所以树与树之间应该是只能串行！那么大家想想，哪里可以并行？！   
**没错，在选择最佳分裂点，进行枚举的时候并行！**（据说恰好这个也是树形成最耗时的阶段）

**Attention：同层级节点可并行。具体的对于某个节点，节点内选择最佳分裂点，候选分裂点计算增益用多线程并行。**—–

较少的离散值作为分割点倒是很简单，比如“是否是单身”来分裂节点计算增益是很easy，但是“月收入”这种feature，取值很多，从5k~50k都有，总不可能每个分割点都来试一下计算分裂增益吧？（比如月收入feature有1000个取值，难道你把这1000个用作分割候选？缺点1：计算量，缺点2：出现叶子节点样本过少，过拟合）我们常用的习惯就是划分区间，那么问题来了，这个区间分割点如何确定（难道平均分割），作者是这么做的：

**方法名字：Weighted Quantile Sketch**   
大家还记得每个样本在节点（将要分裂的节点）处的loss function一阶导数gigi和二阶导数hihi，衡量预测值变化带来的loss function变化，举例来说，将样本“月收入”进行升序排列，5k、5.2k、5.3k、…、52k，分割线为“收入1”、“收入2”、…、“收入j”，满足(每个间隔的样本的hihi之和/总样本的hihi之和）为某个百分比ϵϵ（我这个是近似的说法），那么可以一共分成大约1/ϵ1/ϵ个分裂点。   
数学形式，我再偷懒下（可是latex敲这种公式真的很头疼）：   
  
而且，有适用于分布式的算法设计；

XGBoost还特别设计了**针对稀疏数据的算法**，   
假设样本的第i个特征缺失时，无法利用该特征对样本进行划分，这里的做法是将该样本默认地分到指定的子节点，至于具体地分到哪个节点还需要某算法来计算，

算法的主要思想是，分别假设特征缺失的样本属于右子树和左子树，而且**只在不缺失的样本上迭代**，分别计算缺失样本属于右子树和左子树的增益，选择增益最大的方向为缺失数据的默认方向（咋一看如果缺失情况为3个样本，那么划分的组合方式岂不是有8种？指数级可能性啊，仔细一看，应该是在不缺失样本情况下**分裂后**（有大神的请确认或者修正），把第一个缺失样本放左边计算下loss function和放右边进行比较，同样对付第二个、第三个…缺失样本，这么看来又是可以并行的？？）；

* 可实现后剪枝

**交叉验证**，方便选择最好的参数，early stop，比如你发现30棵树预测已经很好了，不用进一步学习残差了，那么停止建树。

行采样、列采样，随机森林的套路（防止过拟合）

**Shrinkage**，你可以是几个回归树的叶子节点之和为预测值，也可以是加权，比如第一棵树预测值为3.3，label为4.0，第二棵树才学0.7，….再后面的树还学个鬼，所以给他打个折扣，比如3折，那么第二棵树训练的残差为4.0-3.3\*0.3=3.01，这就可以发挥了啦，以此类推，作用是啥，防止过拟合，如果对于“伪残差”学习，那更像**梯度下降里面的学习率**；

* xgboost还支持设置样本权重，这个权重体现在梯度g和二阶梯度h上，是不是有点adaboost的意思，重点关注某些样本

### 看下Xgboost的工程优化

这部分因为没有实战经验，都是论文、博客解读来的，所以也不十分确定，供参考。

Column Block for Parallel Learning   
总的来说：按列切开，升序存放；   
方便并行，同时解决一次性样本读入炸内存的情况

由于将数据按列存储，可以同时访问所有列，那么可以对所有属性同时执行split finding算法，从而并行化split finding（切分点寻找）-**特征间并行**   
可以用多个block(Multiple blocks)分别存储不同的样本集，多个block可以并行计算-**特征内并行**

