MACHINE LEARNING

机器学习

第八章：集成学习

集成学习

- 个体与集成
- Boosting：串行
  - Adaboost
- Bagging与随机森林：并行
- 结合策略
  - 平均法、投票法、学习法
- 多样性
  - 误差-分歧分解、多样性度量、多样性扰动

个体与集成

集成学习

- 通过构建并结合多个学习器，来完成学习任务，提升性能

一般结构

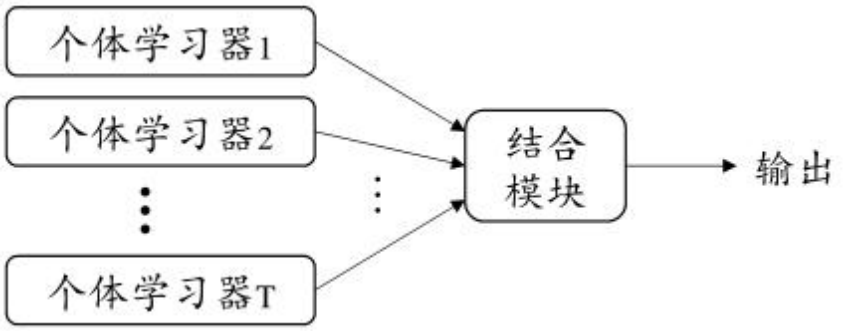
1. 先产生一组“个体学习器”；
2. 再用某种策略，将它们结合起来。

✓ 个体学习器

- 通常 由一个现有的学习算法，从训练数据产生
- 例如， 决策树算法、BP 神经网络算法等

个体与集成

- 集成学习
  - ✓ 集成：
    1. 同质集成
       - 只包含同种类型的个体学习器，个体学习器，称为基学习器。
       - 如，“决策树集成”中全是决策树，”神经网络集成”中全是神经网络
    2. 异质集成
       - 包含不同类型的个体学习器，称为组件学习器。
       - 如，同时包含：决策树和神经网络。



同质集成学习---类别

- 同质集成学习
  - 个体学习器间，存在强依赖关系？
  - 大致可分为两大类：
    1. 个体学习器间　存在强依赖关系、必须串行生成的序列化方法
       - ✓ 代表：Boosting
    2. 个体学习器间　不存在强依赖关系、可同时生成的并行化方法
       - ✓ 代表：Bagging 、随机森林

个体与集成 --- 简单分析（实例分析）

- 集成学习
  - 把多个学习器结合起来，如何能获得：比最好的单一学习器更好的性能呢?

例，二分类问题：3个分类器在三个样本中的表现如下
✓ 分类正确，×分类错误，集成的结果通过投票产生。

	测试例1	测试例2	测试例3
h_1	✓	✓	×
h_2	×	✓	✓
h_3	✓	×	✓
集群	✓	✓	✓

(a) 集群提升性能

	测试例1	测试例2	测试例3
h_1	✓	✓	×
h_2	✓	✓	×
h_3	✓	✓	×
集群	✓	✓	×

(b) 集群不起作用

	测试例1	测试例2	测试例3
h_1	✓	×	×
h_2	×	✓	×
h_3	×	×	✓
集群	×	×	×

(c) 集群起负作用

- 要获得好的集成，个体学习器应好而不同：
  - 个体学习器，要有一定的“准确性”，即学习器不能太坏
  - 个体学习器，要有”多样性”，即学习器间具有差异

个体与集成 - 简单分析（概率分析）

- 考虑二分类问题，假设基分类器的错误率为：

$$P\left(h_i\left(\boldsymbol{x}\right) \neq f\left(\boldsymbol{x}\right)\right) = \epsilon$$

- 假设集成通过简单投票法结合T个分类器，若有超过半数的基分类器正确则分类就正确

$$H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{T} h_i\left(\boldsymbol{x}\right)\right)$$

个体与集成 - 简单分析（概率分析）

- 假设基分类器的错误率相互独立，则由Hoeffding不等式可得集成的错误率为：

$$P(H(\boldsymbol{x}) \neq f(\boldsymbol{x})) = \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k}$$

