

集成学习——XGboost

- 目标函数

- 损失函数（分回归任务和分类任务）+正则化项（叶子节点权重的L2正则和叶子数量的L1正则）
- 回归和分类两个任务损失函数的区别

差异点	回归任务	分类任务
损失函数	均方误差 (MSE)	对数损失 (Log Loss) 或Softmax交叉熵
输出值意义	直接预测连续值	预测概率 (需通过sigmoid或softmax转换)
梯度计算	残差 ($\hat{y}_i - y_i$)	预测概率与真实标签的差值
Hessian计算	常数1	概率的方差 (依赖预测值)

- 关键参数

- booster**: 定义基学习器类型
 - gbtree: 树模型 (可捕获非线性关系)
 - gblinear: 线性回归 (表达线性组合)
- objective**: 指定目标函数 (分类任务和回归任务的损失函数不同)
 - 回归任务目标函数
 - 损失函数+正则项
 - 对应参数: $squarederror = \sum (y_i - \hat{y}_i)^2$
 - 分类任务
 - 二分类: 交叉熵 (对数损失) binary: logistic
 - 多分类: softmax, multi:softmax
- eval_metric**: 评估指标 (目标函数不同, 对应评价指标不同)
 - 回归任务
 - RMSE、MAE
 - 分类任务
 - 二分类: error、logloss、auc (ROC曲线下面积)
 - 多分类: merror (多分类错误率)、mlogloss (多分类对数损失)
- gamma**——叶子节点数量的L1正则项/分裂最小增益阈值
双重意义
 - 叶子节点数量的正则项
 - 来源于XGboost的目标函数 (损失函数+正则项构成)
 - 目标函数 (min) 的正则项展开: γT (当前树叶子节点的个数) + $\frac{1}{2\lambda} ||w_j||^2$ (节点值的平方和)
 - gamma就是控制某一棵树不要太过复杂 (叶子节点不能过多)

每增加一个节点，目标函数（min）就增加一个gamma

- 分裂最小增益阈值

- Gain计算公式

$$\text{Gain} = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma$$

衡量节点分裂带来的损失函数下降的幅度 大于 γ 才分裂

- 判断：Gain > 0 分裂；否则停止分裂
 - gamma 越大，分裂条件越严格
 - lambda 越大，Gain 中分母越大，抑制了节点分裂

- **lambda**——叶子节点权重的L2正则项参数（防止一个叶子捕捉过于敏感信息）

- 来源于XGboost的目标函数（损失函数+正则项构成）
 - 目标函数（min）的正则项展开： γT （当前树叶子节点的个数）+ $1/2\lambda ||w_j||^2$ （节点值的平方和）
 - lambda 用于惩罚叶子节点的权重值，防止某一个叶子（预测值）的权重过大
使模型的预测更加的平滑，避免对噪声的过拟合

- **eta**——学习率（防止一棵树对预测决定作用过大）

- 作用：防止一棵树对结果的决定性作用过大
 - 操作：给每棵树（基学习器）加上一个步长 η
 - 简单理解为：对每棵树的预测结果进行平滑加权，避免单棵树过度拟合噪声

- **subsample**——行采样

- 功能：每棵树训练时，随机抽取部分样本（如70%，参数就是0.7）用于建树
 - 作用：增加树之间的多样性（类似随机森林的Bagging思想）；数据减少可以加快训练
 - 举例：subsample: 0.8
每棵树使用80%样本

- 列采样（特征压缩）

- 分类

- 按树随机：colsample_bytree

根节点按照什么特征划分，整棵树接下来所有的生长都按该特征划分

- 适用于特征间独立性较强的场景（如不同特征代表不同物理意义）

- 按层随机：colsample_bylevel

每棵树的每一层分裂都需要重新选择特征进行分裂

- 适合处理层次化特征（如深度越大的层对应更细粒度的特征）

- 每个节点都重新选择特征后再分裂（精确算法）：colsample_bynode

- 最大化随机性，适合高维稀疏数据（如文本分类）

- 举例：

- colsample_bytree: 0.7

每棵树使用70%特征


- `colsample_bylevel`: 0.7

每层使用70%特征

- 参数调优建议

- 先调`colsample_bytree`: 确定全局特征保留比例。
- 再调`colsample_bylevel`: 控制层次化随机性。
- 最后调`colsample_bynode`: 精细化控制节点级随机性（仅在特征数极大时使用）。
- 这三个参数是层级关系，可以共同作用

假设总共有 M 个原始特征，参数设置如下：



```
python
params = {
    'colsample_bytree': 0.8, # 按树采样保留80%特征 → 剩余 m1 = M × 0.8
    'colsample_bylevel': 0.5, # 按层采样保留50%特征 → 剩余 m2 = m1 × 0.5
    'colsample_bynode': 0.5 # 按节点采样保留50%特征 → 剩余 m3 = m2 × 0.5
}
```

