链滴

从 gbdt 到 xgboost

作者：lai-bluejay

原文链接：https://ld246.com/article/1553313808275

来源网站：链滴

背景：通过对树模型的调研和了解，能够在业务中有把握的使用树模型并进行调优。

目标：掌握树模型的原理，了解GBDT的迭代过程，以及了解每轮迭代可能带来的影响；明白如何构数据和目标函数使之更适合模型和业务。在未来DEBUG模型的时候，能有正确的打开方式。

本文目标：在一定的程度上了解gbdt的来龙去脉，以及gbdt的一个实现--xgboost。

0. 监督学习

目标函数:

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

$L(\theta)$，training loss，表示模型在训练集上表现如何。让模型预测能力更强，使型更匹配底层数据分布。

$\Omega(\Theta)$， regularizationm，表示了模型的复杂度。鼓励模型更简，使得预测更稳定。

可以边阅读边思考，1.损失函数和正则化在模型训练中是这么影响的。

2. 损失函数的不同，对模型训练的影响是什么样的。

目标函数之所以定义为损失函数和正则项两部分，是为了尽可能平衡模型的偏差和方差（Bias Varianc Trade-off）。最小化目标函数意味着同时最小化损失函数和正则项，损失函数最小化表明模型能够好的拟合训练数据，一般也预示着模型能够较好地拟合真实数据（groud truth）；另一方面，对正项的优化鼓励算法学习到较简单的模型，简单模型一般在测试样本上的预测结果比较稳定、方差较小奥坎姆剃刀原则）。也就是说，优化损失函数尽量使模型走出欠拟合的状态，优化正则项尽量使模型免过拟合。

1. 《BoostedTree》

决策树

● 找到最佳分裂点（条件熵，互信息，信息增益等）并分裂。（类别型特征，找分裂的属性；连续型征，找分裂的属性值）直到达到停止条件（单纯的节点、树过大、叶子节点上数据量很少）

tree ensemble

$$
\begin{aligned}
&ALG.1\ \ Gradient\ \ Boost\ \ \backslash\backslash\backslash\backslash \\
&F_0(x) = arg \min_p \sum_{i=1}^N L(y_i, p)\ \ \backslash\backslash\backslash\backslash \\
&for\ m=1\ to\ M\ do:\backslash\backslash\backslash\backslash \\
&\ \ \ \hat{y_i} = \ -[\frac {\partial L(y_i,F(x_i))}{\partial F(x_i)}]_{F(x)}\ \ \text{计算残差, 梯度下降的方向} \\
&\\
&f_m = arg \min_p \sum_{i=1}^N L(y_i, F_{m-1}(x_i)+\eta q(x_i))\ \ \text{基学习器h，优化f，计算长} \backslash\backslash\backslash\backslash \\
&F_m(x) = F_{m-1}(x) + \eta f_m(x)\ \ \text{更新模型} \backslash\backslash\backslash\backslash \\
&\text{end For}
\end{aligned}
$$

即，一堆树结构构成的树集合

$\hat{y} = \sum_{k=1}^M f(x, w_k) \\
$

或者写成:

\(\hat{y_i}= \sum_{k=1}^M f_k(x_i), f_k \in F \)

w_k 或 f_k (x)代表树的结构

这些模型是可加的。additive model.

如何来学习加法模型呢?

解这一优化问题，可以用前向分布算法（forward stagewise algorithm）。因为学习的是加法模型如果能够从前往后，每一步只学习一个基函数及其系数（结构），逐步逼近优化**目标函数**（即损失函），那么就可以简化复杂度。这一学习过程称之为Boosting。具体地，我们从一个常量预测开始，每学习一个新的函数，过程如下：

$$
\begin{split}
\hat{y_i}^0 &= 0
\hat{y_i}
\end{split}
$$

$$
\begin{split}
\hat{y_i}^0 &= 0 \\\
\hat{y_i}^1 &= f_1(x_i) = \hat{y}_i^0 + f_1(x_i) \\\
\hat{y}_i^2 &= f_1(x_i) + f_2(x_i) = \hat{y}_i^1 + f_2(x_i) \\\
& \cdots \\\
\hat{y}_i^t &= \sum_{k=1}^t f_k(x_i) = \hat{y}_i^{t-1} + f_t(x_i) \\\
\end{split}
$$

在这个过程中，最终要的是找到f（weak learner），并使得loss最速下降，并最小化目标函数。

即

$\hat{y}^{(M)} =F_M(x) = \hat{y}^{M-1} + f_M(x)$， $f_M(x)$ 为第M轮迭代学习到的， $\hat{y}^0 = C$

