集成学习

1.集成学习

集成学习的思路大体上可以这样理解：在对新的数据实例进行分类的时候，通过训练好多个分类器，把这些分类器的的分类结果进行某种组合（比如投票）决定分类结果，以取得更好的结果，就是我们生活中那句话"三个臭皮匠顶个诸葛亮"，通过使用多个决策者共同决策一个实例的分类从而提高分类器的泛化能力。

集成学习有两点要求:

a.分类器之间要有差异

b.个分类器的准确度要>0.5

因此,集成学习要解决的关键问题有两个:

a.如何构建有差异性的分类器

b.如何将多个分类器进行整合

下面分别介绍.

1.1构建差异性分类器

(1) 通过处理数据集生成差异性分类器

这种方法实际就是在原有数据集上采用抽样技术获得多个训练数据集，从而生成多个差异性分类器。流行的方法有Bagging和Boosting.

Bagging：通过对原数据集进行有放回的采用构建出大小和原数据集D一样的新数据集D1，D2，D3.....，然后用这些新的数据集训练多个分类器H1，H2，H3.....因为是有放回的,采用所以一些样本可能会出现多次，而其他样本会被忽略，理论上新的样本会包括67%的原训练数据。Bagging通过降低基分类器方差改善了泛化能力，因此Bagging的性能依赖于基分类器的稳定性，如果基分类器是不稳定的，Bagging有助于减低训练数据的随机扰动导致的误差，但是如果基分类器是稳定的，即对数据变化不敏感，那么Bagging方法就得不到性能的提升，甚至会减低，因为新数据集只有63%。

Boostig：提升方法是一个迭代的过程，通过改变样本分布，使得分类器聚集在那些很难分的样本上，对那些容易错分的数据加强学习，增加错分数据的权重，这样错分的数据再下一轮的迭代就有更大的作用（对错分数据进行惩罚）。数据的权重有两个作用，一方面我们可以使用这些权值作为抽样分布，进行对数据的抽样，另一方面分类器可以使用权值学习有利于高权重样本的分类器。

(2) 通过处理数据特征构建差异性分类器

对训练数据抽取不同的输入特征子集分别进行训练，从而构建具有差异性的分类器。一般采用随机子空间，少量余留法（抽取最重要的一些特征），遗传算法等。

(3) 对分类器的处理构建差异性分类器

指的就是通过改变一个算法的参数来生成有差异性的同质分类器，比如改变神经网络的网络拓扑结构就可以构建出不同的分类器。

1.2.对基分类器结果进行整合

学习器结合带来的好处体现在三个方面:

a.由于学习任务的假设空间很大,可能有多个假设在训练集上达到同等性能,此时若使用单个学习器 可能因为误选而导致泛化性能不佳,结合多个学习器可以降低这种风险.

b.从计算角度来看,学习算法可能陷入jubu最优,有的局部最优解对用的学习器的泛化性能可能很差,而通过结合可以降低陷入糟糕局部最优解得风险.

c.从表示的方面来看,某些学习任务的真是假设可能不在当前学习算法的假设空间内,此时若使用单学习器则肯定无

效,通过结合多个学习器,由于假设空间有所扩大,有可能学得更好的近似.

