集成算法：随机森林与 XGBoost 深入探讨

2025 年 7 月 29 日

目录

1 集成学习的基石：偏差与方差

在深入两种算法之前，我们先理解集成学习试图解决的核心问题：**偏差（Bias）**和**方差（Variance）**。

- **偏差**：指模型的预测值与真实值之间的系统性误差。高偏差通常意味着模型**欠拟合**，因为它没有充分捕捉数据的潜在规律（例如，用直线拟合非线性数据）。
- **方差**：指模型在不同训练数据集上训练时，预测结果的波动性。高方差通常意味着模型**过拟合**，因为它对训练数据的微小变化过于敏感，导致在新数据上表现不稳定。

集成学习的目标，就是通过结合多个弱学习器，有效降低模型的偏差或方差，甚至同时降低两者，从而提升整体模型的泛化能力。

2 随机森林 (Random Forest)：降低方差的能手

随机森林是 **Bagging (Bootstrap Aggregating)** 思想的杰出代表。它的核心在于通过引入**随机性**来降低模型的方差。

2.1 核心思想与构建过程

随机森林的构建过程可以概括为以下三步：

1. **自助采样 (Bootstrap Sampling)**：从原始数据集中有放回地随机抽取 N 个训练样本（与原始数据集大小相同）。重复这个过程 K 次，得到 K 个不同的训练子集。由于是有放回采样，每个子集大约包含原始数据集 63.2% 的唯一样本，剩余约 36.8% 的样本称为 **“袋外”（Out-of-Bag, OOB）样本**，可以用于模型评估，无需额外的交叉验证集。

   为什么是 63.2% 的点被取到？

   假设原始数据集有 N 个样本。在进行有放回抽样时，每次抽取一个样本，被抽到的概率是 $1/N$，不被抽到的概率是 $1 - 1/N$。那么，在进

行了 N 次抽样后（即完成一次自助采样），某个特定样本从未被抽到的概率是：

$$P(\text{样本未被抽到}) = \left(1 - \frac{1}{N}\right)^N$$

当 N 趋近于无穷大时，这个表达式的极限是：

$$\lim_{N\to\infty} \left(1 - \frac{1}{N}\right)^N = e^{-1} \approx 0.368$$

这意味着，大约有 36.8% 的原始样本在一次自助采样中不会被抽到（成为袋外样本）。反之，被抽到的样本比例就是 $1 - 0.368 = 0.632$，即大约 **63.2%** 的样本会被抽到作为训练集。这个特性使得袋外（OOB）样本可以作为验证集，有效评估每棵树的性能，并作为整个随机森林的泛化误差估计。

2. **特征随机性 (Feature Randomness)**：在构建每棵决策树时，不是考虑所有特征来寻找最佳分裂点，而是从所有 M 个特征中随机选择一个子集（例如 $\sqrt{M}$ 个或 $\log_2 M$ 个特征）。然后，只在这个随机选择的特征子集中寻找最佳分裂点。

3. **并行构建与集成**：独立并并行地构建 K 棵决策树。每棵树都尽可能完全生长，不进行剪枝。对于最终预测：

   - **分类任务**：采用**多数投票**的方式决定最终类别。
   - **回归任务**：采用所有树预测结果的**平均值**作为最终预测。

2.2 为什么能降低方差？

随机森林通过以下机制有效降低方差：

- **多样性**：
  - **数据多样性**：自助采样使得每棵树看到的数据略有不同。
  - **特征多样性**：每次分裂只考虑部分特征，使得每棵树的结构和关注点不同。

这种多样性使得每棵树捕捉到数据中不同的模式，并且它们各自的错误倾向也各不相同。

- **"去相关化"**：随机性减少了树之间的关联性。如果所有树都学到相似的模式，它们会犯相似的错误，方差依然会很高。但随机森林通过多样性使得树的错误趋于独立。
- **平均效应**：当对多个独立或弱相关的预测器进行平均时，其组合的方差会小于单个预测器的方差。想象一下，如果每棵树都有独立的噪声，那么这些噪声在平均时会相互抵消，从而得到更稳定的结果。

2.3 优缺点

优点：

- **强大的泛化能力**：有效防止过拟合，对噪声和异常值不敏感。
- **并行化**：每棵树独立构建，易于并行处理，训练速度快。
- **易于使用**：参数相对较少，默认参数通常就能取得不错的效果。
- **特征重要性**：可以评估特征的重要性。
- **处理高维数据**：在特征数量远大于样本数量时仍能表现良好。

缺点：

- **可解释性相对较低**：相对于单棵决策树，随机森林是一个"黑箱"模型。
- **计算量大**：需要构建大量的树，对内存和计算资源有一定要求（但可并行）。
- **在某些情况下不如 Boosting 算法**：尤其是在处理结构化数据时，Boosting 算法通常能达到更高的精度。

