集成学习之Boosting —— AdaBoost原理

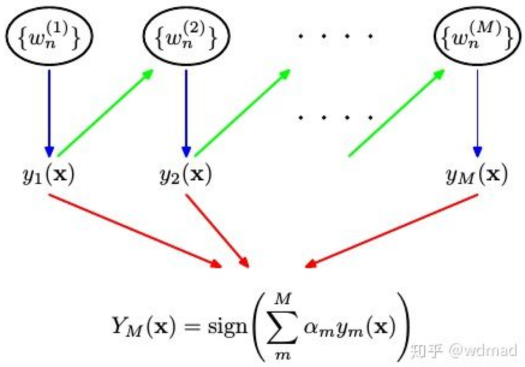
wdmad

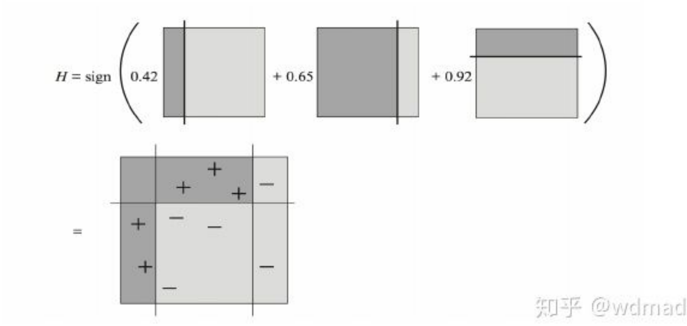
集成学习大致可分为两大类：Bagging和Boosting。Bagging一般使用强学习器，其个体学习器之间不存在强依赖关系，容易并行。Boosting则使用弱分类器，其个体学习器之间存在强依赖关系，是一种序列化方法。Bagging主要关注降低方差，而Boosting主要关注降低偏差。Boosting是一族算法，其主要目标为将弱学习器“提升”为强学习器，大部分Boosting算法都是根据前一个学习器的训练效果对样本分布进行调整，再根据新的样本分布训练下一个学习器，如此迭代M次，最后将一系列弱学习器组合成一个强学习器。而这些Boosting算法的不同点则主要体现在每轮样本分布的调整方式上。本系列文章先讨论Boosting的两大经典算法 —— AdaBoost和Gradient Boosting，再探讨近年来在各大数据科学比赛中大放异彩的XGBoost和LightGBM。

AdaBoost原理概述

AdaBoost是一个具有里程碑意义的算法，因为其是第一个具有适应性的算法，即能适应弱学习器各自的训练误差率，这也是其名称的由来（Ada为Adaptive的简写）。

AdaBoost的具体流程为先对每个样本赋予相同的初始权重，每一轮学习器训练过后都会根据其表现对每个样本的权重进行调整，增加分错样本的权重，这样先前做错的样本在后续就能得到更多关注，按这样的过程重复训练出M个学习器，最后进行加权组合，如下图所示。





这里有两个关键问题：

1. 每轮训练过后如何调整样本权重 w ？
2. 如何确定最后各学习器的权重 α ？

这两个问题可由加法模型和指数损失函数推导出来。

加法模型 (Additive Model) 和指数损失函数 (Exponential Loss)

由上图可以看出，AdaBoost最后得到的强学习器是由一系列的弱学习器的线性组合，此即加法模型：

$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

其中 $G_m(x)$ 为基学习器， α_m 为系数。

在第m步，我们的目标是最小化一个指定的损失函数 $L(y, f(x))$ ，即：

$$\min_{(\alpha_m\, ,\, G_m)} \sum_{i=1}^{N} L(y_i, \sum_{m=1}^{M} \alpha_m G_m(x_i)) \qquad (1.1)$$

其中 $(x_1, y_1), (x_2, y_2), \cdots (x_N, y_N)$ 为训练数据集。

这是个复杂的全局优化问题，通常我们使用其简化版，即假设在第m次迭代中，前m-1次的系数 α 和基学习器 $G(x)$ 都是固定的，则 $f_m(x) = f_{m-1}(x) + \alpha_m G_m(x)$ ，这样在第m步我们只需就当前的 α_m 和 $G_m(x)$ 最小化损失函数。

