第5章：随机森林与集成学习初步

2025年7月

本讲关键问题

（1） 什么是集成学习，它的基本思想是什么？

（2） 什么是随机森林？如何基于决策树构造随机森林？

（3） 随机森林如何提高模型的估计精准度？

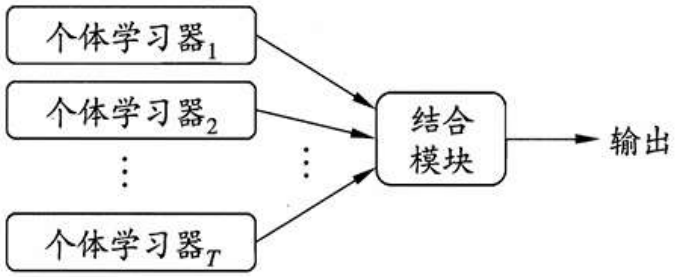
（4） 随机森林在实际应用中有哪些优点和局限？

（5） 如何构造串行的集成学习算法？

关键词：集成学习、并行算法、串行算法、Bagging、随机森林、Boosting。

一、集成学习的基本概念

- 集成学习（Ensemble Learning）是一种通过结合多个模型的预测结果来提高总体预测精准度的机器学习方法。与单个模型相比，集成模型可以减少方差、偏差，从而提升预测的稳定性和准确性。
- 假设现在有T个模型或学习器（Learner），其中每个模型或学习器的预测稳定性和准确性并非令人满意。若我们通过某些方法将这些模型或学习器的预测结果进行汇总，则可能得到更加稳定和精确的预测效果。图 1 展示了一种集成学习的示意图。



一、集成学习的基本概念

- 集成学习是先产生一组模型或学习器，再通过某种策略（结合模块）将他们结合起来，最后产生一个新的预测集成模型或集成学习器。（个体）学习器通常由现有的算法产生，如最小二乘、LASSO、决策树等等。（个体）学习器可以由相同的算法产生，也可以通过不同的算法产生。
- 从直觉上来说，将一系列好坏不等的学习器结合在一起，通常得到的新学习器会比最差的预测精准度高，但会比最好的预测精准度低。
- 思考：在实际问题中，学习器自身的预测精准度以及学习器之间的相关程度会怎么影响到集成学习器最终的预测精准度呢？

一、集成学习的基本概念

考虑一个二分类问题。假设我们现在有三个分类器，分别用h_1, h_2, h_3来表示。在测试集中共有三个样本，分别是测试例1，2，3。用符号√表示该分类器在该测试例上分类正确，反之用×来表示。集成学习的结果由投票法决定，即“少数服从多数”。

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	×	√	√
h_3	√	×	√
集成	√	√	√

(a) 集成提升性能

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	√	√	×
h_3	√	√	×
集成	√	√	×

(b) 集成不起作用

	测试例1	测试例2	测试例3
h_1	√	×	×
h_2	×	√	×
h_3	×	×	√
集成	×	×	×

(c) 集成起负作用

图2：集成学习对预测精准度的提升效果

一、集成学习的基本概念

在图2（a）中，每一个个体学习器的预测精准度都是66%，集成学习器的预测精准度则是100%，通过集成方法提升了预测精准度；而在图2（b）-（c）中，集成方法无效或者甚至有负面效果。

从图2这个简单的例子中我们可以发现，通过集成方法提升预测精准度这一途径依赖于个体学习器的性质：

1. 每个个体学习器自身的预测精准度要比较好（高于50%）；

2. 个体学习器之间要有差异性。

根据个体学习器的生成方式，目前主流的集成学习方法可以大致分成两类：

1. 个体学习器可同时生成的并行算法；

2. 个体学习器之间存在强依赖关系、必须基于序列化生成的串行算法。

二、并行算法：Bagging

- Bagging（Bootstrap Aggregating）是一种基于重抽样的算法，属于并行集成学习方法。
- 其基本思想是将原始数据集随机重抽样生成多个不同的子数据集，对每个子数据集分别训练一个个体学习器，最终将这些个体学习器的结果进行融合。
- 相比于单个个体学习器而言，Bagging的最终结果更为稳健，能够有效降低个体学习器的方差，特别适用于对方差较高的个体学习器（如决策树等）。

