第十二章 集成学习

1.集成学习的种类

（1）Boosting

训练基分类器采用串行的方式，各个基分类器之间有依赖。它的基本思路是将基分类器层层叠加，每一层在训练的时候，对前一层基分类器分错的样本，给予更高的权重。测试时，根据各层分类器的结果的加权得到结果。

（2）Bagging

在训练过程中，各基分类器之间无强依赖，可以进行并行训练，著名的是随机森林。为了基分类器之间互相独立，将训练集分为若干子集。测试时，通过投票的方式做出最后的集体决策。

从消除基分类器的偏差和方差的角度来理解Boosting和Bagging的差异：

基分类器的错误是偏差和方差两种错误之和。偏差表现在训练误差不收敛，方差表现在过拟合。

Boosting方法是通过逐步聚焦于基分类器错误的样本，减小集成分类器的偏差。Bagging方法则是采取分而治之的策略，训练处多个不同模型，然后做综合，来减小集成分类器的方差。

2.集成学习的步骤和例子

集成学习一般可以分为三个步骤：

（1）找到误差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。

合并基分类器的方法有voting和stacking两种。前者才有投票方式，少数服从福偶数。后者采用串行的方式，把前一个基分类器的结果输出到下一个分类器，将所有基分类器的输出结果相加（或用更复杂的算法融合，比如把各基分类器的输出作为特征，使用逻辑回归作为融合模型进行最后的结果预测）作为最终的输出。

3.基分类器

常用的基分类器是决策树，主要原因有以下三点：

（1）决策树可较为方面地将样本的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权重。

（2）决策树的表达能力和泛化能力，可以通过调节树的层数来做折中。

（3）数据样本的扰动对于决策树的影响较大，因此不同子集样本集合生成的决策树基分类器随机性较大，这样的“不稳定学习器“更适合作为基分类器。此外，在决策树节点分裂的时候，随机地选择一个特征子集，从中找出最优分裂属性，很好地引入了随机性。
除了决策树外，神经网络也适合作为基分类器，主要由于神经网络模型也比较“不稳定”，而且还可以通过调整神经元数量、连接方式、网络层数、初始权值等方式引入随机性。

在随机森林的基分类器中，是否可以将决策树替换为线性分类器或K-近邻?
随机森林属于Bagging类的集成学习，主要的好处是集成后的分类器的方差比基分类器的方差小。Bagging所采用的基分类器最好是本身对样本的分布较为敏感的。线性分类器或者KNN都是较为稳定的分类器，本身方差就不大。因此他们不适合用在随机森林或Bagging集成学习中。神志可能因为Bagging的采样方法，而导致他们在训练中更难收敛，从而增大了集成分类器的偏差。

4.偏差与方差

偏差：由采样得到的大小为m的训练数据集训练出的所有模型的输出的平均值和真实模型输出之间的偏差。偏差通常是由于我们的学习算法做了错误的假设所导致的，比如真实模型是某个二次函数，但我们假设模型是一次函数。
方差：由所有采样得到的大小为m的训练集训练出的所有模型的输出的方差。方差通常是由于模型的复杂度相对于训练样本m过高导致的。由方差带来的误差通常体现在测试误差相对于训练误差的增量上。

如何从减小方差和偏差的角度解释Boosting和Bagging的原理?
Bagging能够提高弱分类器性能的原因是降低了方差，Boosting能够提高弱分类器性能的原因是降低了偏差。
Bagging：
Bagging是Boostrap Aggregating的简称，就是再抽样，然后在每个样本上训练出来的模型取平均。相对于随机变量之间的相关性为p的情况下，在随机变量独立的情况下，方差减小到原来的1/n。再从模型的角度理解这个问题，对n个独立不想管的模型的预测结果取平均，方差是原来单个模型的1/n。模型之间不可能完全独立。为了追求模型的独立性，诸多Bagging的方法做了不同的改进。不如在随机森林中，每次选取节点分裂属性时，会随机抽取一个属性子集，而不是从所有属性中选取最优属性，这就是为了避免弱分类器之间过强的相关性。