AdaBoost采用的损失函数为指数损失，形式如下：

$$L(y,\, f(x)) = e^{-yf(x)} \qquad (1.2)$$

结合上文，我们现在的目标是在指数函数最小的情况下求得 α_m 和 $G_m(x)$ 。

$$(\alpha_m, G_m(x)) = \mathop{\arg\min}_{(\alpha, G)} \sum_{i=1}^{N} e^{-y_i f_m(x_i)} = \mathop{\arg\min}_{(\alpha, G)} \sum_{i=1}^{N} e^{-y_i (f_{m-1}(x_i) + \alpha G(x_i))} \qquad (1.3)$$

设 $w_i^{(m)} = e^{-y_i f_{m-1}(x_i)}$ ，由于 $w_i^{(m)}$ 不依赖于 α 和 $G(x)$ ，所以可认为其是第m步训练之前赋予每个样本的权重。然而 $w_i^{(m)}$ 依赖于 $f_{m-1}(x_i)$ ，所以每一轮迭代会改变。

于是式 (1.3) 变为：

$$\sum_{i=1}^{N} w_i^{(m)} e^{-y_i \alpha G(x_i)} = e^{-\alpha} \sum_{y_i = G(x_i)} w_i^{(m)} + e^{\alpha} \sum_{y_i \neq G(x_i)} w_i^{(m)} \qquad (1.4)$$

$$(e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}(y_i \neq G(x_i)) + e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \qquad (1.5)$$

由上面几个式子可以得到AdaBoost算法的几个关键点：

(1) 基学习器 $G_m(x)$ ：

求令式 (1.5) 最小的 $G_m(x)$ 等价于令 $\sum_{i=1}^{N} w_i^{(m)}\mathbb{I}(y_i \neq G(x_i))$ 最小化的 $G_m(x)$ ，因此可认为每一轮基学习器都是通过最小化带权重误差得到。

(2) 下一轮样本权值 $w_i^{(m+1)}$ ：

由
$w_i^{(m+1)} = e^{-y_i f_m(x_i)} = e^{-y_i(f_{m-1}(x_i)+\alpha G(x_i))} = e^{-y_i f_{m-1}(x_i)} e^{-y_i \alpha_m G_m(x_i)} = w_i^{(m)} e^{-y_i \alpha_m G_m(x_i)}$

可以看到对于 $\alpha_m > 0$ ，若 $y_i = G_m(x_i)$ ，则 $w_i^{(m+1)} = w_i^{(m)} e^{-\alpha_m}$ ，表明前一轮被正确分类样本的权值会减小；若 $y_i \neq G_m(x_i)$ ，则 $w_i^{(m+1)} = w_i^{(m)} e^{\alpha_m}$ ，表明前一轮误分类样本的权值会增大。

(3) 各基学习器的系数 α_m ：

设 $G_m(x)$ 在训练集上的带权误差率为 $\epsilon_m = \frac{\sum\limits_{i=1}^{N} w_i^{(m)}\mathbb{I}(y_i \neq G_m(x_i))}{\sum\limits_{i=1}^{N} w_i^{(m)}}$ ，

式 (1.4) 对 α 求导并使导数为0： $-e^{-\alpha} \sum\limits_{y_i = G(x_i)} w_i^{(m)} + e^{\alpha} \sum\limits_{y_i \neq G(x_i)} w_i^{(m)} = 0$ ，

两边同乘以 e^{α} ，得 $e^{2\alpha} = \frac{\sum\limits_{y_i = G(x_i)} w_i^{(m)}}{\sum\limits_{y_i \neq G(x_i)} w_i^{(m)}} = \frac{1-\epsilon_m}{\epsilon_m}$

$\Longrightarrow \quad \alpha_m = \frac{1}{2} ln \frac{1-\epsilon_m}{\epsilon_m}$

