机器学习
Machine Learning

北京航空航天大学计算机学院智能识别与图像处理实验室
IRIP Lab, School of Computer Science and Engineering, Beihang University
黄 迪　刘庆杰

2020年秋季学期
Fall 2020

课前回顾

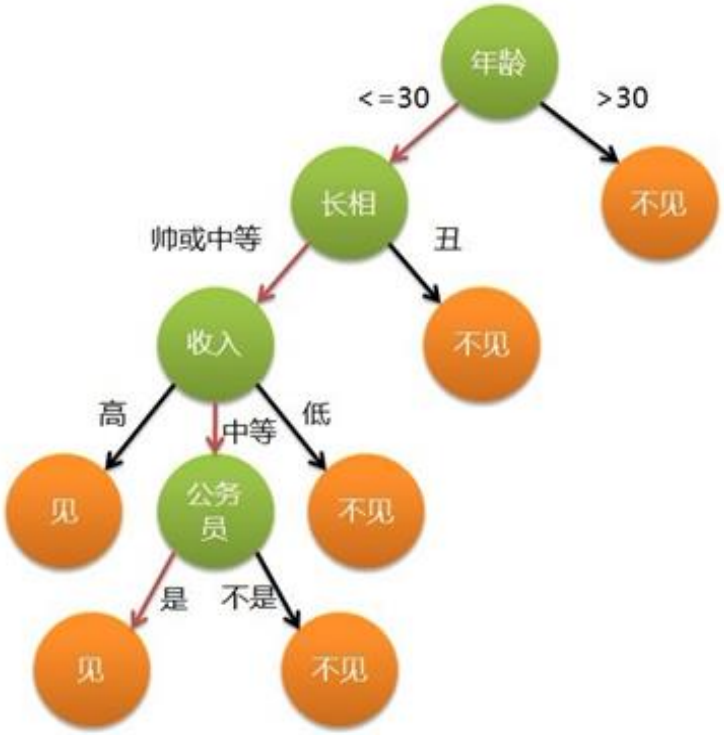
问题背景

- **问题举例**
  - 根据症状或检查结果分类患者
  - 根据起因或现象分类设备故障
  - 根据拖欠支付的可能性分类贷款申请
- **分类问题**
  - 把样例分类到各可能的离散值对应的类别
- **问题特征**
  - 实例由“属性-值”对表示，

    训练数据可以包含缺少属性值的实例
  - 属性可以是连续值或离散值
  - 具有离散的输出值

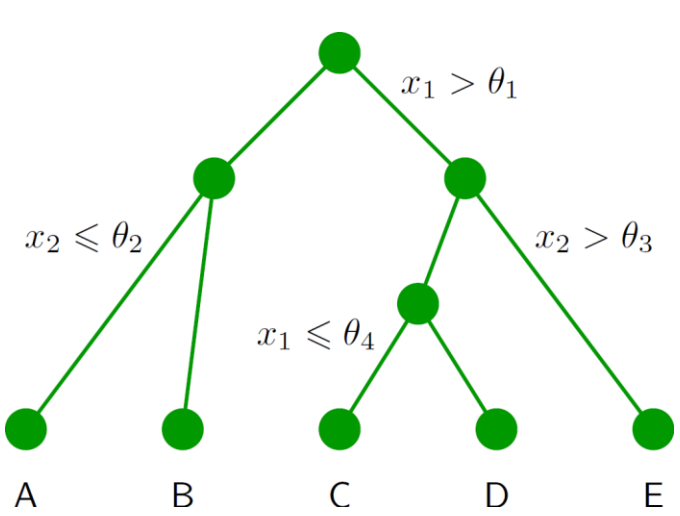
决策树定义

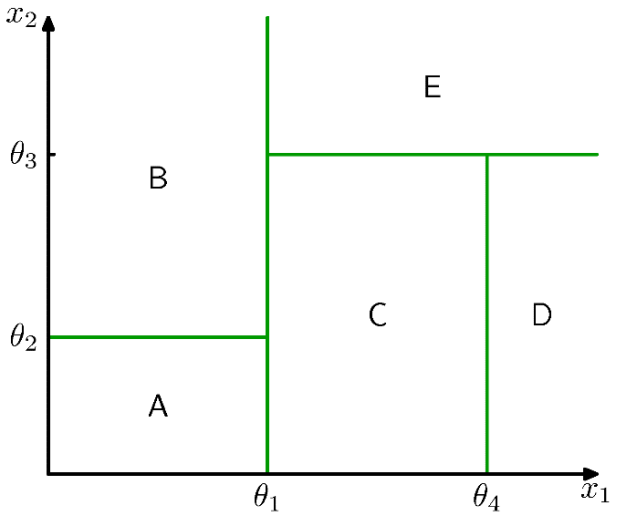
- 决策树(Decision Tree)
  - 决策树是一种树型结构，由结点和有向边组成
  - 结点
    - 内部结点表示一个属性或特征
    - 叶结点代表一种类别
  - 有向边/分支
    - 分支代表一个测试输出



决策树算法

- 基本思想
  - 采用自顶向下的递归方法，以信息熵为度量构造一棵熵值下降最快的树，到叶子结点处的熵值为零，此时每个叶结点中的实例都属于同一类
    - 决策树可以看成一个if-then的规则集合
    - 一个决策树将特征空间划分为不相交的单元(Cell)或区域(Region)





算法流程

- **基本流程分为两步**
  - **第1步：训练，从数据中获取知识进行学习**

利用训练集建立(并精化)一棵决策树，构建决策树模型.

  - **第2步：测试，利用生成的模型对输入数据进行分类**

对测试样本，从根结点依次测试记录的属性值，直至到达某个叶结点，找到该样本所在的类别.

算法流程

- 构建过程的基本流程
  - Step1：选取一个属性作为决策树的根结点，然后就这个属性所有的取值创建树的分支.
  - Step2：用这棵树来对训练数据集进行分类：
    - 如果一个叶结点的所有实例都属于同一类，则以该类为标记标识此叶结点.
    - 如果所有的叶结点都有类标记，则算法终止.
  - Step3：否则，选取一个从该结点到根路径中没有出现过的属性作为标记标识该结点，然后就这个属性的所有取值继续创建树的分支；重复算法步骤2.

主要算法

- 建立决策树的关键，即在当前状态下选择哪个属性作为分类依据

  示例：高？富？帅？　会C++？会图像处理？深度学习？

- **目标**：每个分支结点的样本尽可能属于同一类别，即结点的“**纯度**”**(Purity)**越来越高
- 根据不同的目标函数，建立决策树主要有以下三种算法
  - ID3：　信息增益
  - C4.5：　信息增益率
  - CART：基尼指数

主要算法

- **ID3 (Iterative Dichotomiser 3)算法**
  - 以信息熵为度量，每次优先选取信息增益最大的属性，即能使熵值最小的属性.
  - 熵→随机变量不确定性的度量.
    - 信息熵/经验熵/条件熵/经验条件熵/…

$$H(D) = -\sum_{c=1}^{C} p_c \log_2 p_c$$

$$= -\sum_{c=1}^{C} \frac{D_c}{D} \log_2 \frac{D_c}{D}$$

$$H(Y \mid X) = -\sum_{i=1}^{n} p_i H(Y \mid X = x_i)$$

  - 信息增益→特征对训练数据的信息增益定义为集合的经验熵与特征给定条件下集合的经验条件熵之差

$$G(D,a) = H(D) - H(D \mid a)$$

$$= H(D) - \sum_{n=1}^{N} \frac{|D^n|}{|D|} H(D^n)$$

ID3算法

- ID3算法局限性
  - 信息增益偏好取值多的属性(极限趋近于均匀分布)
  - 可能会受噪声或小样本影响，易出现过拟合问题
  - 无法处理连续值的属性
  - 无法处理属性值不完整的训练数据
  - 无法处理不同代价的属性

属性筛选度量标准

- **信息增益率(Information Gain Ratio)**

$$G_{ratio}(D,a)=\frac{G(D,a)}{H(a)}$$

其中

$$H(a)=-\sum_{n=1}^{N}\frac{|D_n|}{|D|}\log_2\frac{|D_n|}{|D|}$$

称为属性*a*的固有值

***N*越大， *H(a)*通常也越大；因此采用信息增益率，可缓解信息增益准则对可取值数目较多的属性的偏好.**

C4.5算法就采用增益率替代了ID3算法的信息增益

属性筛选度量标准

- **基尼指数(Gini Index)**

$$Gini(D)=\sum_{c=1}^{C}\sum_{c'\neq c}p_c p_c'=1-\sum_{c=1}^{C}p_c^2=1-\sum_{c=1}^{C}\left(\frac{|D_c|}{|D|}\right)^2$$

直观反映了从数据集中随机抽取两个样本，其类别不一致的概率；基尼指数越小，数据集的纯度越高.

- **属性*A*的基尼指数** $Gini(D,a)=\sum_{n=1}^{N}\frac{|D^n|}{|D|}Gini(D^n)$
- **最优属性选择** $a^*=\arg\min_{a\in A}Gini(D,a)$

CART算法就采用基尼指数替代了ID3算法的信息增益

剪枝处理(Pruning)

- 针对过拟合问题
  - 剪枝是主要手段
- 基本策略
  - 预剪枝策略(Pre-pruning)：决策树生成过程中，对每个结点在划分前进行估计，若划分不能带来决策树泛化性能提升，则停止划分并将该节点设为叶结点.
  - 后剪枝策略(Post-pruning)：先利用训练集生成决策树，自底向上对非叶结点进行考察，若将该叶结点对应子树替换为叶结点能带来泛化性能提升，则将该子树替换为叶结点.

预剪枝算法

- **策略特点**
  - **优势**：“剪掉”很多没必要展开的分支，降低了过拟合风险，并且显著减少了决策树的训练时间开销和测试时间开销.
  - **劣势**：有些分支的当前划分有可能不能提高甚至降低泛化性能，但后续划分有可能提高泛化性能；预剪枝禁止这些后续分支的展开，可能会导致欠拟合.

