天善智能
TIANSHAN SOFT

机器学习模型在信贷风控中的应用之

XGBoost模型

讲师：安迪生

课程简介

- 在分类任务中，除了逻辑回归、决策树、SVM等简单模型外，还有例如像随机森林之类的集成模型。本节课介绍集成模型中的一类梯度提升模型。GBDT是其中一个典型的模型，而XGBoost模型又是GBDT的升级版。 XGBoost模型在工业界取得了重要的成功，在很多领域展现了良好的性能。

目录

- Gradient Boosting的概念
- GBDT模型简介
- GBDT的升级版：XGBoost
- XGBoost模型在信贷风控中的应用

Gradient Boosting的概念

- 从梯度下降法说起

在求解函数$f(w)$最值的问题中，梯度下降法是基本的数值方法之一。以求最小值为例说明基本步骤：

① 初始化w: $= w^{(0)}$

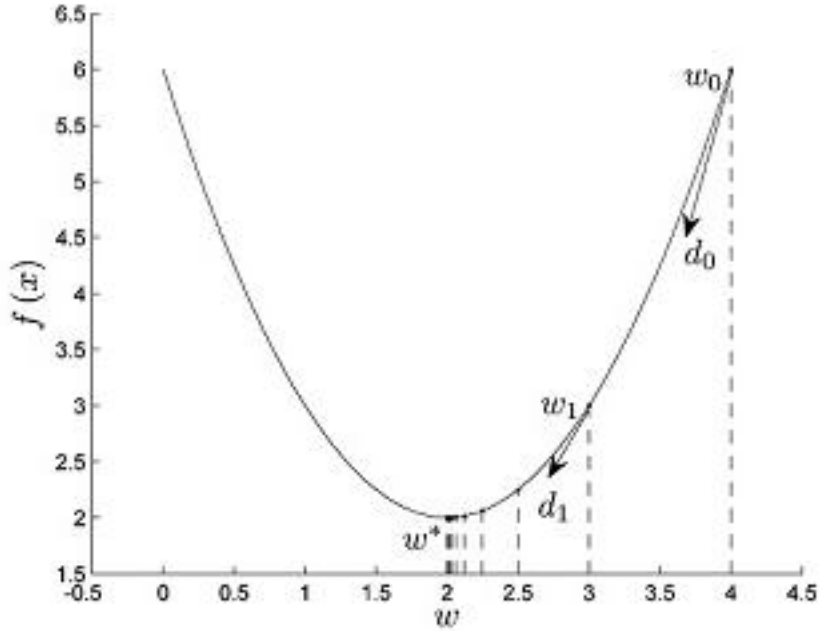
② for i in range[0:N]

求解梯度$d_i = -\frac{\partial f}{\partial w}|_{w^{(i)}}$

更新$w^{(i+1)} = w^{(i)} + \lambda_i d_i$, λ_i为当前的步长

则最终解

$$w^* = w^{(0)} + \sum_{i=1}^{N} \lambda_i d_i$$



Gradient Boosting的概念

- 在函数空间求解最优化——Gradient Boosting

如果将变量扩展到函数空间（相当于泛函），该如何求解最优化的解呢？

考虑给定数据集$\{\boldsymbol{x_i}, y_i | i = 1,2, \dots, M\}$，建立$x$对$y$的回归模型$f(x)$，需优化损失函数$L\big(y, f(x)\big)$。即求解

$$f^*(x) = arg \min_f L(y, f(\boldsymbol{x}))$$

与梯度下降法一致，基本的步骤仍然为：

① 初始化$f := f^0(\boldsymbol{x})$

② for i in range$(0, \mathrm{N})$

$$f^{i+1}(\boldsymbol{x}) = f^i(\boldsymbol{x}) + \lambda_i g_i(\boldsymbol{x}),$$

$$where \ \ g_i(\boldsymbol{x}) = -\frac{\partial L}{\partial f}|_{f=f^i(x)}, \ \lambda_i = \arg\min_\lambda L(y, f^i(\boldsymbol{x}) + \lambda g_i(\boldsymbol{x}))$$

