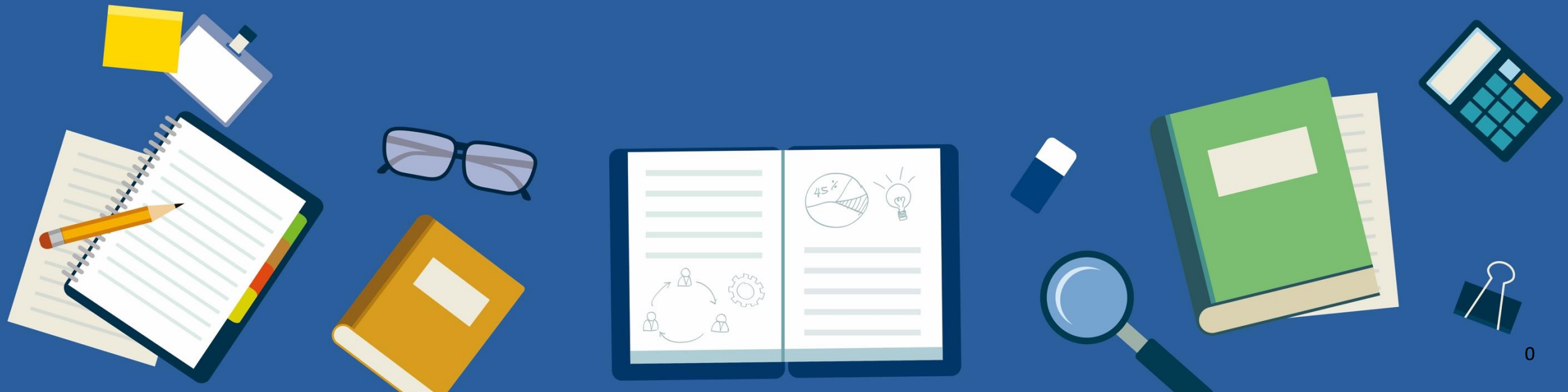
博雅大数据学院
BOYA BIG DATA COLLEGE

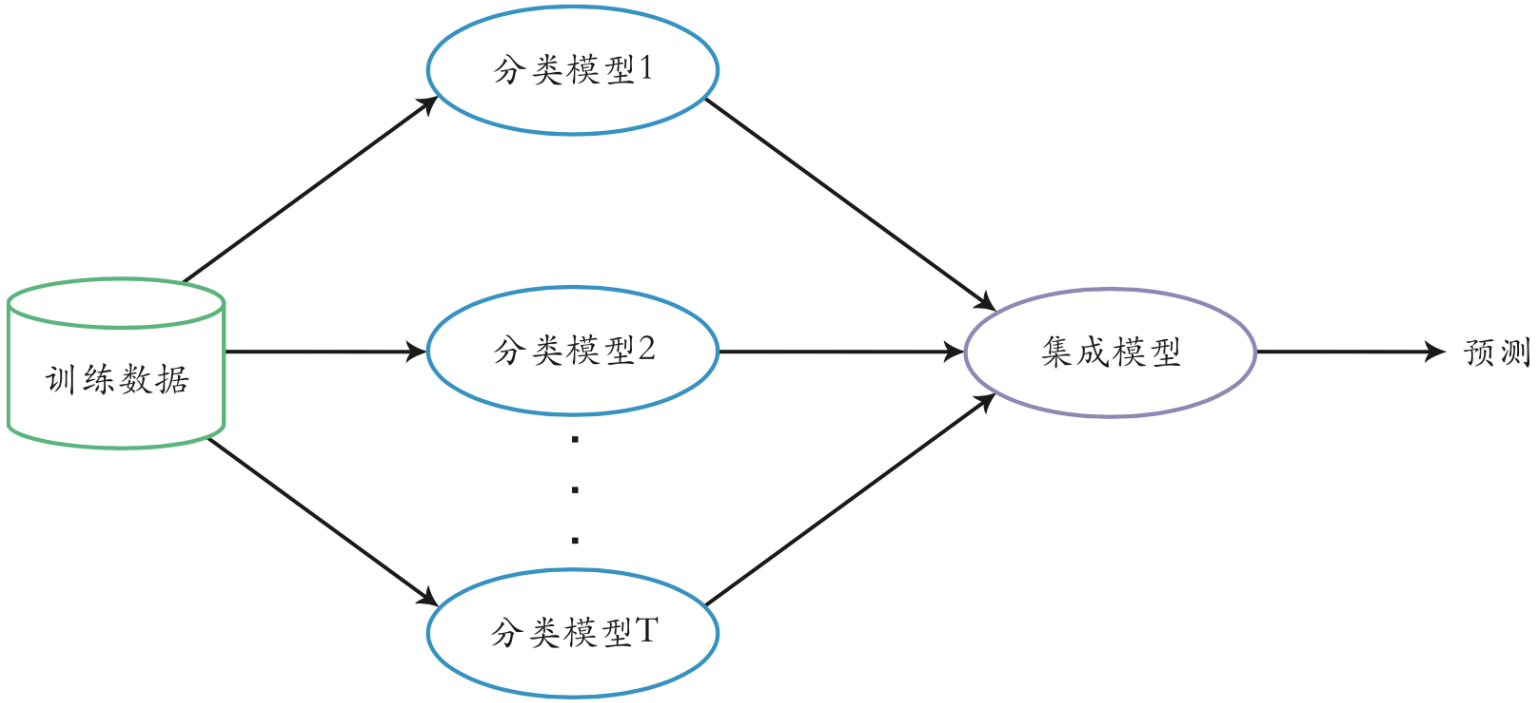
第五部分
集成模型



目录

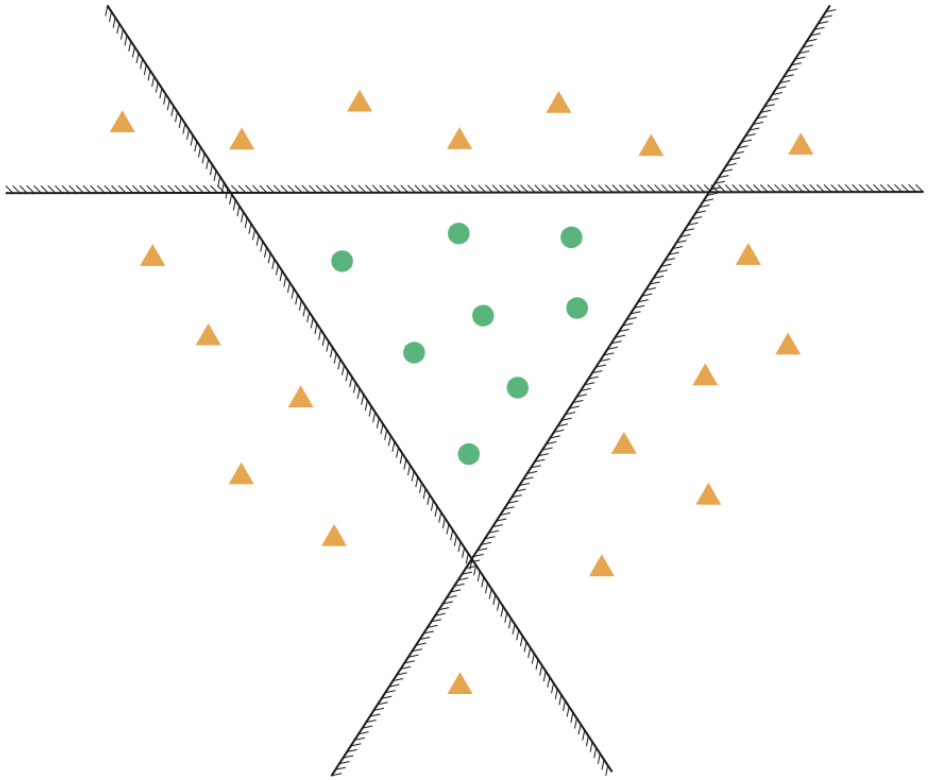
集成方法概述

- “三个臭皮匠，顶个诸葛亮”
- 集成多个模型的能力，得到比单一模型更佳的效果



为什么能够提高效果?