$$\leq \exp\left(-\frac{1}{2} T (1-2\epsilon)^2\right)$$

- 上式显示，在一定条件下，随着集成分类器数目的增加，集成的错误率将指数级下降，最终趋向于0

个体与集成 – 简单分析

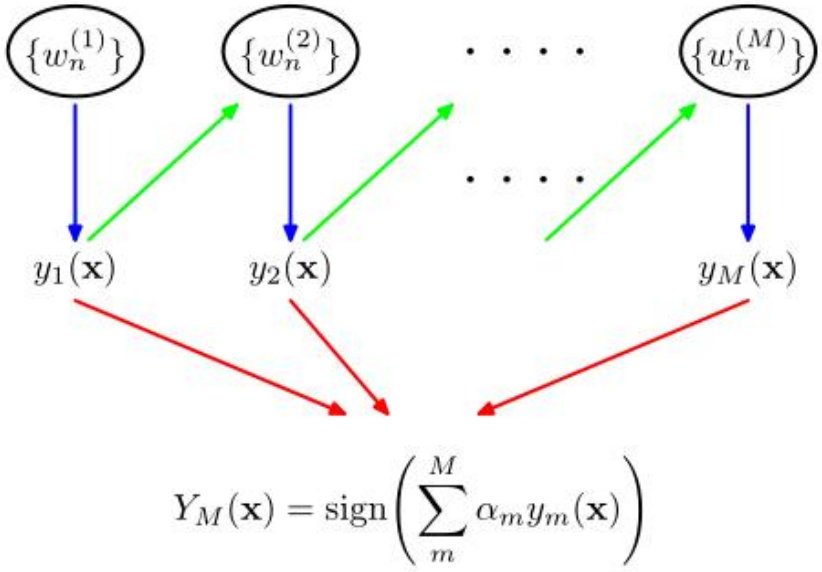
- 上面的分析有一个关键假设：基学习器的误差相互独立
- 现实任务中，个体学习器是为解决同一个问题训练出来的，显然不可能互相独立
- 事实上，个体学习器的“准确性”和“多样性”本身就存在冲突
- 如何产生“好而不同”的个体学习器是集成学习研究的核心

集成学习

- 个体与集成
- Boosting：串行
  - Adaboost
- Bagging与随机森林：并行
- 结合策略
  - 平均法、投票法、学习法
- 多样性
  - 误差-分歧分解、多样性度量、多样性扰动

Boosting

- 个体学习器存在强依赖关系
- 串行生成
  - 逐个生成基学习器
  - 每次：调整训练数据的样本分布



工作机制

1. 先，从初始训练集，训练出一个基学习器；
2. 再，根据基学习器的表现，对训练样本分布，进行调整，使得先前基学习器，做错的训练样本，在后续受到更多关注；
3. 然后，基于调整后的样本分布，训练下一个基学习器

如此重复2、3进行，逐个生成基学习器，直至基学习器数目，达到事先指定的值T ， 最终将这 T 个基学习器，进行加权结合。

- Boosting族算法最著名的代表：AdaBoost

Boosting - Boosting算法

Input: Sample distribution $\mathcal{D}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

Process:

1. $\mathcal{D}_1 = \mathcal{D}$. % Initialize distribution
2. **for** $t = 1, \ldots, T$:
3. $h_t = \mathfrak{L}(\mathcal{D}_t)$; % Train a weak learner from distribution $\mathcal{D}_t$
4. $\epsilon_t = P_{\boldsymbol{x} \sim D_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; % Evaluate the error of h_t
5. $\mathcal{D}_{t+1} = Adjust\_Distribution(\mathcal{D}_t, \epsilon_t)$
6. **end**

Output: $H(\boldsymbol{x}) = Combine\_Outputs(\{h_1(\boldsymbol{x}), \ldots, h_t(\boldsymbol{x})\})$

- Boosting族算法最著名的代表是AdaBoost

Boosting – AdaBoost算法

- 基学习器的线性组合

$$H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$$

- 过程
  - 初始化样本权值分布 $\mathcal{D}_1(\boldsymbol{x}) = 1/m.$
  - 基于分布$\boldsymbol{D}_t$从数据集D中训练出分类器$\boldsymbol{h}_t$
  - 估计$\boldsymbol{h}_t$的误差
  - 确定分类器的权重$\boldsymbol{\alpha}_t$
  - 更新样本权值分布，其中$\boldsymbol{Z}_t$是规范化因子，以确保$\boldsymbol{D}_{t+1}$ 是一个分布

Boosting – AdaBoost算法

在 AdaBoost算法 中, 第一个基分类器 h_1 是通过直接将基学习算法用于初始数据分布而得; 此后迭代地生成 h_t 和 α_t, 当基分类器 h_t 基于分布 $\mathcal{D}_t$ 产生后, 该基分类器的权重 α_t 应使得 $\alpha_t h_t$ 最小化指数损失函数

输入: 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

过程:

1: $\mathcal{D}_1(\boldsymbol{x}) = 1/m$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $\quad h_t = \mathfrak{L}(D, \mathcal{D}_t)$;
4: $\quad \epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$;
5: $\quad$ **if** $\epsilon_t > 0.5$ **then break**
6: $\quad \alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$;
7: $\quad \mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases}$
$\qquad = \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}$
8: **end for**

