GBDT ⟹ Gradient Boost Decision Tree. (梯度提升树)

⇓ ⇓

Gradient Descent + Boosting

Boost : many weak predicted model ensemble into one ⟹ 弱学习机

Strong model. ⇓ Single Decision Tree. Model

⇓

Boost Tree

Scale: Boosting ⟹ (Specify) Gradient Boosting ⟹ (Specify) GBDT

Difference between GBDT and Random Forest: 集成模型

① Drop single decision tree.

In Random Forest, each decision tree is trained independently and it doesn't affect the performance of the model if we randomly / selectively drop one tree.

In GBDT, a sequence of decision tree each improve the decision of all future trees. And if we drop one tree, the performance of the model will be affected (the first tree affect more than the last tree) △ Adaboost 算法 ⟹ 最具代表性的提升树算法

② Combine two models:

RandomForest can be constructed in parallel while GBDT model can only be constructed in sequence manner.

③ When the number of decision tree increase.

For RandomForest, it doesn't affect a lot. Because the trees don't have relationship among them.

For GBDT, it does matter. The model will be more powerful if the #tree ↑.

提升的思想：F_{m+1} 通过提升值纠正 F_m 的误差变得更完美.

GBDT思想：提升树由多棵基础决策树构成，每一棵决策树均依赖于前一棵决策树分类结果而设置不同的权重

（预测样本错误 ⇒ 增大样本权重；正确 ⇒ 降低样本权重.）

每次构造新的提升树均是对原本的决策树的优化。

Adaboost Algorithm:

$$F_m(x) = \sum_{m=1}^{n} \alpha_m \cdot f_m(x)$$

$$F_m(x) = F_{m-1}(x) + \underbrace{h(x)}_{\text{估计量}} = F_{m-1}(x) + \underbrace{\nu}_{\text{学率，默认为1}} \cdot \underbrace{\alpha_m}_{\text{权重}} \cdot \underbrace{f_m(x)}_{\text{第 m 棵基础决策树}}$$

△ Why Gradient Boost ⇔ Gradient Descent（以平方损失函数为例）

估计值/提升值 $h_{m+1}(x)$ $= y - F_m(x)$ $\left(\underbrace{\frac{1}{2}\cdot(y-F_m(x))^2}_{MSE}\right)'$

即损失函数的导数，因为梯度下降为 $\theta - \alpha_{m+1}\frac{\partial J(\theta)}{\partial \theta}$

而梯度提升为 $F_m(x) + \alpha_{m+1}\cdot h_{m+1}(x)$，因为提升其实是 MSE 的负梯度.

因此GBDT模型的估计值/提升值为损失函数的负梯度值，即间接地求得了损失函数梯度值。（当我们损失函数为一般函数，梯度值不好求时，原本回归等机器学习模型不再适用，因此引入GBDT.）

确定的量：
- 样本点权重 $w_{m,i}$
- 基础决策树 $f(x)$ 选择

How: \ 决策树权重 α_m.

Cost function:

$$L(y, F(x)) = e^{-yF(x)} \xrightarrow{n} \text{提升层数}$$

$$= e^{\left(-y \sum_{m=1} \alpha_m \cdot f_m(x)\right)}$$

$$= e^{-y\left(F_{m-1}(x) + \alpha_m \cdot f_m(x)\right)}$$

确定 $\alpha_m \cdot f_m(x)$:

$$\left(\alpha_m, f_m(x)\right) = \underset{\alpha \cdot f(x)}{\operatorname{argmin}} \sum_{i=1}^{n} e^{\left[-y_i\left(F_{m-1}(x_i) + \alpha_m \cdot f_m(x^i)\right)\right]} \Longrightarrow \text{所有样本数之}$$

$y_i F_{m-1}(x_i)$: 已知, 与 $\alpha_m \cdot f_m(x)$ 无关

$$= \underset{\alpha \cdot f(x)}{\operatorname{argmin}} \sum_{i=1}^{n} e^{-y_i \cdot \alpha_m \cdot f_m(x^i)} \cdot W_{mi}$$

$$= \begin{cases} \underset{\alpha \cdot f(x)}{\operatorname{argmin}} \sum_{i=1}^{n} e^{-\alpha_m} \cdot W_{mi}, & \text{第 } m \text{ 棵树预测准确} \Rightarrow y_i \cdot f_m(x_i) = 1 \\ \underset{\alpha \cdot f(x)}{\operatorname{argmin}} \sum_{i=1}^{n} e^{\alpha_m} \cdot W_{mi}, & \text{第 } m \text{ 棵树预测错误} \Rightarrow y_i \cdot f_m(x_i) = -1 \end{cases}$$

此时仅与 α_m 有关, 因此得先求最佳 $f_m(x)$ 使 cost function

最小.

$$f_m^*(x) = \underbrace{\arg\min}_{f} \sum_{i=1}^{N} w_{mi} \cdot I(y_i \neq f_m(x))$$

f: 所有决策树空间 f 中

选则第 m 轮基础决策树，其使加权训练样本分类错误率最小.