后剪枝算法

- **策略特点**
  - **优势**：测试了所有分支，比预剪枝决策树保留了更多分支，降低了欠拟合的风险，泛化性能一般优于预剪枝决策树.
  - **劣势**：后剪枝过程在生成完全决策树后再进行，且要自底向上对所有非叶节点逐一评估；因此，决策树的训练时间开销要高于未剪枝决策树和预剪枝决策树.

连续值处理

- 基本思想：采用二分法(Bi-Partition)进行离散化
  - 给定样本集D和连续属性a($a \in A$)，假定a在D上有N个不同取值，将这些值从大到小排序得到$\{a_1, a_2, \ldots, a_N\}$
  - 基于划分点t，可将D分为子集D_t^+和D_t^-，其中D_t^+ (D_t^-)包含了属性值A不小(大)于t的样本子集
  - t在$[a_n, a_{n+1})$上的任意取值的划分结果都相同

连续值处理

- **基本思想：采用二分法(Bi-Partition)进行离散化**
  - **候选划分点集合** $T_a = \left\{ \frac{a_n + a_{n+1}}{2} \,\middle|\, 1 \le n \le N-1 \right\}$
  - **信息增益** $G(D,a) = \max_{t \in T_a} G(D,a,t)$

$$= \max_{t \in T_a} \left(H(D) - \sum_{\lambda \in \{+,-\}} \frac{|D_t^\lambda|}{|D|} H(D_t^\lambda) \right)$$

其中，$G(D,A,t)$ 是样本集D基于划分点t二分后的信息增益．我们需选择使 $G(D,A,t)$ 最大的划分点t.

缺失值处理

- **定义**
  - $\tilde{D}$：**D**中在属性**a**上没有缺失值的样本子集
  - 属性**a**($a \in A$) 有**N**个可取值$\{a_1, a_2, \ldots, a_N\}$
  - $\tilde{D}^n$：$\tilde{D}$中在属性**a**上取值为a_n的样本子集
  - $\tilde{D}_c$：$\tilde{D}$中属于第**c**($c \in C$)类的样本子集
  - 无缺失值样本所占比例 $\rho = \dfrac{\sum_{x \in \tilde{D}} \omega_x}{\sum_{x \in D} \omega_x}$
  - 无缺失样本中第**c**类比例 $\tilde{p}_c = \dfrac{\sum_{x \in \tilde{D}_c} \omega_x}{\sum_{x \in \tilde{D}} \omega_x}, (1 \le c \le C), \sum_{c=1}^{C} \tilde{p}_c = 1$
  - 无缺失样本中属性**a**上取值为a_n的比例 $\tilde{r}_n = \dfrac{\sum_{x \in \tilde{D}^n} \omega_x}{\sum_{x \in \tilde{D}} \omega_x}, (1 \le n \le N), \sum_{n=1}^{N} \tilde{r}_n = 1$

缺失值处理

- **信息增益** $G(D,A)=\rho\times G(\tilde{D},A)$

$$=\rho\times\left(H(\tilde{D})-\sum_{n=1}^{N}\tilde{r}_n H(\tilde{D}_n)\right)$$

其中 $H(\tilde{D})=-\sum_{c=1}^{C}\tilde{p}_c\log_2\tilde{p}_c$

原则：让样本以不同概率划分到不同的子结点去

- 若样本x在属性a上的取值已知，则将x划入与其取值对应的子结点，且样本权值保持为ω_x
- 若样本x在属性a上的取值未知，则将x划入所有子结点，且样本权值在与属性值对应的子结点中调整为 $\tilde{r}_n\cdot\omega_x$

不同代价属性的处理

- **问题描述**
  - 不同的属性测量具有不同的代价
  - 医疗诊断为例：
    - 属性：体温、血压、血常规、活检
    - 代价不同：所需时间、费用、友好性等
- **基本思想**
  - 在属性筛选度量标准中考虑属性的不同代价
  - 优先选择低代价属性的决策树
  - 必要时才依赖高代价属性

不同代价属性的处理

- **属性筛选度量标准1** [*Tan et al.*, 1990, 1993]

$$G_{Cost}(D,a)=\frac{G(D,a)}{Cost(a)}$$

- **属性筛选度量标准2** [*Nunez et al.*, 1988, 1991]

$$G_{Cost}(D,a)=\frac{2^{G(D,a)}-1}{(Cost(a)+1)^{\omega}}$$

其中$Cost(a)$为属性a的代价

$\omega\in[0,1]$为常数，决定代价对于信息增益的相对重要性

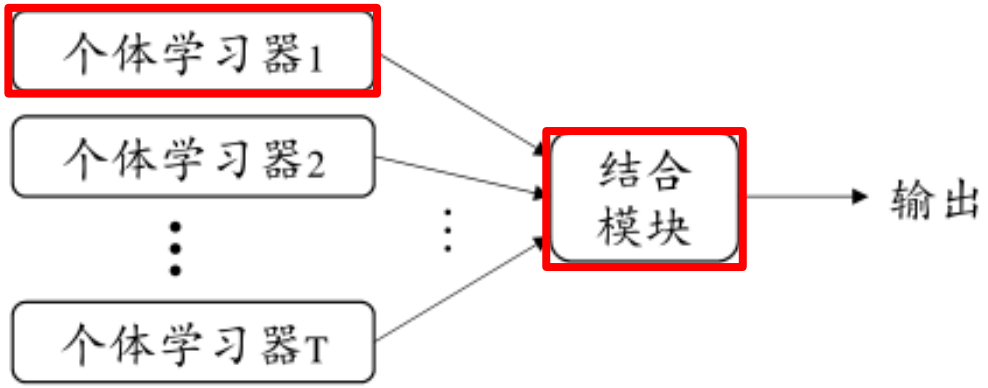
集成学习

Ensemble Methods

什么是集成学习?

- 集成学习(Ensemble Learning)
  - 通过构建并结合多个分类器完成学习任务
  - 也称为多分类器系统(Multi-Classifier System)、基于委员会的学习(Committee based Learning)等

同质或异质



三个臭皮匠，胜过诸葛亮！

什么是集成学习?

- 通过将多个学习器进行集成，常可获得比单一学习器显著优越的泛化性能，这对弱分类器尤为明显
  - **弱分类器**：准确率仅比随机猜测略高的分类器
  - **强分类器**：准确率高并能在多项式时间内完成的分类器

个体与集成

- 多个学习器一定比单一学习器性能好吗？ NO
  - 简单示例：在二分类问题中，假定3个分类器在三个样本中的表现如下所示，其中√表示分类正确，×号表示分类错误，集成结果通过投票(Voting)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

  - 如何获得比最好的单一学习器更好的性能？

集成个体：好而不同

个体与集成

- 简单分析
  - 考虑二分类问题 $y \in \{-1, +1\}$ 和真实函数 f，假定基分类器的错误率为 ϵ，即对每个基分类器 h_i 有：

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

  - 假设集成通过简单投票法结合 T 个基分类器，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
  - 假设基分类器的错误率相互独立，则由Hoeffding不等式可知，集成的错误率为：

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

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

  - 可见，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
  - 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！
  - 个体学习器的“准确性”和“多样性”存在冲突

集成学习的研究核心：

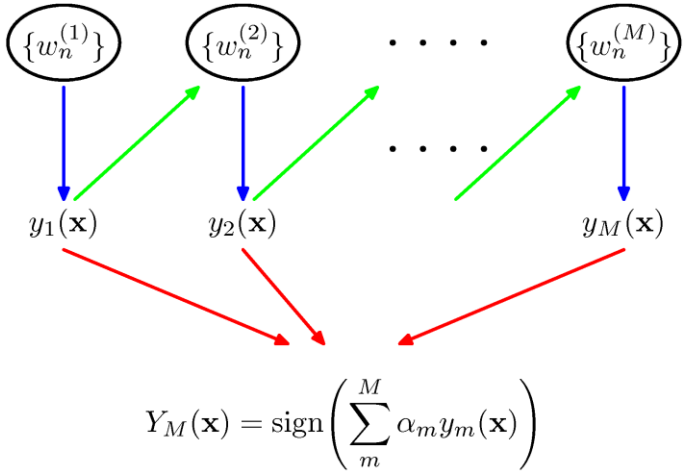
如何产生并结合“好而不同”的个体学习器

集成学习方法

- 根据个体学习器生成方式不同，形成两大类方法
  - 串行化方法：个体学习器间存在强依赖关系
    - 典型算法：Boosting，Adaboost
  - 并行化方法：个体学习器间不存在强依赖关系
    - 典型算法：Bagging、随机森林(Random Forest)

串行化方法 Boosting

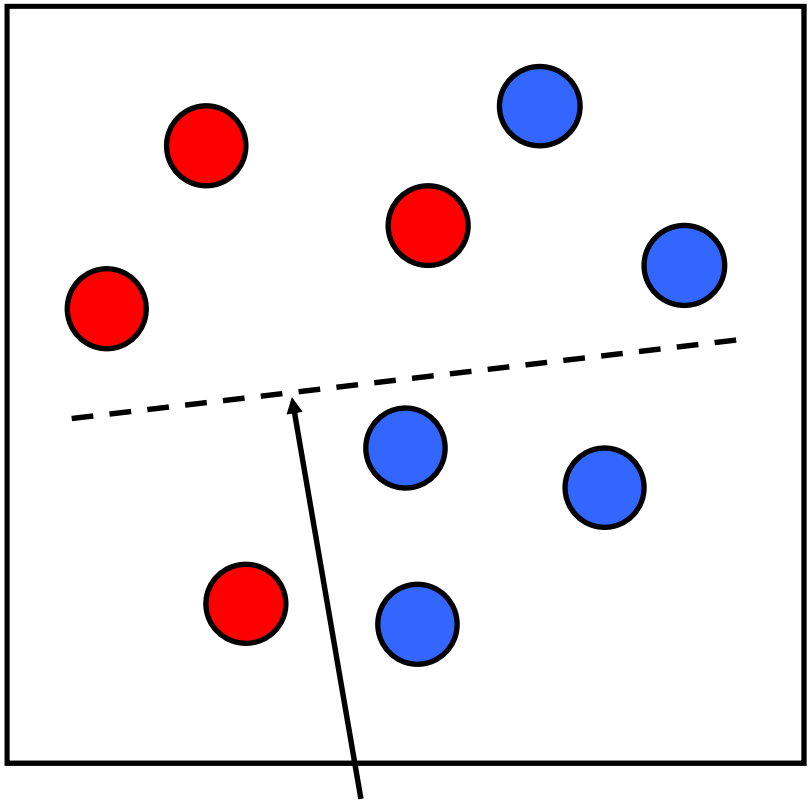
- 一族可将弱学习器提升为强学习器的算法
  - 个体学习器存在强依赖关系
  - 串行生成
  - 每次调整训练数据的样本分布