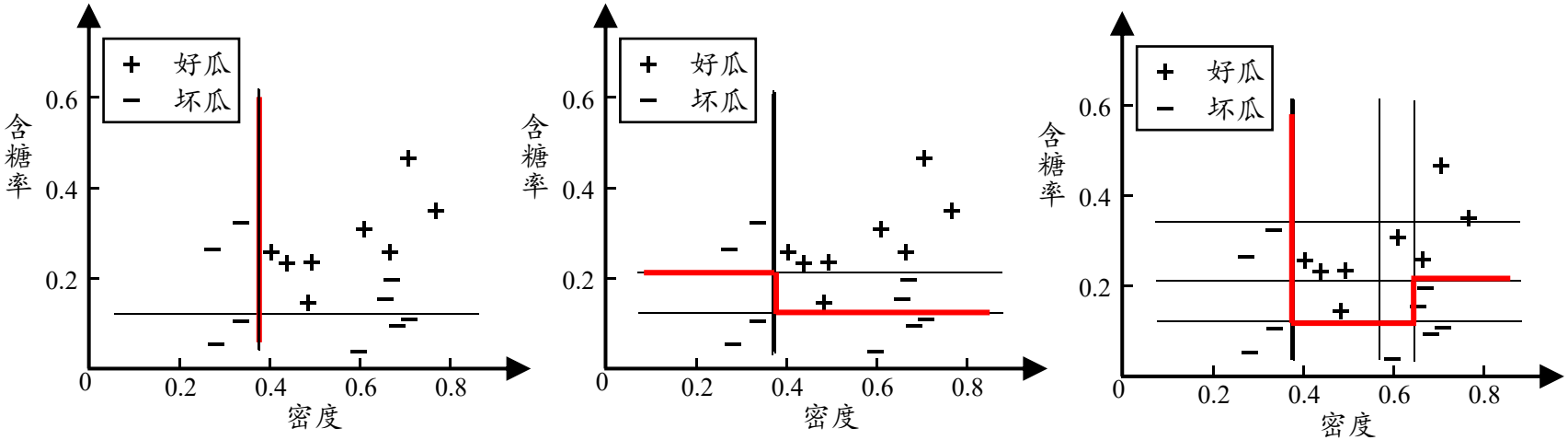
输出: $H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) \right)$

Boosting – AdaBoost注意事项

- Boosting算法要求基学习器能对特定的数据分布进行学习
  - 重赋权法：在训练过程的每一轮中，根据样本分布为每个训练样本重新赋予一个权重。
  - 重采样法：对无法接受带权样本的基学习算法，则可通过“重采样法”处理，即在每一轮学习中，根据样本分布对训练集重新进行采样，再用重采样而得得样本集对基学习器进行训练

Boosting – AdaBoost实验

- 以决策树桩为基学习器，在表4.5的西瓜数据集3.0α上运行AdaBoost，不同规模的集成基学习器所对应的分类边界



(a) 3个基学习器　(b) 5个基学习器　(c) 11个基学习器

Boosting

- 优点

  偏差-方差的角度：偏差（算法效率）；方差（算法稳定性）

  Boosting 算法关注降低偏差（算法精度），可对泛化性能相当弱的学习器，构造出很强的集成

- 缺点

  在训练的每一轮，都要检查当前生成的基学习器，是否满足基本条件？一旦条件不满足，则当前基学习器即被抛弃，且学习过程停止。

  因此，初始设置的学习轮数T 也许遥远未达到，学习过程却停止了，导致最终集成中只包含很少的基学习器，算法性能不佳。

集成学习

- 个体与集成
- Boosting：串行
  - Adaboost
- Bagging与随机森林：
  - 个体学习器间不存在强依赖关系、基于自助采样法
  - 可同时生成的并行化方法
- 结合策略
  - 平均法、投票法、学习法
- 多样性
  - 误差-分歧分解、多样性度量、多样性扰动

Bagging与随机森林

- 欲得到泛化性能强的集成，集成中的个体学习器应尽可能相互独立，在现实任务中无法做到

- 可设法使基学习器，尽可能具有较大的差异。
- 给定训练数据集，一种可能的做法：
  1. 对训练样本进行采样，产生出若干个不同的子集。
  2. 再从每个数据子集中，训练出一个基学习器。
  - ✓ 由于训练数据不同，获得的基学习器，可望具有比较大的差异。

Bagging与随机森林

- 为获得好的集成，同时还希望：个体学习器不能太差
  - ✓ 如果采样出的每个子集都完全不同，则每个基学习器只用到了一小部分训练数据，甚至不足以进行有效学习
  - ✓ 这显然无法确保：产生出比较好的基学习器。
- 为解决这个问题
  - 可考虑使用相互有交叠的采样子集。

Bagging

- 给定训练数据集，交叠的采样子集
- 基学习器：决策树
- 构造训练数据--自助采样法
  - 每个基学习器，使用初始训练集中约63.2% 的样本，进行训练
  - 剩下约36.8% 的样本（包外样本），可作验证集，提升泛化性能
    - ✓ 基学习器是决策树：包外样本，可以辅助剪枝
    - ✓ 基学习器是神经网络：包外样本，可辅助早期停止，减小过拟合风险

Bagging

- 构造训练数据--自助采样法
- 基本流程
  - 首先，采样出T 个含m 个训练样本的采样集
  - 然后，基于每个采样集，训练出一个基学习器
  - 最后，将这些基学习器，进行结合
- 预测输出
  - 分类任务：简单投票法。
  - 回归任务：简单平均法。

Bagging与随机森林 - Bagging算法

输入: 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

过程:

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$
3: **end for**

$\mathcal{D}_{bs}$ 是自助采样产生的样本分布.