目标函数:

$$L(\psi) = \sum_il(y_i, \hat{y}_i) + \sum_k(\Omega(f_k))$$

其中$\Omega(f) = \gamma T + \frac{1}{2}\lambda ||\omega||^2$

$=\gamma T+\frac{1}{2}\lambda||f(x)||^2$

这里f为给定结构q下树的结构，ω为叶子节点权重，T为叶子节点数。对于优化第t轮

$$min L^{(t)} = \sum_il(y_i, \hat{y}^{(t-1)} + f_t(x_i)) + \Omega(f(t))$$

由于可导凸函数，在t-1处进行泰勒近似二阶展开，Δ 为 $f_t(x_i)$。

为什么可以近似。推导过程参考http://projecteuclid.org/download/pdf_1/euclid.aos/1016218223（可补充）

$$L^{(t)} \approx \sum_i(l(y_i, \hat{y}^{(t-1)}) + g_if_t(x_i) + \frac{1}{2} h_i f^2_t(x_i)) + \Omega \tag 1$$

g_i 和 h_i分别为一阶导和二阶导。（**这也是和常见的GBDT有区别的地方之一**）

参见square loss的情况下，梯度下降最速的方向为一阶导。二阶导恒为常数2.

第t轮的目标函数为

$$\begin{split}
Obj^{(t)} &= \sum_{i=1}^nl(y_i, \hat{y}_i^t) + \sum_{i=i}^t \Omega(f_i) \\\\
&= \sum_{i=1}^n l\left(y_i, \hat{y}_i^{t-1} + f_t(x_i) \right) + \Omega(f_t) + constant\\\\
& \approx \sum_i(l(y_i, \hat{y}^{(t-1)}) + g_if_t(x_i) + \frac{1}{2} h_i f^2_t(x_i)) + \Omega + constant
\end{split}\tag{2}$$

以square loss为例:

$$\begin{split}
Obj^{(t)} &= \sum_{i=1}^n \left(y_i - (\hat{y}_i^{t-1} + f_t(x_i)) \right)^2 + \Omega(f_t) + constant \\\\
&= \sum_{i=1}^n \left[2(\hat{y}_i^{t-1} - y_i)f_t(x_i) + f_t(x_i)^2 \right] + \Omega(f_t) + constant
\end{split}\tag{3}$$

g_i=\partial_{\hat{y}^{t-1}}(\hat{y}^{t-1} - y_i)^2 = 2(\hat{y}^{t-1} - y_i) ， h_i=\partial_{\hat{y}^{t-1}}^2(\hat{y}^{t-1} - y_i)^2 = 2其中 (\hat{y}_i^{t-1} - y_i) 为残差。

以log loss为例:

$$\begin{split}
\text{设定sigmoid变换} p = \frac {1}{1+e^{-x_i}}\\\\
\text{设定loss function} l = \prod_{i=1}^M p_i^{y_i}(1- p_i)^{1-y_i}\\\\
\text{对loss function 进行负ln变换} l = \sum_{i=1}
^M(ln(1+e^{x_i}) - y_i * x_i)\\\\
\end{split}$$

$$\begin{split}
Obj^{(t)} &= \sum_{i=1}^n \left(y_i - (\hat{y}_i^{t-1} + f_t(x_i)) \right)^2 + \Omega(f_t) + constant \\\\
&= \sum_{i=1}^n \left[(p_i- y_i)f_t(x_i) + p_i(1-p_i)f_t(x_i)^2 \right] + \Omega(f_t) + constant
\end{split}\tag{4}$$

g_i=\partial_{\hat{y}^{t-1}}l(y_i- \hat{y}_i^{t-1})= (\frac {1}{1+e^{-x_i}}- y_i) =p_i - y_i，h_i=\partial_{\hat{y}^{t-1}}^2l(y_i- \hat{y}_i^{t-1}) = \frac {e^{-x_i}}{(1+e^{-x_i)^2}} = p_i(1-p_i)

对于树模型来说，\Omega就是树的复杂度，可以考虑树的数量，叶子节点的数量，树的深度，或者子节点对应的score的L2

该部分推导对所有可加模型都使用。例如，对于多数据源的boosting，如果不希望引入的新数据，响原有模型的训练，实际上就是把确定的树结构q(x)，换成不同的模型结构m(x)，只要保证是可加可。或者再抽象一层，即如果有新加入的数据源\模型\树\特征空间，如果不想影响原有的模型训练均可按照此方式推导模型的合理性。

shrinkage: 即 F_k(x) = F_{k-1}(x) + {v} * f_k(x)， v是学习率,

展开写(一阶泰勒展开)。这也是常见的GBDT的实现。

$$F(x_{k+1}) = F(x_k + x_{k+1} - x_k) \approx F(x_k) + \nabla F(x_k) (x_{k+1} - x_k)$$