可以看出， ϵ_m 越小，最后得到的 α_m 就越大，表明在最后的线性组合中，准确率越高的基学习器会被赋予较大的系数。

AdaBoost算法

在了解了 $G_m(x)$ ， $w_i^{(m+1)}$ 和 α_m 的由来后，AdaBoost算法的整体流程也就呼之欲出了：

> 输入：训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \cdots (x_N, y_N)\}$ ， $y \in \{-1, +1\}$ ，基学习器 $G_m(x)$ ，训练轮数M
>
> 1. 初始化权值分布： $w_i^{(1)} = \frac{1}{N}, \quad i = 1, 2, 3, \cdots N$
>
> 2. for m=1 to M:
> (a) 使用带有权值分布的训练集学习得到基学习器$G_m(x)$:
> $$G_m(x) = \mathop{\arg\min}_{G(x)} \sum_{i=1}^{N} w_i^{(m)}\mathbb{I}(y_i \neq G(x_i))$$
> (b) 计算 $G_m(x)$ 在训练集上的误差率：
> $$\epsilon_m = \frac{\sum\limits_{i=1}^{N} w_i^{(m)}\mathbb{I}(y_i \neq G_m(x_i))}{\sum\limits_{i=1}^{N} w_i^{(m)}}$$
> (c) 计算 $G_m(x)$ 的系数： $\alpha_m = \frac{1}{2} ln \frac{1-\epsilon_m}{\epsilon_m}$
>
> (d) 更新样本权重分布： $w_i^{(m+1)} = \frac{w_i^{(m)} e^{-y_i \alpha_m G_m(x_i)}}{\cdots}$ ， $i = 1, 2, 3 \cdots N$

$w_i^{(m+1)}$ 构成一个分布。

3. 输出最终模型： $G(x) = sign\left[\sum m = 1^M \alpha_m G_m(x)\right]$

AdaBoost采用指数损失的原因

若将指数损失表示为期望值的形式：

$$E(e^{-yf(x)}|x) = P(y=1|x)e^{-f(x)} + P(y=-1|x)e^{f(x)}$$

由于是最小化指数损失，则将上式求导并令其为0：

$$\frac{\partial E(e^{-yf(x)}|x)}{\partial f(x)} = -P(y=1|x)e^{-f(x)} + P(y=-1|x)e^{f(x)} = 0 \quad ,$$

$$f(x) = \frac{1}{2}log\frac{P(y=1|x)}{P(y=-1|x)}，\quad 或写为 P(y=1|x) = \frac{1}{1+e^{-2f(x)}}$$

仔细看，这不就是logistic regression吗？二者只差系数 $\frac{1}{2}$ ，因此每一轮最小化指数损失其实就是在训练一个logistic regression模型，以逼近对数几率 (log odds)。

于是，

$$\begin{aligned} sign(f(x)) &= sign(\frac{1}{2}log\frac{P(y=1|x)}{P(y=-1|x)}) \\ &= \begin{cases} 1 & P(y=1|x) > p(y=-1|x) \\ -1 & P(y=1|x) < P(y=-1|x) \end{cases} \\ &= \arg\max_{y\in\{-1,1\}} P(y|x) \end{aligned}$$

这意味着 $sign(f(x))$ 达到了贝叶斯最优错误率，即对于每个样本 x 都选择后验概率最大的类别。若指数损失最小化，则分类错误率也将最小化。这说明指数损失函数是分类任务原本0-1损失函数的一致性替代函数。由于这个替代函数是单调连续可微函数，因此用它代替0-1损失函数作为优化目标。

Real AdaBoost

上文推导出的AdaBoost算法被称为"**Discrete AdaBoost**"，因为其各个基学习器 $G_m(x)$ 的输出为{-1,1}。如果将基学习器的输出改为一个类别概率，则产生了Real AdaBoost。Real AdaBoost通常在更少的轮数达到更高的精度，像scikit-learn中的AdaBoostClassifier就是默认优先使用Real AdaBoost (SAMME.R)，不过Real AdaBoost中的基学习器必须支持概率估计。

推导与Discrete AdaBoost类似，仍最小化指数损失：

$$\begin{aligned} E(e^{-yf_m(x)}|x) &= E(e^{-y(f_{m-1}(x)+G(x))}|x) \\ &= E(w \cdot e^{-yG(x)}|x) \\ &= e^{-G(x)}P_w(y=1|x) + e^{G(x)}P_w(y=-1|x) \end{aligned}$$