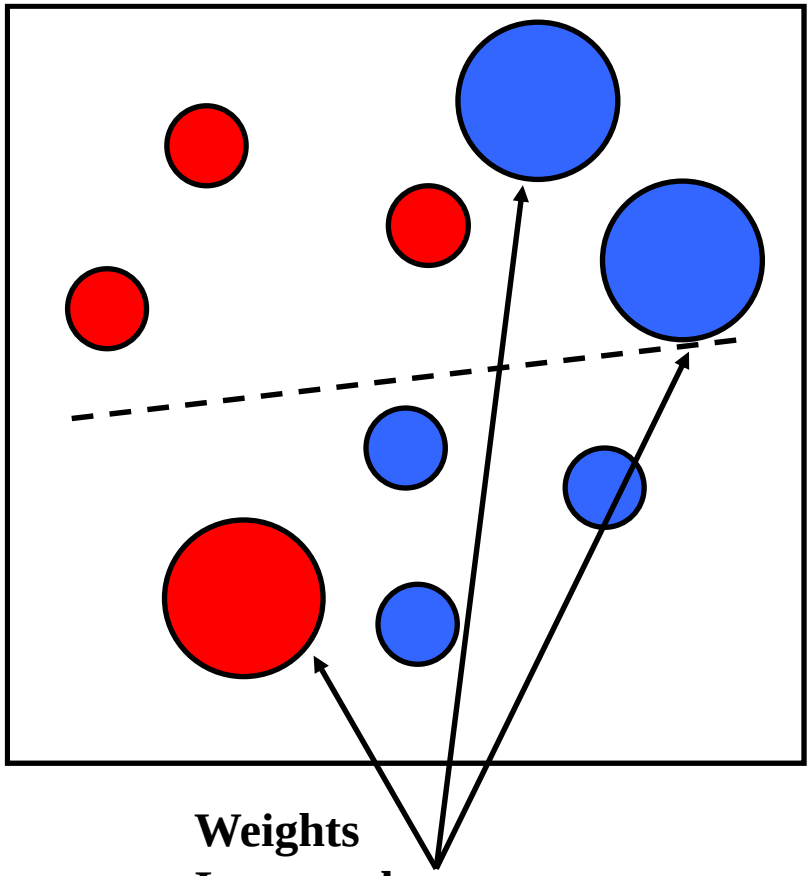


- **基本思想**：

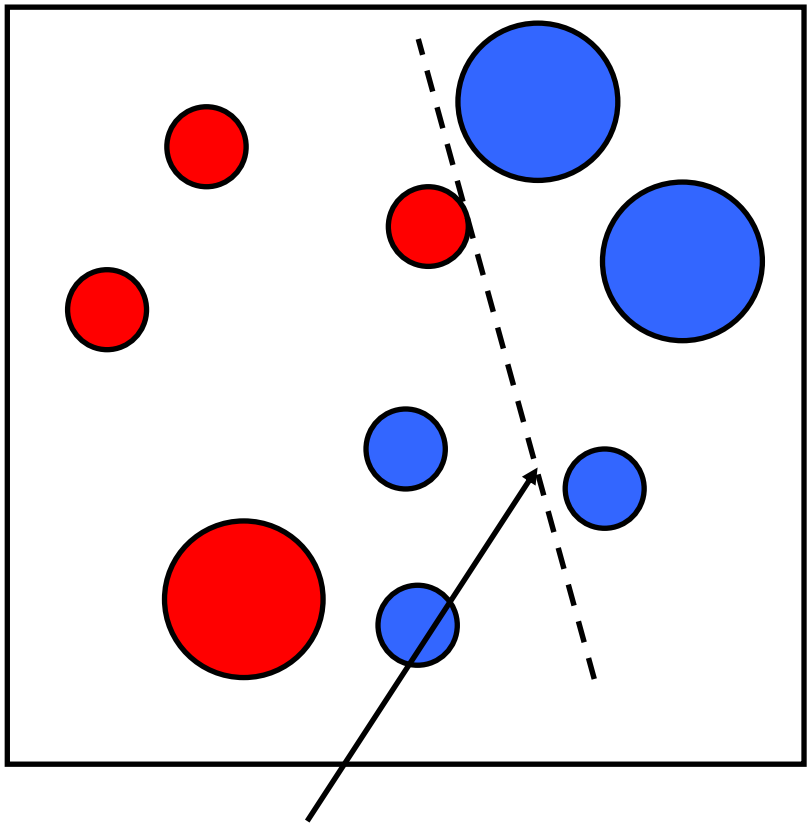
先从初始数据集训练一个基学习器，再根据其对训练样本分布进行调整，使先前错分样本在后续受到更多关注，然后基于调整后的样本分布训练下一个基学习器；重复进行直至基学习器数目达到预先指定值。最终将这些基学习器加权结合