Boosting：

Boosting在训练好一个弱分类器后，我们需要计算弱分类器的错误或者残差，作为下一个分类器的输入。这个过程本身就是在不断减小损失函数，来使模型不断逼近“靶心”，使得模型偏差不断降低。但Boosting的过程不会显著降低方差。这是因为Boosting的训练过程使得各弱分类器之间是强相关的，缺乏独立性，所以并不会对降低方差有作用。

5.梯度提升决策树的基本原理

GBDT的基本原理：

GBDT的基本思想是根据当前模型损失函数的负梯度信息来训练模型新加入的弱分类器，然后将训练好的弱分类器以累加的形式结合到现有模型中。基本流程，在每一轮迭代中，首先计算出当前模型在所有样本上的负梯度，然后以该值为目标训练一个新的弱分类器进行拟合并计算出该弱分类器的权重，最终实现对该模型的更新。

梯度提升与梯度下降的区别和联系是什么？

GBDT使用梯度提升算法作为训练方法，而在逻辑回归或者神经网络的训练过程中往往采用梯度下降作为训练方法，二者之间有什么联系和区别呢?

下表是梯度提升算法和梯度下降算法的对比情况。可以发现，两者都是在每一轮迭代中，利用损失函数相对于模型的负梯度方向的信息来对当前模型进行更新，只不过在梯度下降中，模型是以参数化形式表示，从而模型的更新等价于参数的更新。而在梯度提升中，模型并不需要进行参数化表示，而是直接定义在函数空间中，从而大大扩展了可以使用的模型种类。

算法名称	模型定义空间	优化规则	损失函数
梯度下降	参数空间	$\theta_t = \theta_{t-1} + \Delta\theta_t$	$\mathrm{L} = \sum_t l(y_t, f(\theta_t))$
梯度提升	函数空间	$f_t(x) = f_{t-1}(x) + \Delta f_t(x)$	$L = \sum_t l(y_t, f(x_t))$

https://zhuanlan.zhihu.com/p/86354141

https://www.cnblogs.com/hitwhhw09/p/4715030.html

所谓的梯度“上升”和“下降”，一方面指的是你要计算的结果是函数的极大值还是极小值。计算极小值，就用梯度下降，计算极大值，就是梯度上升；另一方面，运用上升法的时候参数是不断增加的，下降法是参数是不断减小的。但是，在这个过程中，“梯度”本身都是下降的。

GBDT的优点和局限性：

优点：

（1）预测阶段的计算速度块，树与树之间可并行化计算。

（2）在分布稠密的数据集上，泛化能力和表达能力都很好。

（3）采用决策树作为弱分类器使得GBDT模型具有较好的解释性和鲁棒性，能够自动发现特征间的高阶关系，并且也**不需要对数据进行特殊的预处理如归一化**等。

局限性：

（1）GBDT在高维稀疏的数据集上，表现不如SVM或者神经网络。

（2）GBDT在处理文本分了特征问题上，相对其他模型的优势不如它在处理数值特征时明显。

（3）训练过程需要串行训练，只能在决策树内部采用一些局部并行的手段提高训练速度。

6.XGBoost与GBDT的联系与区别

（1）GBDT是机器学习算法，XGBoost是该算法的工程实现。

（2）原始的GBDT算法基于经验损失函数的负梯度来构造新的决策树，只是在决策树构建完成后再进行剪枝。而XGBoost在决策树构建阶段就加入了正则化项。

（3）在使用CART树作为基分类器时，XGBoost显式地加入了正则化项来控制模型的复杂度，有利于防止过拟合。

（4）GBDT在训练模型时只使用了代价函数的一阶导数信息，XGBoost对代价函数进行二阶泰勒展开式，可以同时使用一阶导数和二阶导数。

（5）传统的GBDT采用CART作为基分类器，XGB支持多种类型的基分类器，比如线性分类器。

（6）传统的GBDT在每轮迭代时使用全部的数据，XGB则采用了与随机森林相似的策略，支持对数据进行采样。

（7）传统的GBDT没有设计对缺失值进行处理，XGB能够自动学习出缺失值的处理策略。