对 $G(x)$ 求导： $G(x) = \frac{1}{2}log\frac{P_w(y=1|x)}{\dots}$

> 1. 初始化权重分布：$w_i^{(1)} = \frac1N, \qquad i = 1,2 \cdots N$
>
> 2. for m=1 to M:
>
> (a) 使用带权重分布的训练集训练基学习器，得到类别概率
> $P_m(x) = P_w(y=1|x) \;\in [0,1]$
> (b) $G_m(x) = \frac{1}{2} log \frac{P_m(x)}{1-P_m(x)} \;\in R$
> (c) 更新权重分布：$w_i^{(m+1)} = \frac{w_i^{(m)} e^{-y_i G_m(x_i)}}{Z^{(m)}}$
>
> 3. 输出最终模型：$G(x) = sign\left[\sum\limits_{m=1}^M G_m(x)\right]$

正则化 (Regularization) 和其他

1、Shrinkage

对每个基学习器乘以一个系数 ν (0 < ν <1)，使其对最终模型的贡献减小，从而防止学的太快产生过拟合。 ν 又称学习率，即scikit-learn中的 **learning rate**。于是上文的加法模型就从：

$$f_m(x) = f_{m-1}(x) + \alpha_m G_m(x)$$

变为：

$$f_m(x) = f_{m-1}(x) + \nu \cdot \alpha_m G_m(x)$$

一般 ν 要和迭代次数M结合起来使用，较小的 ν 意味着需要较大的M。ESL中提到的策略是先将 ν 设得很小 (ν < 0.1)，再通过early stopping选择M，不过现实中也常用cross-validation进行选择。

2、Early stopping

将数据集划分为训练集和测试集，在训练过程中不断检查在测试集上的表现，如果测试集上的准确率下降到一定阈值之下，则停止训练，选用当前的迭代次数M，这同样是防止过拟合的手段。

3、Weight Trimming

weight trimming不是正则化的方法，其主要目的是提高训练速度。在AdaBoost的每一轮基学习器训练过程中，只有小部分样本的权重较大，因而能产生较大的影响，而其他大部分权重小的样本则对训练影响甚微。Weight trimming的思想是每一轮迭代中删除那些低权重的样本，只用高权重样本进行训练。具体是设定一个阈值 (比如90%或99%)，再将所有样本按权重排序，计算权重的累积和，累积和大于阈值的权重 (样本) 被舍弃，不会用于训练。注意每一轮训练完成后所有样本的权重依然会被重新计算，这意味着之前被舍弃的样本在之后的迭代中如果权重增加，可能会重新用于训练。

根据 [Friedman et al., 2000] 中的描述，weighting trimming可以提高计算效率，且不会牺牲准确率，下一篇中将通过实现来验证。

终末

AdaBoost的原始论文

并非使用了上文中的推导方法，而是基于PAC学习框架下进行解释的。上文的将AdaBoost视为 "加法模型 + 指数损失" 的观点，由斯坦福的几位统计系大牛 [Friedman et al., 2000] 提出，因而这一派也被称为"统计视角"。沿着这个路子，若将指数损失替换为其他损失函数并依此不断训练新学习器，则诞生了Gradient Boosting算法，是为下一篇的主题。

Reference :

1. Friedman, J., Hastie, T. and Tibshirani, R.

Additive logistic regression: a statistical view of boosting
statweb.stanford.edu

2. Friedman, J., Hastie, T. and Tibshirani, R. *The Elements of Staistical Learning*

3. 周志华. 《机器学习》

4. 李航. 《统计学习方法》

5. Schapire R. E.

Expalining AdaBoost
rob.schapire.net

/

编辑于 2018-06-07

集成学习 boosting adaboost

The anwser 19 天前

写得真不错，全篇感觉没有一句废话，简洁高效数学公式齐全

赞

wdmad (作者) 回复 The anwser 18 天前

你看的其他文章都废话很多么。。。。

赞 查看对话

The anwser 回复 wdmad (作者) 18 天前

你这篇文章一看就明了话真不多还有代码实现，不是说也介绍xgboost和lightgbm，怎么不写了

赞 查看对话