- 增强模型的表达能力
  - 单个感知机无法正确分类数据
  - 集成三个感知机能正确分类数据



为什么能够提高效果?

- 降低误差
  - 假设单个分类器误差 p，分类器之间独立，T个分类器采用投票进行预测，得到集成模型 H
  - 集成分类器误差为

$$Error_H = \sum_{K \leq \frac{T}{2}} C_T^k p^{T-k} (1-p)^k$$

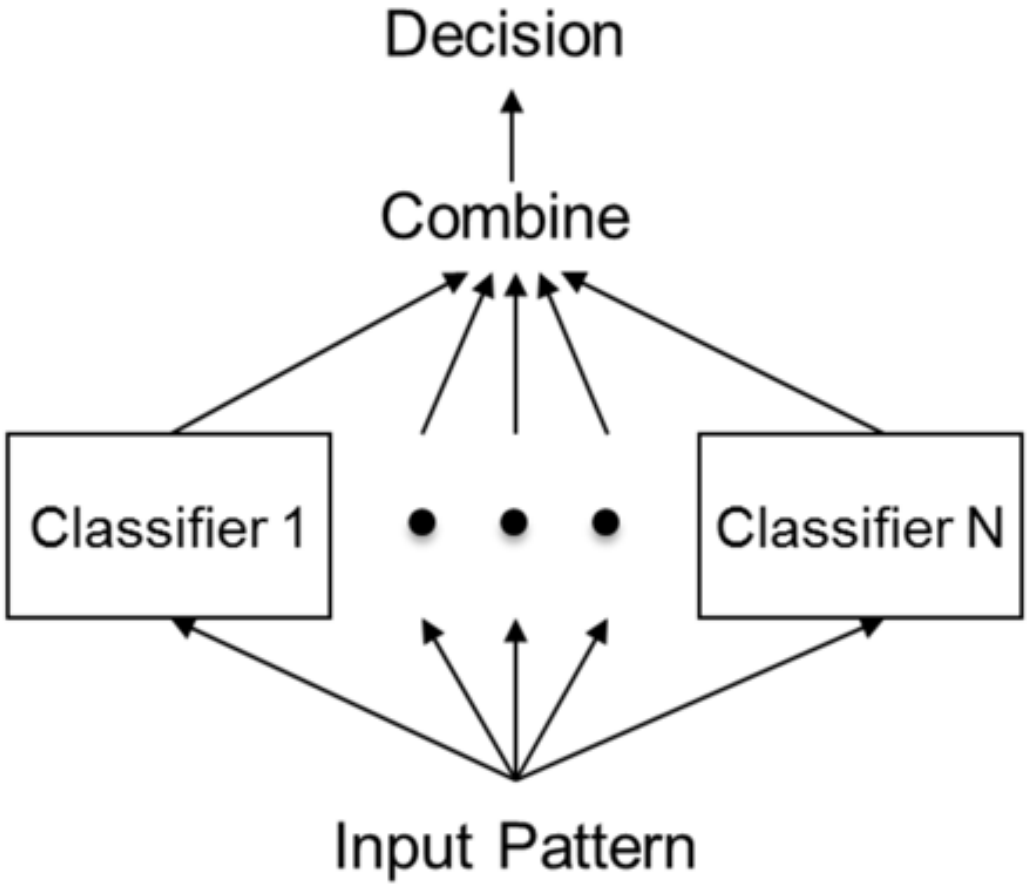
- T=5, p=0.1 时，$Error_H$<0.01

基模型集成策略

- 多数投票方法 (majority vote)
- 平均 (averaging)
- 加权平均 (weighted averaging)：如AdaBoost

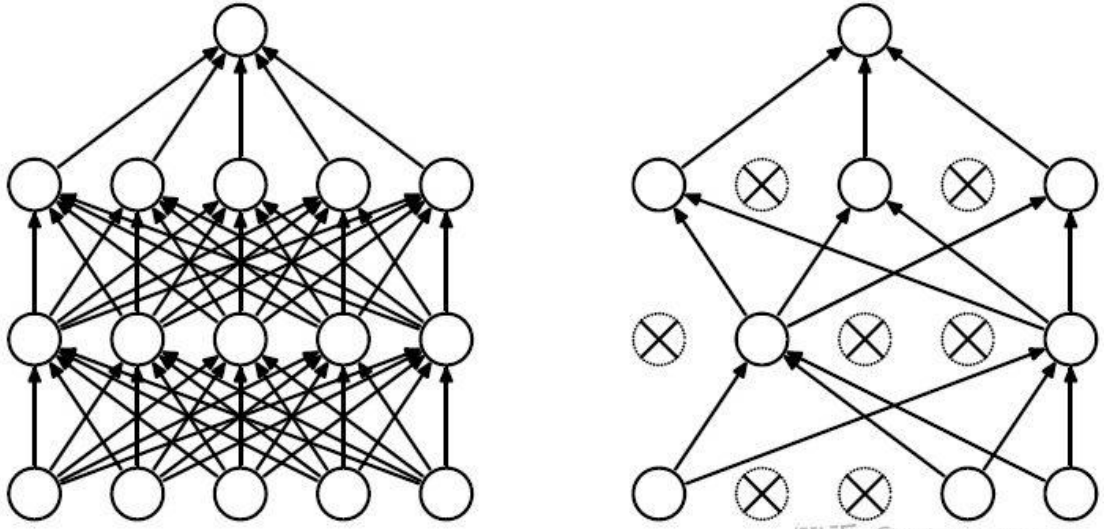
典型的集成方法

- Bagging（Bootstrap Aggregating）
  - 对样本或特征随机取样，学习产生多个独立的模型，然后平均所有模型的预测值
  - 主要减小方差
  - 典型代表随机森林
- Boosting
  - 串行训练多个模型，后面的模型是基于前面模型的训练结果（误差）
  - 主要减小偏差
  - 代表是AdaBoost



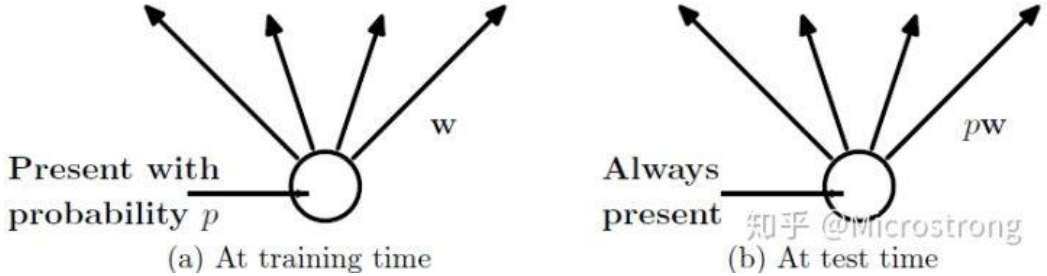
集成思想在神经网络中的应用

- Bagging -> Dropout
  - 训练时随机丢掉一些神经元



(a) Standard Neural Net　　(b) After applying dropout.