Boosting-基本思想示意

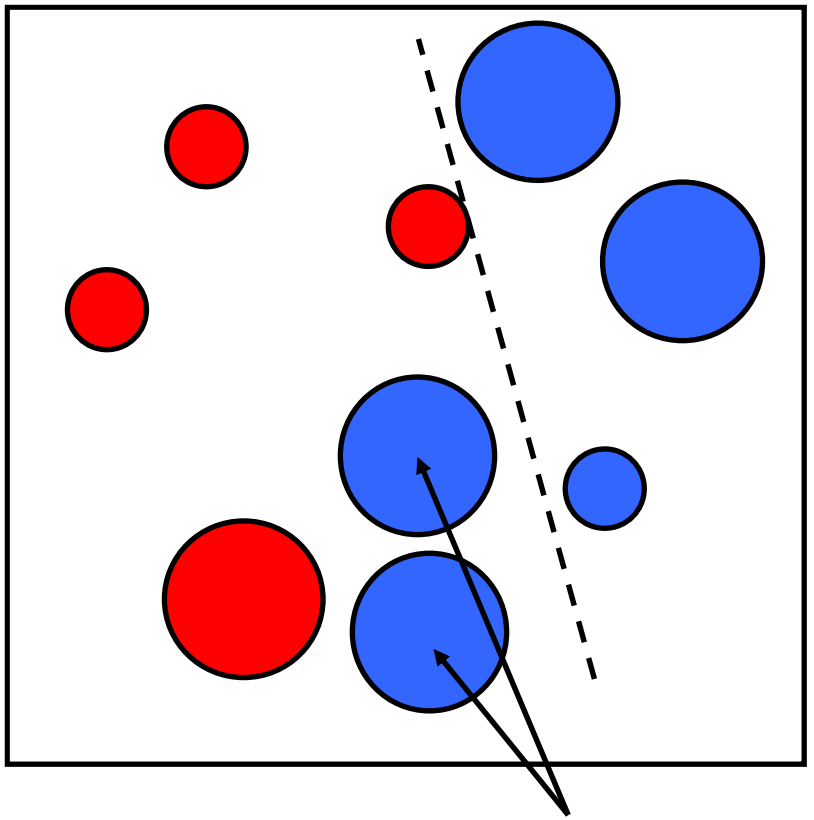




Boosting-基本思想示意

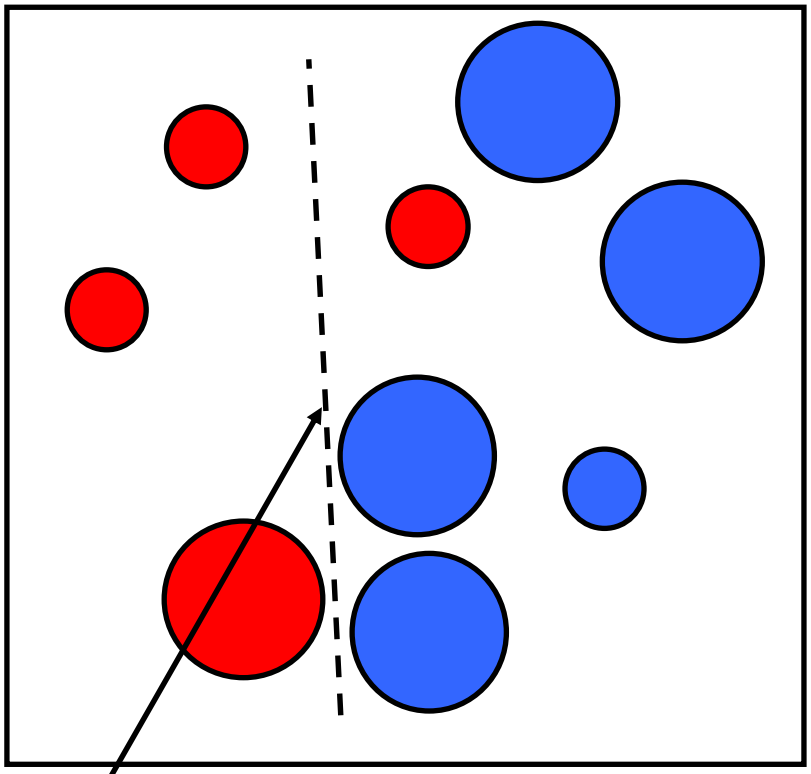


Weak Classifier 2

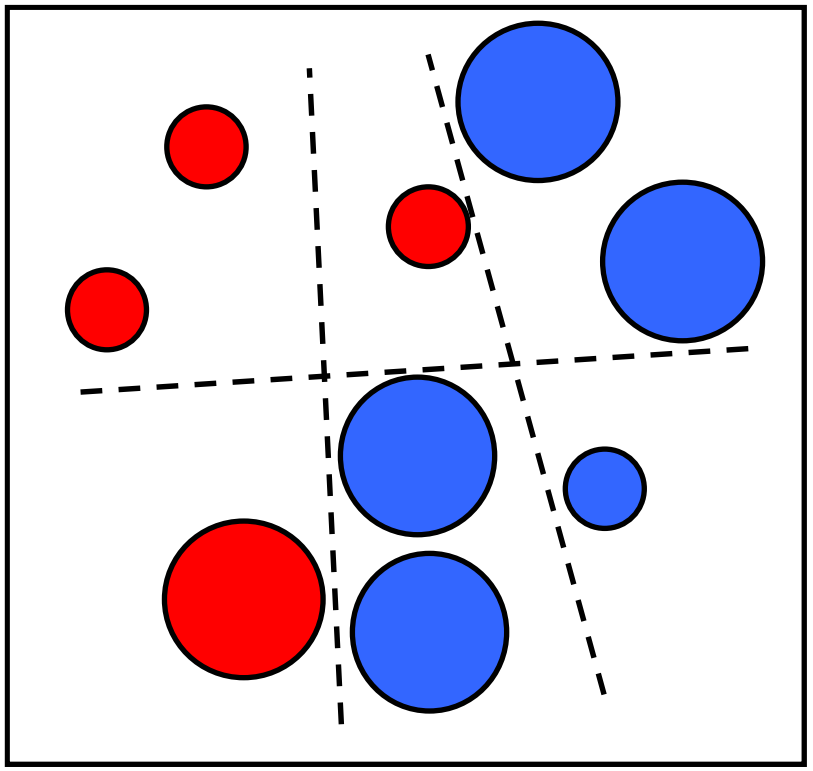


Weights Increased

Boosting-基本思想示意



Weak
Classifier 3



Final classifier is
a combination of weak
classifiers

Adaboost算法

- **Boosting族算法最著名的代表【1997年Freund和Schapire提出】**

f：真实函数
$y \in \{-1, +1\}$

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：

初始化样本权值分布

1: $\mathcal{D}_1(\boldsymbol{x}) = 1/m.$

2: **for** $t = 1, 2, \ldots, T$ **do**

基于分布D_t训练分类器h_t

3: $h_t = \mathfrak{L}(D, \mathcal{D}_t);$

估计h_t误差

4: $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}));$

5: **if** $\epsilon_t > 0.5$ **then break**

确定分类器h_t的权重

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

更新样本分布
(Z_t规范化因子)

7: $$\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases}$$
$$= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}$$

8: **end for**

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

Boosting算法

- 基学习器学习特定的数据分布
  - 重赋权法(Re-weighting)

  在每轮根据样本分布为每个训练样本重新赋予权重
  - 重采样法(Re-sampling)

  在每轮根据样本分布对训练集重新采样形成新的训练集

  注意：Boosting每轮检查当前生成的基学习器是否满足优于随机猜测的基本条件，若不满足，此基学习器被抛弃，学习过程停止。如果与预先设定的学习轮数差距较大，会导致整体性能不佳。采用重采样策略，则可“重启动”避免训练过早停止。

Boosting算法

- 特点总结：
  - 主要关注降低偏差，可基于泛化性能相当弱的学习器构造出很强的集成
  - Boosting中每个模型是弱模型，偏差高，方差低
  - Boosting的基本思想是用贪心法最小化损失函数，因此能降低偏差
  - 但是由于模型的相关性很强，因此不能显著降低方差

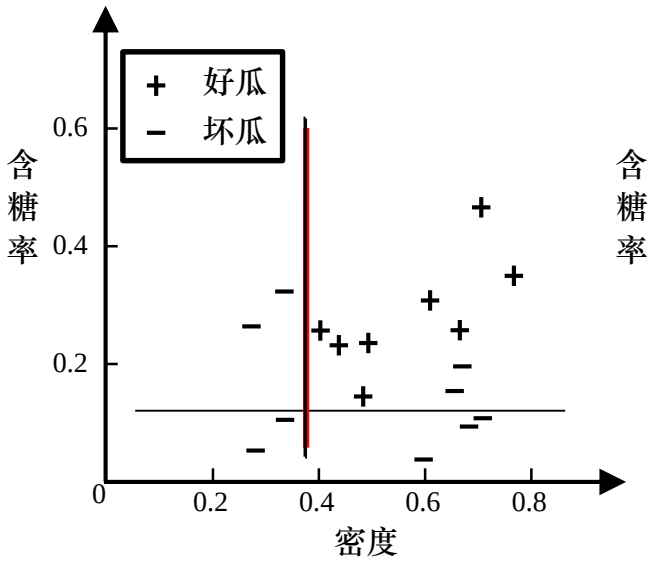
Boosting算法

- 示例-西瓜分类

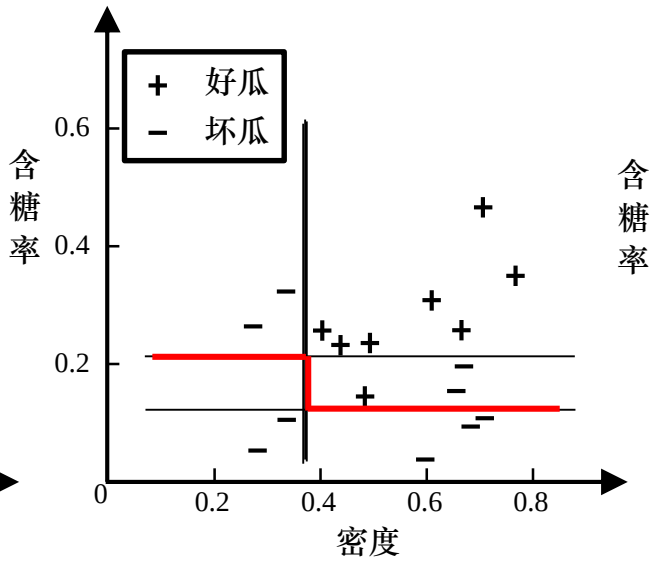
编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	0.360	0.370	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	0.719	0.103	否

