18. GBDT常见问题★

1．gbdt 的算法的流程?

gbdt通过多轮迭代,每轮迭代产生一个弱分类器，每个分类器在上一轮分类器的**残差**基础上进行训练。对弱分类器的要求一般是足够简单，并且是低方差和高偏差的。因为训练的过程是通过降低偏差来不断提高最终分类器的精度，（此处是可以证明的）。弱分类器一般会选择为CART树。由于上述高偏差和简单的要求 每个分类回归树的深度不会很深。最终的总分类器是将每轮训练得到的弱分类器加权求和得到的（也就是加法模型）。

模型最终可以描述为：

$$F_m(x) = \sum_{m=1}^{M} T(x;\theta_m)$$

模型一共训练M轮，每轮产生一个弱分类器 T(x;θm)。弱分类器的**损失函数**

$$\hat{\theta}_m = \arg\min_{\theta_m} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + T(x_i;\theta_m))$$

Fm−1(x)为当前的模型，gbdt 通过经验风险极小化来确定下一个弱分类器的参数。具体到损失函数本身的选择也就是L的选择，有平方损失函数，0-1损失函数，对数损失函数等等。如果我们选择平方损失函数，那么这个差值其实就是我们平常所说的残差。但是其实我们真正关注的，1.是希望损失函数能够不断的减小，2.是希望损失函数能够尽可能快的减小。所以如何尽可能快的减小呢?

让损失函数沿着梯度方向的下降。这个就是gbdt 的 gb的核心了。**利用损失函数的负梯度在当前模型的值作为残差的近似值去拟合一个回归树**。gbdt 每轮迭代的时候，都去拟合损失函数在当前模型下的负梯度。这样每轮训练的时候都能够让损失函数尽可能快的减小，尽快的收敛达到局部最优解或者全局最优解。

2．gbdt 如何选择特征？

gbdt选择特征的细节其实是想问你CART Tree生成的过程。这里有一个前提，gbdt的弱分类器默认选择的是CART TREE。其实也可以选择其他弱分类器的，选择的前提是低方差和高偏差。框架服从boosting 框架即可。下面我们具体来说CART TREE(是一种二叉树) 如何生成。CART TREE 生成的过程其实就是一个选择特征的过程。假设我们目前总共有 M 个特征。第一步我们需要从中选择出一个特征 j，做为二叉树的第一个节点。然后对特征 j 的值选择一个切分点 m. 一个 样本的特征j的值 如果小于m，则分为一类，如果大于m,则分为另外一类。如此便构建了CART 树的一个节点。其他节点的生成过程和这个是一样的。现在的问题是在每轮迭代的时候，如何选择这个特征 j,以及如何选择特征 j 的切分点 m。原始的gbdt的做法非常的暴力，首先遍历每个特征，然后对每个特征遍历它所有可能的切分点，找到最优特征 m 的最优切分点 j。对于特征 j，我们遍历特征 j 所有特征值的切分点 s。找到可以让下面这个式子最小的特征 j 以及切分点s.

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

3．gbdt 如何构建特征？

其实说gbdt 能够构建特征并非很准确，gbdt 本身是不能产生特征的，但是我们可以利用gbdt去产生特征的组合。在CTR预估中，工业界一般会采用逻辑回归 <http://www.cnblogs.com/ModifyRong/p/7739955.html#3825035> 去进行处理,在我的上一篇博文当中已经说过，逻辑回归本身是适合处理线性可分的数据，如果我们想让逻辑回归处理非线性的数据，其中一种方式便是组合不同特征，增强逻辑回归对非线性分布的拟合能力。

长久以来，我们都是通过人工的先验知识或者实验来获得有效的组合特征，但是很多时候，使用人工经验知识来组合特征过于耗费人力，造成了机器学习当中一个很奇特的现象：有多少人工就有多少智能。关键是这样通过人工去组合特征并不一定能够提升模型的效果。所以我们的从业者或者学界一直都有一个趋势便是通过算法自动，高效的寻找到有效的特征组合。Facebook 在2014年 发表的一篇论文便是这种尝试下的产物，利用gbdt去产生有效的特征组合，以便用于逻辑回归的训练，提升模型最终的效果。