  - 测试时使用所有神经元



(a) At training time　　(b) At test time

- Boosting -> ResNet
  - 输出 = 输入 + 非线性层
  - 建立起从输入到输出的直接shortcut

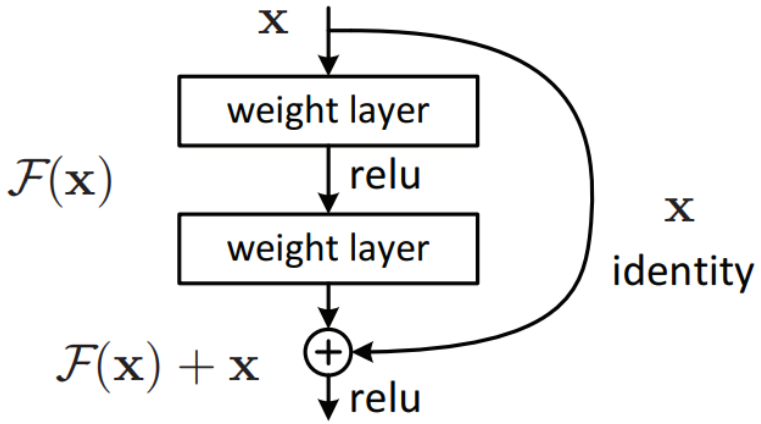


Figure 2. Residual learning: a building block.

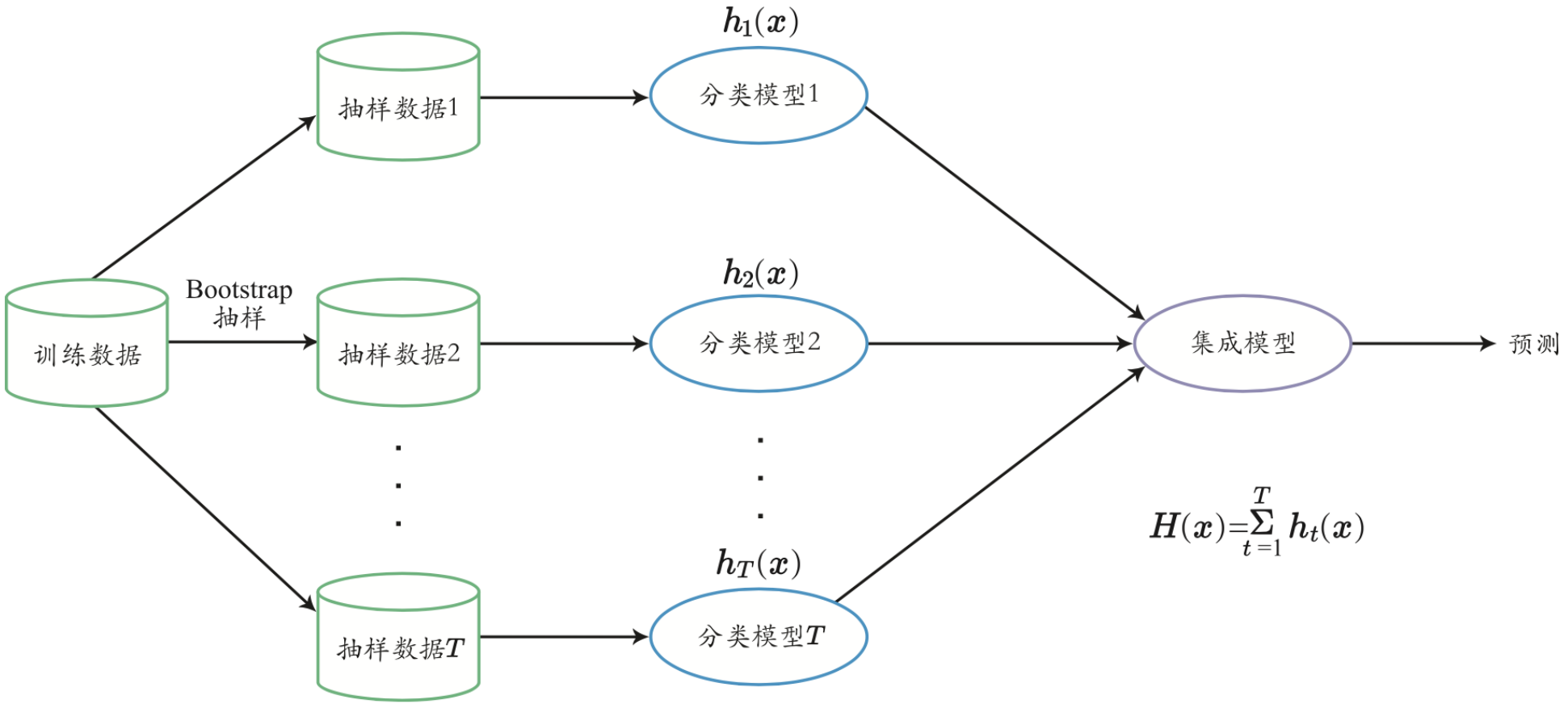
  - <=>非线性层拟合两层之间的差值

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$

目录

Bagging

- Bagging是“Bootstrap aggregating”的缩写
- 该方法的核心包括自助抽样(Bootstrap)和平均(Aggregating)



Bagging自助抽样过程

- 通过有放回抽样的方式，获取与原始数据集 D 大小一样的样本集 D_t
- 给定 n 个样本的数据集 D，对于迭代$(t = 1,2,\dots,T)$，用来创建分类器M_t；基于基分类器的投票返回类预测

- 一个样本有$1-\frac{1}{n}$ 的概率不会被选到，则一个样本不会被抽到的概率是

$$\lim_{n\to\infty}(1-\frac{1}{n})^n = 0.368$$

- 意味着训练集包含63.2%的不同的样本

Bagging提升过程

- 每个 D_t 学习得到一个分类器 M_t
- 对于分类问题，每个分类器返回类预测，采用多数投票的方法
- 对于回归问题，采用平均值的方法
- 有效性原因: 通过降低由于不稳定学习器 (unstable learners) 所造成的方差来降低错误率

泛化误差 $E[(y - \hat{f}(\boldsymbol{x}))^2]$

$$= Var(\boldsymbol{x}) + bias^2(\boldsymbol{x}) + variance(\boldsymbol{x})$$

Irreducible Error	偏差平方	方差
真实值与观测值之间的方差	真实值与模型预测值期望之间的方差	模型预测值与预测值期望之间的方差

Bagging算法

算法 2 Bagging 算法

输入: 训练集 $D = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n)\}$，学习算法 h.

输出: 集成模型 $H(\boldsymbol{x})$.

1: **for** $t = 1$ to T **do**

2: 使用自助抽样，从训练数据集 D 获得抽样数据集 D_t;

3: 基于 D_t，使用学习算法训练得到一个基础模型 $h_t(\boldsymbol{x})$;

4: 输出最终的分类器: $H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} h_t(\boldsymbol{x})\right)$.

- 预测：使用组合分类器对样本$\boldsymbol{x}$分类

Bagging的优势

- 特别适合用来提高那些方差大但偏差小的基模型(决策树，神经网络等)的预测性能
- 单个模型不稳定：对训练数据轻微的改变就能够造成分类器性能很明显的变化
- 使用Bagging可以综合投票结果，从而提升稳定性以及准确率
- 便于并行化。多个抽样数据的获取及基模型的训练互相没有关联，可以方便地进行并行计算

随机森林

- 最典型的Bagging算法：“随机”是其核心，“森林”意在说明它是通过组合多棵决策树来构建模型
- 主要特点：
  - 对样本进行有放回抽样
  - 对特征进行随机抽样
  - 基本分类器最常见的为决策树
- 它在许多实际问题中得到了广泛的应用，例如市场营销、股票市场分析、金融欺诈检测、基因组数据分析和疾病风险预测