输出: $H(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$

Bagging

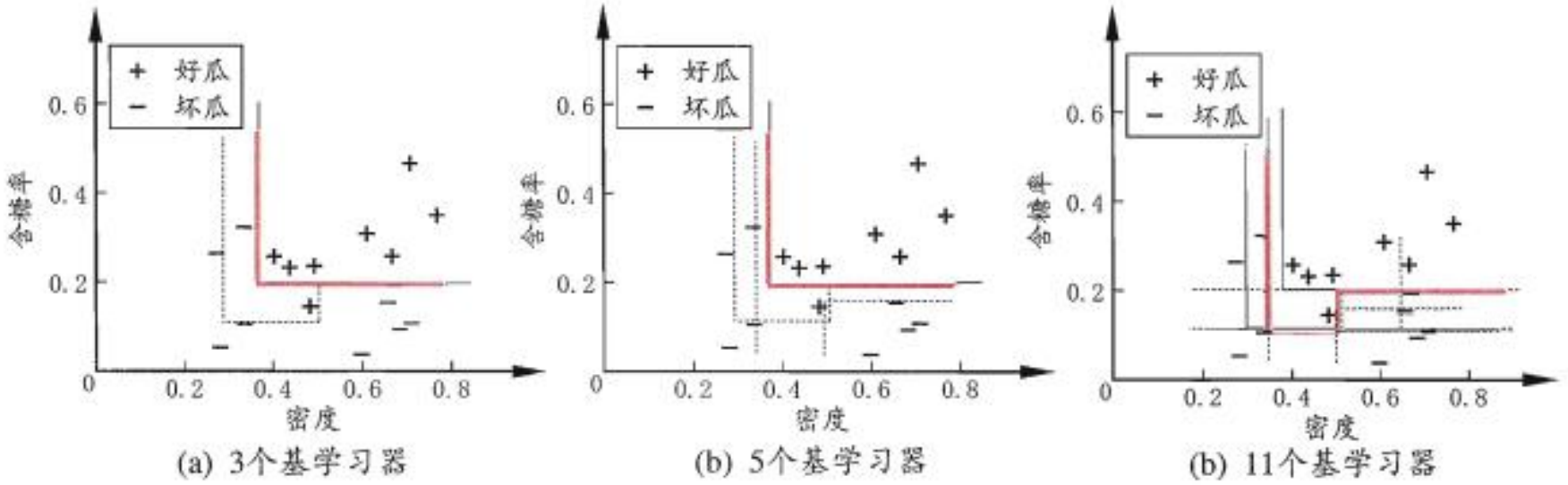
Wolpert and Macready, 1999]. 为此需记录每个基学习器所使用的训练样本. 不妨令 D_t 表示 h_t 实际使用的训练样本集, 令 $H^{oob}(\boldsymbol{x})$ 表示对样本 $\boldsymbol{x}$ 的包外预测, 即仅考虑那些未使用 $\boldsymbol{x}$ 训练的基学习器在 $\boldsymbol{x}$ 上的预测, 有

$$H^{oob}(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y) \cdot \mathbb{I}(\boldsymbol{x} \notin D_t) \ , \tag{8.20}$$

则 Bagging 泛化误差的包外估计为

$$\epsilon^{oob} = \frac{1}{|D|} \sum_{(\boldsymbol{x},y) \in D} \mathbb{I}(H^{oob}(\boldsymbol{x}) \neq y) \ . \tag{8.21}$$

Bagging



- 优点
  - 从偏差-方差的角度：偏差（算法效率）；方差（算法稳定性）

    Bagging 算法关注降低方差（算法稳定性），在不剪枝的决策树、神经网络等易受样本影响的学习器上，效果更好
  - 标准AdaBoost ：只适用于二分类任务

    Bagging ：可直接用于多分类、回归等任务

随机森林

- 随机森林
  - Bagging的一个扩展变种
  - Bagging：以决策树为基学习器，构建Bagging 集成
  - 随机森林：在决策树的训练过程中，引入了随机选择

