机器学习常见知识汇总之集成学习

田晓彬
xiaobin9652@163.com

集成学习

1.常用的集成学习方法以及有何异同

集成学习的目标是希望通过将多个学习器的结合，得到比单一学习器显著优越的泛化性能。常用的集成学习方法分为两种：Boosting和Bagging方法。

1. Boosting方法训练基分类器时采用串行的方法，将基分类器层层叠加。每一层在训练的时候，对样本进行重新赋予权重，前面一层分类错误的样本给予更高的权重。最后根据各层基分类器的结果加权得到最终结果。Boosting算法通过聚焦基分类器分错的样本，主要降低集成分类器的偏差，因此Boosting算法能基于泛化能力相当弱的学习器构建出很强的集成。著名的Boosting算法是AdaBoost算法。
2. 与Bagging的串行训练方式不同，Bagging使用并行训练，各个分类器之间无相互依赖。给定一个包含 m 个样本的数据集，首先有放回的随机抽取 m 个样本。重复此过程 T 次，可以得到 T 个包含 m 个样本的训练集，然后基于每个训练集训练出一个基学习器，最后再将这些基学习器进行结合。Bagging通常对分类任务使用简单投票法，对回归任务使用简单平均法。
   bagging的采样方法使得数据集剩下部分样本，这样使得基学习器可以使用剩下的样本作为验证集。Bagging算法通过对数据集的重采样，训练出不同的分类器做综合来降低集成分类器的方差，因此它在不剪枝决策树、神经网络等易受样本扰动的学习器上效果尤为明显。常见的Bagging算法有随机森林。

2.基分类器的基本步骤

集成学习一般可以分为三个步骤：

1. 找到相互独立的基分类器；
2. 训练基分类器；
3. 合并基分类器的结果。

在基分类器的选取中，最常用的为决策树，主要有以下的原因：

1. 决策树可以较为方便地将样本权重整合到训练过程中，而不需要使用过采样的方法来调整样本的权重；
2. 决策树的表达能力和泛化能力可以通过控制树的层数来进行折中；
3. 数据样本的扰动对于决策树的影响较大，因此不同子样本集生成的决策树的随机性较大，更适合作为基分类器。此外还可以在决策树节点分裂的时候，随机选取一个特征子集进行分裂来引入随机性。

除了决策树，神经网络模型也适合作为基分类器。主要因为神经网络模型并不稳定，而且可以通过调整神经元数量，连接方式、网络层数、初始权重等方式来引入随机性。

3.偏差与方差

在集成学习中，Bagging能提升弱分类器性能的原因是降低了方差，Boosting能提升弱分类器性能的原因是降低了偏差。

从模型的角度来说，对n个独立不相关的模型的预测结果取平均，方差是原来单个模型的$\frac{1}{n}$。当然模型之间不可能完全独立，所以Bagging使用的不同方法来增加模型的独立性。随机森林算法在每次选取节点分裂属性的时候，会随机抽取一个属性子集，这就可以避免弱分类器之间过强的相关性。通过训练数据的重采样也能够带来弱分类器之间一定的独立性，从而降低Bagging后模型的方差。

Boosting方法在训练好一个弱分类器后，计算出当前弱分类器的错误或者残差，作为下一个弱分类器的输入。这个过程实际上在不断地减小损失函数，使得模型偏差不断降低。而因为Bagging的训练过程决定了各个弱分类器之间必定是强相关的，所以并不会降低方差。

Adaboost算法

Boosting算法中最著名的代表是AdaBoost算法。AdaBoost（自适应增强）算法是一种提升方法，将多个弱分类器当作基分类器，组合成一个强分类器。假设训练样本有N个，有T个基分类器，则基分类器的训练过程如下所示：

1. 初始化采样分布，即训练样本的权重分布$D_1(i) = \frac{1}{N}$；
2. 从训练集中，按照数据的权重分布$D(t)$，采样出子集$S_t = \{x_i, y_i | i = 1, ...N_t\}$；
3. 使用S_t训练出基分类器h_t；
4. 计算h_t的错误率：$\epsilon_t = \frac{\sum_{i=1}^{N_t} I[h_t(x_i) \neq y_i] D_t(x_i)}{N_t}$，其中 I 是判别函数；
5. 计算基分类器h_t的权重$\alpha_t = \log \frac{(1-\epsilon_t)}{\epsilon_t}$;
6. 更新训练样本的权重分布，

$$D_{t+1} = \begin{cases} D_t(i)\text{或者}\frac{D_t(i)(1-\epsilon_t)}{\epsilon_t} & , h_t(x_i) \neq y_i \\ \frac{D_t(x_i)\epsilon_t}{(1-\epsilon_t)} & , h_t(x_i) = y_i \end{cases}$$

,并将其进行归一化；

7. 重复2-6步骤T次，训练得到每个基分类器和其权重；
8. 合并基分类器，$y = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$。