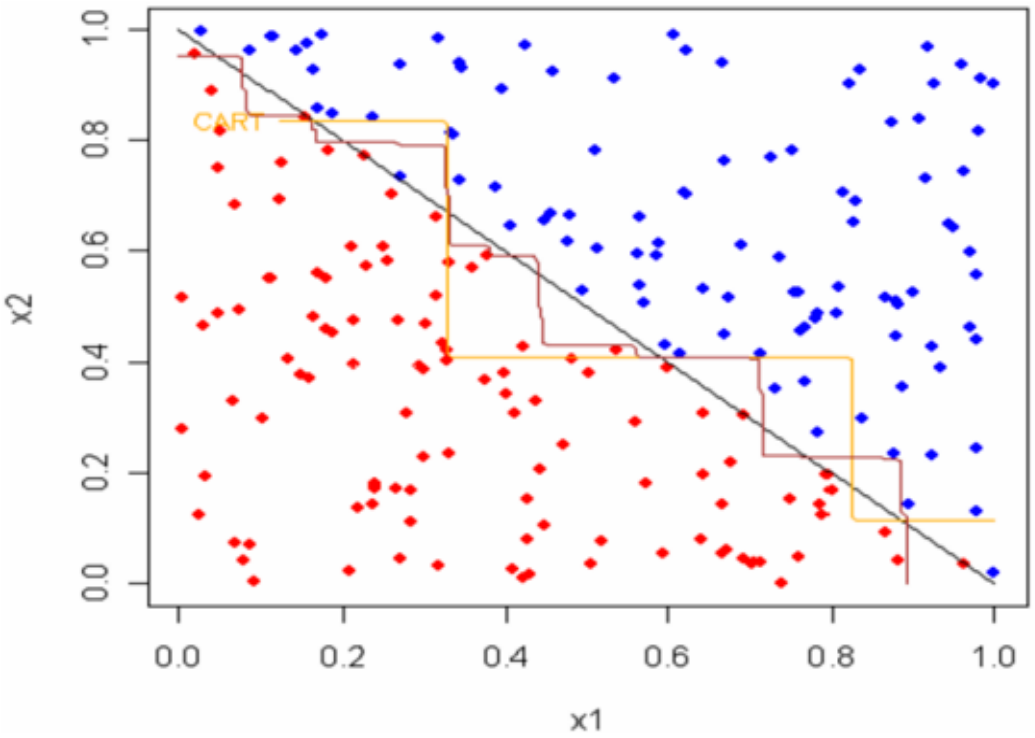
决策树的局限性

- 局部最优问题

决策树由于使用贪婪算法的思想，即在每次分割时选择当前情况下带来信息增益最高的特征进行分割，容易陷入局部最优

- 分类边界问题

单棵的决策树在确定分类边界时，由于决策特征只涉及单个特征的逻辑判断，导致决策边界是平行于坐标轴的直线，这就限制了决策树对分类边界的表达能力，导致模型的准确性较差



橙色和棕色的线代表单个决策树的分类边界，黑色表示真实的分类边界

随机森林的算法原理

- 随机森林使用并汇总多棵决策树进行预测，所以即使每棵树的决策能力很弱，由它们组合起来形成的随机森林的决策能力也会较强
- 假设使用三棵决策树组合成随机森林，每棵树各不相同且预测结果相互独立，每棵树的预测错误率为40%
- 那么两棵树以及两棵树以上预测错误的概率下降为：三棵全部错误 + 两棵树错误一个棵树正确$= 0.4^3 + 3 * 0.4^2 * (1 - 0.4) = 0.352$

随机森林的算法原理

- 在上述过程中，需要注意一点是错误率降低的前提假设：每棵树各不相同且预测结果相互独立
- 随机森林在构建每棵树的时候，为了保证各棵树之间的独立性，通常会采用两到三层的随机性
  - 随机有放回的抽取样本
  - 随机选取 m 个特征
  - 随机选择特征取值进行分割(不遍历特征所有取值)

随机森林的算法过程

算法 3 随机森林算法

输入: 训练集 D，特征维度为 d，随机选取特征数量 m，决策树学习算法 h.

输出: 集成模型 $H(\boldsymbol{x})$.

1: **for** $t = 1$ to T **do**

2: 使用 Bootstrap 抽样，从训练数据集 D 获得大小为 n 的抽样数据集 D_t;

3: 从 d 个特征中随机选取 m 个特征，基于 D_t 中随机选取的 m 个特征，使用决策树模型学习得到一棵决策树 $h_t(\boldsymbol{x})$;

4: 如果是回归问题，则输出集成模型: $H(\boldsymbol{x}) = \frac{1}{T}(\sum_{t=1}^{T} h_t(\boldsymbol{x}))$;

5: 如果是分类问题，则输出集成模型: $H(\boldsymbol{x}) = \text{majority\_vote}(\{h_t(\boldsymbol{x})\}_{t=1}^{T})$.

随机选取多少个特征?

- 大的 m 值可以使得决策树的准确率提高，但是多棵决策树之间的相关性却很高
- 方法一：交叉验证方法选取最佳 m 值
- 方法二：经验设置。通常选取 $m = \log_2 d + 1$，其中d为特征维度

随机森林的性能评估

- 分类间隔 (margin)：正确分类某样本的决策树的比例减去错误分类样本决策树的比例
  - 假设对样本 A 有75%的树分类正确， 那么分类间隔就是75% - 25% = 50
  - 通过平均随机森林在各个样本上的分类间隔得到随机森林的分类间隔
  - 实际中，我们希望分类间隔越大越好，因为大的分类间隔表示我们的分类效果比较稳定，泛化效果更好

随机森林的性能评估

- 袋外误差 (Out-Of-Bag Error, OOB)：随机森林对袋外样本的预测错误率
- 袋外样本：对于每一棵树而言，没有被抽样到训练集中的样本
- 计算方式：
  - 对每个样本，计算把该样本作为袋外样本的树对该样本的分类情况
  - 以简单多数投票作为该样本的分类结果
  - 以误分样本个数占样本总数的比率作为随机森林的袋外误差

随机森林与特征选择

- 随机森林能够给出特征的重要性度量，帮助进行特征选择
  - 计算特征的平均信息增益大小。

    训练决策树时，可以算出每个特征在每棵树中有多大的信息增益，算出每个特征在随机森林中的平均信息增益，作为该特征的重要性
  - 计算每个特征对模型准确率的影响。

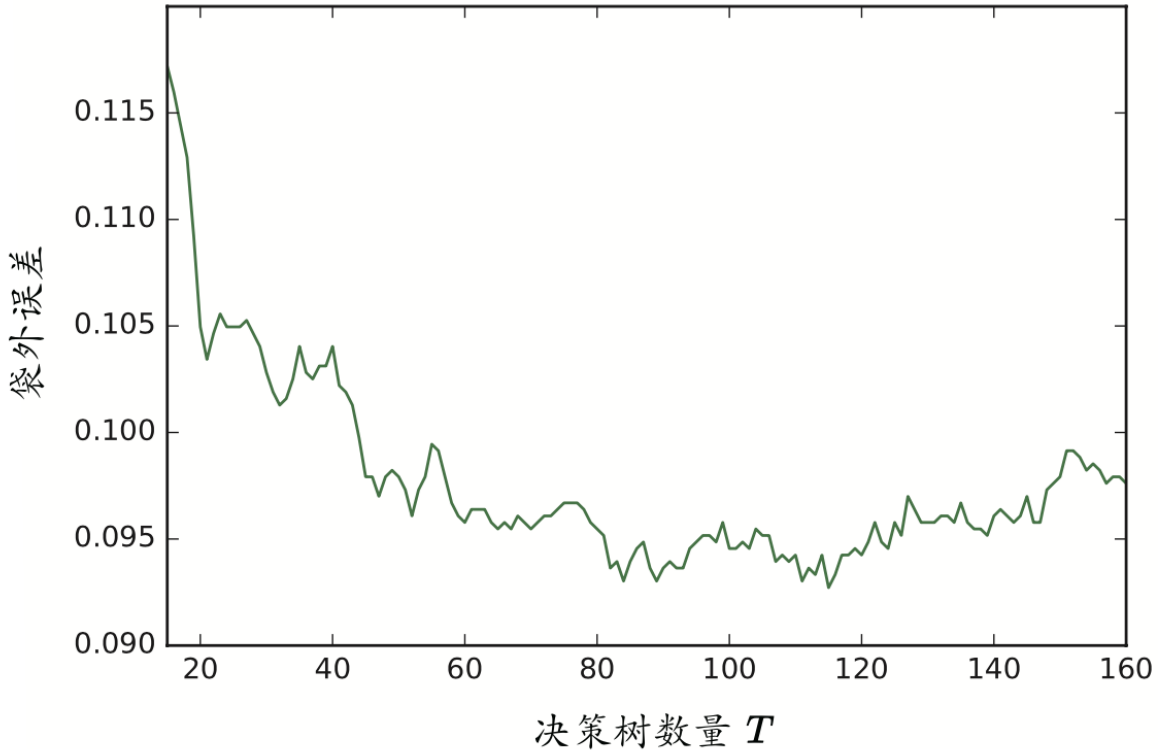
    通过打乱样本中某一特征的特征值顺序，产生新样本。新样本放入建立好的随机森林模型计算准确率，对于重要的特征来说，打乱顺序会极大的降低模型的准确率 。