随机森林

- 随机森林
  1. 采样的随机性 ：基于自助采样法，构造训练数据
  2. 属性选择的随机性

➢ 传统决策树

✓ 选择划分属性时，在当前结点的属性集合（假定有d个属性）中，选择一个最优属性

随机森林

- 随机森林
  1. 采样的随机性 ：基于自助采样法，构造训练数据
  2. 属性选择的随机性

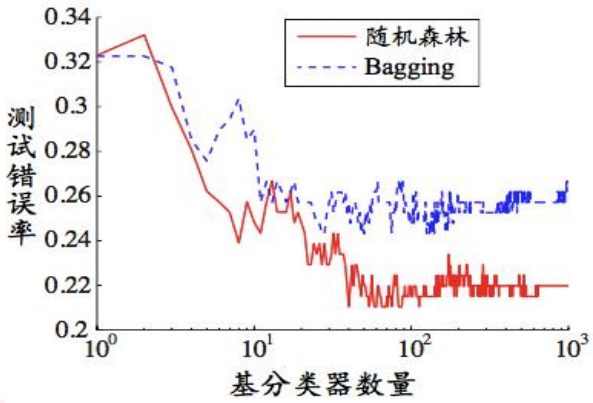
     对基决策树的每个结点，
     1. 首先，从该结点的属性集合中，随机选择一个包含k个属性的子集
        - 参数k 控制了随机性的引入程度

          若 $k = d$ ，基决策树的构建与传统决策树相同；

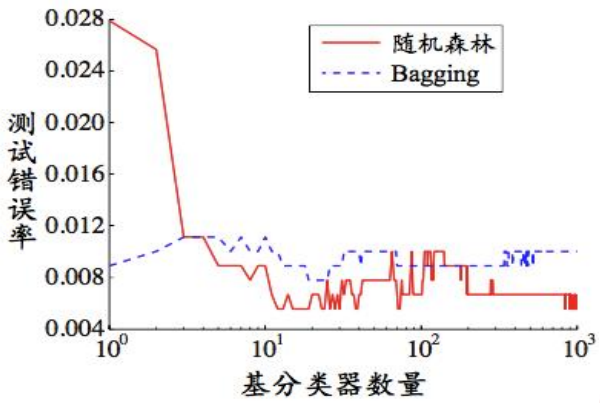
          若 $k = 1$ ，随机选择一个属性，用于划分
     2. 然后，从这个子集中，选择一个最优属性，用于划分。

随机森林 VS Bagging

- 基学习器多样性
  - Bagging ：通过自助采样法，实现样本扰动，构建多样性
  - 随机森林：通过样本扰动+属性扰动，构建多样性
- 收敛性： 相似
- 泛化性能：随机森林更好
  - 由于引入属性扰动，个体学习器的性能，往往有所降低
  - 随机森林的初始性能相对较差， 特别是只包含一个基学习器时
  - 随机森林通常会收敛到 更低泛化误差



(a) glass 数据集



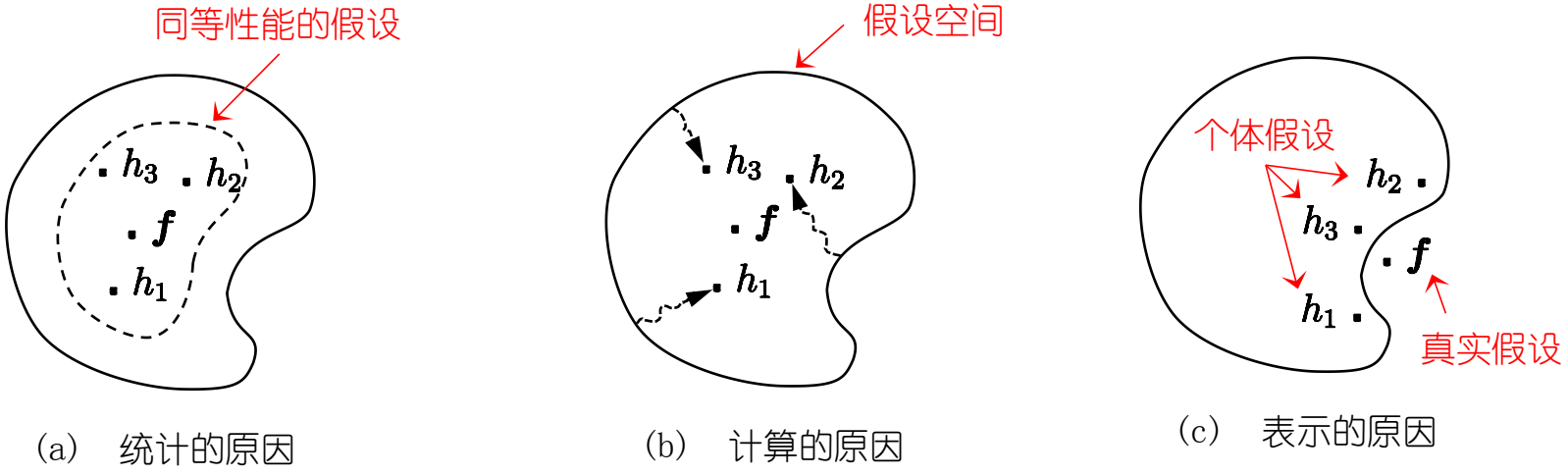
(b) auto-mpg 数据集

集成学习

- 个体与集成
- Boosting：串行
  - Adaboost
- Bagging与随机森林：并行
- 结合策略
  - 平均法、投票法、学习法
- 多样性
  - 误差-分歧分解、多样性度量、多样性扰动