从上面的步骤中，我们可以明显地看到AdaBoost的思想，对正确分类的样本降低了权重，对于错误分类的样本提高权重或者保持权重不变。并且在最后的合并阶段，根据错误率对基分类器进行加权融合，使错误率低的基分类器得到更高的权重。

GBDT

Gradient Boosting是Boosting算法中非常流行的模型，使用CART作为弱分类器的Gradient Boosting模型叫做GBDT。和Adaboost一样，GBDT也是一种加性模型，可以表示为$y = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$。但是Adaboost模型每次训练完一个基分类器后，会将分类错误样本的权重增大，这样在训练下一个基分类器时会侧重错误样本，进而达到模型性能的提升。而GBDT是根据当前基分类器损失函数的负梯度信息来训练新加入的弱分类器，从而得到下一个基分类器及其权重。

在每一次迭代中，GBDT算法都需要计算得到当前基分类器的损失函数$L_t = \sum_{i=1}^{N} l(y_i, F_{t-1}(x_i) + f_t(x_i))$,并求得$L_t$的负梯度。其中$F_{t-1}(x_i)$为现有的前$t-1$个基分类器的最优解，$f_t(x_i)$为第$t$个基分类器的解。当使用平方损失函数时，损失函数的负梯度和残差的方向相同，而残差的方向即为全局最优的方向。当损失函数不为平方损失函数时，损失函数的负方向为局部最优方向。

GBDT在预测阶段计算速度快，树与树之间可以并行化计算；在分布稠密的数据及上，GBDT具有较好的泛化能力和表达能力；同时因为GBDT使用决策树作为弱分类器，所以具有较好的解释能力。但GBDT也存在着训练过程串行使得训练速度较慢、在高维稀疏数据集和文本分类特征上表现并不好的缺点。

XGBoost

原始的GBDT算法基于经验损失函数的负梯度来构建新的决策树，在决策树构建完成后再进行剪枝。而XGBoost则在决策树构建的时候就加入了正则项，即XGBoost的损失函数定义为：

$$L_t = \sum_{i=1}^{N} l(y_i, F_{t-1}(x_i) + f_t(x_i)) + \Omega(f_t)$$

其中正则化项定义为

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

其中T为当前树中叶子节点的个数，w_j表示第j个叶子节点的预测值。对该损失函数在F_{t-1}处进行二姐泰勒展开可以得到

$$L_t \approx \hat{L}_t = \sum_{j=1}^{T} [G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2] + \gamma T$$

其中 T 为决策树f_t中叶子节点的个数，$G_j = \sum_{i \in I_j} \Delta_{F_{t-1}} l(y_i, F_{t-1}(x_i))$, $H_j = \sum_{j \in I_j} \Delta^2_{F_{t-1}} l(y_i, F_{t-1}(x_i))$, I_j表示所有属于叶子节点j的样本的索引集合。

假设决策树的结构已知，通过令损失函数相对于w_j的导数为0可以求出在最小化损失函数时各个叶子节点上的预测值

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

通过将预测值带入进损失函数可以得到损失函数的最小值为

$$\hat{L}_t^* = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

在对当前决策树进行分裂时，可以得到分裂前后损失函数的差值，XGBoost采用最大化这个差值来进行决策树的构建。通过遍历所有的特征取值，来寻找使得损失函数前后相差最大时对应的分裂方式。

GBDT和XGBoost的区别可以总结成一下几个方面：

1. GBDT是一种机器学习算法，而XGBoost是该算法的工程实现；
2. 在使用CART作为基分类器时，XGBoost在损失函数中加入了正则化项来控制模型的复杂度，有利于防止过拟合，进而提升模型的泛化能力；
3. GBDT在模型训练时只使用了代价函数的一阶导数信息，XGBoot对代价函数进行了二阶泰勒展开，可以同时使用一阶或者二阶导数；
4. GBDT通常使用CART作为基分类器，XGBoost支持多种类型的基分类器，如线性分类器；
5. GBDT通常在每轮迭代中使用全部的数据，XGBoost采用了与随机森林相似的策略，支持对数据进行采样；
6. GBDT通常不能对缺失值进行处理，XGBoost能够自动学习出缺失值的处理策略。

随机森林

随机森林（RF）是一种Bagging算法的扩展变体。RF是一种以决策树为基学习器构建Bagging集成的算法，并且在每一个决策树的训练过程中引入了随机属性选择。传统的决策树在分裂时选择当前节点属性集中最优的划分属性，但是在RF算法中，对于决策树的每一个节点，先从该节点的属性集合中随机选择包含k个属性的属性子集，然后再从这个子集中选择一个最优属性用于划分。参数k控制了随机性的引入程度，k值越小随机性越高。一般情况下，将设置为$k = \log_2 d$。

RF算法不仅使用样本扰动来带来基学习器的随机性，并且通过添加属性的随机性来使基学习器之间的差异性进一步增加，这就使得最终集成的泛化性能进一步提升。