Boosting算法

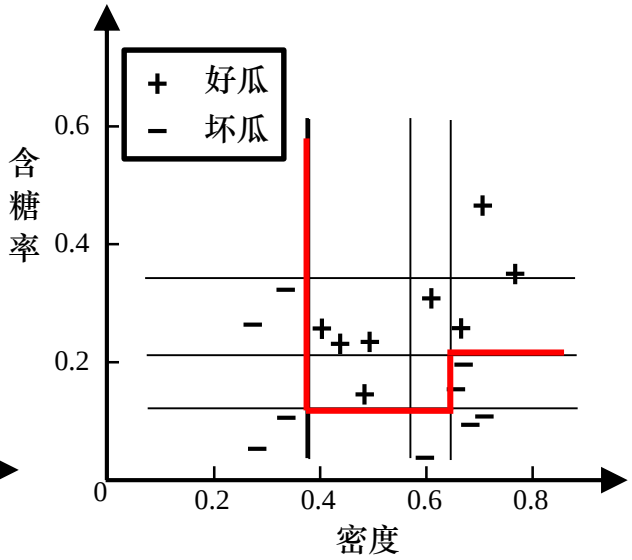
- 示例-西瓜分类



(a) 3个基学习器



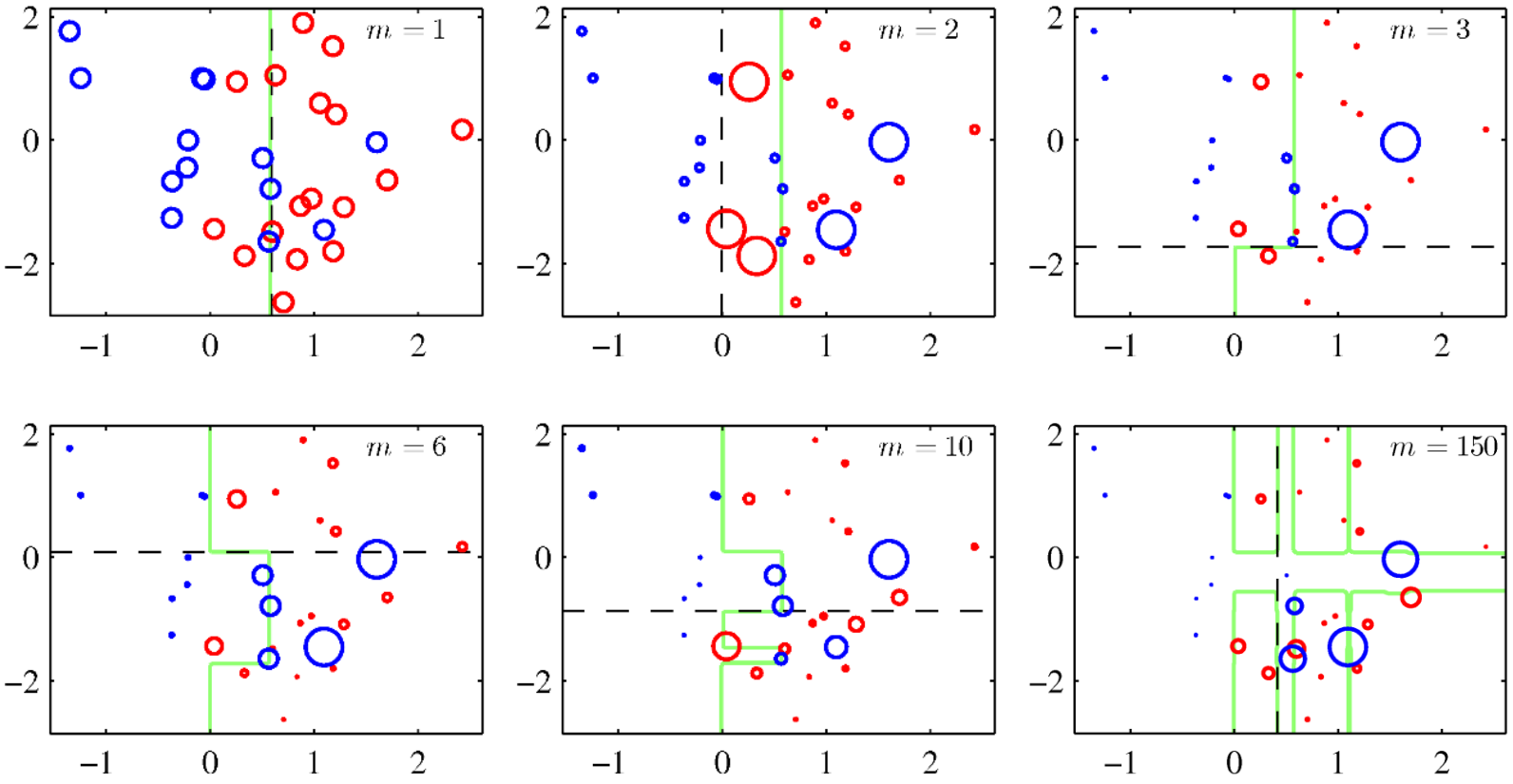
(b) 5个基学习器



(c) 11个基学习器

Boosting算法

- 示例



并行化方法　Bagging算法

- 并行式集成学习最著名的代表性方法【1996年Breiman提出】
  - 名字由Bootstrap AGGregatING缩写而来
  - 基于自助法采样 (Bootstrap Sampling)

给定包含m个样本的数据集D，对其进行采样产生数据集D'：

每次随机从D中挑选一个样本，将其拷贝至D'，这个过程重复执行m次后，就得到了包含m个样本的数据集D'。显然，D中有一部分样本会在D'中多次出现，而有一部分样本则不会出现。

样本在m次采样中始终不被采到的概率是 $(1-\frac{1}{m})^m$，其极限：

$$\lim_{m\to\infty}(1-\frac{1}{m})^m \mapsto \frac{1}{e} \approx 0.368$$

将D'用做训练集; $D\backslash D'$用作测试集

产生不同训练集对集成学习有好处，但改变数据分布会引入偏差。

【1993年Efron和Tibshirani提出】

Bagging算法

- **并行式集成学习最著名的代表性方法【1996年Breiman提出】**

基本思想:

利用自助法采样可构造*T*个含*m*个训练样本的采样集，基于每个采样集训练出一个基学习器，再将它们进行结合(在对预测输出结合时，通常对分类任务使用简单投票法，对回归任务使用简单平均法)。

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.

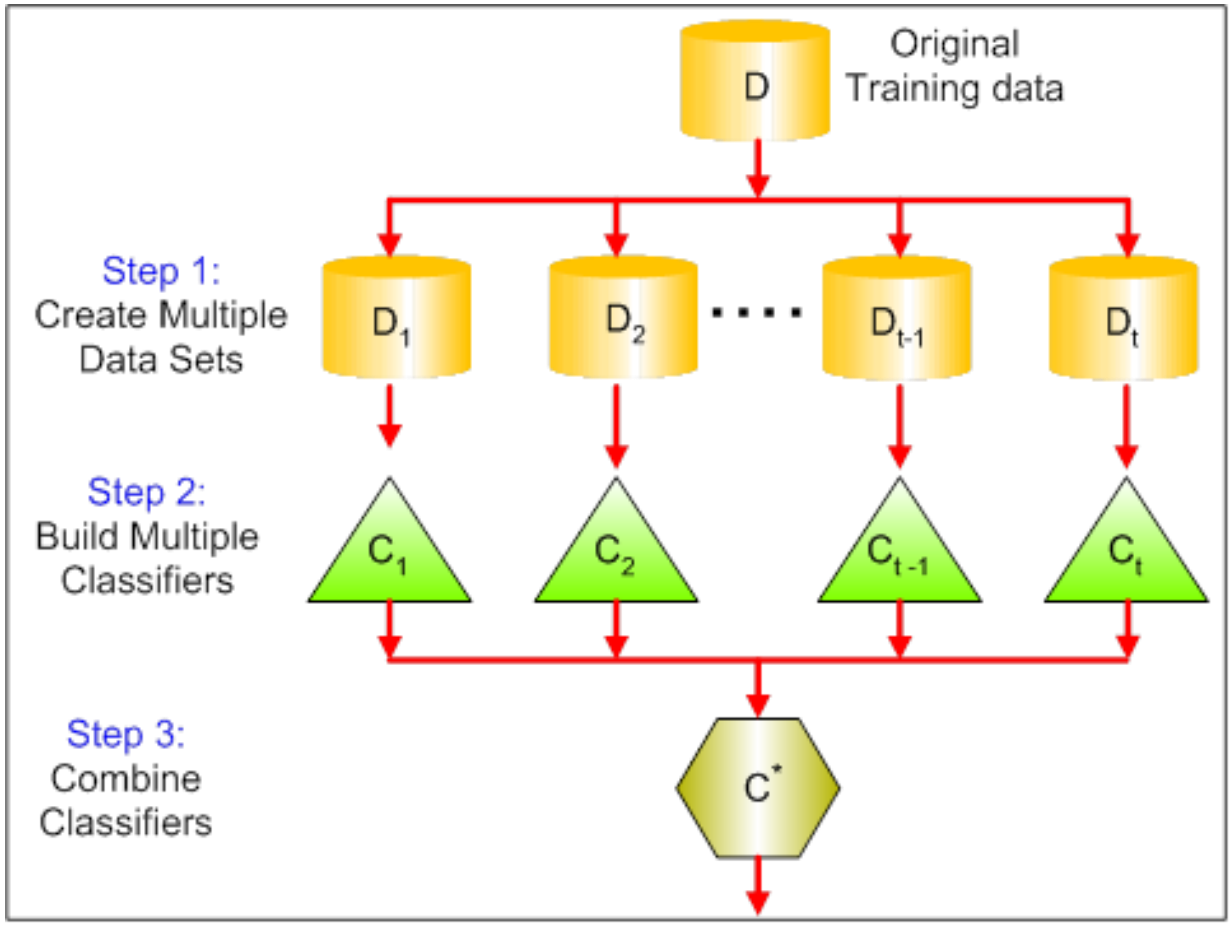
过程:

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

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

Bagging算法

- 并行式集成学习最著名的代表性方法【1996年Breiman提出】



Bagging算法

- **算法特点**
  - **时间复杂度低**：集成与直接训练一个学习器复杂度同阶

假定基学习器的计算复杂度为$O(m)$，采样与投票/平均过程的复杂度为$O(s)$，则Bagging的复杂度大致为$T(O(m)+O(s))$；

  - 可以直接用于多分类、回归等任务；
  - 可包外估计(Out-of-Bag Estimate)泛化性能.

$H^{oob}(x)$表示对样本x的包外预测，即仅考虑那些未使用样本x训练的基学习器在x上的预测：

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

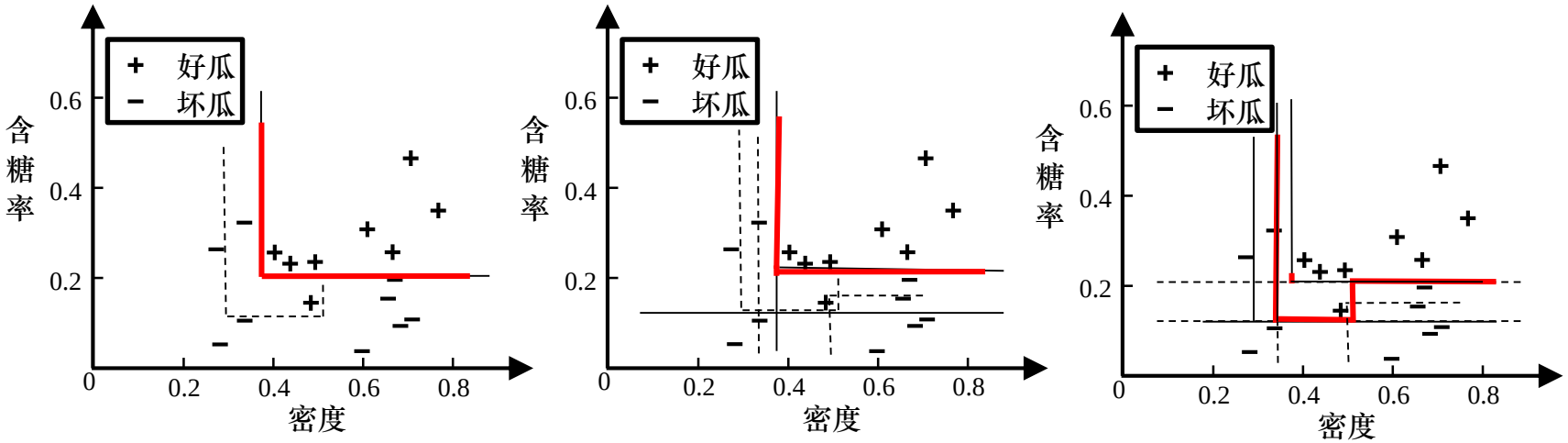
泛化误差的包外估计为：$\epsilon^{oob} = \frac{1}{|D|} \sum_{(\boldsymbol{x},y) \in D} \mathbb{I}(H^{oob}(\boldsymbol{x}) \neq y)$

Bagging算法

- 特点总结
  - 主要关注降低方差，在易受样本扰动的学习器上效用更为明显(如不剪枝的决策树、神经网络等)
  - Bagging中的模型是强模型，偏差低，方差高
  - 在Bagging中，每个模型的偏差方差近似相同，但是互相相关性不太高，因此一般不能降低偏差，而一定程度上能降低方差

Bagging算法

- 示例-西瓜分类



(a) 3个基学习器　(b) 5个基学习器　(c) 11个基学习器

并行化方法 随机森林算法

- Bagging方法的一种扩展变体 【2001年Breiman提出】
  - Random Forest，简称RF
  - 以决策树为基学习器
  - 训练过程引入随机属性选择
- **基本思想**:
  - 对基决策树的每个结点，先从该结点的(d个)属性集合中随机选择一个包含 k 个属性的子集，再从这个子集选择一个最优属性用于划分，一般情况下推荐 $k=\log_2 d$

随机森林算法

- **Bagging方法的一种扩展变体** 【2001年Breiman提出】

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Feature subset size K.