结合策略

- 学习器的组合，可以从三个方面带来好处

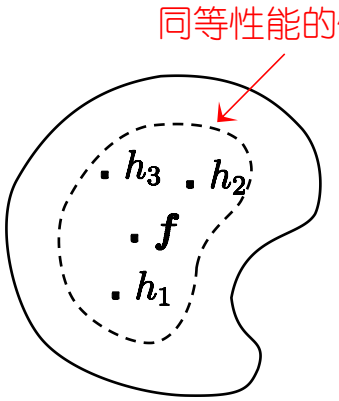


(a) 统计的原因　(b) 计算的原因　(c) 表示的原因

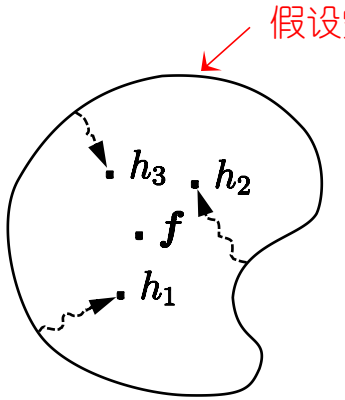
- 统计方面
  - 由于学习任务的假设空间往往很大，可能有多个假设在训练集上达到同等性能。
  - 若使用单学习器，可能因误选，而导致泛化性能不佳。
  - 结合多个学习器，则会减小这一风险