二、并行算法：Bagging

Bagging的思想来源于“自助法（Bootstrap）”，即通过随机有放回抽样生成多个样本集合，再利用这些集合构建多个学习器进行投票或平均，以减少学习器的方差。Bagging的关键在于样本的随机性和学习器的并行性：

a. 随机性：通过从原始数据集中重复随机抽样，生成多个数据子集，每个子集约包含原始数据集样本的70%。

b. 并行性：学习器彼此独立并行训练，不同学习器之间没有依赖关系，这一特点使Bagging能够在多核环境下加速训练过程。

c. 多样性：由于每个学习器是使用不同的样本子集训练的，能带来模型的多样性，有助于提升集成学习器的预测精准度。

二、并行算法：Bagging

算法1：

- 输入：训练数据集$D = \{Y_i, X_i\}_{i=1}^{n}$，学习器数量$K$，学习器类型$f$。
- 自助抽样：对数据集D进行有放回抽样，重复K次，生成K个样本量为n的新数据集，记为$D_1, D_2, \ldots, D_K$。
- 训练学习器：对于每一个数据集D_j，训练一个学习器f_j，最终得到新数据集对应的K个学习器$f_1, f_2, \ldots, f_K$。
- 学习器集成：
  - 分类任务：对每个学习器的预测结果进行投票（多数表决），作为最终的预测结果。
  - 回归任务：对每个学习器的预测结果求平均值，作为最终的预测结果。
- 输出：得到的最终集成学习器。

二、并行算法：Bagging

- 如何进行样本外预测？
  - 交叉验证
  - 外包法（Out-of-Bag）
- 如何解释变量的重要性？
  - RSS（回归树）
  - 基尼系数（分类树）

二、并行算法：随机森林

- 随机森林（Random Forest）是一种并行集成学习方法，通过构建多棵决策树并结合Bagging策略来提高学习器的预测能力。它通过在Bagging基础上进一步引入**随机特征选择**来增加学习器的多样性。
- 随机森林进一步在Bagging的基础上增强了多样性。在每棵决策树的构建过程中，随机选择一部分特征用于节点分裂，这使得每棵树不仅数据不同，特征空间也不同，从而减少了树之间的相关性，提升了集成学习器的预测精准度。