3 XGBoost (eXtreme Gradient Boosting)：降低偏差的利器

XGBoost 是 **Boosting** 思想的代表，尤其侧重于通过迭代优化来**降低模型的偏差**，同时通过强大的正则化和工程优化来控制方差。

3.1 核心思想与迭代优化

XGBoost 继承了梯度提升的“串行”构建模式，每棵新树都旨在纠正前面所有树的累积残差。但它在以下方面进行了重大改进：

1. **广义损失函数与二阶泰勒展开**：传统的梯度提升通常依赖于一阶导数（梯度），例如，对于均方误差，残差就是一阶梯度。XGBoost 更进一步，对损失函数进行二**阶泰勒展开**，从而同时利用损失函数的一阶梯度 g_i 和二阶梯度 h_i。目标函数在添加第 t 棵树 f_t 时，可以近似表示为：
$$\text{Obj}^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$
其中 $\Omega(f_t)$ 是正则化项。这种二阶信息的使用，使得优化过程更加精确和灵活，能够支持任何二阶可导的损失函数。

2. **正则化**：XGBoost 在目标函数中显式地加入了正则化项 $\Omega(f_t)$，惩罚模型的复杂度：
$$\Omega(f) = \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2$$
   - γT：惩罚叶子节点的数量 T，鼓励生成更小的树。
   - $\frac{1}{2}\lambda \sum w_j^2$：L2 正则化项，惩罚叶子节点上的预测分数 w_j，防止分数过大。

   正则化是 XGBoost 防止过拟合的关键，使得它在保持高精度的同时，仍具有良好的泛化能力。

3. **分裂查找算法**：
   - **精确贪婪算法**：类似于传统的决策树，遍历所有特征的所有可能分裂点，选择能够最大化增益的分裂点。增益的计算公式为：
$$\text{Gain} = \frac{1}{2} \left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I_P} g_i)^2}{\sum_{i \in I_P} h_i + \lambda} \right] - \gamma$$
     其中 I_L, I_R, I_P 分别代表左右子节点和父节点样本集。
   - **近似贪婪算法**：对于大规模数据，精确算法计算量过大。XGBoost 提出了近似算法，通过特征分位数（Quantile Sketch）将特征值离散化，只在这些分位数点上评估分裂，大大降低计算复杂度。

- **稀疏感知分裂算法**：XGBoost 对稀疏数据（如缺失值、零值）进行了优化，它能够智能地将稀疏值分配到左子树或右子树，以最大化增益。

4. **学习率 (Shrinkage)**：每棵新树的预测结果会乘以一个学习率 η（通常在 0 到 1 之间），然后再加到整体模型中。这可以减缓学习过程，使得模型更加稳健，进一步防止过拟合。

5. **列采样 (Column Subsampling)**：类似随机森林，XGBoost 也支持在每次构建树时随机选择部分特征，这不仅能防止过拟合，还能加速训练。

3.2 优缺点

优点：

- **极致的性能和精度**：在许多数据集和竞赛中表现出领先的性能。
- **强大的泛化能力**：内置正则化机制有效防止过拟合。
- **处理大规模数据**：支持并行和分布式计算，以及近似算法，能够处理亿级别的数据。
- **灵活性**：支持自定义损失函数，适用于分类、回归、排序等多种任务。
- **处理缺失值**：内置缺失值处理机制。
- **可扩展性**：支持多种平台和语言。

缺点：

- **训练速度相对较慢**：Boosting 的串行特性使其不能像 Bagging 那样完全并行。
- **参数众多且调参复杂**：模型参数较多，调优需要经验和时间。
- **模型可解释性差**：同样是“黑箱”模型。

表 1: 随机森林与 XGBoost 对比

特性	随机森林 (Bagging)	XGBoost (Boosting)
训练方式	并行训练，树之间独立	串行训练，后一棵树基于前一棵树的误差
目的	降低方差，减少过拟合	降低偏差，逐步提升模型精度
组合方式	投票（分类），平均（回归）	累加每棵树的预测结果并加权
弱学习器	通常是深度较深的决策树（但独立）	通常是深度较浅的决策树（关注残差）
代表算法	Bagging，随机森林	AdaBoost，GBDT，XGBoost，LightGBM

4 随机森林 vs. XGBoost：何时选择？

理解了它们的深层原理，我们就能更好地判断何时选择哪种算法：

- **选择随机森林的情况**：
  - **对速度要求高，且可以接受略低的精度**：随机森林可以高效并行，训练速度快。
  - **数据量非常大，难以完全加载到内存**：Bagging 机制可以处理这种情况。
  - **需要降低模型的方差，避免过拟合**：随机森林在这方面表现突出。
  - **简单易用，不希望花费太多时间调参**：随机森林通常默认参数效果不错。
- **选择 XGBoost 的情况**：
  - **追求极致的模型精度**：尤其是在结构化数据上，XGBoost 往往能取得更高的分数。
  - **数据质量较高，对异常值不那么敏感**（尽管有正则化）。**愿意投入时间进行参数调优**：正确的调参能极大发挥 XGBoost 的潜力。
  - **需要利用特征的一阶和二阶梯度信息进行更精细的优化**。

在实际项目中，两者都常常被尝试。通常，**XGBoost 会作为首选的树模型尝试**，因为它在竞赛中表现优异。但如果计算资源有限或对模型可解释性有更高要求时，随机森林也是一个非常好的选择。很多时候，通过**模型融合（Ensemble Stacking/Blending）**，将这两种模型的预测结果结合起来，还能进一步提升性能。