1、对于回归预测（数值预测）

（1）简单平均（Simple Average），就是取各个分类器结果的平均值。

（2）加权平均（Weight Average），加权平均。

$$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}),$$

2、对于分类（类别预测）

（1）简单投票（Major vote）：就是每个分类器的权重大小一样，少数服从多数，类别得票数超过一半的作为分类结果

（2）加权投票（Weight Vote）：每个分类器权重不一。

（3）概率投票（Soft vote）：有的分类器的输出是有概率信息的，因此可用概率投票。

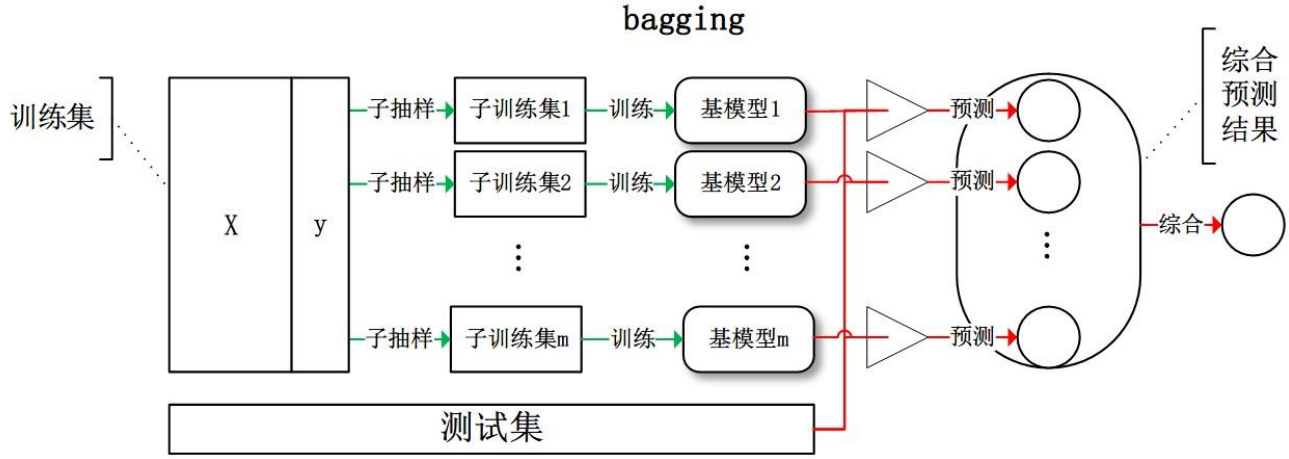
2.Bagging和Boosting

Bagging的代表是随机森林,Boosting的代表是AdaBoost,GBDT.

2.1Bagging和随机森林(RF)

Bagging (Bootstrap Aggregation)采用自主采样的方式对原数据集进行有放回的采用构建出大小和原数据集D一样的新数据集,在各数据集上分别训练分类器.

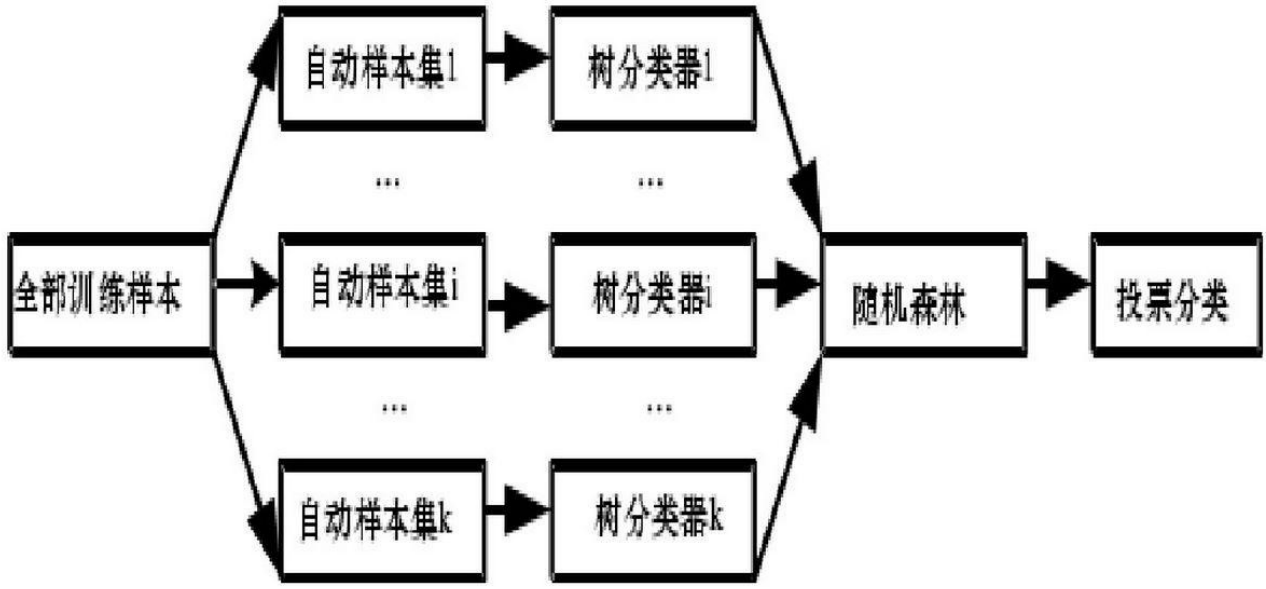
Boostrap简单来说就是有放回的重采样.



可以看出,Bagging产生的多个学习器是可并行化的,互相之间没有影响,可同时生成.