二、并行算法：随机森林

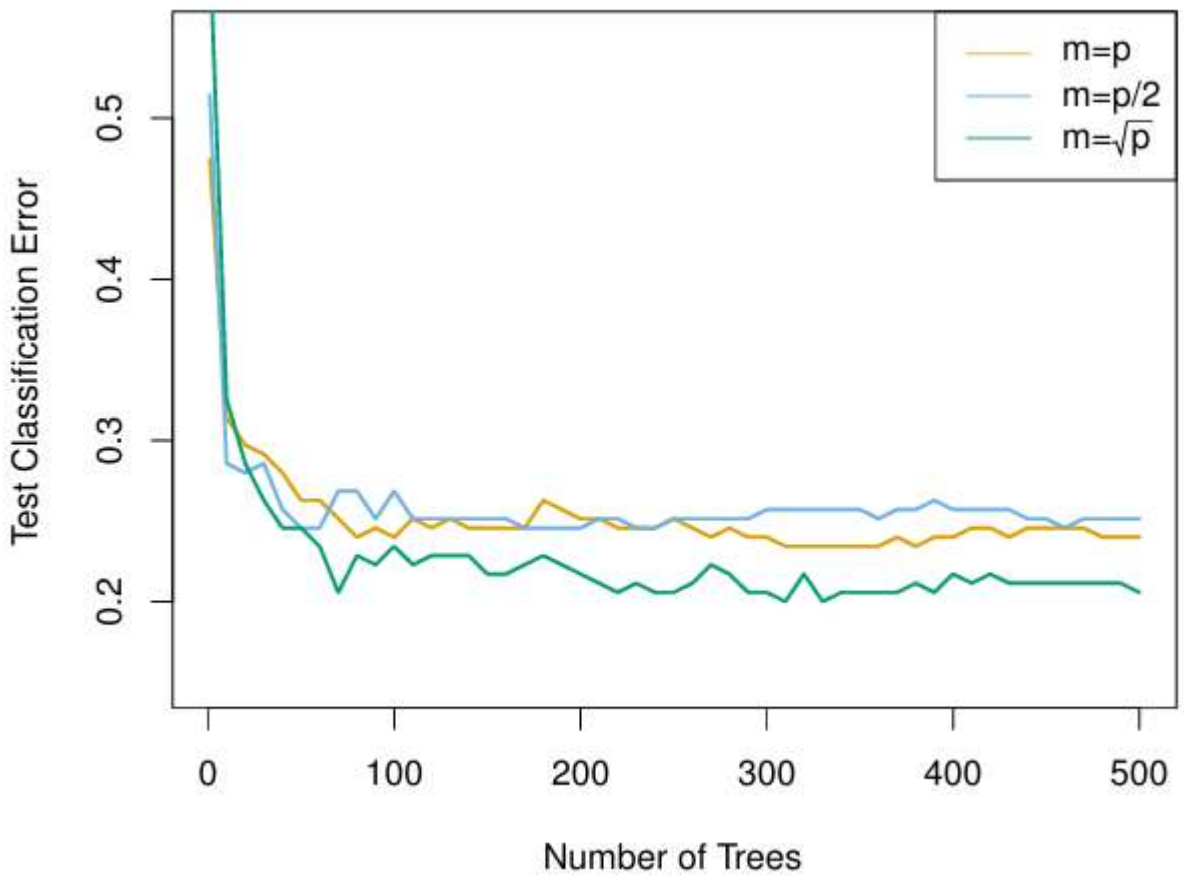
算法2：

- 输入：训练数据集$D=\{Y_i,X_i\}_{i=1}^n$，决策树数量K，特征X的子集大小p。
- 自助抽样：对数据集D进行有放回抽样，重复K次，生成K个样本量为n的新数据集，记为$D_1,D_2,\dots,D_K$。
- 训练决策树：对于每一个数据集D_j，选择其中m个X进行训练，生成一颗决策树f_j，最终得到新数据集对应的K颗决策树$f_1,f_2,\dots,f_K$。
- 决策树集成：
  - 分类任务：对每颗决策树的预测结果进行投票（多数表决），作为最终的预测结果。
  - 回归任务：对每颗决策树的预测结果求平均值，作为最终的预测结果。
- 输出：得到的最终随机森林。

二、并行算法：随机森林

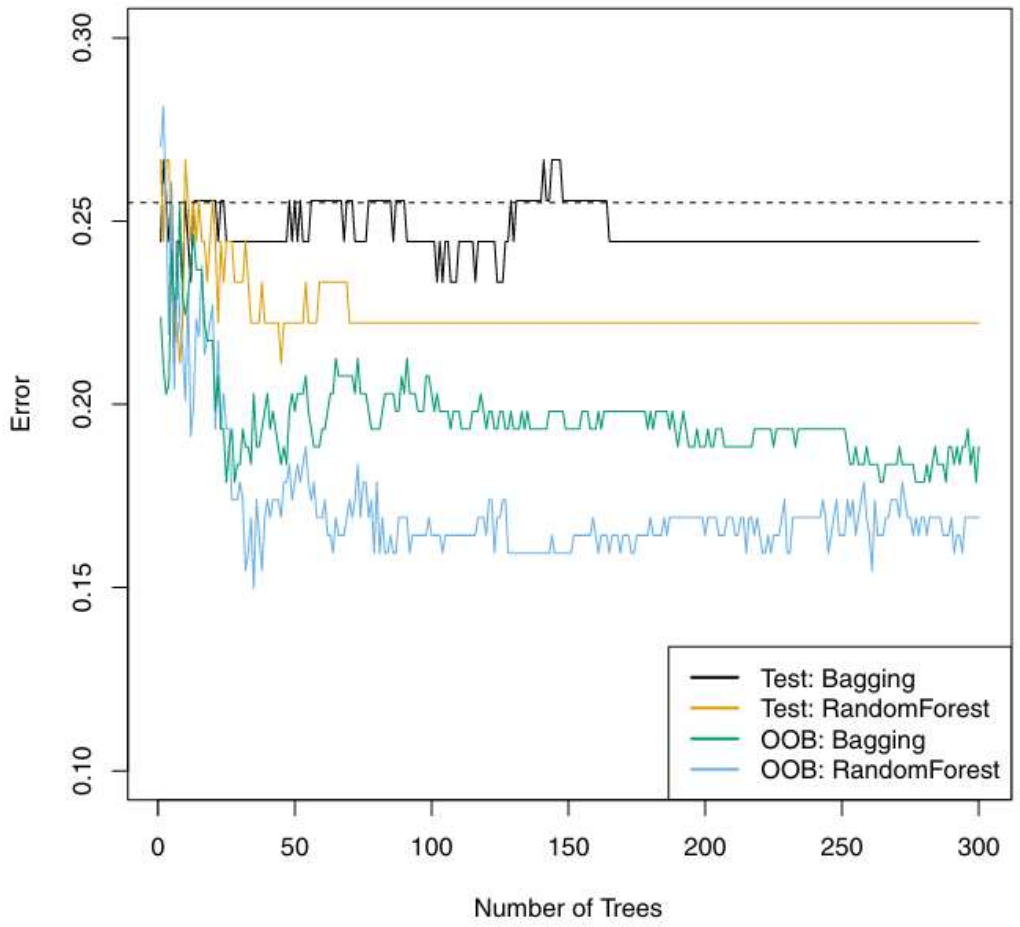
- 随机森林（Random Forest）是一种并行集成学习方法，通过构建多棵决策树并结合Bagging策略来提高学习器的预测能力。它通过在Bagging基础上进一步引入**随机特征选择**来增加学习器的多样性。
- 随机森林进一步在Bagging的基础上增强了多样性。在每棵决策树的构建过程中，随机选择一部分特征用于节点分裂，这使得每棵树不仅数据不同，特征空间也不同，从而减少了树之间的相关性，提升了集成学习器的预测精准度。

二、并行算法：随机森林



- $p = 500.$
- 单颗树的分类错误率高达45.7%!