要使第k+1轮迭代后，F下降。即 满足 $\nabla F(x_k) (x_{k+1} - x_k) < 0$

给定一个值η，使得

$$\begin{split}
\nabla F(x_k) (x_{k+1} - x_k) < -\frac{1}{\eta} \\\\
&即 \nabla F(x_k) (x_k - x_{k+1}) < \frac{1}{\eta}\\\\
& x_{k+1} = x_k - \eta \nabla F(x_k)
\end{split}\tag{5}$$

η即为熟知的学习率，使得学到的每颗决策树的影响权重降低。因此当给定学习率越小时，迭代的长变小。

梯度提升决策树 GBDT

###目标函数:

回顾一下目标函数的定义：

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

$L(\theta)$，training loss，表示模型在训练集上表现如何。让模型预测能力更强，使模型更匹配底层据分布。

$\Omega(\Theta)$， regularizationm，表示了模型的复杂度。鼓励模型更简单，使得预测更稳定。

损失函数必须是可微的凸函数。

GBM属于树的可加模型，对于函数空间F内的函数f而言，即确定了了每颗树的结构q(x)。由于每个weak learner都是决策树，对于决策树而言，要找到最佳的分裂点属于NP难问题，因此大部分情况都是发式的方式去寻找。对于gbdt，则可以把每个启发式的方法映射到目标函数上，即

- 通过信息增益来分裂 --> training loss
- 对树进行剪枝 --> regularization by nodes
- 设定最大深度 --> 每个weak learner的空间
- 叶子节点的值平滑 --> 叶子节点的显式L2正则。

###公式推导

目标函数：

$$L(\psi) = \sum_i l(y_i, \hat{y}_i) + \sum_k(\Omega(f_k))$$

其中

$$\Omega(f) = \gamma T + \frac{1}{2} \lambda ||\omega||^2$$

$$=\gamma T+\frac{1}{2}\lambda||f(x)||^2$$

这里f为给定结构q下树的结构，\omega为叶子节点权重，T为叶子节点数。对于优化第t轮

$$min\ L_t\ = \sum_i l(y_i, \hat{y}^{(t-1)} + f_t(x_i)) + \Omega(f(t))$$

由于可导凸函数，在t-1处进行泰勒近似二阶展开，\Delta 为 f_t(x_i)。

$$L\approx\sum_i[l(y_i, \hat{y}_{t-1}) + g_if_t(x_i) + \frac{1}{2}\ h_i\ f^2_t(x_i)) + \Omega$$

g_i 和 h_i分别为一阶导和二阶导。

对于第i轮迭代, l(y_i, \hat{y}^{(t-1)}) 为常数项。

定义 I_j = \{i|q(x_i)=j\}，为给定树结构q的叶子节点j上的数据集

移除常数项之后，即优化:

$$\begin{split}
\hat{L}^{(t)} = \sum_{i=1}^n[g_if_t(x_i) + \frac{1}{2}\ h_i f^2_t(x_i)) + \lambda T + \frac {1}{2} \lambda \sum_{j=1}^T \omega_j^2\\
=\sum_{j=1}^T[(\sum_{i\in I_j}g_i)\omega_j +\frac{1}{2}(\sum_{i\in I_j}h_i + \lambda) \omega_j^2\ ) + \gamma T
\end{split}$$

对于给定的结构q(x) 即\omega_j 一定, 对\omega_j 求偏导。

$$\frac {\partial}{\partial\omega_j} = \sum g_i + (\sum h_i + \lambda)\omega_j$$

令公式(7)=0。即

$$\omega_j^* = - \frac{\sum_{i \in I_j}g_i}{\sum_{i \in I_j}h_i+ \lambda} \tag 8$$

代入\hat{L}^{(t)}中，得到给定树结构q(x)的最小值

$$\hat{L}^{(t)}_{q} = - \frac{1}{2}\sum_{j=1}^T \frac {(\sum_{i \in I_j}g_i)^2}{\sum_{i\in I_j}h_i + \lambda} + \gamma T \tag 9$$

设，对于集合I，在某个叶子节点分裂后变为

I_L 和 I_R，对于该集合的loss，希望分裂后loss能降低最多

，即

$$max\ L - (L_{I_L} + L_{I_R})$$

即

$$\begin{split}
\max L_{split} = \frac{1}{2}[l_L+l_R - l) - \gamma T \\\\
= \frac{1}{2} [\frac{\sum_{I_L} g_i^2}{(\sum_{I_L}h_i)+\lambda}\ + \frac{\sum_{I_R} g_i^2}{(\sum_{I_R}h_i)+\lambda}\ - \frac{\sum_{I} g_i^2}{(\sum_{I}h_i)+\lambda}] - \gamma
\end{split}$$