算法分析

- 假设单棵决策树的方差为 $\sigma^2(\boldsymbol{x})$，任意两颗决策树之间的相关性为 $\rho(\boldsymbol{x})$，随机森林一共由 T 棵决策树组成，则集成的随机森林模型的方差 $\mathrm{var}(\boldsymbol{x})$ 为

$$\mathrm{var}(\boldsymbol{x}) = \rho(\boldsymbol{x})\sigma^2(\boldsymbol{x}) + \frac{1-\rho(\boldsymbol{x})}{T}\sigma^2(\boldsymbol{x})$$

随机　　森林



随机森林的优缺点

- 优点：
  - 能够处理很高维度的数据，并且不用做特征选择
  - 对特征之间存在的多重共线性不敏感，并且能够在一定程度上处理缺失数据和不均衡数据
  - 在训练完后能够给出哪些特征比较重要
  - 容易做成并行化方法

随机森林的优缺点

- 缺点：
  - 处理噪音较大的小样本和低维数据集的问题上会过度拟合
  - 相对于决策树，预测速度较慢
  - 相对于决策树，模型可解释性较差

随机森林的其他功能

- 计算样本的相似度
  - 在建立随机森林的时候，记录样本两两之间出现在同一叶子节点的次数，生成相似性矩阵 (proximity matrix)；
  - 如果越多的出现在同一叶节点，说明这两个样本越相似；如果越少的出现在同一叶节点，说明两个样本差异越大。

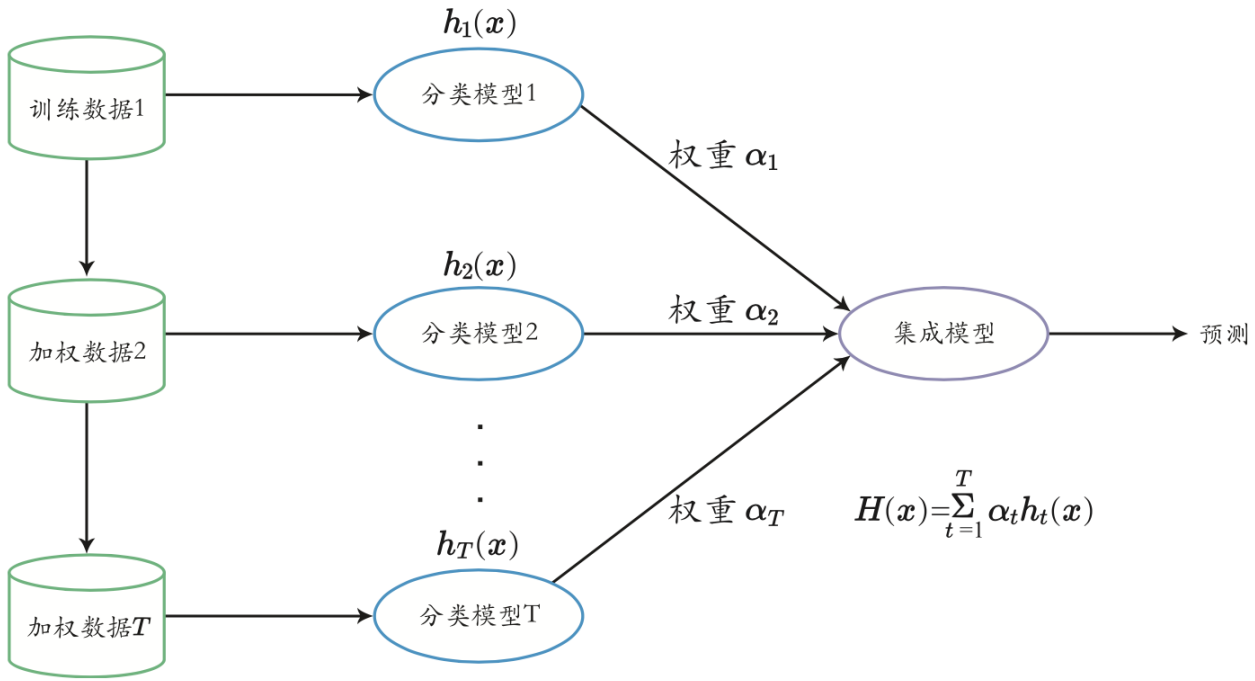
目录

Boosting

- 以分类为例，Boosting假设基模型是弱分类器，即分类性能比随机猜测稍好的分类器
- 目标是通过一定的策略提升弱分类器效果，得到一个综合的强分类器
- 基本思想:
  - 从训练数据集中训练得到一个弱分类器，下一个弱分类器基于前一个弱分类器的表现，重点关注前一个弱分类分错的样本，尽量将前一个弱分类器分错的样本分对
- 不同于Bagging并行的方式，Boosting是以串行方式训练获得强分类器

Boosting

- 在Boosting方法中，权重赋给每一个训练样本，迭代地学习 T 个分类器
- 学习得到分类器$\mathbf{h_t}$之后，更新权重，使得其后的分类器$\mathbf{h_{t+1}}$更关注$\mathbf{h_t}$误分类的训练样本
- 最终的分类器 H集成每个个体分类器的预测结果
- 需要预先设置弱分类器误差的上界



AdaBoost

- AdaBoost (Adaptive Boosting)是最有代表性的Boosting算法；
- AdaBoost的核心思想是利用同一训练样本的不同加权版本，训练一组弱分类器，然后把这些弱分类器以加权的形式集成起来，形成一个最终的强分类器：
  - 在每一步迭代过程中，会给训练集中的样本赋予一个权重 $(w_1, w_2, \ldots, w_n)$；
  - 样本的初始权重都一样，设置为 $1/n$；
  - 在每一步迭代过程中，被当前弱分类器分错的样本的权重会相应得到提高，被当前弱分类器分对的样本的权重则会相应降低；
  - 弱分类器的权重则根据当前分类器的加权错误率来确定。

AdaBoost的特点

- 通过改变样本权重的方式训练新的弱分类器，后一个弱分类器基于前一个分类器的结果来训练；
- 能够自动学习多个弱分类器集成时的分类器权重；
- 不需要预先设置弱分类器误差上界。

AdaBoost算法流程

算法 4 AdaBoost 算法

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n)\}$，弱学习算法 h.

输出： 集成分类器 $H(\boldsymbol{x})$.