$I(y_i \neq f_m(x))$: $=1$ 当 $y_i \neq f_m(x)$; $=0$ 当 $y_i = f_m(x)$

求出 $f_m^*(x)$ 后，上式：

$$L(y, F(x)) = \sum_{y_i = f_m(x_i)} w_{mi} \cdot e^{-d_m} + \sum_{y_i \neq f_m(x_i)} w_{mi} \cdot e^{d_m}$$

第 m 棵树预测准确 $y_i \cdot f_m(x_i) = 1$

第 m 棵树预测错误 $\Rightarrow y_i \cdot f_m(x_i) = -1$

$$= \sum_{i=1}^{n} w_{mi} \cdot e^{(-d_m)} + \left(e^{d_m} - e^{-d_m}\right) \sum_{y_i \neq f_m(x_i)}^{n} w_{mi}$$

$$\Downarrow$$

$$\sum_{i=1}^{n} w_{mi} \cdot I(y_i \neq f_m(x_i))$$

$$= \sum_{i=1}^{n} w_{mi} \cdot e^{-\alpha_m} + \left(e^{\alpha_m} - e^{-\alpha_m}\right) \cdot \sum_{i=1}^{n} w_{mi} \cdot I(y_i \neq f_m(x_i))$$

对 $L(y, F(x))$ 求偏导：

$$\Rightarrow \frac{\partial L(y, F(x))}{\partial \alpha_m} = 0 \Rightarrow \alpha_m^* = \frac{1}{2}\left(\log\frac{1-e_m}{e_m}\right)$$

e_m：错误率

$$e_m = \sum_{i=1}^{n} w_{mi} \cdot I(y_i \neq f_m(x))$$

因此，我们可得：

$$\begin{cases} f_m^*(x) = \arg\min \sum_{i=1}^{n} \cdot w_{mi} \cdot I(y_i \neq f_m(x_i)) \\ e_m = \sum_{i=1}^{n} \cdot w_{mi} \cdot I(y_i \neq f_m(x_i)) \\ \alpha_m^* = \frac{1}{2} \cdot \log\left(\frac{1-e_m}{e_m}\right) \end{cases}$$

$\Rightarrow$ 根据 $f_m(x)$ 的预测结果，得到下轮样本点

权重：

$$w_{m+1,i}^{*} = \begin{cases} \dfrac{w_{m1} \cdot e^{(-\alpha m^{*})}}{\sum_{i=1}^{n} w_{m1} \cdot e^{(-\alpha m^{*})}}, & f_m(x_i)^{*} = y_i \\ \dfrac{w_{m1} \cdot e^{\alpha m^{*}}}{\sum_{i=1}^{n} w_{m1} \cdot e^{\alpha m^{*}}}, & f_m(x_i)^{*} \neq y_i \end{cases}$$

GBDT Algorithm:

$$F_M(x) = F_{M-1}(x) + h_m(x)$$

$$= F_{M-1}(x) + \alpha m \cdot f_m(x)$$

$$= F_{M-1}(x) + \sum_{j=1}^{J} \cdot C_{mj} \cdot I(x_i \in R_{mj})$$

M层提升树 ← M

J → 样本个数

$$= \sum_{m=1}^{M} \sum_{j=1}^{J} \cdot C_{mj} \cdot I(x_i \in R_{mj})$$

$f_m(x)$ 在样本结点 j 上预测值

上采样

SMOTE Algorithm ⇒ 合成少数过采样技术

!! 过采样 欠采样 组合采样

(Synthetic Minoritye Over-sampling Technique)

Steps:

I. Choose minoritye sample a.
Use KNN to get k nearest neighbour.

II. According to sampling rate N,
choose $B = N \cdot \frac{\#a}{\text{num of a}}$ from k sample

III. For each b in B, create new sample
$c = a + |a-b| * rand(0,1)$, for b in B.

Do until we balance the data.

In python:

随机数种子

SMOTE(ratio='auto', random_state=None, k_neighbours=5.

all ⇒过采样

近邻个数，即k

not minority ⇒ 大众样本
majority ⇒ 对多数的样本采样

m_neighbours = 10)

即 B，近邻中 再选取个数.

GBDT

Improve ⇒ XGBoost

Advantage:

XGBoost是由传统的GBDT模型发展而来的，GBDT模型在求解最优化问题时应用了一阶导技术，而XGBoost则使用损失函数的一阶和二阶导，而且可以自定义损失函数，只要损失函数可一阶和二阶求导。

XGBoost算法相比于GBDT算法还有其他优点，例如支持并行计算，大大提高算法的运行效率；XGBoost在损失函数中加入了正则项，用来控制模型的复杂度，进而可以防止模型的过拟合；XGBoost除了支持CART基础模型，还支持线性基础模型；XGBoost采用了随机森林的思想，对字段进行抽样，既可以防止过拟合，也可以降低模型的计算量。