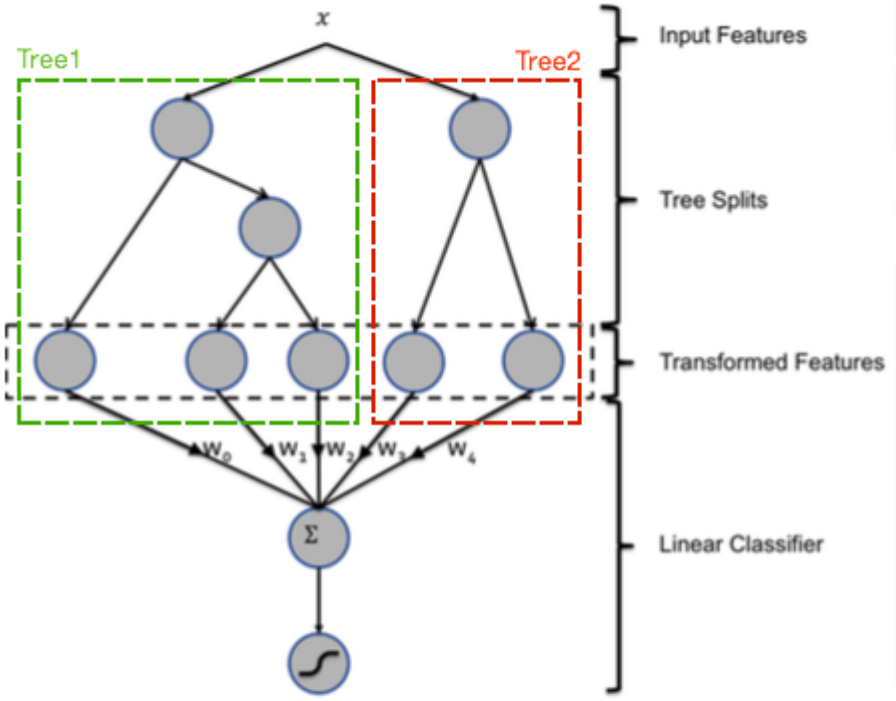

Figure 1: Hybrid model structure. Input features are transformed by means of boosted decision trees. The output of each individual tree is treated as a categorical input feature to a sparse linear classifier. Boosted decision trees prove to be very powerful feature transforms.

如图 2所示，我们 使用 GBDT 生成了两棵树，两颗树一共有五个叶子节点。我们将样本 X 输入到两颗树当中去，样本X 落在了第一棵树的第二个叶子节点，第二颗树的第一个叶子节点，于是我们便可以依次构建一个五纬的特征向量，每一个纬度代表了一个叶子节点，样本落在这个叶子节点上面的话那么值为1，没有落在该叶子节点的话，那么值为 0。于是对于该样本，我们可以得到一个向量[0,1,0,1,0] 作为该样本的组合特征，和原来的特征一起输入到逻辑回归当中进行训练。实验证明这样会得到比较显著的效果提升。

4．GBDT使用CART回归树而不是分类树的原因

GBDT主要是利用残差逼近的方式，这就意味每棵树的值是连续的可叠加的，这一点和回归树输出连续值不谋而合，如果采用分类树，那么残差逼近进行叠加就会使得这种叠加没有意义，比如男+男+女=到底是男是女。这个是GBDT基本原理决定的。

5．gbdt 如何用于分类?

首先明确一点，gbdt 无论用于分类还是回归一直都是使用的CART 回归树。不会因为我们所选择的任务是分类任务就选用分类树，这里面的核心是因为gbdt 每轮的训练是在上一轮的训练的残差基础之上进行训练的。这里的残差就是当前模型的负梯度值 。这个要求每轮迭代的时候，弱分类器的输出的结果相减是有意义的。残差相减是有意义的。

如果选用的弱分类器是分类树，类别相减是没有意义的。上一轮输出的是样本 x 属于 A类，本一轮训练输出的是样本 x 属于 B类。 A 和 B 很多时候甚至都没有比较的意义，A 类- B类是

没有意义的。

我们具体到分类这个任务上面来，我们假设样本 X 总共有 K类。来了一个样本 x，我们需要使用gbdt来判断 x 属于样本的哪一类。

第一步 我们在训练的时候，是针对样本 X 每个可能的类都训练一个分类回归树。举例说明，目前样本有三类，也就是 K = 3。样本 x 属于 第二类。那么针对该样本 x 的分类结果，其实我们可以用一个三维向量 [0,1,0] 来表示。0表示样本不属于该类，1表示样本属于该类。由于样本已经属于第二类了，所以第二类对应的向量维度为1，其他位置为0。

针对样本有 三类的情况，我们实质上是在每轮的训练的时候是同时训练三颗树。第一颗树针对样本x的第一类，输入为（x,0）。第二颗树输入针对 样本x 的第二类，输入为（x,1）。第三颗树针对样本x 的第三类，输入为（x，0）