1: 初始化样本权重 $\boldsymbol{w}^1$: $w_i^1 = \frac{1}{n}, i = 1, 2, ..., n$; ▷ 初始化数据具有相同的权重

2: **for** $t = 1$ to T **do**

3: 使用训练数据集 D 及权重 $\boldsymbol{w}^t$，训练一个分类器 $h_t(\boldsymbol{x})$;

4: 计算当前弱分类器的误差:

$$\epsilon_t = \frac{\sum_{i=1}^{n} w_i^t I(y_i \neq h_t(\boldsymbol{x}_i))}{\sum_{i=1}^{n} w_i^t};$$

5: 根据当前弱分类器的误差，计算权重:

$$\alpha_t = \frac{1}{2} \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right);$$

6: 按如下方法更新样本权重为 $\boldsymbol{w}^{t+1}$: ▷ 提高 $h_t(\boldsymbol{x})$ 分错样本的权重

$$w_i^{t+1} \leftarrow \frac{w_i^t \cdot e^{-\alpha_t y_i h_t(\boldsymbol{x}_i)}}{Z_t},$$

其中 Z_t 为归一化因子;

7: 输出最终的分类器: $H(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$.

分类器权重更新

- AdaBoost中每一个弱分类器的权重α_t是根据该分类器在当前权重的训练样本下的分类效果来确定的
- 弱分类器的分类效果越好，该弱分类器的权重越高
- 具体而言，弱分类器的分类效果使用加权错误率ϵ_t来进行评估，ϵ_t的取值等价于弱分类器错误分类的样本的权重之和与训练集所有样本总权重之比

$$\epsilon_t = \frac{\sum_{i=1}^{n} w_i^t I(y_i \neq h_t(\boldsymbol{x}_i))}{\sum_{i=1}^{n} w_i^t}$$

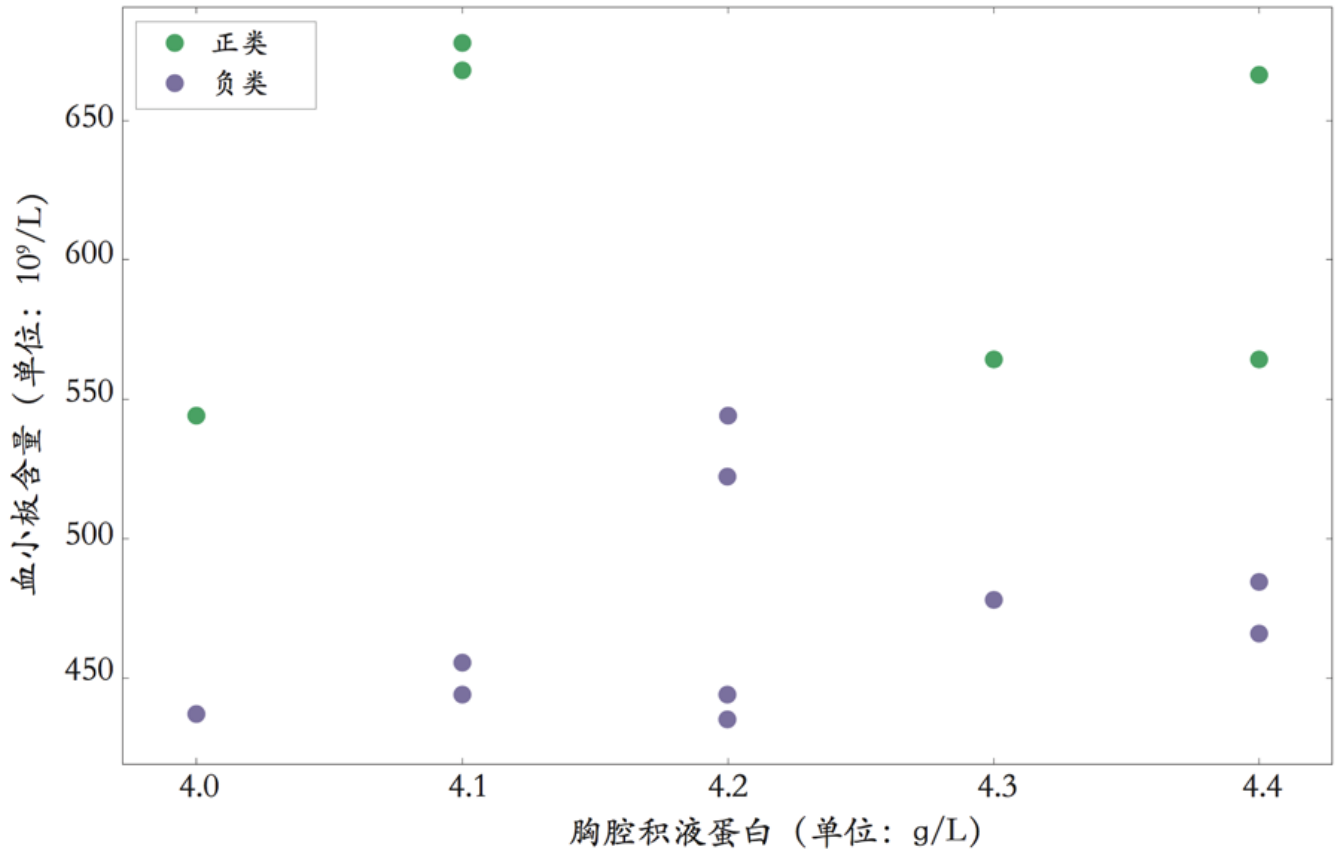
样本权重更新

- 初始阶段，训练集中每一个样本具有相等的权重 $\frac{1}{n}$
- 当弱分类器对样本正确分类时，$y_i h_t(x_i) = 1$，样本权重乘以小于1的因子 $\exp(-\alpha_t)$，相当于降低了正确分类的样本的权重
- 当弱分类器对样本错误分类时，$y_i h_t(x_i) = -1$，样本权重乘以大于1的因子 $\exp(\alpha_t)$，相当于提高了错误样本的权重

$$w_i^{t+1} \leftarrow \frac{w_i^t \cdot e^{-\alpha_t y_i h_t(x_i)}}{Z_t}, \qquad \alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right);$$