RF:

构造流程



RF中Random包含两个方面:

(1)训练样本选择方面的Random：Bootstrap方法随机选择子样本

(2)特征选择方面的Random：在传统的决策树构建过程中,每一步选用的特征都是通过信息增益等选取的全局最优特征,但RF中是在属性集中随机选择k个属性的子集，每个树节点分裂时，从这个子集中选择最优的.K控制了随机性的引入程度,若k=总的属性数量d,则与传统决策树的构建相同,若k=1,则是随机选用一个属性进行划分,随机性达到最大,一般选择k=log2(d)

1.用Random(训练样本用Bootstrap方法，选择分离叶子节点用上面的2)的方式构造一棵决策树(CART)

2.用1的方法构造很多决策树,每棵决策树都最大可能地进行生长而不进行剪枝，许多决策树构成一片森林，决策树之间没有联系

3.测试数据进入每一棵决策树，每棵树做出自己的判断，然后进行投票选出最终所属类别(默认每棵树权重一致)

RF只对Bagging做了微小的改动,但是与Bagging中基学习器的多样性仅通过样本扰动而来不同,RF中基学习器的多样性还来自属性扰动,这就使得最终集成的泛化性能可通过个体学习器之间差异度的增加而进一步提升.

RF的收敛性与Bagging相似,特别是集成中只包含一个基学习器时,这是因为引入属性扰动,个体学习器的性能往往有所降低.但随着基学习器数目的增加,RF通常会收敛到更低的泛化误差.

RF的训练效率要优于Bagging,因为个体决策树的构建过程中,Bagging在选择划分属性时要对节点的所有属性进行考察,而RF只需考察属性子集.

2.2Boosting,Adaboost和GBDT,XGBoost

Boosting

Boosting 不进行随机采样，它的策略是通过当前的基学习器的预测结果，改变样本的分布，使得上一个学习器预测错误的样本的权重 更高，这样在下一个学习器中就会被重点关照。

AdaBoot

参见统计学习方法P138

1.首先，初始化训练数据的权值分布。每一个训练样本最开始时都被赋予相同的权值：1/N。

$$D_1=(w_{11},w_{12}\cdots w_{1i}\cdots,w_{1N}),\ w_{1i}=\frac{1}{N},\ i=1,2,\cdots,N$$

2.迭代，训练m个基本分类器，不断加到最终的模型上

a. 使用具有权值分布Dm的训练数据集学习，得到基本分类器

$$G_m(x):\ \chi\to\{-1,+1\}$$

b. 计算Gm(x)在训练数据集上的分类误差率

$$e_m=P(G_m(x_i)\neq y_i)=\sum_{i=1}^{N}w_{mi}I(G_m(x_i)\neq y_i)$$

由上述式子可知，Gm(x)在训练数据集上的误差率em就是被Gm(x)误分类样本的权值之和

c. 计算Gm(x)的系数，am表示Gm(x)在最终分类器中的重要程度（目的：得到基本分类器在最终分类器中所占的权重。

$$\alpha_m=\frac{1}{2}\log\frac{1-e_m}{e_m}$$

由上述式子可知，em <= 1/2时，am >= 0，且am随着em的减小而增大，意味着分类误差率越小的基本分类器在最终分类器中的作用越大。

d. 更新训练数据集的权值分布（目的：得到样本的新的权值分布），用于下一轮迭代

$$D_{m+1}=(w_{m+1,1},w_{m+1,2}\cdots w_{m+1,i}\cdots,w_{m+1,N})$$

$$w_{m+1,i}=\frac{w_{mi}}{Z_m}\exp(-\alpha_m y_i G_m(x_i)),\ i=1,2,\cdots,N$$

其中，Zm是规范化因子，使得Dm+1成为一个概率分布

$$Z_m = \sum_{i=1}^{N} w_{mi} \exp(-\alpha_m y_i G_m(x_i))$$

3. 组合各个弱分类器

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

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

AdaBoost也可以通过加法模型通过前向算法推导得到，此时损失函数是指数损失

GBDT

https://www.zybuluo.com/yxd/note/611571

讲得很好

提升树是指以加法模型与前向分步算法，以决策树为基学习器的模型。

当提升树解决二分类问题时，其实就是AdaBoost的一个特例

当提升树使用CART树通过平方损失函数求解，此时，下一个学习器是在拟合上一个之前模型的残差。

但是，当提升树使用其他损失函数时，往往每一步优化并不简单，由此引入了梯度提升树。

GBDT:

GBDT无论分类问题还是回归问题均使用CART做基学习器。

在回归问题中，如果采用平方差损失，下一个学习器学习的就是残差，其他损失函数也可以用，不过学习的是损失函数的负梯度。

对分类问题来说，有两种方案：

a. 采用指数损失函数，此时退化为AdaBoost

b. 借鉴逻辑回归的思想，采用对数损失函数将CART计算的值压缩到0-1之间。

拟合负梯度的思想可以看下面的图，或者上边贴的博客。

XGBoost