二、并行算法：随机森林



$$m = \sqrt{p}.$$

三、串行算法：Boosting

- Boosting是一种迭代的集成方法，常用于解决分类问题。
- 它通过不断生成弱分类器（通常是简单的分类器，如决策树桩），并根据前一轮分类器的错误率调整后续分类器的训练重点，从而逐步提升整体分类效果。
- Boosting通过关注先前分类器未能正确分类的样本，将分类器的重点放在难分类的样本上，因此具有较强的分类能力。
- Boosting的原理是将多个弱分类器组合成一个强分类器，使得最终模型的预测能力远远优于单个弱分类器。

三、串行算法：Boosting

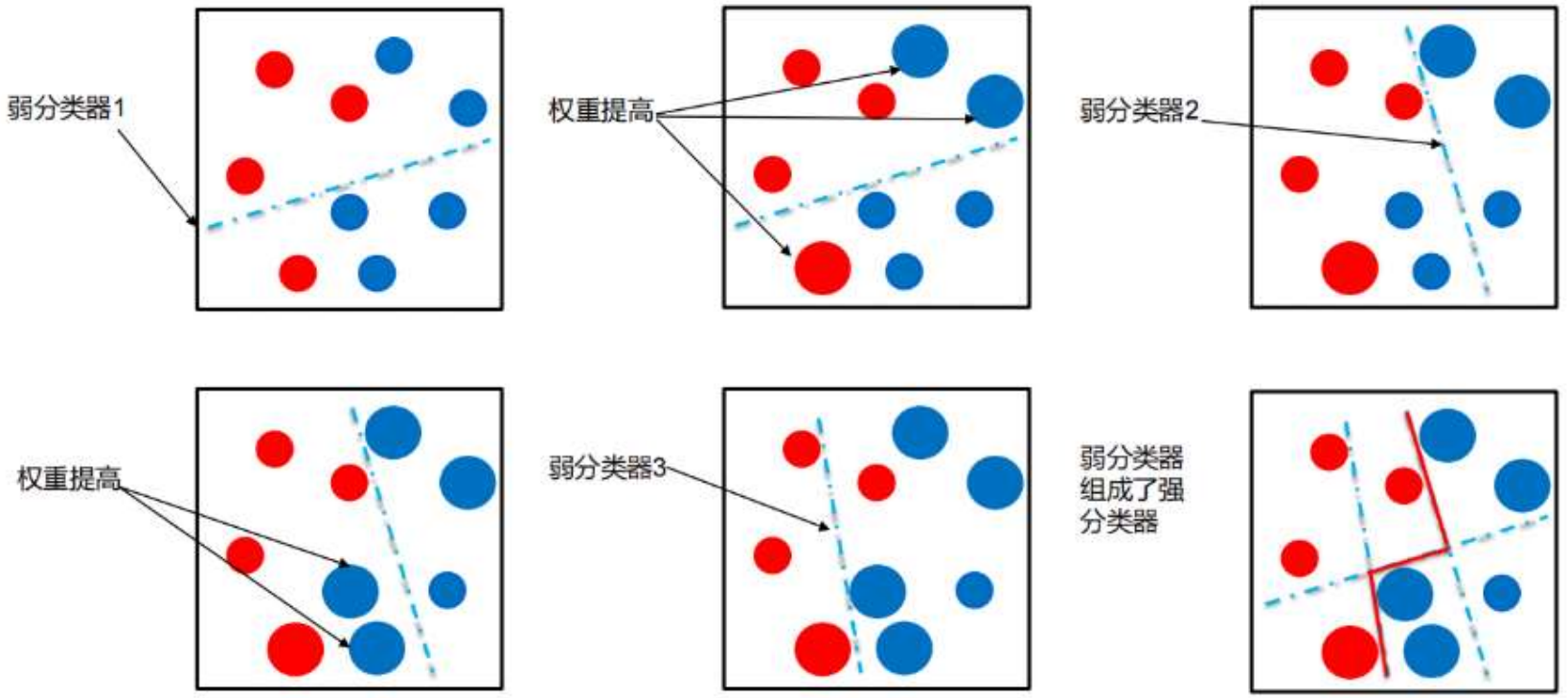
- Boosting主要通过以下三步实现：
- 初始化：给训练数据分配相同的初始权重。
- 迭代训练：在每一轮训练中：
  - 根据数据的权重训练一个弱分类器。
  - 计算弱分类器的分类错误率，根据错误率计算该分类器在最终模型中的权重。
  - 调整训练数据的权重，使得误分类的样本权重增加，从而在下一轮训练中更关注这些样本。
- 最终模型输出：将各轮训练得到的弱分类器按权重进行加权组合，得到最终的预测结果。

•根据算法不同，Boosting主要分为AdaBoost（Adaptive Boosting）、Gradient Boosting以及XGBoost（eXtreme Gradient Boosting）。

三、串行算法：AdaBoost

- 初始化样本权重：对每个样本赋予初始权重，通常设为$\omega_{0,i} = 1/n$。
- 迭代训练：对每一轮t：
  - 根据样本权重训练弱分类器$h_t(x)$。
  - 计算分类器的错误率ε_t，并用它来确定分类器的权重α_t：$\alpha_t = 0.5ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$.
  - 根据分类结果更新样本的权重，增大误分类样本的权重，从而在下一轮中给予更多关注：$\omega_{t+1,i} = \omega_{t,i}e^{\alpha_t \times \text{误分类项}}$.
- 组合模型：将各轮训练得到的弱分类器按权重进行加权，最终的分类结果为：$H(x) = sign(\sum_{t=1}^{T}\alpha_t h_t(x))$.