在这里每颗树的训练过程其实就是就是我们之前已经提到过的CATR TREE 的生成过程。在此处我们参照之前的生成树的程序 即可以就解出三颗树，以及三颗树对x 类别的预测值 f1(x),f2(x),f3(x)。那么在此类训练中，我们仿照多分类的逻辑回归 ，使用softmax 来产生概率，则属于类别 1 的概率：

$$p_1 = exp(f_1(x)) / \sum_{k=1}^{3} exp(f_k(x))$$

并且我们我们可以针对类别1 求出 残差y11(x)=0−p1(x)；类别2 求出残差 y22(x)=1−p2(x)；类别3 求出残差y33(x)=0−p3(x)

然后开始第二轮训练 针对第一类 输入为（x,y11(x)），针对第二类输入为（x,y22(x)), 针对 第三类输入为 (x,y33(x)).继续训练出三颗树。一直迭代M轮。每轮构建 3颗树。

所以当K =3。我们其实应该有三个式子：

$$F_{1M}(x) = \sum_{m=1}^{M} \hat{C_{1m}} I(x \epsilon R_{1m})$$

$$F_{2M}(x) = \sum_{m=1}^{M} \hat{C_{2m}} I(x \epsilon R_{2m})$$

$$F_{3M}(x) = \sum_{m=1}^{M} \hat{C_{3m}} I(x \epsilon R_{3m})$$

当训练完毕以后，新来一个样本 x1 ，我们需要预测该样本的类别的时候，便可以有这三个式子产生三个值，f1(x),f2(x),f3(x)。样本属于 某个类别c的概率为 ：

$$p_c = exp(f_c(x)) / \sum_{k=1}^{3} exp(f_k(x))$$

6．gbdt 通过什么方式减少误差？

每棵树都是在拟合当前模型的预测值和真实值之间的误差，GBDT是通过不断迭代来使得误差减小的过程。

7．gbdt的效果相比于传统的LR，SVM效果为什么好一些？

- GBDT是集成模型
- GBDT基于树模型，继承了cart树模型的优点：缺失值、连续值、不平衡数据、计算量、剪枝策略

8．gbdt 如何加速训练?

预排序可以加速查找最佳分裂点

9．gbdt的参数有哪些，如何调参？

- n_estimators：最大弱学习器的个数，太小欠拟合，太大过拟合
- learning_rate：学习率，太大过拟合，一般很小0.1，和n_estimators一起调
- subsample：子采样，防止过拟合，太小欠拟合。GBDT中是不放回采样
- max_features：最大特征数
- max_depth：最大树深，太大过拟合
- min_samples_split：内部节点再划分所需最小样本数，越大越防过拟合
- min_weight_fraction_leaf：叶子节点最小的样本权重和。如果存在较多样本有缺失值，或者分类树样本的分布类别偏差很大，就会引入样本权重，这时我们就要注意这个值了。越大越防过拟合
- max_leaf_nodes：最大叶子节点数，太大过拟合

10．gbdt的优缺点？

优点:

- 预测阶段的计算速度快，树与树之间可并行化计算。
- 在分布稠密的数据集上，泛化能力和表达能力都很好。
- 采用决策树作为弱分类器使得GBDT模型具有较好的解释性和鲁棒性
- 能够自动发现特征间的高阶关系。

缺点:

- GBDT在高维稀疏的数据集上，表现不如支持向量机或者神经网络。
- GBDT在处理文本分类特征问题上，相对其他模型的优势不如它在处理数值特征时明显。
- 训练过程需要串行训练，只能在决策树内部采用一些局部并行的手段提高训练速度。

11．梯度提升和梯度下降的区别和联系是什么?

梯度下降:

梯度下降法(Gradient Decent,Steepest Decent)用于求解无约束最优化问题：

$$x^* = \arg\min_{x \in \mathbb{R}^n} f(x) \tag{1.1}$$

记第 k 次迭代值为 $x^{(k)}$ ，初始值为 $x^{(0)}$ 。

更新公式：

$$x^{(k+1)} \leftarrow x^{(k)} - \mu^{(k)} \cdot \nabla f(x^{(k)}) \tag{1.2}$$

$\nabla f(x^{(k)})$ 是 f 在点 $x^{(k)}$ 的梯度。 $\mu^{(k)} > 0$ 是学习率或步长，可以是定值，也可以是使得 f 下降最大的那个步长(一维搜索，line search)：

$$\mu^{(k)} = \arg\min_{\mu} f(x^{(k)} - \mu \cdot \nabla f(x^{(k)})) \tag{1.3}$$