结合策略

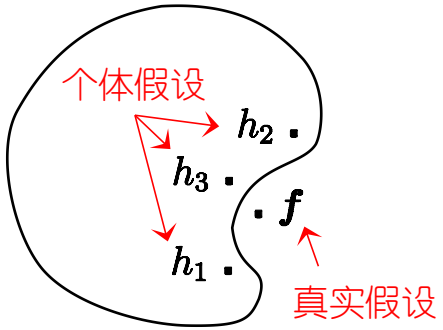
- 学习器的组合，可以从三个方面带来好处



(a) 统计的原因



(b) 计算的原因



(c) 表示的原因

- 统计方面

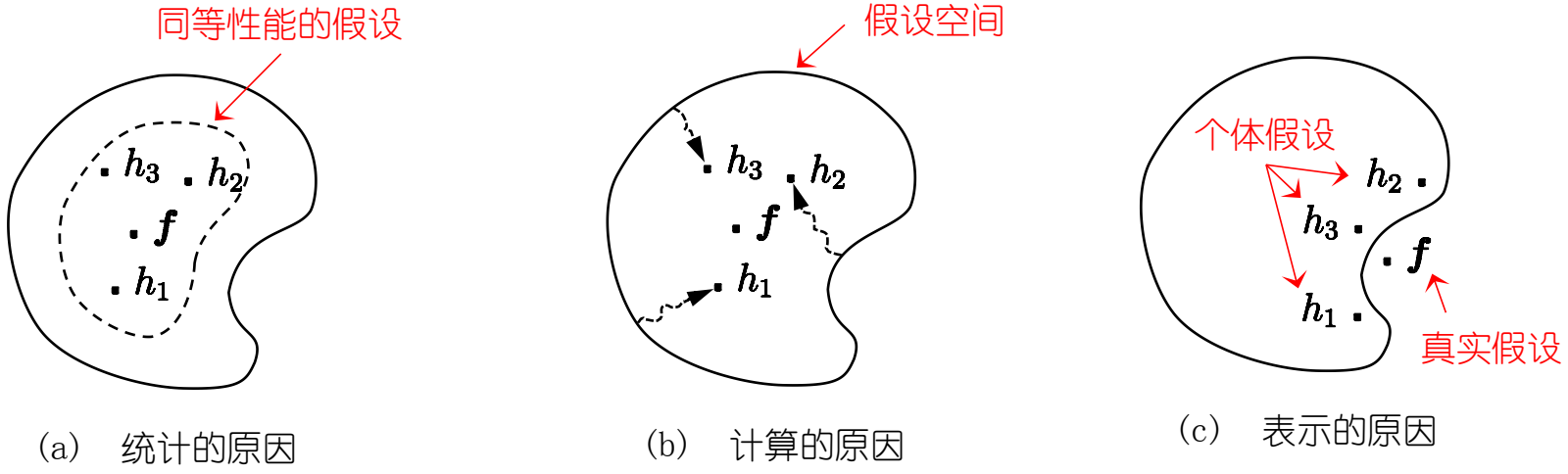
  减少 多个同等性能假设的误选择。

- 计算方面

  降低 陷入糟糕局部极小点的风险。

结合策略

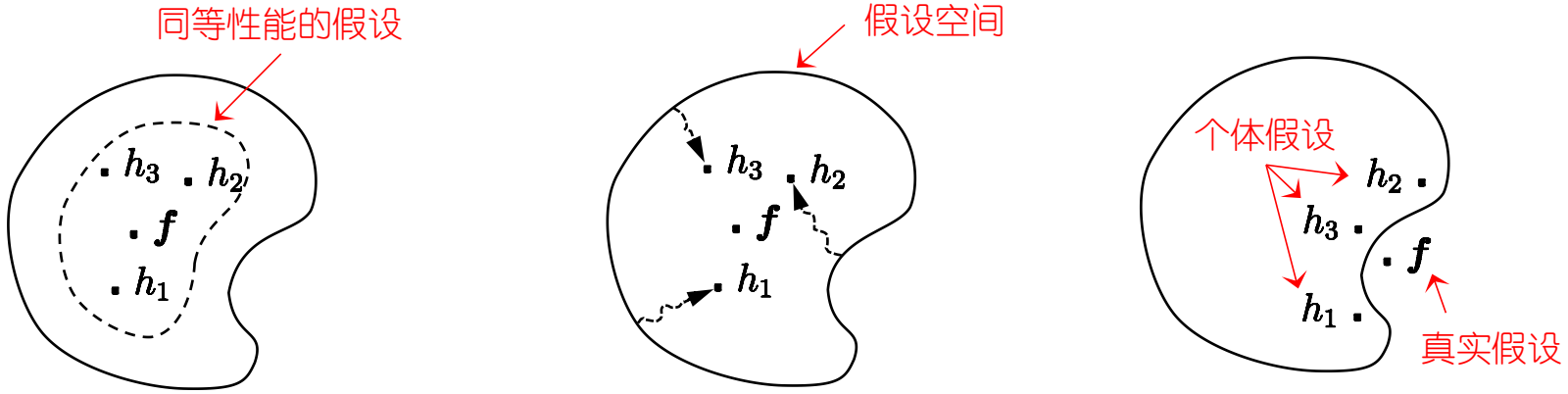
- 学习器的组合，可以从三个方面带来好处



- 统计方面：减少 多个同等性能假设的误选择。
- 计算方面：降低 陷入糟糕局部极小点的风险。
- 表示方面
  - 某些学习任务的真实假设，可能不在当前算法所考虑的假设空间中。
  - 若使用单学习器，肯定无效。
  - 通过结合多个学习器，相应的假设空间有所扩大，有可能学得更好的近似处

结合策略

- 学习器的组合，可以从三个方面带来好处



(a) 统计的原因　(b) 计算的原因　(c) 表示的原因

- 统计方面
  - ✓ 减少 多个同等性能假设的误选择。
- 计算方面
  - ✓ 降低 陷入糟糕局部极小点的风险。
- 表示方面
  - ✓ 扩大 假设空间，有可能学得更好的近似处。

结合策略 - 平均法

- 集成包含T 个基学习器$\{h_1, h_2, \cdots, h_T\}$，其中$h_i$ 在示例 x上的输出为$h_i(x)$。
- 常见 平均法 策略：

1. 简单平均法 $$H(\boldsymbol{x}) = \frac{1}{T}\sum_{i=1}^{T} h_i(\boldsymbol{x}).$$

2. 加权平均法 $$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}), \quad w_i \geq 0 \ \text{and} \ \sum_{i=1}^{T} w_i = 1.$$

➢ 加权平均法，未必一定优于简单平均法

- 加权平均法的权重，一般是从训练数据中学习而得，现实任务中的训练样本，通常不充分或存在噪声，这将使得学出的权重不完全可靠。
- 规模比较大的集成，要学习的权重比较多，较容易导致过拟合。

➢ 个体学习器性能相差较大时，宜使用加权平均法

➢ 个体学习器性能相近时，宜使用简单平均法。

结合策略 - 投票法

- 分类任务

学习器h_i，从类别标记集合$\{c_1, c_2, \cdots, c_N\}$中，预测出一个标记

- 最常见的结合策略：投票法

h_i在样本m上的预测输出为一个N维向量（$h_i^1(x)$；$h_i^2(x)$；…；$h_i^N(x)$）

其中，$h_i^j(x)$是h_i在类别标记c_j上的输出

1. 绝对多数投票法

若某标记得票过半数，则预测为该标记

2. 相对多数投票法

若同时有多个标记获最高票，则从中随机选取一个

3. 加权投票法

预测标记，基于概率论模型。

结合策略 - 学习法

- 训练数据很多时，一种更为强大的结合策略：“学习法”
  - ✓ 通过另一个学习器，来进行结合。
- 个体学习器称为初级学习器
- 用于结合的学习器称为次/元级学习器