三、串行算法：AdaBoost



三、串行算法： Gradient Boosting

Gradient Boosting是一种通过梯度下降优化方法实现的Boosting算法，主要用于回归和分类任务。其核心思想是通过不断地拟合残差来降低模型误差。

1. 初始化模型：选择一个初始预测值，一般为$h_0(x) = 0$。令$r_i = Y_i$。
2. 迭代训练：对每一轮t，重复以下步骤：

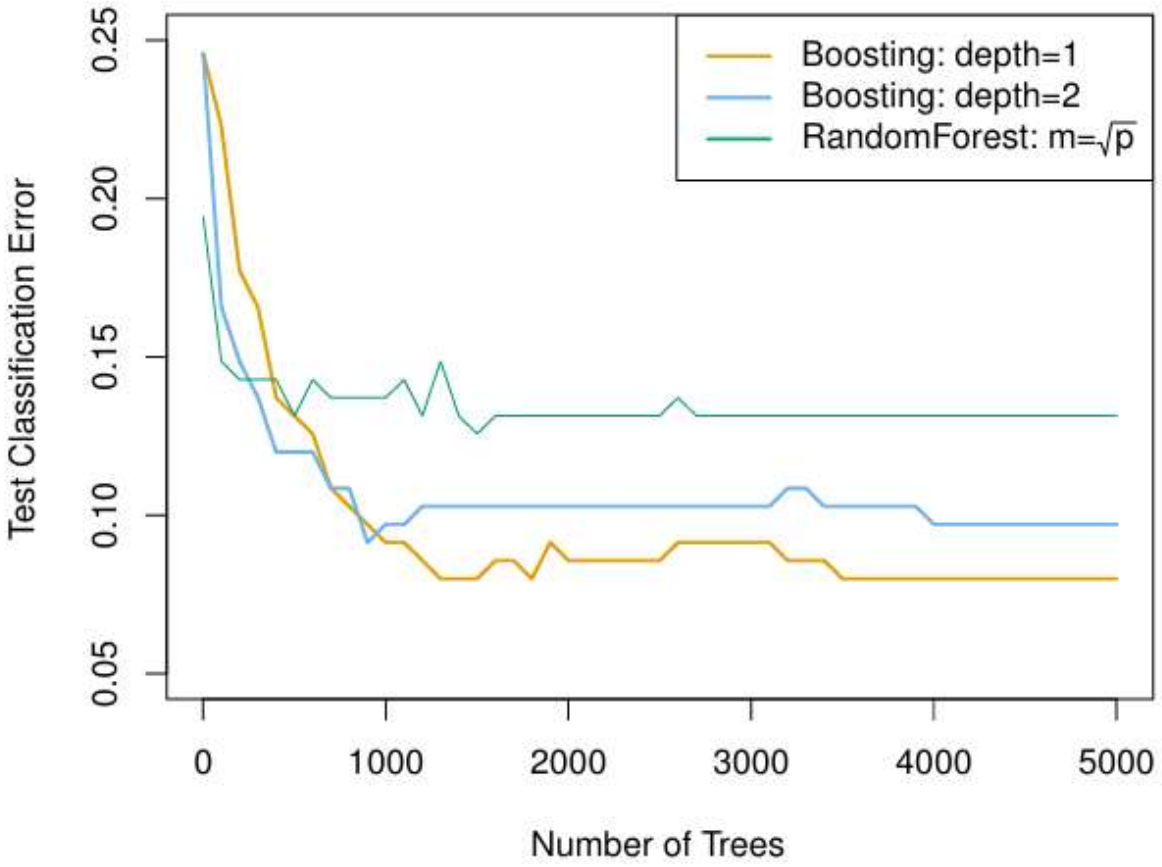
    (i)利用深度为d的决策树拟合样本(X, r)，记为$h_t(x)$ 。