通常的理解是，更新公式中的 x 是模型参数 w ，但其实 x 可以不仅仅可以是模型参数，后文会说明这一点。

梯度提升：

给定训练集 $D = (\mathbf{x}_i, y_i), i = 1, 2, \ldots, n$ ，任务是训练一个函数 F 来拟合这个数据，损失函数是平方误差：

$$J = \sum_{i=1}^{m} (y_i - F(\mathbf{x}_i))^2 \tag{2.1}$$

一开始，我们训练了一个函数 h_0 ，赋给 F ， $F = h_0$ 。每个样本产生的误差，也叫残差(residual)，是 $y_i - F(\mathbf{x}_i)$ 。

接下来，我们要优化 F ，条件是不能改变 F 已有的内容，只能往 F 上继续添加函数。那么顺其自然就想到了再训练一个函数 h_1 来填补误差，即 h_1 的目标从 y_i 变成了 $y_i - F(\mathbf{x}_i)$ 。然后把 h_1 添加到 F ， $F = F + h_1 = h_0 + h_1$ 。然后误差变为新的 $y_i - F(\mathbf{x}_i)$ 。

以此类推，每次迭代都训练一个新的函数 h_m ，目标是误差 $y_i - F(\mathbf{x}_i)$ ，然后把新的函数 h_m 添加到 F 。

停止迭代后， F 就是若干个函数的和:

$$F(\mathbf{x}) = \sum_{i=1}^{m} h_m(\mathbf{x}) \tag{2.2}$$

这就是梯度提升的直观例子，每次迭代训练一个弱学习器，目标是误差，最后将若干个弱学习器加到一起，作为最终的模型。

梯度提升和梯度下降法的关系就是：殊途同归。梯度提升的目的也是梯度下降，只不过相比于普通梯度下降通过**直接更新参数**来梯度下降，梯度提升通过**累加弱学习器**来梯度下降。

12．为什么GBDT中要拟合残差?

首先，GBDT拟合的不是残差，而是负梯度。只是当损失函数为平方损失的时候，负梯度正好为残差。

13．为什么GBDT中要拟合负梯度?

GBDT通过弱学习器的累加实现强学习器，每个弱学习器需要不断减小损失函数，即每加一个树都应该减小损失函数。用 $F_k = \sum_{i=1}^{k} f_i$ 代表k棵树累加的结果，第i棵树的输出表示为 f_i ，则损失函数的泰勒一阶展开为:

$$Loss(F_k) = Loss(F_{k-1} + f_k) \approx Loss(F_{k-1}) + f_k * \frac{\partial Loss(F)}{\partial F}|_{F=F_{k-1}}$$

这里 $Loss(F_{k-1})$ 和 $\frac{\partial Loss(F)}{\partial F}|_{F=F_{k-1}}$ 均为已知的值，因为之前k-1棵树已经生成好了，要求 $Loss(F_k) < Loss(F_{k-1})$ ，则应使 $f_k = -\frac{\partial Loss(F)}{\partial F}|_{F=F_{k-1}}$ ，这样损失函数变为:

$$Loss(F_k) = Loss(F_{k-1} + f_k) \approx Loss(F_{k-1}) - (\frac{\partial Loss(F)}{\partial F}|_{F=F_{k-1}})^2 < Loss(F_{k-1})$$

因此每棵树都应拟合损失函数对前k-1棵树的负梯度，不断减小损失函数。

14．GBDT对异常值敏感吗

GBDT对异常值比较敏感，原因是当前的错误会延续给下一棵树。

15．GBDT的“梯度提升”体现在哪个阶段

前面提到，在构建cart树时使用了损失函数的负梯度，而不是所谓的残差=真值-预测值；实际上是一种更宽广的概念，但是在平方损失的情况下，上面等式是成立的。另外使用损失函数的梯度可以保证损失函数最小值。所以GBDT的梯度提升体现在构建cart树的所需的负梯度阶段，其利用最速下降的近似方法。