- 最终每个节点分裂时可用的特征数：

$$m_{\text{final}} = M \times 0.8 \times 0.5 \times 0.5 = M \times 0.2$$

组合方式是“逐步细化采样”，即先按树采样 → 再按层采样 → 最后按节点采样，形成三级随机性

- 早停轮数

- 在模型没必要训练时及时结束训练

模型性能（比如损失值）不再提升甚至开始下降时，立即停止训练

- 没必要训练的情况

- 验证集上的Loss在模型多次迭代后，没有下降
- 验证集上的Loss开始上升
- 验证集上的准确率在模型多次迭代后，没有上升
- 验证集上的准确率开始下降

- 早停法

以性能以损失（min）为衡量标准为例

- 手段：通过监控验证集的表现
-

假设置 `early_stopping_rounds=50`，流程如下：

1. 初始化：我们希望越小越好
 - 记录最佳验证集损失值 `best_loss = ∞`
 - 计数器 `counter = 0`
2. 迭代检查：
每训练一轮（树）后：
 - 计算当前验证集损失 `current_loss`
 - 如果 `current_loss < best_loss`：
 - 更新 `best_loss = current_loss`
 - 重置计数器 `counter = 0`
 - 记录当前模型状态
 - 否则：
 - `counter += 1`
3. 终止条件：
 - 当 `counter ≥ 50` 时，说明连续50轮验证集损失没有改善
 - 停止训练，并回滚到 `best_loss` 对应的模型状态

• 随机种子

- 功能：控制随机过程的确定性
 - 预测中的随机过程有：交叉验证时数据集的随机分割；随机列/行的特征采样；当多个特征具有相同增益的时候，随机选择分裂的特征；
- 作用：固定seed可确保多次运行结果一致，便于实验复现（可重复性）

• eta和早停轮数的设置

- 原则
 - 低学习率（0.05），单棵树的权重更新幅度较小，模型需要更多迭代轮次（即更多的树）才能收敛到最优解，此时，若早停轮数设置过小（如 `early_stopping_rounds=20`），可能导致模型在尚未充分学习时提前终止，造成欠拟合
 - 高学习率（0.3），单棵树的权重更新幅度较大，模型收敛速度加快，可能在较少轮次内达到最优性能。此时，若早停轮数设置过大（如 `early_stopping_rounds=100`），可能错过最佳停止点，导致过拟合

3. 参数调优的经验法则

在实践中，学习率与早停轮数的设置需遵循以下原则：

1. 固定学习率，动态调整早停轮数：
 - 通常先设定较小的学习率（如 `eta=0.05~0.1`），以确保模型充分学习。
 - 通过交叉验证观察验证集损失的收敛曲线，确定合理的早停轮数（如 `early_stopping_rounds=50~100`）。
 - 若损失曲线波动较大（常见于噪声数据或小样本场景），可适当增大早停轮数以平滑噪声影响。
2. 学习率与早停轮数的经验关系：
 - 当学习率减半时，所需迭代轮次大致翻倍，此时早停轮数可同比增加（但非严格线性）。
 - 例如，若 `eta=0.1` 时最优轮次为 100 轮（早停轮数设为50），则 `eta=0.05` 时最优轮次可能增至 200 轮（早停轮数可设为80~100）。

● 预测中防止数据泄露很重要！！！！

在分割数据为训练集/测试集之前的对数据集进行数据准备技术操作可能会导致数据泄露

● 数据泄露的表现

- 训练模型时直接/间接使用了测试集数据，导致模型记住数据
- 间接使用例子
 - 对数据进行归一化时，需要计算每个变量的最大值和最小值, 并利用这些值去缩放变量. 之后才将数据集分为训练数据集和测试数据集。
因为使用全局最小值和最大值进行了缩放，模型会掌握更多有关变量全局分布的信息。

● 数据泄露避免方式

原则：数据准备工作只能在训练数据集中进行

● 防止数据泄露的数据准备方法顺序

- 先分割数据
- 在训练数据集上进行数据准备
包括数据转换、特征选择、降维、特征工程
- 将数据准备技术应用于训练和测试数据集
 - 如何应用？

测试集被动接受训练集生成的参数的过程

● 数据预处理阶段

- 使用fit-transform分离机制（scikit-learn里面有transform函数）

```
# 原始数据分割
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# 仅在训练集计算参数
scaler = StandardScaler().fit(X_train) # 训练集fit()

# 用训练集参数转换所有数据
X_train_scaled = scaler.transform(X_train) # 训练集transform()
X_test_scaled = scaler.transform(X_test) # 测试集transform()
```

在训练集上训练，得到参数
再应用标准化参数到训练/测试集上面

● 特征工程阶段

●

技术类型	训练集操作	测试集操作	实例
特征选择	基于 <u>训练集选择特征子集</u>	强制使用相同特征子集	方差阈值、卡方检验
降维技术	在训练集训练降维模型	应用训练好的降维模型	PCA、LDA
编码技术	在训练集构建编码映射	沿用相同编码映射	One-Hot、标签编码

● 特殊场景

- 时间序列数据：必须按时间顺序分割，禁止随机shuffle

- 交叉验证：需在训练集内部做嵌套交叉验证
- 数据增强：仅应用于训练集，测试集保持原始分布
- 评估模型