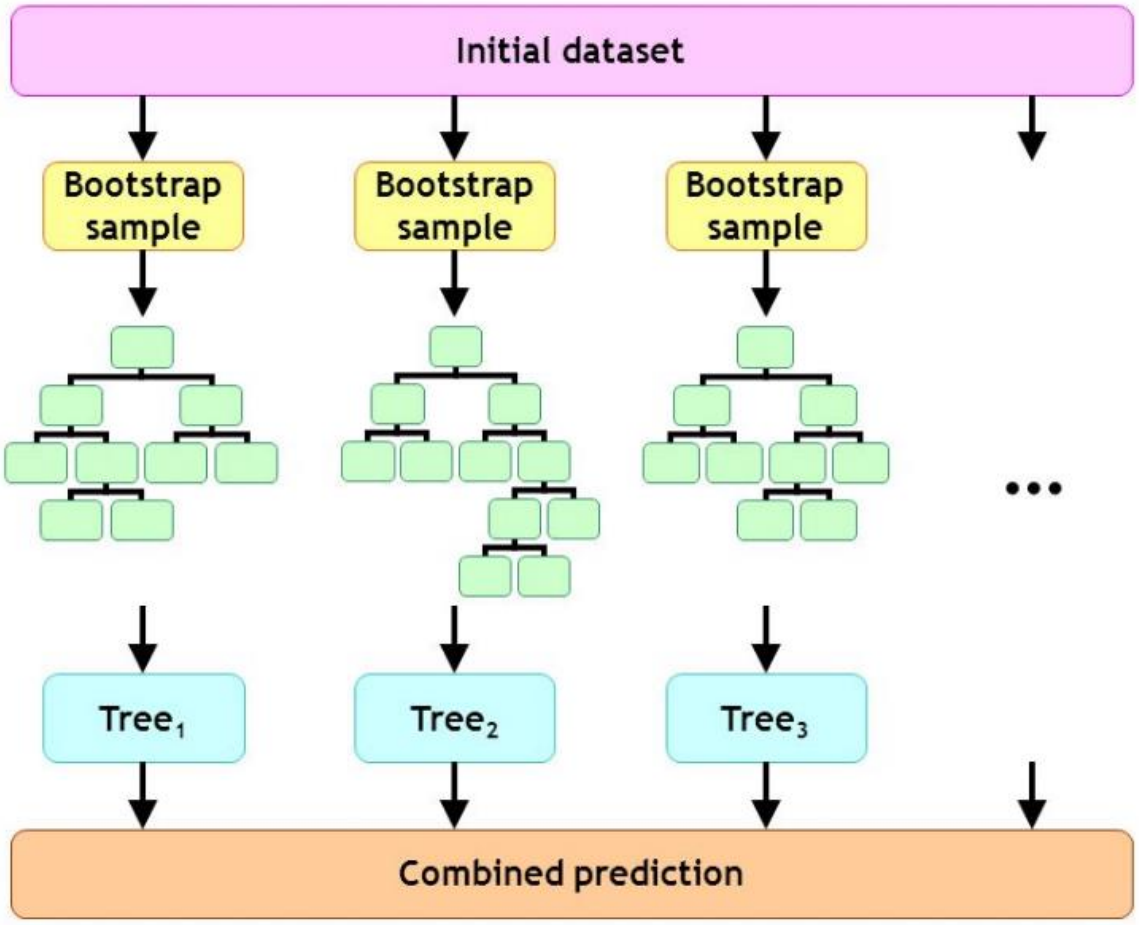
Process:

1. $N \leftarrow$ create a tree node based on D;
2. **if** *all instances in the same class* **then return** N
3. $\mathcal{F} \leftarrow$ the set of features that can be split further;
4. **if** $\mathcal{F}$ *is empty* **then return** N
5. $\tilde{\mathcal{F}} \leftarrow$ select K features from $\mathcal{F}$ randomly;
6. $N.f \leftarrow$ the feature which has the best split point in $\tilde{\mathcal{F}}$;
7. $N.p \leftarrow$ the best split point on $N.f$;
8. $D_l \leftarrow$ subset of D with values on $N.f$ smaller than $N.p$;
9. $D_r \leftarrow$ subset of D with values on $N.f$ no smaller than $N.p$;
10. $N_l \leftarrow$ call the process with parameters (D_l, K);
11. $N_r \leftarrow$ call the process with parameters (D_r, K);
12. **return** N

Output: A random decision tree

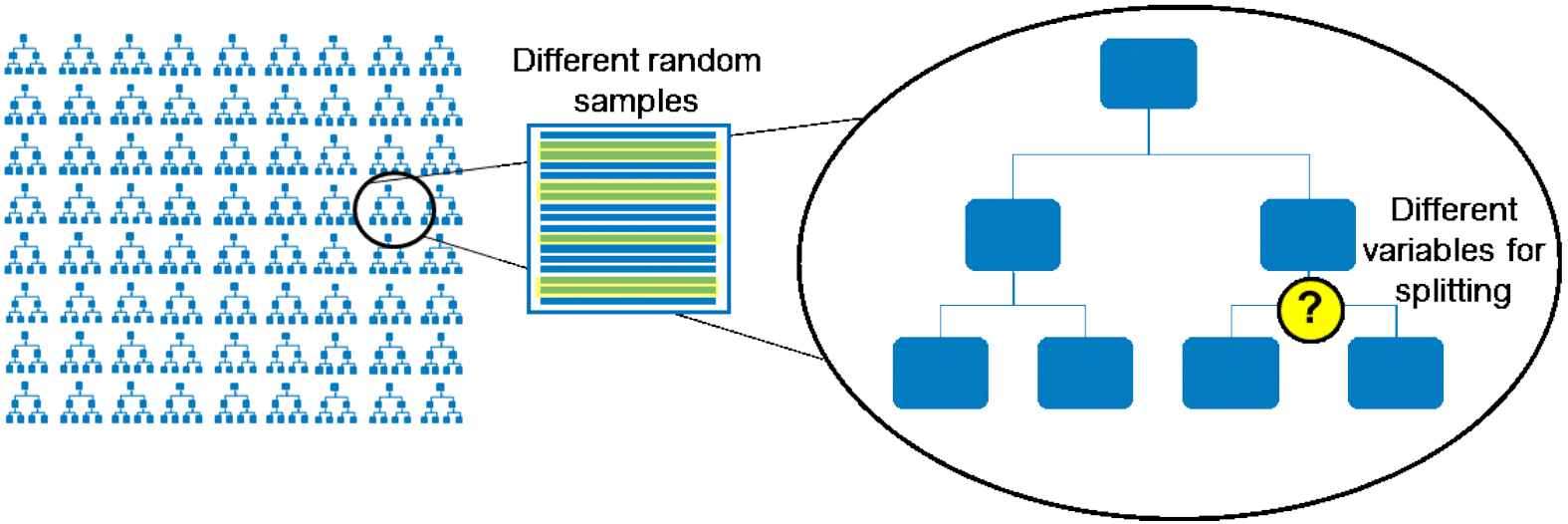
随机森林算法

- **Bagging方法的一种扩展变体** 【2001年Breiman提出】



随机森林算法

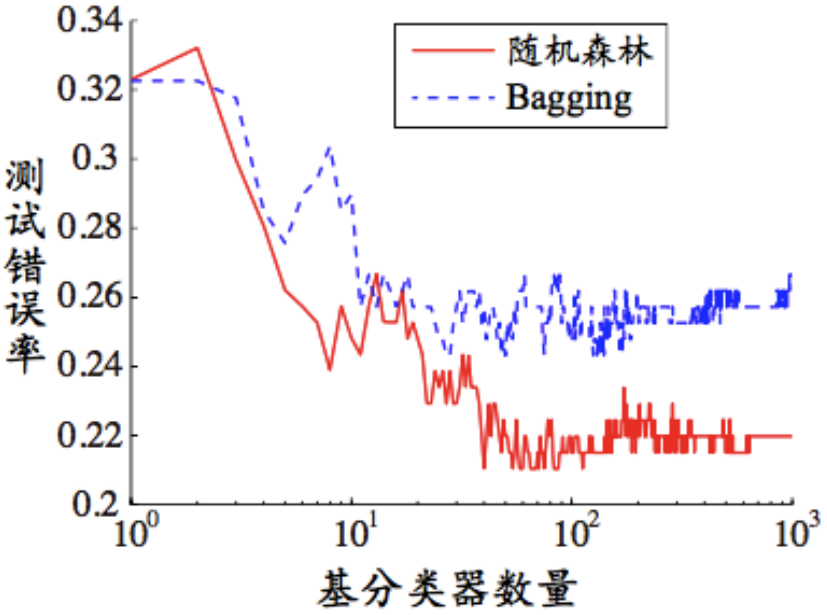
- 算法特点
  - 基学习器多样性通过样本扰动和属性扰动实现
  - 算法简单、容易实现、计算开销小
  - 性能强大，被誉为“代表集成学习技术水平的方法”



