第四章 集成模型

一、集成框架

降低模型对未来数据预测出现错误的概率
两种计算的方法：

- 传统方法：构建一个非常出色的模型
- 集成学习（较新的方法）：构建多个模型并取其平均值
  不仅是学习单个模型，而是学习一组模型
  将多个模型的预测合并起来

二、袋装法Bagging（Bootstrap aggregating)

1. **算法步骤**
   - Bootstrap Sampling：从原始训练数据中随机抽取M个子集（允许重复抽取），以确保基础模型能够基于不同的子集进行训练，因为某些样本可能会在新的子集中多次出现，而其他样本可能会被省略。 它降低了模型过拟合的风险，并提高了模型的准确性。为了使得模型计算上更加高效，花费更少的时间，基础的模型并行训练
   - Aggregation：一旦所有的模型被训练，他被用来对从未见过的数据做预测，针对给出的例子所预测的类标号是基于多数表决投票，由大多数投票得到的是模型的预测结果
   - OOB(out of bag) Evaluation：在用Bagging进行训练的时候，一些样本会被排除在特定基础模型的训练子集之外，OOB采样可以用来评估模型表现，不需要交叉验证
   - Final Prediction：经过聚合所有基础模型的预测，Bagging对每个样例产生一个最终的预测结果
2. 算法思想
   想法：获取独立分类器
   自助抽样：给定包含n个训练样本的集合S_n，通过有放回地抽取n个样本来创建S_n
   创建m个基础数据集：$S_n^1, S_n^2, \ldots, S_n^m$
   装袋
   - 在每个S上分别训练不同的分类器
   - 通过多数票表决来对新实例进行分类
3. 为什么要降低方差?
   n个决策树，每个决策树的输出是X_i，所有输出的方差是$Var(X_i)$
   假定在装袋之后输出的平均值是$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$，方差是$Var(\bar{X}) = Var(\frac{\sum_{n=1}^{n} X_i}{n})$

根据方差的变化公式

$$
\begin{aligned}
Var(\bar{X}) &= Var\left(\frac{\sum_{n=1}^{n} X_i}{n}\right) \\
&= \frac{1}{n^2}\left(Var\left(\sum_{n=1}^{n} X_i\right)\right) \\
&= \frac{1}{n^2}(Var(X_1) + Var(X_2) + \cdots + Var(X_n)) \\
&= \frac{1}{n^2} \times n \times \sigma^2 \\
&= \frac{\sigma^2}{n} < Var(X_i)
\end{aligned}
$$

4. “袋装法”何时适用？
   “装袋法”通常会降低分类器的方差。
   - 通过投票，集成分类器对有噪声的样本更具鲁棒性。
   - “装袋”方法在以下这类分类器中最为有用：
     不稳定：训练集中的微小变化会导致产生截然不同的模型 容易出现过拟合的情况

三、随机森林

1. 算法思想
   - 获取独立分类器
   - 重新采集训练数据是不够的
   - 泛化程度取决于个体树之间的相关性
   - 每一个树都取决于随机向量的值，而这个值是独立抽取出来的
   - 每次分割的时候限制使用的特征
2. **随机森林随机的两个来源**
   - 有放回式随机抽样
   - 随机特征子集，在每个节点，都会从d个特征中随机抽取k个特征的最佳分割
3. **随机森林步骤**
   - 从训练集中有放回抽取得到n个大小相同的样本子集，生成随机森林
   - 在满足停止条件之前，重复执行以下操作：
   - 从 F 个特征中随机选择 m 个特征，
   - 使用信息增益挑选用于分隔的最佳特征，
   - 然后将节点分割成子节点。
   - 根据这组树的多数表决结果进行预测
4. **随机森林的错误率取决于**
   森林中任意两棵树之间的相关性：更高的相关性将提升错误率
   森林中每个独立树的错误率：一个更低误差的树会降低树的错误率
5. 选取特征数m
   当m增加时候，每棵树的偏差会降低，增加准确率，增加方差

6. **随机森林变量重要性**
   - 对于变量v的重要性，对每棵树将训练集之外的案例输入这棵树，找到正确分类输出数量C_A
   - 随后对变量v，随机打乱数据里面这个属性的值，重新推理，得到打乱属性v的值之后的正确分类数C_P
   - 计算两者差值$C_A - C_P$，作为该属性对分类产生的影响得分
   - 对分类结果产生较大影响的属性更加重要