* Blocks for Out-of-core Computation   
  数据大时分成多个block存在磁盘上，在计算过程中，用另外的线程读取数据，但是由于磁盘IO速度太慢，通常更不上计算的速度，   
  将block按列压缩，对于行索引，只保存第一个索引值，然后只保存该数据与第一个索引值之差(offset)，一共用16个bits来保存 offset，因此，一个block一般有216216个样本。
* …

### 与GDBT、深度学习对比下

Xgboost第一感觉就是**防止过拟合+各种支持分布式/并行**，所以一般传言这种大杀器效果好（集成学习的高配）+训练效率高（分布式），与深度学习相比，对样本量和特征数据类型要求没那么苛刻，适用范围广。

说下GBDT：有两种描述版本，把GBDT说成一个迭代残差树，认为每一棵迭代树都在学习前N-1棵树的残差；把GBDT说成一个梯度迭代树，使用梯度迭代下降法求解，认为每一棵迭代树都在学习前N-1棵树的梯度下降值。有说法说前者是后者在loss function为平方误差下的特殊情况。这里说下我的理解，仍然举个例子：第一棵树形成之后，有预测值y^iy^i，真实值（label）为yiyi，前者版本表示下一棵回归树根据样本(xi,yi−y^i)(xi,yi−y^i)进行学习，后者的意思是计算loss function在第一棵树预测值附近的梯度负值作为新的label，也就是对应xgboost中的−gi−gi

这里真心有个疑问：   
Xgboost在下一棵树拟合的是残差还是负梯度，还是说是一阶导数+二阶导数，−gi(1+hi)−gi(1+hi)?可能人蠢，没看太懂，换句话说GBDT残差树群有一种拟合的（输入样本）是(xi,yi−y^i)(xi,yi−y^i),还一种拟合的是(xi,−gi)(xi,−gi)，Xgboost呢？

Xgboost和深度学习的关系，陈天奇在Quora上的解答如下：   
不同的机器学习模型适用于不同类型的任务。深度神经网络通过对**时空位置建模**，能够很好地捕获图像、语音、文本等高维数据。而基于树模型的XGBoost则能很好地**处理表格数据**，同时还拥有一些深度神经网络所没有的特性（如：模型的可解释性、输入数据的不变性、更易于调参等）。   
这两类模型都很重要，并广泛用于数据科学竞赛和工业界。举例来说，几乎所有采用机器学习技术的公司都在使用tree boosting，同时XGBoost已经给业界带来了很大的影响。

### 参考文献/博文

（有的可能需要vpn）   
原论文-[https://arxiv.org/pdf/1603.02754v1.pdf](https://arxiv.org/pdf/1603.02754v1.pdf" \t "_blank)   
博客1-[http://blog.csdn.net/a358463121/article/details/68617389](http://blog.csdn.net/a358463121/article/details/68617389" \t "_blank)   
博客2-[http://blog.csdn.net/sb19931201/article/details/52557382](http://blog.csdn.net/sb19931201/article/details/52557382" \t "_blank)   
博客3-[http://blog.csdn.net/xiaocong1990/article/details/55107239](http://blog.csdn.net/xiaocong1990/article/details/55107239" \t "_blank)   
博客4-   
[http://blog.csdn.net/totoro1745/article/details/53328725?utm\_source=itdadao&utm\_medium=referral](http://blog.csdn.net/totoro1745/article/details/53328725?utm_source=itdadao&utm_medium=referral" \t "_blank)   
资料5-[http://matafight.github.io/2017/03/14/XGBoost-%E7%AE%80%E4%BB%8B/](http://matafight.github.io/2017/03/14/XGBoost-%E7%AE%80%E4%BB%8B/" \t "_blank)   
资料6-[https://wenku.baidu.com/view/44778c9c312b3169a551a460.html](https://wenku.baidu.com/view/44778c9c312b3169a551a460.html" \t "_blank)

--------------------- 本文来自 滚滚的纸盒子 的CSDN 博客 ，全文地址请点击：https://blog.csdn.net/github\_38414650/article/details/76061893?utm\_source=copy