\end{split} \tag {10}

前面部分为训练loss的减少，\gamma为正则项。
即每轮的信息增益为bias和Variance的平衡，在预测能力和简单程度进行平衡。

- pre-stopping
  - 在最佳分裂点收益为负时，停止分裂
  - 但继续分裂可能使得之后的分裂仍然有收益。（在xgb中，logloss的增加可能带来auc的增长）
- post-prunning
  - 生成满树之后，把分裂收益为负值的叶子给 剪枝。

即为每轮分裂需要找到分裂点，是的目标loss下降最快

寻找分裂点的方法

2.1 基本的贪心算法

遍历所有可能的分裂点，并计算。代表：sklearn，单机的XGB

2.2 近似算法

背景

- 内存不能满足计算性能要求；
- 分布式计算

算法依赖：

- 根据特征分布，依靠百分位数，确定候选集
- 将候选集的连续值，映射到不同的桶中，聚合

2.xgboost的改进

简单列了一些改进，会对其中一些展开描述。

- 二阶泰勒展开
- 叶子节点的显示L2正则
- 通过近似的方式寻找分裂点
- 计算缺失值的分裂方向
- block + cache的计算时间复杂度减少
- 引入RF的subsample
- 工程上，抽象了obj function等。

2.1 近似算法 - 加权分位数方案

对于数据集D_k = \{(x_{1k}, h_1), \dots (x_{nk}, h_n)\} 定义排位函数r_k(z), 代表特征k的值小于z的例。

目标是找到候选分裂点\{s_{k1},\dots,s_{kl}\}，使得分裂点内的排位差值小于给定的近似因子\epsilon. 即

|r_k(s_{k,j}) - r_k(s_{k, j+1}) < \epsilon, s_{k1} = \min_i x_{ik}, s_{kl} = \max_i x_{ik}

即，会找到\frac {1}{\epsilon}个候选集。每个点的权重即二阶导h_i. 将公式(2) 改写为如下形式：

\begin{split}
Obj^{(t)} &= \sum_{i=1}^nl(y_i, \hat{y}_i^t) + \sum_{i=i}^t \Omega(f_i) \\\
&= \sum_{i=1}^n l\left(y_i, \hat{y}_i^{t-1} + f_t(x_i) \right) + \Omega(f_t) + constant\\\
& \approx \sum_i(g_if_t(x_i) + \frac{1}{2} h_i f^2_t(x_i)) + \Omega + constant \\\
&= \sum_i\frac{1}{2} h_i(f_t(x_i) - \frac {g_i}{h_i})^2 + \Omega + constant
\end{split}\tag{10}

h_i 即f_t(x_i) 对label \frac{g_i}{h_i}的平方误差的权重。因此称为有权重的分位数方案。

在论文里给出了理论上的证明。简单概况，就是

“是在寻找split point的时候，不会枚举所有的特征值，而会对特征值进行聚合统计，然后形成若干bucket，只将bucket边界上的特征值作为split point的候选，从而获得性能提升。”

在实现版本中，也支持histogram的方式减少计算量。

2.2 sparsity aware split

xgboost对于缺失值的处理。

在传统找分裂点的时候，模型会找到最优的分裂点，使得信息增益最大，再进行分裂。

由于现实数据，特征向量往往是稀疏的，简单几个原因：

- 缺失值很常见
- 统计量为0的实体很多。或者说，很多实例在某些特征空间上的属性就是无。
- 人为的特征工程，如one-hot.

xgboost在寻找数据分裂点时，只统计无缺失的对象。同时在计算分裂点的信息增益时，记录缺失值裂的方向不同时，如何获得最大收益。即记录分裂点的两个属性：分裂的特征值和缺失值的方向。

2.3 其他性能优化

- data事先排好序并以block的形式存储，利于并行计算
- cache-aware, out-of-core computation，这个我不太懂。。
- 支持分布式计算可以运行在MPI，YARN上，得益于底层支持容错的分布式通信框架rabit。openMP
- 实现版本中还对drop-out进行了支持。
- 并行策略。

I). inter-feature exact parallelism （特征级精确并行） II). inter-feature approximate parallelism（特征级近似并行，基于特征分bin计算，减少了枚举所有特征分裂点的开销） III). intra-featur parallelism （特征内并行） IV). inter-node parallelism （多机并行）

参考文献：

1. GREEDY FUNCTION APPROXIMATION:

A GRADIENT BOOSTING MACHINE1

2.