四、Adaboosting

1. **算法思路：**
   - 所有数据点都被赋予同等权重，随机选取一个训练子集。
   - 通过基于上一次训练的准确预测结果来选择训练集，从而迭代地训练 AdaBoost 模型。
   - 会给予错误分类的观测值更高的权重，以便在下一轮迭代中，这些观测值能获得较高的分类概率。
   - 在每次迭代中，根据训练分类器的准确率为其分配权重。准确率越高的分类器将获得更高的权重。
   - 这个过程会不断重复进行，直至达到指定的最大估计数量为止。
   - 最终，将所有弱学习器整合为一个单一的模型。它更重视那些表现更佳且错误率更低的弱学习器。
2. 如何得到不同的分类器
   - 在不同的训练数据集上训练
   - 如何有不同的训练数据集：
     在训练数据上重新选取来组成一个新的数据集
     在训练数据上重新分配权重来形成新的数据集
   - 如何基于带权的数据集来获取不同的分类器
3. **算法步骤**

   输入：
   - 训练样本集 $(x^n, y^n)_{n=1}^N$，其中 $y^n \in -1, +1$
   - 弱分类器数量 (T)

     步骤
   - **初始化样本权重**

$$w_1^n = \frac{1}{N}, \quad n = 1, 2, \ldots, N$$

   - **对于每个弱分类器 $t = 1, 2, \ldots, T$：**
     在加权样本上训练弱分类器 $f_t(x)$

计算弱分类器的加权错误率

$$\epsilon_t = \frac{\sum_{n=1}^{N} w_t^n \cdot \mathbf{1}(f_t(x^n) \neq y^n)}{\sum_{n=1}^{N} w_t^n}$$

计算弱分类器权重

$$\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$$

其中ϵ_t 越小 → α_t 越大 → 该弱分类器权重越高
更新样本权重

$$w_{t+1}^n = w_t^n \cdot \exp(-\alpha_t y^n f_t(x^n))$$

归一化权重

$$w_{t+1}^n = \frac{w_{t+1}^n}{\sum_{m=1}^{N} w_{t+1}^m}$$

- **组合最终分类器**

$$F(x) = \text{sign}\Big(\sum_{t=1}^{T} \alpha_t f_t(x)\Big)$$

最终分类器是各弱分类器的加权投票
弱分类器越多，整体分类误差通常越低

4. **步骤总结**
首先对初始化权重后数据的树计算分类错误率，我们选取分类错误率最小的作为第一个分类器，
随后计算分类器的权重$\alpha_t = \frac{1}{2}ln(\frac{1-\varepsilon_t}{\varepsilon_t})$，
然后使用分类器权重更新数据权重$w_{t+1}^n = w_t^n \times exp(-\alpha_t y^n f_t(x_n))$，分对就乘以$\exp(\alpha)$，分错就乘以$exp(-\alpha)$，
最后得到更新后的数据权重进入下一轮计算，
直到计算指定轮次后将若干弱分类器组合起来$F(x) = sign(\sum_{t=1}^{T} \alpha_t f_t(x))$
5. **AdaBoosting的优点**
   - 提高准确性（减少偏差）：集成方法能够通过将多个较弱模型的准确率进行组合并取平均值（用于回归任务）或对它们进行投票（用于分类任务）来提高模型的准确性，从而提升最终模型的性能。
   - 抗过拟合能力：通过调整错误分类的输入数据的权重，提升算法能够降低过拟合的风险。
   - 更有效的不平衡数据处理：增强学习能够通过更加关注那些被错误分类的数据点的方式来处理不平衡数据。
   - 更好的可解释性：增强技术能够通过将模型的决策过程分解为多个步骤来提高模型的可解释性。
6. 总结
Boosting是一个串行技术

第一种算法是通过替换的方式在与原始数据集大小相同的数据集上进行训练的。
而后续的算法则是通过拟合第一种模型的残差来构建的，这样就给那些被前一种模型预测得不准确的观测值赋予了更高的权重。
依靠一系列较弱的学习器。
需要调整的参数并不多（除了运行次数之外）
它具有灵活性（我们可以将任何学习算法进行组合）
增强型算法容易受到异常值的影响
无需事先了解弱学习器的相关知识。

7. 比较

Boosting	**Bagging**
Boosting的主要目的是为了减少偏差而不是方差	Bagging的主要目的是减少方差而不是偏差
在每一层中模型的权重都会根据其表现进行调整	所有模型权重相同
新模型会被旧模型的准确性影响	每个模型都是相互独立的