16．为什么GBDT的树深度较RF通常都比较浅

对于机器学习来说，泛化误差可以理解为两部分，分别是偏差（bias）和方差（variance）；偏差指的是算法的期望预测与真实预测之间的偏差程度，反应了模型本身的拟合能力；方差度量了同等大小的训练集的变动导致学习性能的变化，刻画了数据扰动所导致的影响。当模型越复杂时，拟合的程度就越高，模型的训练偏差就越小；但此时如果换一组数据可能模型的变化就会很大，即模型的方差很大，所以模型过于复杂的时候会导致过拟合。对于RF来说由于并行训练很多不同的分类器的目的就是降低这个方差（variance）。所以对于每个基分类器来说，目标就是如何降低这个偏差（bias），所以我们会采用深度很深甚至不剪枝的决策树。而对于GBDT来说由于利用的是残差逼近的方式，即在上一轮的基础上更加拟合原数据，所以可以保证偏差（bias），所以对于每个基分类器来说，问题就在于如何选择 variance 更小的分类器，即更简单的分类器，所以我们选择了深度很浅的决策树。

17．GBDT需要对样本进行归一化吗

需要，因为GBDT的树是在上一颗树的基础上通过梯度下降求解最优解，归一化能收敛的更快，GBDT通过减少偏差来提高性能，而随机森林本来就是通过减少方差提高性能的，树之间建立关系是独立的，不需要归一化。

18．GBDT如何做正则化

- Shrinkage，即在每一轮迭代获取最终学习器的时候按照一定的**步长**进行更新。

  $ft(x)=ft-1(x)+\eta ht(x)$

  其中$0<\eta\leq 1$是步长，对于同样的训练集学习效果，较小的η意味着需要更多的迭代次数；通常用步长和迭代最大次数一起来决定算法的拟合效果。

- subsample（子采样），取值为(0,1]，采用的**不放回采样**，如果取值为1，则全部样本都使用，等于没有使用子采样；如果取值小于1，则只有一部分样本会去做GBDT的决策树拟合，选择小于1的比例可以减少方差，即防止过拟合，但是会增加样本拟合的偏差，因此取值不能太低，推荐在[0.5, 0.8]之间。
- CART回归树进行正则化**剪枝**，这里不详细介绍。

19．GBDT哪些部分可以并行

- 计算每个样本的负梯度；
- 分裂挑选最佳特征及其分割点时，对特征计算相应的误差及均值时；
- 更新每个样本的负梯度时；
- 最后预测过程中，每个样本将之前的所有树的结果累加的时候。

20．GBDT与AdaBoost的区别

GBDT与Adboost最主要的区别在于两者如何识别模型的问题。Adaboost用错分数据点来识别问题，通过调整错分数据点的权重来改进模型。GBDT通过负梯度来识别问题，通过计算负梯度来改进模型。

21．比较LR和GBDT。什么情景下GBDT不如LR

区别

- LR是线性模型，可解释性强，很容易并行化，但学习能力有限，需要大量的人工特征工

程。

- GBDT是非线性模型，具有天然的特征组合优势，特征表达能力强，但是树和树之间无法并行训练，而且树模型很容易过拟合。

当在高维稀疏特征的场景下，LR的效果一般会比GBDT好

- 现在的模型普遍都带有正则项，而LR等线性模型的正则项是对权重的惩罚，也就是W1一旦过大，惩罚就会很大，进一步压缩W1的值，使他不至于过大。但是树模型则不一样，树模型的惩罚项通常为叶子节点数和深度等。在高维稀疏特征的时候，线性模型会比非线性模型效果好的原因：带正则化的线性模型比较不容易对稀疏特征过拟合

22．RF(随机森林)与GBDT之间的区别与联系

相同点：

- 都是由多棵树组成，最终的结果都是由多棵树一起决定。
- RF和GBDT在使用CART树时，可以是分类树或者回归树。

不同点：

- 组成随机森林的树可以并行生成，而GBDT是串行生成
- 随机森林的结果是多数表决表决的，而GBDT则是多棵树累加之和
- RF对异常值不敏感，原因是多棵树表决，而GBDT对异常值比较敏感，原因是当前的错误会延续给下一棵树。
- 随机森林是减少模型的方差，而GBDT是减少模型的偏差
- 随机森林不需要进行特征归一化。而GBDT则需要进行特征归一化

- https://zhuanlan.zhihu.com/p/351781576 <https://zhuanlan.zhihu.com/p/351781576>
- https://zhuanlan.zhihu.com/p/385261343 <https://zhuanlan.zhihu.com/p/385261343>
- https://blog.csdn.net/qy724728631/article/details/82023701 <https://blog.csdn.net/qy724728631/article/details/82023701>
- https://blog.csdn.net/xwl198937/article/details/79749048 <https://blog.csdn.net/xwl198937/article/details/79749048>