则最后的模型为 $F^*(x) = f^0(\boldsymbol{x}) + \sum \lambda_i g_i(\boldsymbol{x})$

Gradient Boosting的概念

- Gradient Boosting Decision Tree, 梯度提升树

如果将上述介绍的模型定义成CART模型，即可得到Gradient Boosting Decision Tree（GBDT）。

（当然也可以选择其他分类器或回归模型，满足上述的boosting的框架即可）

特点

- 基于简单决策树的组合模型
- 沿着梯度下降的方向进行提升
- 只接受数值型连续变量

优点

- 准确度高
- 不易过拟合

GBDT模型简介

- GBDT模型的原理(以分类树为例)

结构

用$F(x)=\sum_k^K f_k(x)$来逼近y，y是二分类标签，K是分类树个数

损失函数(Loss Function)

- 第k步累计函数的损失 = 加上第k棵树后的精度损失(Training Loss) + 加上第k棵树后的复杂度惩罚(Penalty on Complexity)
- 待求变量：第k棵树
- 目的：让第k步累计函数的损失最小(梯度法结合泰勒展式)
- 结束：将第k棵树加到之前的模型中

GBDT模型简介

- GBDT模型的原理(以分类树为例)

损失函数：负二项对数函数

$$l(y,F)=\log(1+\exp(-2yF)),y=\pm1$$

$$where\ F=\frac{1}{2}\log(\frac{P(y=1|x)}{P(y=-1|x)})$$

（此处不考虑模型复杂度）

关于相加性

- 预测的类别是不能相加的
- 概率也是不能相加的（否则会超过1）
- 但是log odds ratio 是可以相加的

GBDT模型简介

- 分类问题中的梯度提升

先定义初始化模型$F^{(0)}(x)$

计算损失函数在$F^{(0)}(x)$处的负梯度（因为要最小化损失函数，故而计算负梯度）：

$$\tilde{y} := -\frac{\partial l}{\partial F}\Big|_{F=F^{(0)}(x)} = \frac{2y}{1+\exp(2yF^{(0)}(x))}$$

构建**回归树**$f^1(x)=\tilde{y}$, 并计算回归树的每个叶子节点的取值：

$$\gamma_{1,j} = \arg\min_{\gamma} l\left(y, F^{(0)}(x)+\gamma\right) = \arg\min_{\gamma} \sum_{x\in R_{1,j}} \log\left(1+\exp\left(-2y\left(F^{(0)}(x)+\gamma\right)\right)\right)$$

$\gamma_{1,j}$的近似解

$$\gamma_{1,j} = \frac{\sum_{x\in R_{1,j}} \tilde{y}}{\sum_{x\in R_{1,j}} |\tilde{y}|(2-|\tilde{y}|)}$$

GBDT模型简介

- 分类问题中的梯度提升(续)

对F的更新为：

$$F^{(1)}(x) = F^{(0)}(x) + \sum_{j=1}^{J} \eta * \gamma_{1,j} I(x \in R_{1,j})$$

此处$R_{1,j}$表示第一个对梯度拟合的回归树的叶子节点。 η是步长，也称为收缩因子

迭代下去可以得到$F^{(M)}(x)$

注意

- 模型$F^{(m)}(x)$不是分类树，即$F^{(m)}(x)$的结果不是类别，而是概率的转换：

$$P(R_{m,j} = 1) = \frac{1}{1 + \exp(-F^{(m)}(x))}, x \in R_{m,j}$$

- 模型$f^{(m)}(x)$也不是分类树，而是损失函数对$F^{(m-1)}(x)$的梯度
- 训练梯度树$f^{(m)}(x)$是顺序训练的，即$f^{(m)}(x)$的训练依赖于$f^{(m-1)}(x)$, $f^{(m-2)}(x)$,....等等
- 选择不同的损失函数，得到的梯度会略有不同