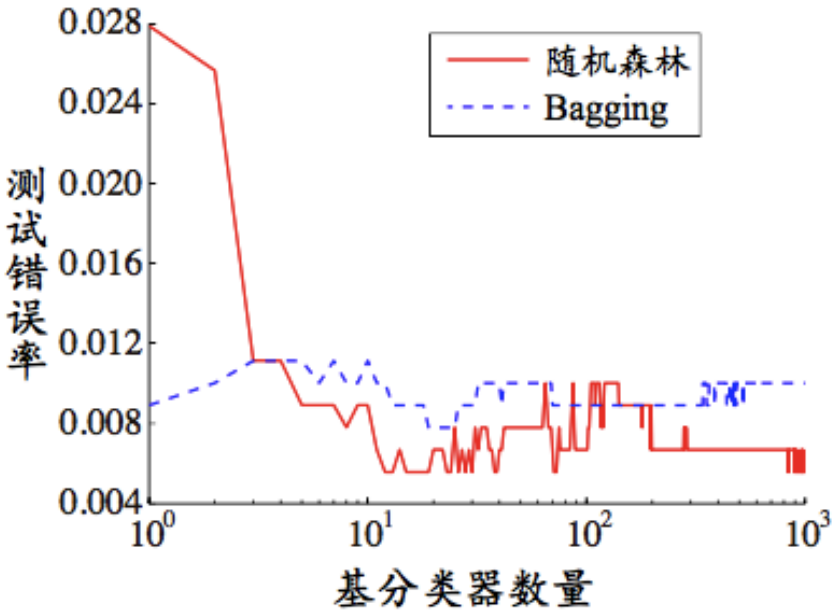
Bagging vs. RF

- RF与Bagging收敛性相似

UCI数据



(a) glass 数据集

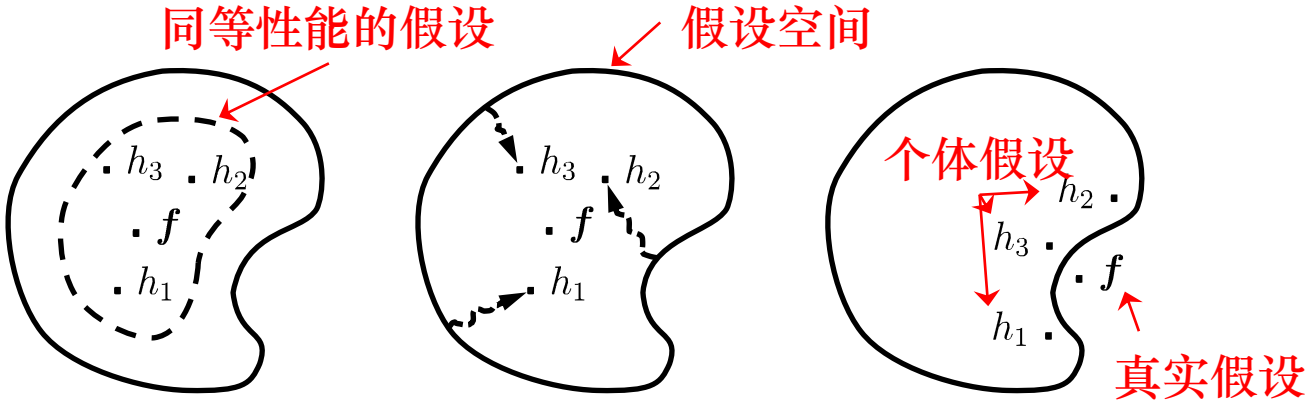


(b) auto-mpg 数据集

- RF训练效率优于Bagging(随机型 vs. 确定型)

结合策略

- 学习器的组合有三个方面的好处
  - 统计方面：减小误选假设空间导致泛化性能不佳的几率
  - 计算方面：降低陷入坏局部极小点影响泛化性能的风险
  - 表示方面：扩大假设空间学习对于真实空间更好的近似



(a) 统计的原因　(b) 计算的原因　(c) 表示的原因

结合策略—平均法

- 平均法(Averaging)是数值型输出最常见的结合策略
  - 简单平均法(Simple Averaging)　个体学习器性能相近时适用

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

  - 加权平均法(Weighted Averaging) 个体学习器性能迥异时适用

【1993年Perrone和Cooper正式将其用于集成学习】

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

加权平均法是集成学习的基本出发点，各种结合方法都可视为其特例或变体，不同的集成学习方法是通过不同的方式确定加权平均法中基学习器的权重

结合策略—投票法

硬投票（类标签）和软投票（类概率）

- 投票法(Voting)是标签型输出最常见的结合策略

  标记集合$\{c_1, c_2, ..., c_N\}$，h_i在样本x上的预测$\{h_i^1(x), h_i^2(x), ..., h_i^N(x)\}$

  - 绝对多数投票法(Majority Voting)：得票超半数

$$H(\boldsymbol{x}) = \begin{cases} c_j & \text{if } \sum_{i=1}^{T} h_i^j(\boldsymbol{x}) > \frac{1}{2} \sum_{k=1}^{l} \sum_{i=1}^{T} h_i^k(\boldsymbol{x}) \\ \text{rejection} & \text{otherwise}. \end{cases}$$

  - 相对多数投票法(Plurality Voting)：得票最多

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} h_i^j(\boldsymbol{x})}$$

  - 加权投票法(Weighted Voting)：加权后得票最多

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} w_i h_i^j(\boldsymbol{x})}$$

结合策略—学习法

- 当训练数据很多时采用另一个学习器进行结合

  初级学习器 vs. 次级学习器或元学习器(Meta-learner)

  – Stacking是学习法的典型代表【1992年Wolpert提出】

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
First-level learning algorithms $\mathfrak{L}_1, \ldots, \mathfrak{L}_T$;
Second-level learning algorithm $\mathfrak{L}$.

Process:

从初始数据集训练初级学习器

```
1.  for t = 1, ..., T:      % Train a first-level learner by applying the
2.    h_t = L_t(D);         % first-level learning algorithm L_t
3.  end
```

生成次级数据集：初级学习器的输出被当作样例输入特征，继承初始样本标记.

```
4.  D' = ∅;                 % Generate a new data set
5.  for i = 1, ..., m:
6.    for t = 1, ..., T:
7.      z_it = h_t(x_i);
8.    end
9.    D' = D' ∪ ((z_i1, ..., z_iT), y_i);
10. end
```

从次级数据集训练次级学习器

```
11. h' = L(D');             % Train the second-level learner h' by applying
                            % the second-level learning algorithm L to the
                            % new data set D'.
```

Output: $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x}))$

多样性

- 误差-分歧分解
  - 对于个体学习器“好而不同”的理论分析

令$\overline{E}=\sum_{i=1}^{T} w_i E_i$，表示个体学习器泛化误差的加权均值

$\overline{A}=\sum_{i=1}^{T} w_i A_i$，表示个体学习器的加权分歧值

有
$$E=\overline{E}-\overline{A}$$

这个式子显示：个体学习器精确性越高、多样性越大，则集成效果越好，这个分析称为误差-分歧分解(Error Ambiguity Decomposition)

【1995年Krogh和Vedelsby给出】

注意：

现实任务中很难直接把$\bar{E}-\bar{A}$作为优化目标进行求解，原因：(1)它们定义在整个样本空间上；(2)$\bar{A}$不是一个可直接操作的多样性度量。此外，上面的推导只适用于回归学习，难以推广到分类学习任务.

多样性

- **多样性度量(Diversity Measure)**
  - 用于度量集成中个体学习器的多样性
  - 考虑个体学习器的两两相似/不相似性

给定数据集$D = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$，对二分类任务$y_i \in \{-1, +1\}$

分类器h_i与h_j的预测结果联立表(Contingency Table)为:

	$h_i = +1$	$h_i = -1$
$h_j = +1$	a	c
$h_j = -1$	b	d

$$a + b + c + d = m$$

多样性

- **多样性度量(Diversity Measure)**
  - **不合度量(Disagreement Measure)** $dis_{ij} = \frac{b+c}{m}$

    值域[0, 1]，值越大多样性越大.
  - **相关系数(Correlation Coefficient)** $\rho_{ij} = \frac{ad-bc}{\sqrt{(a+b)(a+c)(c+d)(b+d)}}$

    值域[-1, 1]，学习器无关值为0；正相关值为正，否则为负.
  - ***Q*-统计量(Q-Statistic)** $Q_{ij} = \frac{ad-bc}{ad+bc}$

    Q_{ij}**与相关系数** ρ_{ij} **符号相同，且** $|Q_{ij}| \leq |\rho_{ij}|$ **.**
  - ***k*-统计量(Kappa-Statistic)** $\kappa = \frac{p_1 - p_2}{1 - p_2}$

    学习器完全一致值为1；偶然一致值为0，一致概率低于偶然取负值.