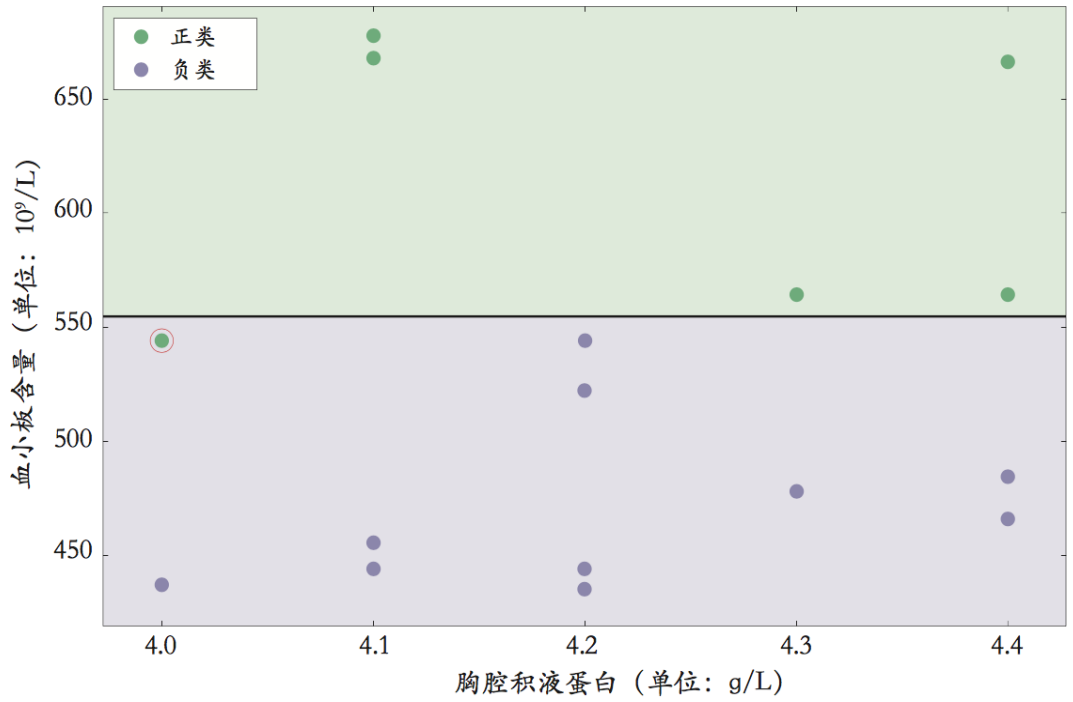
AdaBoost示例

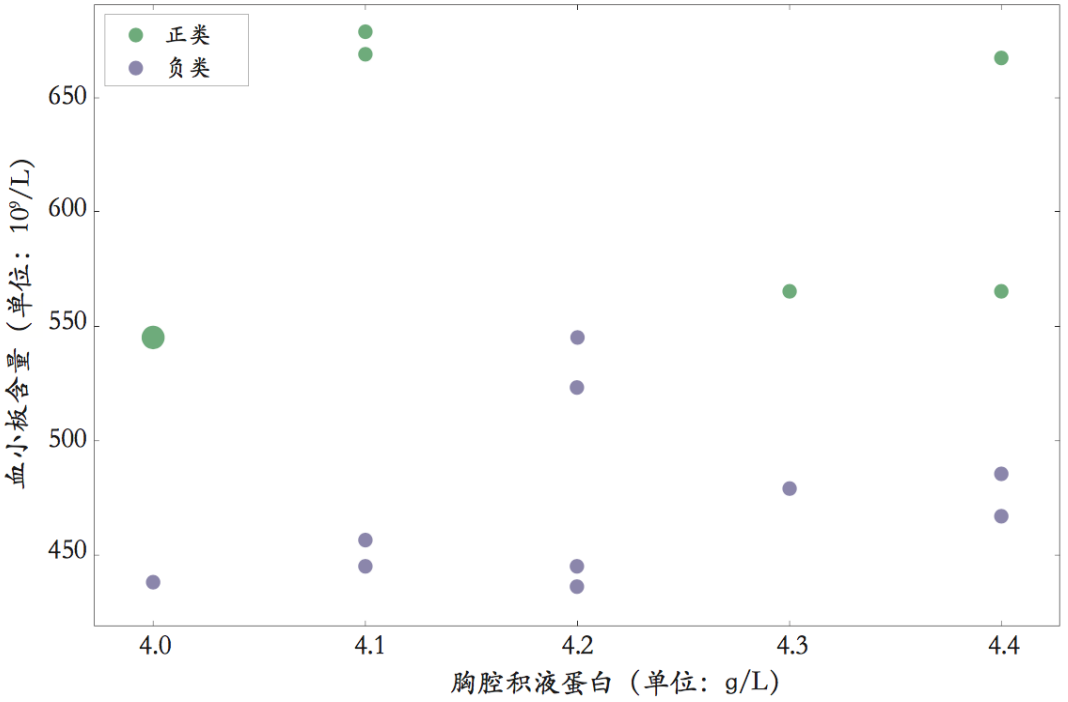
- 初始样本状态，使用只包含一个特征的决策树作为基分类器



算法举例

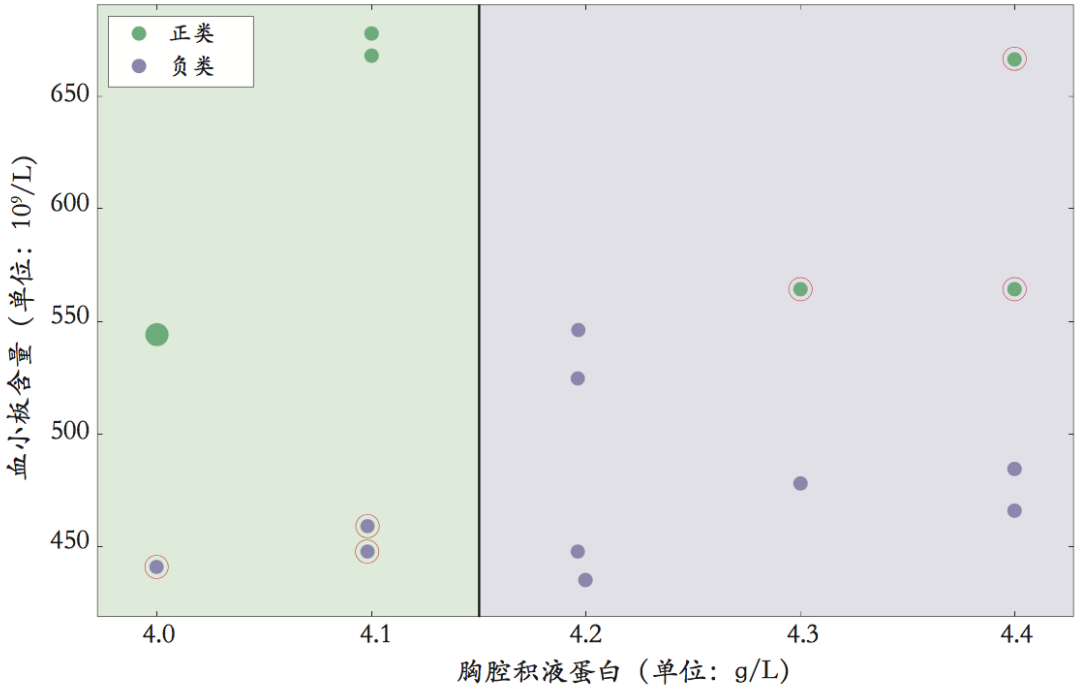
- 第1轮学习到第一个基分类器 $h_1(x)$，分类器错误率 ε_1 为0.11，权重 α_1 为2.08，调整样本权重