    (ii)基于压缩方式来更新预测模型$h(x) \leftarrow h(x) + \lambda h_t(x)$。

    (iii)更新残差$r_i \leftarrow r_i - \lambda h_t(x)$ 。

3.组合模型： $H(x) = \sum_{t=1}^{T} \lambda h_t(x)$。

三、串行算法：Boosting



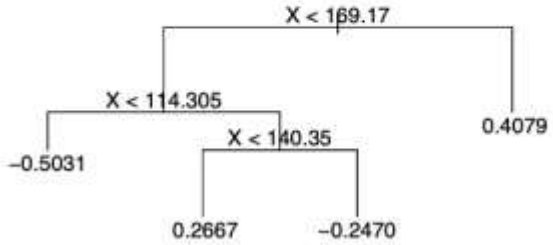
《面向机器学习的计量经济学》

三、串行算法：BART

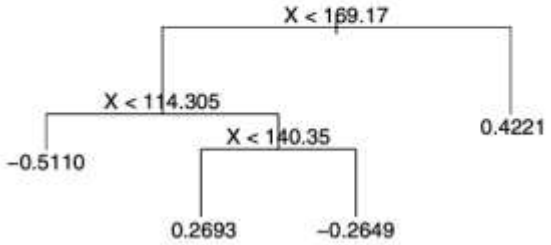
- BART（Bayesian Additive Regression Trees，贝叶斯加总回归树）是一种基于贝叶斯方法的机器学习算法，主要用于回归和分类任务。其核心思想是将预测函数建模为一组回归树的组合，并在贝叶斯框架下进行推断。
- BART 通过使用类似随机森林的多棵树结构来捕捉非线性关系，同时采用贝叶斯方法进行正则化，以防止过拟合。

三、串行算法：BART

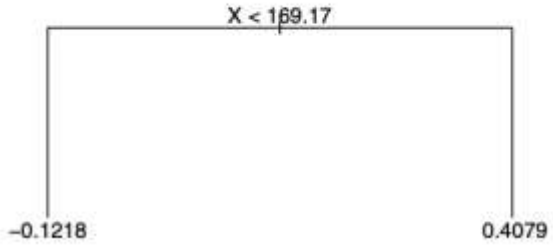
(a): $\hat{f}_k^{b-1}(X)$



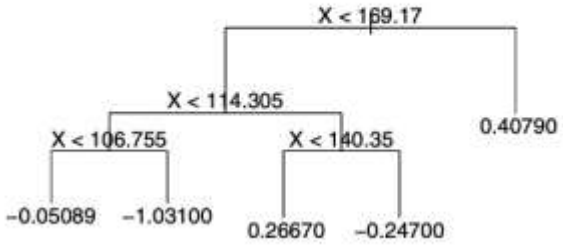
(b): Possibility #1 for $\hat{f}_k^b(X)$



(c): Possibility #2 for $\hat{f}_k^b(X)$



(d): Possibility #3 for $\hat{f}_k^b(X)$



三、串行算法：BART

Algorithm 8.3 *Bayesian Additive Regression Trees*

1. Let $\hat{f}_1^1(x) = \hat{f}_2^1(x) = \cdots = \hat{f}_K^1(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i$.
2. Compute $\hat{f}^1(x) = \sum_{k=1}^{K} \hat{f}_k^1(x) = \frac{1}{n}\sum_{i=1}^{n} y_i$.
3. For $b = 2, \ldots, B$:
   (a) For $k = 1, 2, \ldots, K$:
      i. For $i = 1, \ldots, n$, compute the current partial residual

$$r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^b(x_i) - \sum_{k'>k} \hat{f}_{k'}^{b-1}(x_i).$$

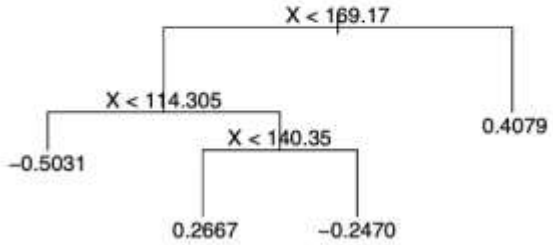
      ii. Fit a new tree, $\hat{f}_k^b(x)$, to r_i, by randomly perturbing the kth tree from the previous iteration, $\hat{f}_k^{b-1}(x)$. Perturbations that improve the fit are favored.

   (b) Compute $\hat{f}^b(x) = \sum_{k=1}^{K} \hat{f}_k^b(x)$.
4. Compute the mean after L burn-in samples,

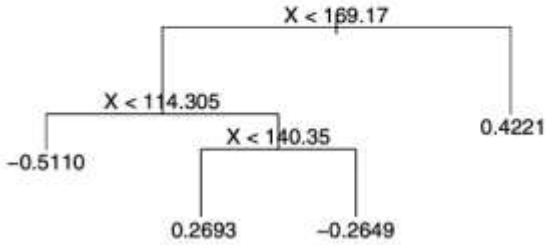
$$\hat{f}(x) = \frac{1}{B-L}\sum_{b=L+1}^{B} \hat{f}^b(x).$$

三、串行算法：BART

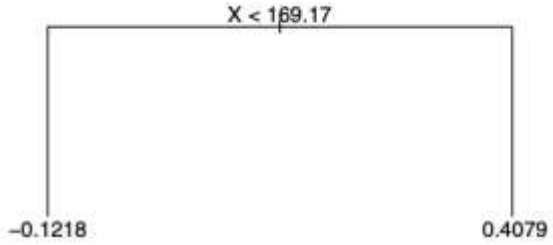
(a): $\hat{f}_k^{b-1}(X)$



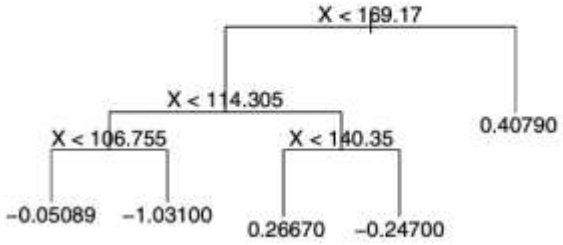
(b): Possibility #1 for $\hat{f}_k^b(X)$