- Stacking是学习法的典型代表

- 贝叶斯模型平均(BMA)
  - ✓ 基于后验概率，为不同模型赋予权重，加权平均法的一种特殊
  - ✓ 对数据噪声，敏感。

Stacking ，通常优于贝叶斯，其鲁棒性更好。
贝叶斯，对模型近似误差非常敏感。

集成学习

- 个体与集成
- Boosting：串行
  - Adaboost
- Bagging与随机森林：并行
- 结合策略
  - 平均法、投票法、学习法
- 多样性
  - 误差-分歧分解、多样性度量、多样性扰动

多样性 – 描述：误差-分歧分解

- 集成学习的目标
  - 构建泛化能力强的集成，个体学习器，应好而不同（具有多样性）。

- 集成的分歧：个体学习器h_i的分歧的加权平均
  - ✓ 分歧：代表了个体学习器在样本x上的不一致性
  - ✓ 分歧在一定程度上，反映了个体学习器的多样性

- 集成的误差：个体学习器h_i的误差的加权平均

➢ 集成学习的目标，拆分为分歧项和误差项，称为误差-分歧分解

- 个体学习器，精确性越高、多样性越大，则集成效果越好。

多样性 - 度量

- 多样性度量
  - ✓ 度量集成中，个体学习器的多样性
  - ✓ 即估算个体学习器的多样化程度。

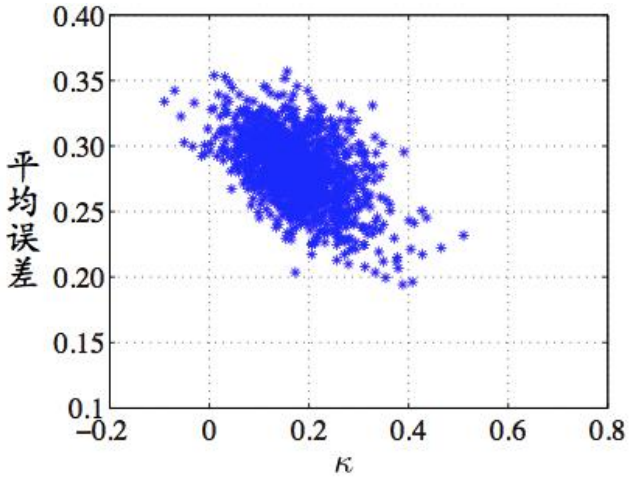
- 典型做法
  - ✓ 考虑 个体分类器的
    - 两两 相似性
    - 两两 不相似性

多样性 - 多样性度量

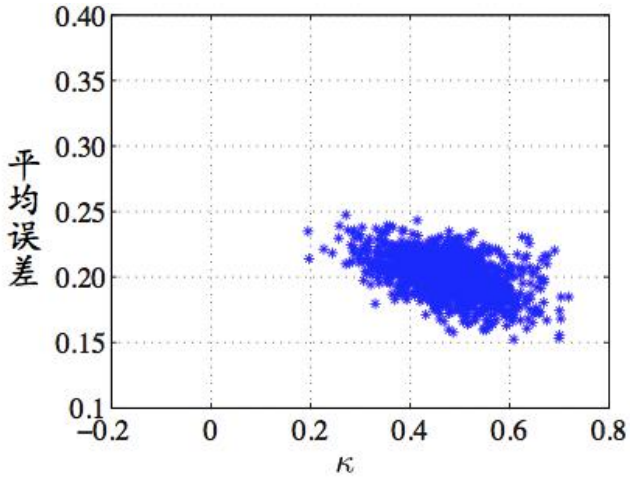
- 多样性度量：度量集成中个体学习器的多样性
  即估算个体学习器的多样化程度。
- 典型做法是考虑 个体分类器的 两两 相似/不相似性。

- k-误差图

每一对分类器，作为图上的一个点
横坐标：这对分类器的k值
纵坐标：这对分类器的平均误差



(a) AdaBoost 集成



(b) Bagging 集成

- 数据点云的位置越高，则个体分类器准确性越低；
  数据点云的位置越靠右，则个体学习器的多样性越小。
- 数据点云的位置越低、靠左，则个体学习器的多样性越好

多样性 - 多样性增强

- 集成学习中，需有效地生成多样性大的个体学习器。
- 一般思路：在学习过程中，引入随机性。
- 常见的增强个体学习器的多样性的方法
  - 数据样本扰动
    - ✓ 通常是基于采样法，对“不稳定基学习器”很有效
  - 输入属性扰动
    - ✓ 基于不同子空间，训练个体学习器
  - 输出表示扰动
    - ✓ 对输出表示进行操纵。
    - ✓ 例如，随机改变一些训练样本的标记；将多分类任务，拆解为一系列二分类任务，来训练基学习器
  - 算法参数扰动
    - ✓ 随机设置不同参数，训练出多个学习器，最终仅选其中一个学习器，进行使用。

总结

- 个体与集成
- Boosting：串行
  - Adaboost
- Bagging与随机森林：并行
- 结合策略
  - 平均法、投票法、学习法
- 多样性
  - 误差-分歧分解、多样性度量、多样性扰动