GBDT模型简介

- 一个简单的例子

x	1	2	3	4	5	6	7	8	9	10
y	-1	-1	-1	1	1	-1	-1	-1	1	1

在负二项对数函数下，求解梯度提升树（树深为1），步长为0.2

初始化：$F^0(x) = \frac{1}{2}\log\left(\frac{P(y=1|x)}{P(y=-1|x)}\right) = \frac{1}{2}\log\left(\frac{0.4}{0.6}\right) = -0.2027$

x	1	2	3	4	5	6	7	8	9	10
y	-1	-1	-1	1	1	-1	-1	-1	1	1
$F^0(x)$	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027

GBDT模型简介

- 一个简单的例子(续)

拟合第1棵树：

$$\tilde{y} = \frac{2y}{1 + \exp(2yF^{(0)}(x))}$$

x	1	2	3	4	5	6	7	8	9	10
y	-1	-1	-1	1	1	-1	-1	-1	1	1
$F^0(x)$	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027
$\tilde{y}$	-1.2	-1.2	-1.2	0.8	0.8	-1.2	-1.2	-1.2	0.8	0.8

以$\tilde{y}$为目标拟合一棵单层CART树

GBDT模型简介

- 一个简单的例子(续)

在叶子节点$x <= 8.5$上，返回的值是-0.31252,在叶子节点$x > 8.5$上，返回的值是1.25

根据$F^{(1)}(x) = F^{(0)}(x) + \sum_{j=1}^{J} \eta * \gamma_{1,j} I(x \in R_{1,j})$，有

x	1	2	3	4	5	6	7	8	9	10
y	-1	-1	-1	1	1	-1	-1	-1	1	1
$F^0(x)$	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027
$\tilde{y}$	-1.2	-1.2	-1.2	0.8	0.8	-1.2	-1.2	-1.2	0.8	0.8
γ	-0.3125	-0.3125	-0.3125	-0.3125	-0.3125	-0.3125	-0.3125	-0.3125	1.2500	1.2500
$F^{(1)}(x)$	-0.2652	-0.2652	-0.2652	-0.2652	-0.2652	-0.2652	-0.2652	-0.2652	0.0472	0.0472

GBDT模型简介

- 一个简单的例子(续)

得到$F^{(1)}(x)$后，计算损失函数的梯度： $\tilde{y} = \frac{2y}{1+\exp(2yF^{(1)}(x))}$

x	1	2	3	4	5	6	7	8	9	10
y	-1	-1	-1	1	1	-1	-1	-1	1	1
$F^0(x)$	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027
$F^{(1)}(x)$	-0.2652	-0.2652	-0.2652	-0.2652	-0.2652	-0.2652	-0.2652	-0.2652	0.0472	0.0472
$\tilde{y}$	-0.7408	-0.7408	-0.7408	-0.7408	-0.7408	-0.7408	-0.7408	-0.7408	0.9527	0.9527

拟合x对$\tilde{y}$的回归树

GBDT模型简介

- 一个简单的例子(续)

此时叶子节点的估计值为：

在叶子节点$x \leq 3.5$上，返回的值是-0.7941,在叶子节点$x > 3.5$上，返回的值是0.3305.

再计算$F^{(2)}(x)$。

x	1	2	3	4	5	6	7	8	9	10
y	-1	-1	-1	1	1	-1	-1	-1	1	1
$F^0(x)$	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027
$F^{(1)}(x)$	-0.2652	-0.2652	-0.2652	-0.2652	-0.2652	-0.2652	-0.2652	-0.2652	0.0472	0.0472
$\tilde{y}$	-0.7408	-0.7408	-0.7408	-0.7408	-0.7408	-0.7408	-0.7408	-0.7408	0.9527	0.9527
γ	-0.7941	-0.7941	-0.7941	0.3305	0.3305	0.3305	0.3305	0.3305	0.3305	0.3305
$F^{(2)}(x)$	-0.4240	-0.4240	-0.4240	-0.1991	-0.1991	-0.1991	-0.1991	-0.1991	0.1133	0.1133