$$p_1 = \frac{a+d}{m}, \quad p_2 = \frac{(a+b)(a+c)+(c+d)(b+d)}{m^2}$$

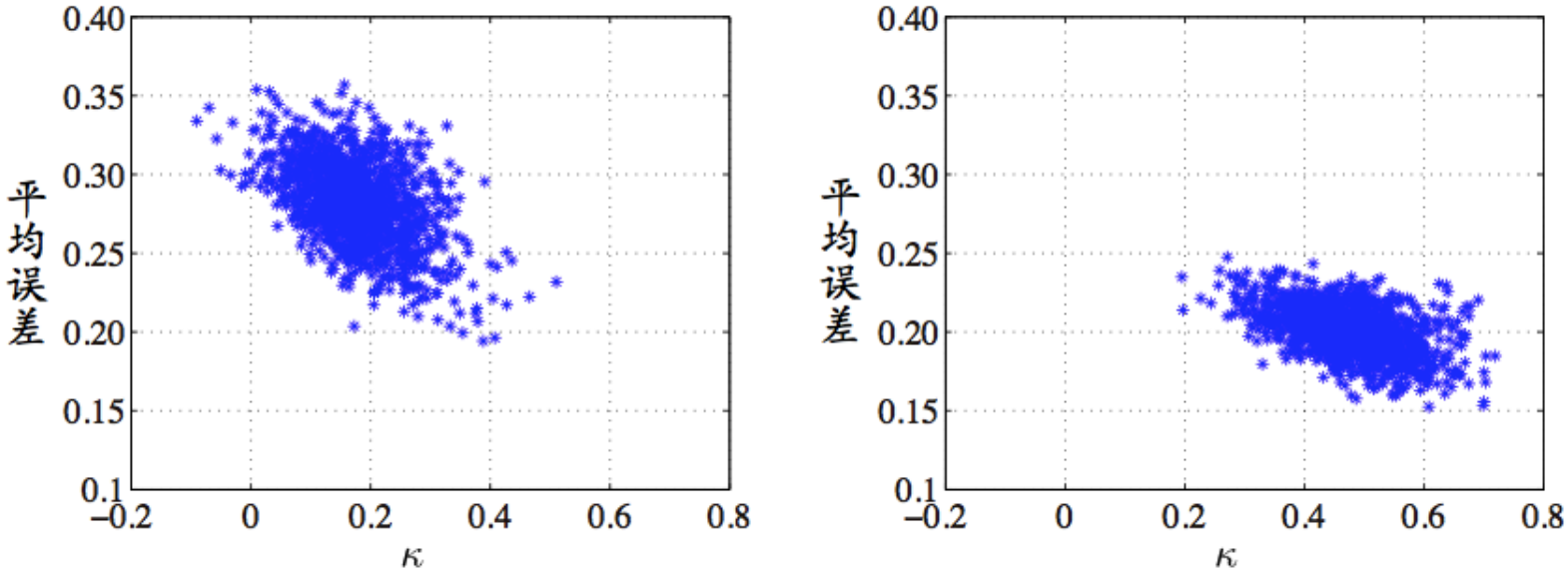
p_1**一致概率；** p_2**偶然一致概率**

多样性

- **多样性度量(Diversity Measure)**

***k*-误差图**

UCI数据tic-tac-toe图，每个集成含50棵C4.5决策树.



(a) AdaBoost 集成　　(b) Bagging 集成

横坐标是学习器的*k*值，纵坐标是其平均误差。数据点云位置越高，个体学习器准确性越低；点云位置越靠右，个体学习器多样性越小.

多样性

- 多样性增强：在学习过程引入随机性
  - 数据样本扰动
  - 输入属性扰动
  - 输出表示扰动
  - 算法参数扰动
  - 不同的多样性增强机制也可一起使用
    - Adaboost：加入了**数据样本扰动**
    - 随机森林：同时加入了**数据样本扰动和输入属性扰动**

Adaboost算法

- **推导**

基于“加性模型”(Additive Model)，即基学习器线性组合

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

最小化指数损失函数(Exponential Loss Function) :

[Friedman *et al*. 2000]

$$\ell_{\exp}(H \mid \mathcal{D}) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}}[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}]$$

Adaboost算法

若$H(x)$能令指数损失函数最小化，考虑上式对$H(x)$的偏导

$$\frac{\partial \ell_{\exp}(H \mid \mathcal{D})}{\partial H(\boldsymbol{x})} = -e^{-H(\boldsymbol{x})} P(f(\boldsymbol{x}) = 1 \mid \boldsymbol{x}) + e^{H(\boldsymbol{x})} P(f(\boldsymbol{x}) = -1 \mid \boldsymbol{x})$$

令其为零求得 $$H(\boldsymbol{x}) = \frac{1}{2} \ln \frac{P(f(x) = 1 \mid \boldsymbol{x})}{P(f(x) = -1 \mid \boldsymbol{x})}$$

因此有 $$\operatorname{sign}(H(\boldsymbol{x})) = \operatorname{sign}\left(\frac{1}{2} \ln \frac{P(f(x) = 1 \mid \boldsymbol{x})}{P(f(x) = -1 \mid \boldsymbol{x})}\right)$$

$$= \begin{cases} 1, & P(f(x) = 1 \mid \boldsymbol{x}) > P(f(x) = -1 \mid \boldsymbol{x}) \\ -1, & P(f(x) = 1 \mid \boldsymbol{x}) < P(f(x) = -1 \mid \boldsymbol{x}) \end{cases}$$

$$= \underset{y \in \{-1,1\}}{\arg\max} P(f(x) = y \mid \boldsymbol{x}) ,$$

sign($H(x)$)达到了贝叶斯最优错误率，说明指数损失函数是分类任务原来0/1损失函数的一致的替代损失函数

Adaboost算法

第一个分类器h_1是直接将基学习算法用于初始数据分布得到的，此后迭代生成h_t和α_t，当基分类器h_t基于分布D_t产生后，其权重α_t应使得$\alpha_t h_t$最小化指数损失函数

$$\begin{aligned}\ell_{\exp}\left(\alpha_t h_t \mid \mathcal{D}_t\right) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}\right] \\ &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[e^{-\alpha_t}\mathbb{I}\left(f\left(\boldsymbol{x}\right)=h_t\left(\boldsymbol{x}\right)\right)+e^{\alpha_t}\mathbb{I}\left(f\left(\boldsymbol{x}\right)\neq h_t\left(\boldsymbol{x}\right)\right)\right] \\ &= e^{-\alpha_t}P_{\boldsymbol{x}\sim\mathcal{D}_t}\left(f\left(\boldsymbol{x}\right)=h_t\left(\boldsymbol{x}\right)\right)+e^{\alpha_t}P_{\boldsymbol{x}\sim\mathcal{D}_t}\left(f\left(\boldsymbol{x}\right)\neq h_t\left(\boldsymbol{x}\right)\right) \\ &= e^{-\alpha_t}\left(1-\epsilon_t\right)+e^{\alpha_t}\epsilon_t\end{aligned}$$

其中 $\epsilon_t = P_{\boldsymbol{x}\sim\mathcal{D}_t}\left(h_t(\boldsymbol{x})\neq f(\boldsymbol{x})\right)$. 令指数损失函数的导数为0:

$$\frac{\partial \ell_{\exp}(\alpha_t h_t \mid \mathcal{D}_t)}{\partial \alpha_t} = -e^{-\alpha_t}(1-\epsilon_t)+e^{\alpha_t}\epsilon_t \Rightarrow \alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

Adaboost算法

获得H_{t-1}之后对样本分布进行调整，使下一轮基学习器h_t能纠正H_{t-1}的一些错误，理想的h_t能纠正全部错误，即最小化：

$$\begin{aligned}\ell_{\exp}(H_{t-1}+h_t \mid \mathcal{D}) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})(H_{t-1}(\boldsymbol{x})+h_t(\boldsymbol{x}))}] \\ &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}e^{-f(\boldsymbol{x})h_t(\boldsymbol{x})}]\end{aligned}$$

由于 $f^2(x)=h_t^2(x)=1$，上式用 $e^{-f(x)h_t(x)}$ 泰勒展开式近似

$$\begin{aligned}\ell_{\exp}(H_{t-1}+h_t \mid \mathcal{D}) &\simeq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1-f(\boldsymbol{x})h_t(\boldsymbol{x})+\frac{f^2(\boldsymbol{x})h_t^2(\boldsymbol{x})}{2}\right)\right] \\ &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1-f(\boldsymbol{x})h_t(\boldsymbol{x})+\frac{1}{2}\right)\right]\end{aligned}$$

Adaboost算法

于是，理想的基学习器

$$
\begin{aligned}
h_t(\boldsymbol{x}) &= \arg\min_h \ell_{\exp}(H_{t-1}+h \mid \mathcal{D}) \\
&= \arg\min_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1-f(\boldsymbol{x})h(\boldsymbol{x})+\frac{1}{2}\right)\right] \\
&= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}f(\boldsymbol{x})h(\boldsymbol{x})\right] \\
&= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}f(\boldsymbol{x})h(\boldsymbol{x})\right],
\end{aligned}
$$

注意到 $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]$ 是一个常数，令 D_t 表示一个分布：

$$
\mathcal{D}_t(\boldsymbol{x}) = \frac{\mathcal{D}(\boldsymbol{x})e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}
$$

Adaboost算法

根据数学期望的定义，这等价于令

$$h_t(\boldsymbol{x}) = \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}} \left[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]} f(\boldsymbol{x})h(\boldsymbol{x}) \right]$$

$$= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t} \left[f(\boldsymbol{x})h(\boldsymbol{x}) \right] .$$

由$f(x), h(x) \in \{-1, +1\}$，有

$$f(\boldsymbol{x})h(\boldsymbol{x}) = 1 - 2\,\mathbb{I}\big(f(\boldsymbol{x}) \neq h(\boldsymbol{x})\big)$$

则理想的基学习器

$$h_t(\boldsymbol{x}) = \arg\min_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t} \left[\mathbb{I}\big(f(\boldsymbol{x}) \neq h(\boldsymbol{x})\big) \right]$$

理想的h_t将在分布D_t下最小化分类误差.

Adaboost算法

弱分类器基于分布D_t训练，且针对D_t的分类误差应小于0.5。这在一定程度上类似“残差逼近”的思想。考虑到D_t和D_{t+1}的关系，有

$$\begin{aligned}\mathcal{D}_{t+1}(\boldsymbol{x}) &= \frac{\mathcal{D}(\boldsymbol{x})\, e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]} \\ &= \frac{\mathcal{D}(\boldsymbol{x})\, e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})} e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]} \\ &= \mathcal{D}_t(\boldsymbol{x}) \cdot e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})} \frac{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]}\end{aligned}$$

$$\begin{aligned}\mathcal{D}_{t+1}(\boldsymbol{x}) &= \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases} \\ &= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}\end{aligned}$$