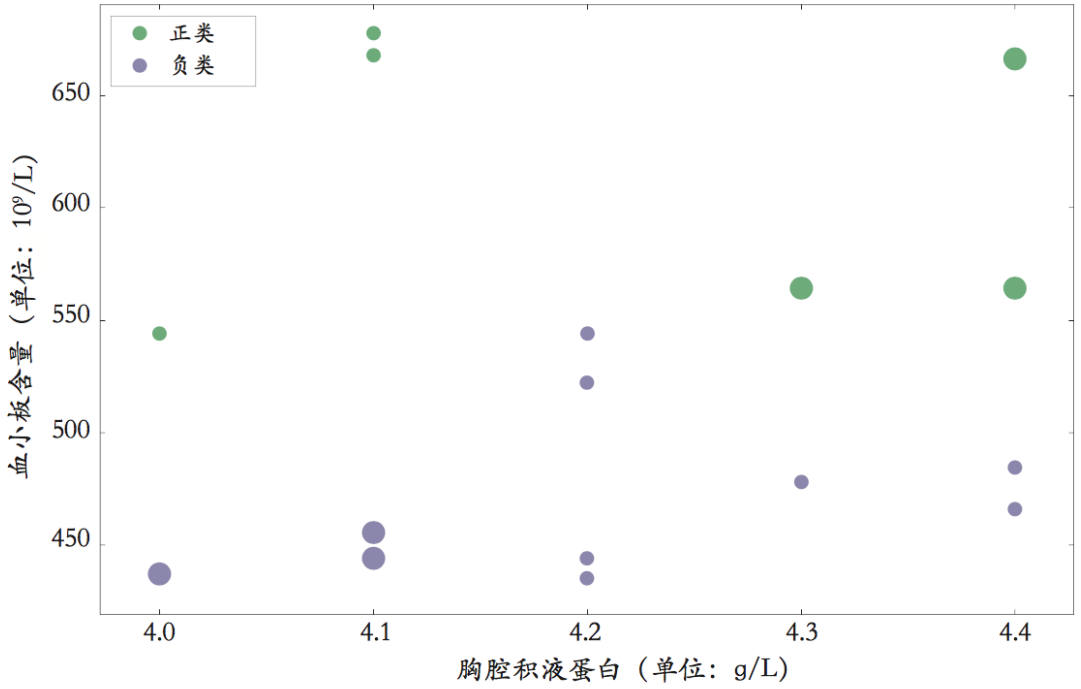




算法举例

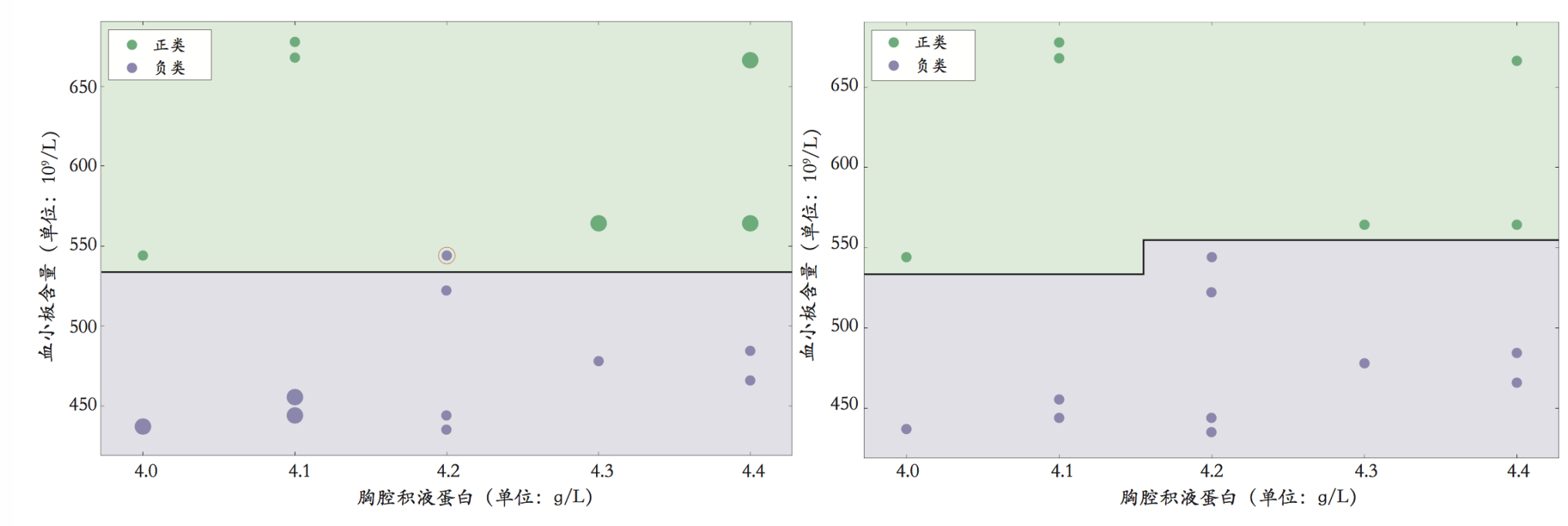
- 第2轮学习到第一个基分类器 $h_2(x)$，分类器错误率 ε_2为0.19，权重 α_2为1.47，调整样本权重





算法举例

- 第3轮学习到第一个基分类器 $h_3(x)$，分类器错误率 ε_3为0.17，权重 α_3为1.56



- 由于综合的错误率达到 0，停止迭代，最终的分类器为

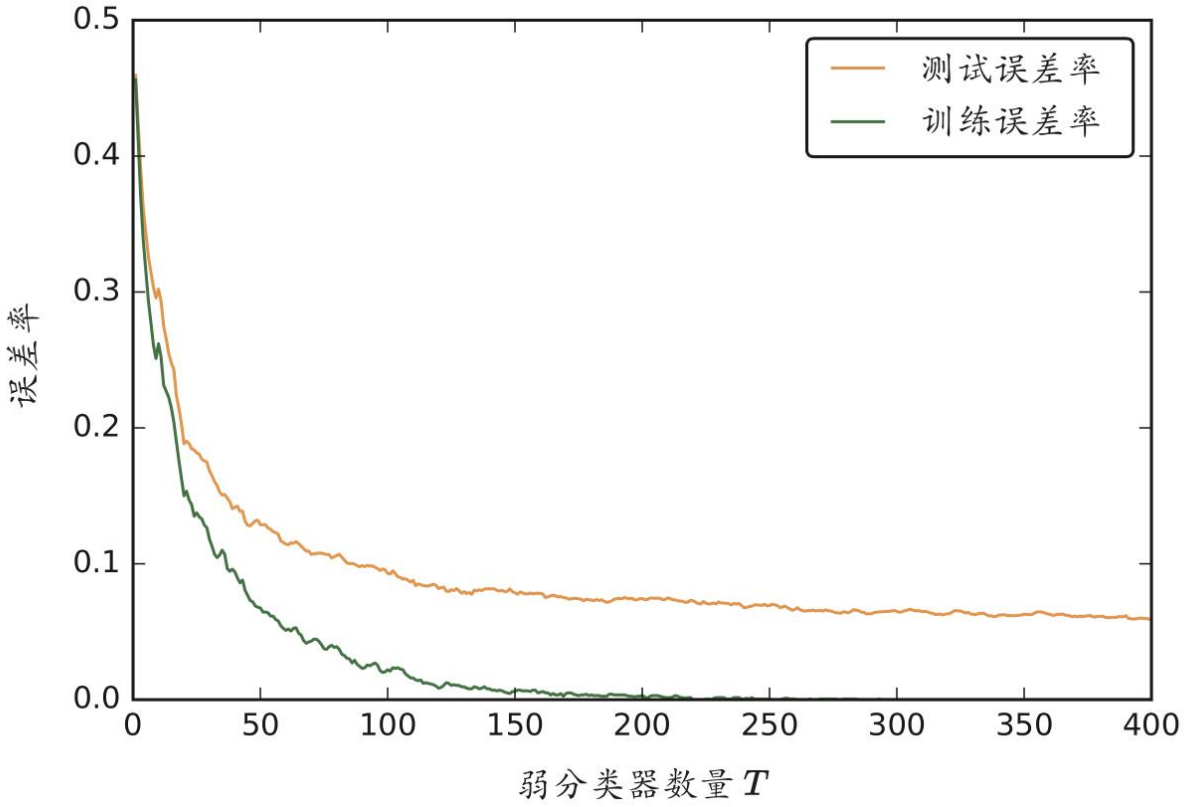
$$H(x) = \text{sign}(2.08h_1(x) + 1.47h_2(x) + 1.56h_3(x))$$

误差分析

- 泛化误差是对模型的测试集进行误差评估时使用的，泛化误差的计算公式为

$$\text{Error}_{\text{test}} = \mathbb{E}_{\boldsymbol{x},y}(L(y_i, H(\boldsymbol{x}_i)))$$

- 理论上随着弱分类器数目T的增大，泛化误差上界会增大。经验性结果表明，即使T很大也不容易过度拟合。



AdaBoost的优缺点

- 优点
  - 防止过拟合
  - 不仅能减少方差还能减小偏差
  - 除了弱分类器的数目T以外，没有额外的参数需要调参
  - AdaBoost算法提供的是一种框架，弱分类器可以选择决策树或者其他分类算法
- 缺点
  - 由于集成了多个弱分类器，模型的可解释性降低
  - 当弱分类器太复杂或者效果太差时，容易导致过拟合
  - 对于异常值比较敏感

目录