GBDT模型简介

- 一个简单的例子(续)

如果只生成2棵树，则迭代终止。此时得到$F^{(2)}(x)$并非最终的分类器，而是需要以此计算概率：

$$P(y=1|x)=\frac{\exp(2F^{(2)}(x))}{1+\exp(2F^{(2)}(x))}, P(y=-1|x)=\frac{1}{1+\exp(2F^{(2)}(x))}$$

x	1	2	3	4	5	6	7	8	9	10
y	-1	-1	-1	1	1	-1	-1	-1	1	1
$F^0(x)$	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027	-0.2027
$F^{(1)}(x)$	-0.3486	-0.3486	-0.3486	-0.3486	-0.3486	-0.3486	-0.3486	-0.3486	-0.0361	-0.0361
$F^{(2)}(x)$	-0.4240	-0.4240	-0.4240	-0.1991	-0.1991	-0.1991	-0.1991	-0.1991	0.1133	0.1133
$P(y=1\|x)$	0.2998	0.2998	0.2998	0.4017	0.4017	0.4017	0.4017	0.4017	0.5564	0.5564

GBDT的升级版：XGBoost

- XGBoost模型

GBDT模型是将损失函数进行线性逼近，本质是对损失函数做1阶泰勒展开：

$$Loss\left(y, F^k(x) + f(x)\right) = Loss\left(y, F^k(x)\right) + \frac{\partial Loss}{\partial F}\Big|_{F^k(x)} \times f(x) + o(f(x))$$

如果用多项式代替线性，即将泰勒展式展开到高阶项（例如2阶），就得到精度更高的下降法。XGBoost（**Ext**reme **G**radient **Boost**ing）模型即采用2阶泰勒展式，同时考虑了模型的复杂度：

$$Loss\left(y, F^k(x) + f(x)\right) = Loss\left(y, F^k(x)\right) + gf(x) + \frac{1}{2}hf^2(x) + o\left(f^2(x)\right) + \Omega(f)$$
$$\cong Loss\left(y, F^k(x)\right) + gf(x) + \frac{1}{2}hf^2(x) + \Omega(f)$$

其中，

$$g = \frac{\partial Loss}{\partial F}\Big|_{F^k(x)}, h = \frac{\partial^2 Loss}{\partial F^2}\Big|_{F^k(x)}$$

由于$Loss\left(y, F^k(x)\right)$是常数，所以对Loss的最小化等价于对$\tilde{l} = gf(x) + \frac{1}{2}hf(x) + \Omega(f)$的最小化

GBDT的升级版：XGBoost

- XGBoost模型(续)

$gf(x)+\frac{1}{2}hf^2(x)$代表模型在预测精度上的“伪残差”，$\Omega(f)$代表模型的复杂度。有多种方式衡量这一复杂度，在XGBoost中复杂度的表示为：

$$\Omega(f)=\gamma T+\frac{\lambda}{2}\sum w_j^2$$

其中，T表示CART中叶子节点的个数，w_j表示叶子节点的取值。将叶子节点的取值也放入复杂度中可以有效避免过拟合。

综上，第t棵树$f_t(x)$加入后总的损失函数（忽略常数项后）可以近似为：

$$\tilde{l}^{(t)}=gf(x)+\frac{1}{2}hf(x)++\Omega(f)=\sum_{i=1}^{n}[g_i f_t(x_i)+\frac{1}{2}h_i f_t^2(x_i)]+\gamma T+\frac{\lambda}{2}\sum_{j=1}^{T}w_j^2$$

$$=\sum_{j=1}^{T}[\left(\sum_{x_i\in I_j}g_i\right)w_j+\frac{1}{2}(\left(\sum_{x_i\in I_j}h_i+\lambda\right)w_j^2)]+\gamma T$$

GBDT的升级版：XGBoost

- XGBoost模型(续)