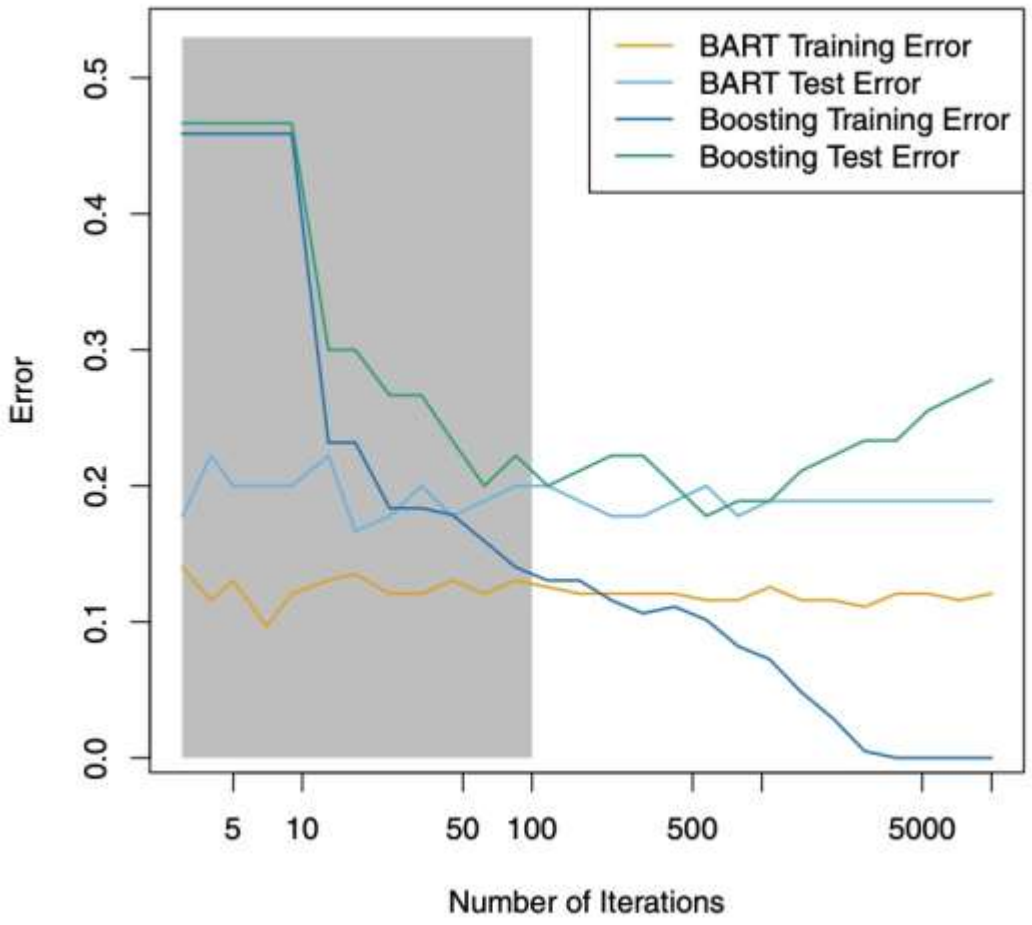
(c): Possibility #2 for $\hat{f}_k^b(X)$



(d): Possibility #3 for $\hat{f}_k^b(X)$



三、串行算法：BART



《面向机器学习的计量经济学》

四、决策树实例——工业增加值预测

- 再次研究企业生产函数这一问题。本章中我们使用决策树模型以及2004年工业企业数据库中的数据来估计企业的生产函数：
- $lnY_i = g(lnL_i, lnK_i, X_i, e_i)$,
- 其中Y_i表示第i个企业的工业增加值；
- L_i表示第i个企业的从业人员规模；
- K_i表示第i个企业的固定资产；
- X_i表示第i个企业的一些其他控制变量；
- e_i表示第i个企业的不可观测扰动项。
- 与Lasso一章的设定不同，本章中函数g为非线形设定。
- 之后我们可以用2005-2007年的数据来拟合，考察集成学习算法在样本外的拟合效果，并与Lasso，Ridge和Elastic net方法进行比较。