为了让$\tilde{l}^{(t)}$最小化，w的取值是：

$$w^* = -\frac{\sum_{x_i \in I_j} g_i}{\sum_{x_i \in I_j} h_i + \lambda}$$

此时$\tilde{l}^{(t)}$取最小值（预测精度部分）

$$\tilde{l}^{(t)} = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{x_i \in I_j} g_i\right)^2}{\sum_{x_i \in I_j} h_i + \lambda} + \gamma T$$

GBDT的升级版：XGBoost

- XGBoost模型：树的结构

如何确定树的结构

假设树的结构已经确定，则$\tilde{l}^{(t)}$可视为树的得分。我们的任务就是要找出得分最小的树。理想情况下可以枚举出所有可能的结构从中挑选得分最小的结构。但是当变量或者样本较多时，枚举法是几乎不可能实现的。因此采用近似算法去寻找较优的树：

① 贪婪法：生长一棵树时，寻找最优的特征作为当前节点，以及最优的分裂点形成子树

② 近似法：在贪婪法中用特征的分位点代替每种可能的取值。降低了精准度但是极大提高了寻找的速度

GBDT的升级版：XGBoost

- XGBoost模型：树的结构（续）

贪婪法

注意到一棵树在生长的时候，预测精度在上升（即预测损失函数下降）的同时，复杂度会提高。因此在分裂节点时需要评估复杂度的提高对整个损失函数带来的抵消有多大。假设某节点对应的样本集合为I，分裂后形成的左右分支的样本集合分别为I_L和L_R。则分裂后损失函数的变化为：

$$L_{split} = \frac{1}{2}\left[\frac{\left(\sum_{x_i \in I_L} g_i\right)^2}{\sum_{x_i \in I_L} h_i + \lambda} + \frac{\left(\sum_{x_i \in I_R} g_i\right)^2}{\sum_{x_i \in I_j} h_R + \lambda} - \frac{\left(\sum_{x_i \in I} g_i\right)^2}{\sum_{x_i \in I} h_i + \lambda}\right] - \gamma$$

其中$-\gamma$表示分裂前该分支有1个叶子节点，选择其中一个分裂后共有2个叶子节点γ是叶子节点的惩罚系数。L_{split}越大表明分裂后的效果越好。

GBDT的升级版：XGBoost

- XGBoost模型：树的结构（续）

贪婪法

实践中，我们贪婪的增加树的叶子结点数目:

① 从深度为0的树开始

② 对于树的每个叶子节点，尝试增加一个分裂点：

$$L_{split} = \frac{1}{2}\left[\frac{\left(\sum_{x_i \in I_L} g_i\right)^2}{\sum_{x_i \in I_L} h_i + \lambda} + \frac{\left(\sum_{x_i \in I_R} g_i\right)^2}{\sum_{x_i \in I_j} h_R + \lambda} - \frac{\left(\sum_{x_i \in I} g_i\right)^2}{\sum_{x_i \in I} h_i + \lambda}\right] - \gamma$$

对于每次扩展，我们还是要**穷举所有可能的分割方案**

一．对每个特征，通过特征值将实例进行排序

二．运用线性扫描来寻找该特征的最优分裂点

三．对所有特征，采用最佳分裂点

GBDT的升级版：XGBoost

• XGBoost模型：树的结构（续）

近似法

当数据太多不能装载到内存时，不能进行精确搜索分裂，只能近似。 根据特征分布的百分位数，提出特征的一些候选分裂点。将连续特征值映射到桶里（候选点对应的分裂），然后根据桶里样本的统计量，从这些候选中选择最佳分裂点。根据候选提出的时间，分为:

① **全局近似**：

在构造树的初始阶段提出所有的候选分裂点，然后对下面的各个层次采用相同的候选特征分裂点。

特点：提出候选的次数少，但每次的候选数目多（因为候选不更新）

② **局部近似**：

在每次分裂都重新提出候选特征分裂点。

特点：对层次较深的树更适合

GBDT的升级版：XGBoost

- XGBoost模型：其他的优化

该模型在计算的时候，提出了一些加速计算的优化方法，包括：

- 考虑了训练数据为稀疏值的情况，可以为缺失值或者指定的值指定分支的默认方向，这能大大提升算法的效率。
- 特征列排序后以块的形式存储在内存中，在迭代中可以重复使用；虽然boosting算法迭代必须串行，但是在处理每个特征列时可以做到**并行**。
- 按照特征列方式存储能优化寻找最佳的分割点，但是当以行计算梯度数据时会导致内存的不连续访问，降低算法效率。可先将数据收集到线程内部的缓冲区，然后再计算，提高算法的效率。
- 还考虑了当数据量比较大，内存不够时怎么有效的使用磁盘，主要是结合多线程、数据压缩、分片的方法，尽可能的提高算法的效率。

GBDT的升级版：XGBoost

- 特征重要性评估

XGBoost能够给出变量重要性的估计：变量的全局重要性通过其在单棵决策树的重要性的平均值来衡量的：

$$\widehat{J_k}^2 = \frac{1}{M}\sum_{m=1}^{M}\widehat{J_k}^2(T_m)$$

其中，M是树的个数，变量在单棵树中的重要度如下：

$$\widehat{J_k}^2(T) = \sum_{t=1}^{L-1}\widehat{i_t}^2 1(v_t = k)$$

L是叶子节点数量，L-1则是非叶节点数量（CART是二叉树），v_t是和节点t有关的特征，$\widehat{i_t}^2$是t分裂后的不纯度的减少值

GBDT的升级版：XGBoost

- 综述

XGBoost模型的主要步骤为：

1. 初始化：$F^{(0)}(x)$

2. $for\ i\ in\ 0{:}N$

    ① 计算损失函数对于当前$F^{(i)}(x)$的一阶和二阶导数g_i，h_{i}

    ② 通过贪婪或者近似法生成CART树$f^{(i)}(x)$

    ③ 更新：$F^{(i+1)}(x) = F^{(i)}(x) + \eta f^{(i)}(x)$

$end\ for$

- 和GBDT模型的区别

➢ 最显著的区别是，最小化损失函数时采用二阶多项式进行逼近，比GBDT的精度要高

➢ 其他一些工程化方面的优化，例如近似搜索、线程缓冲区存储

XGBoost模型在信贷风控中的应用

- 案例

在构造XGBoost模型用于违约预测之前，依然需要对数据做预处理工作和特征工程，可按照上一节课中介绍神经网络模型的方法进行。

需要注意的是：

① 极端值的处理不是必须的，这是因为构建每棵树模型时，CART模型对于极端值是不敏感的。这一步可以忽略。

② XGBoost模型本身可以处理缺失值，即对缺失值单独分枝。但是python包中相应的的模块不能读取缺失数据，因而要对缺失值做预处理。

XGBoost模型在信贷风控中的应用

- 案例（续）

max_depth：树的最大深度，用来避免过拟合

min_child_weight：最小样本权重的和，用于避免过拟合，

gamma：在某节点分裂时，只有分裂后损失函数的值的下降幅度超过gamma ，才会分裂这个节点。

subsample：控制对于每棵树，随机采样的比例

reg_alpha：权重的L1正则化项

n_estimators：迭代次数，或者生成的梯度树的个数

XGBoost模型在信贷风控中的应用

- 案例（续）

完成参数调优后，我们查看特征重要性，发现存在特征，其重要性几乎为0。可以将这部分特征过滤掉，重新构建模型。发现新构模型的AUC几乎没有变化。而入模的特征数从400减少为158.

更多商业智能BI和大数据精品视频尽在 www.hellobi.com

BI、商业智能
数据挖掘 大数据
数据分析师
R语言 Python
机器学习
深度学习
人工智能
Hive Hadoop
Tableau
BIEE ETL
数据科学家
PowerBI

天善智能
TIANSHAN SOFT