四、决策树实例——工业增加值预测

表 5.1 2004 年数据的描述性统计

变量	均值	标准差	最大值	最小值
log(工业增加值+1)	8.657	1.4109	16.033	1.609
log(固定资产合计+1)	8.209	1.9788	18.116	0
Log(从业人员总数+1)	214.6	1.0249	9.599	6
log(主营收入+1)	10.006	1.2983	17.559	6.897
log(应交增值税+1)	6.037	2.2782	14.587	0
log(资产总计+1)	9.930	1.4509	18.180	6.006
log(实收资本+1)	7.822	2.3933	15.790	0
log(直接材料+1)	9.329	1.5484	17.380	0
log(应付工资总额+1)	7.317	1.2462	13.509	4.111
log(管理费用+1)	7.201	1.6886	13.820	0

四、决策树实例——工业增加值预测

表 2 不同估计方法得到的均方误差（MSE）

	决策树（预剪枝）	决策树（后剪枝）	Lasso	Ridge	Elastic net（$\alpha = 0.5$）
2005 年	0.5329	0.5291	0.3683	0.2869	0.3696
2006 年	0.4357	0.4349	0.2682	0.2800	0.2677
2007 年	0.5937	0.5698	0.3911	0.4294	0.3953

不同估计方法得到的均方误差（MSE）

	Bagging	随机森林	GMBboost	XGboost
2005 年	0. 0724	0. 0718	0. 2690	0. 0677
2006 年	0. 2448	0. 2429	0. 2524	0. 2598
2007 年	0. 4189	0. 4197	0. 4300	0. 4241

参数设置：

袋装法和随机森林：选择树的数量为ntree=1000,终结点的最小规模为nodesize=5,其中袋装法使用所有变量进行分裂，mtry=length(x);随机森林用最小化袋外误差选择用于节点分裂的变量个数。

五、各类集成学习之间的比较

1. Bagging（Bootstrap Aggregating）

基本思想：从原始数据集中有放回地抽样多个子集，对每个子集训练模型（常用的是决策树），最后对结果进行平均（回归）或投票（分类）。

优点：

- 降低方差，减少过拟合；
- 训练过程可以并行；
- 对异常值不敏感。

缺点：

- 减少方差但不减少偏差；
- 各子模型间没有交互，不能彼此纠正错误。

五、各类集成学习之间的比较

2. 随机森林（Random Forest）

基本思想：是Bagging的扩展，不仅对样本做随机抽样，同时在构建每棵树时对特征进行随机选择，增加模型多样性。

优点：

- 在Bagging的基础上进一步降低相关性，提高性能；
- 对高维数据和缺失值表现良好；
- 不易过拟合，鲁棒性强；
- 可以评估变量重要性（Feature Importance）。

缺点：

- 模型可解释性差；
- 对小数据集可能过拟合；
- 难以捕捉复杂非线性关系。

五、各类集成学习之间的比较

3. Adaboost (Adaptive Boosting)

基本思想：序列式训练多个弱分类器（通常是浅层树），每一轮都增加对前一轮错分样本的权重，最终加权合并所有模型。

优点：

- 能有效减少偏差；
- 对训练误差非常敏感，适用于复杂关系；
- 可以提供特征的重要性。

缺点：

- 对噪声和异常值敏感（因为不断强调难分样本）；
- 不易并行，训练速度慢；
- 容易过拟合小数据集。

五、各类集成学习之间的比较

4. Gradient Boosting（梯度提升）

基本思想：通过迭代的方式，每一轮拟合前一轮残差，使用梯度下降方向进行损失最小化。XGBoost、LightGBM 和 CatBoost 都是它的高效实现。

优点：

- 高准确率，适应性强；
- 支持自定义损失函数；
- 对非线性、异方差、复杂交互非常有效。

缺点：

- 参数调节复杂；
- 对异常值敏感；
- 训练时间长，难以并行（但现代实现已有改进）；
- 更容易过拟合。

五、各类集成学习之间的比较

5. BART (Bayesian Additive Regression Trees)

基本思想：以贝叶斯方法对回归树的加性模型建模，引入后验分布，对结果进行不确定性量化。

优点：

- 提供预测不确定性（置信区间）；
- 对非线性和变量间复杂交互建模效果好；
- 通常无需手动调参，自动控制复杂度；
- 在中等规模数据下表现优异。

缺点：

- 计算代价较高，训练速度慢；
- 不易扩展到大规模数据；
- 实现较少，工程支持不如GBM类方法丰富。

五、各类集成学习之间的比较

方法	偏差	方差	可解释性	并行性	对异常值鲁棒性	是否提供不确定性	调参难度
Bagging	✘ 高	✔ 低	✘ 低	✔ 高	✔ 好	✘ 否	✔ 低
随机森林	✘ 中	✔ 低	✘ 低	✔ 高	✔ 好	✘ 否	✔ 低
Adaboost	✔ 低	✘ 高	✘ 低	✘ 差	✘ 差	✘ 否	✘ 中
Gradient Boosting	✔ 低	✘ 高	✘ 低	✘ 差	✘ 差	✘ 否	✘ 高
BART	✔ 低	✔ 控	✘ 中	✘ 差	✔ 好	✔ 是	✔ 低

谢 谢!