模式识别

主讲：崔林艳&邹征夏

单位：宇航学院

开课专业：飞行器控制与信息工程

第四章 非线性分类器

CONTENTS PAGE

4.5 分类器集成的基本原理

4.5.1 弱分类器概念

4.5.2 个体与集成

4.5.3 集成学习法特点

4.5 分类器集成的基本原理

4.5.1 弱分类器概念

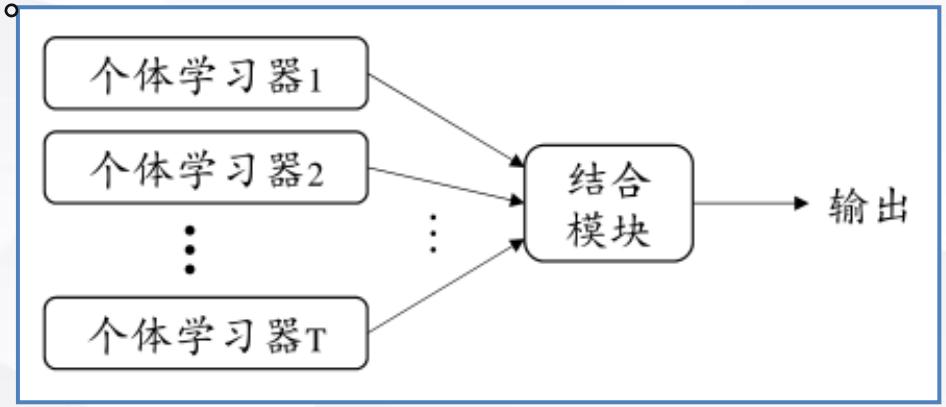
- 通常情况下，对于二分类问题，弱分类器是指识别错误率略小于1/2，即准确率仅比随机猜测略高的分类器。
- 而强分类器是指识别准确率很高，并且能在多项式时间内完成的分类器。
- 问题：仅有一些弱分类器，能否获得强分类器的结果？

注：多项式时间指的是解决（计算）问题需要的时间（复杂度）与问题的规模之间是多项式关系

4.5 分类器集成的基本原理

4.5.2 个体与集成

- 在有监督学习算法中，有时很难学习出一个稳定且在各个方面表现都较好的模型，而是只能得到多个有偏好的模型（仅在某些方面表现比较好，也叫弱监督模型）。
- 集成学习：组合多个弱监督模型以期得到一个更好更全面的强监督模型。
- 分类器集成，其实就是集成学习，通过构建并结合多个弱分类器来完成分类任务。一般结构是：先产生一组“个体学习器”，再用某种策略将它们结合起来。



- 集成学习潜在的思想：即便某一个弱分类器得到了错误的预测，其他的弱分类器也可以将错误纠正回来，从而提升分类性能。

三个臭皮匠顶个诸葛亮

4.5 分类器集成的基本原理

4.5.2 个体与集成

➢ 由个体学习器集成的算法分为两类

- 同质：集成中只包含同种类型的个体学习器，其中的个体学习器亦称为“基学习器”，相应的算法称为“基学习算法”。例如：“决策树集成”仅包含决策树，“神经网络集成”全是神经网络。

- 异质：集成中包含不同类型的个体学习器，其中的个体学习器称为“组建学习器”。

要获得好的集成，个体学习器应“好而不同”，即个体学习器要有一定的“准确性”并且要有多样性（个体学习器间具有差异）。

4.5 分类器集成的基本原理

4.5.2 个体与集成

➢ 举例分析

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图所示，其中√ 表示分类正确，X 号表示分类错误，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

(a)：每个分类器只有66.6%的精度，集成学习达到了100%的精度
(b)：三个分类器相同，导致集成性能没有提高
(c)：每个分类器只有33.3%的精度，导致集成学习的效果变得更糟

集成个体应：好而不同

4.5 分类器集成的基本原理

4.5.2 个体与集成

➢ 错误率分析

考虑二分类问题$y \in \{+1,-1\}$和真实分类器$f(x)$，**假设基分类器$h_i(x)$的错误率均为ε**：

$$P\left(h_i\left(x\right) \neq f\left(x\right)\right) = \varepsilon$$

假设集成策略采用简单投票法结合T个（设为奇数）分类器，票多者胜：

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

假设基分类器的误差相互独立，由Hoeffding不等式可知，则集成分类器H错误率为：

$$P\left(H\left(x\right) \neq f\left(x\right)\right) = \sum_{t=0}^{(T-1)/2} C_T^t \varepsilon^{T-t}\left(1-\varepsilon\right)^t \leq \exp\left(-\frac{1}{2}\left(1-2\varepsilon\right)^2 T\right)$$

如果基分类器的误差相互独立，随着模型数量T的增加，集成模型的错误率将指数级下降，并逐渐趋近于0

4.5 分类器集成的基本原理

4.5.2 个体与集成

$$P\left(H\left(x\right)\neq f\left(x\right)\right)=\sum_{t=0}^{(T-1)/2}C_T^t\varepsilon^{T-t}\left(1-\varepsilon\right)^t\leq\exp\left(-\frac{1}{2}\left(1-2\varepsilon\right)^2T\right)$$

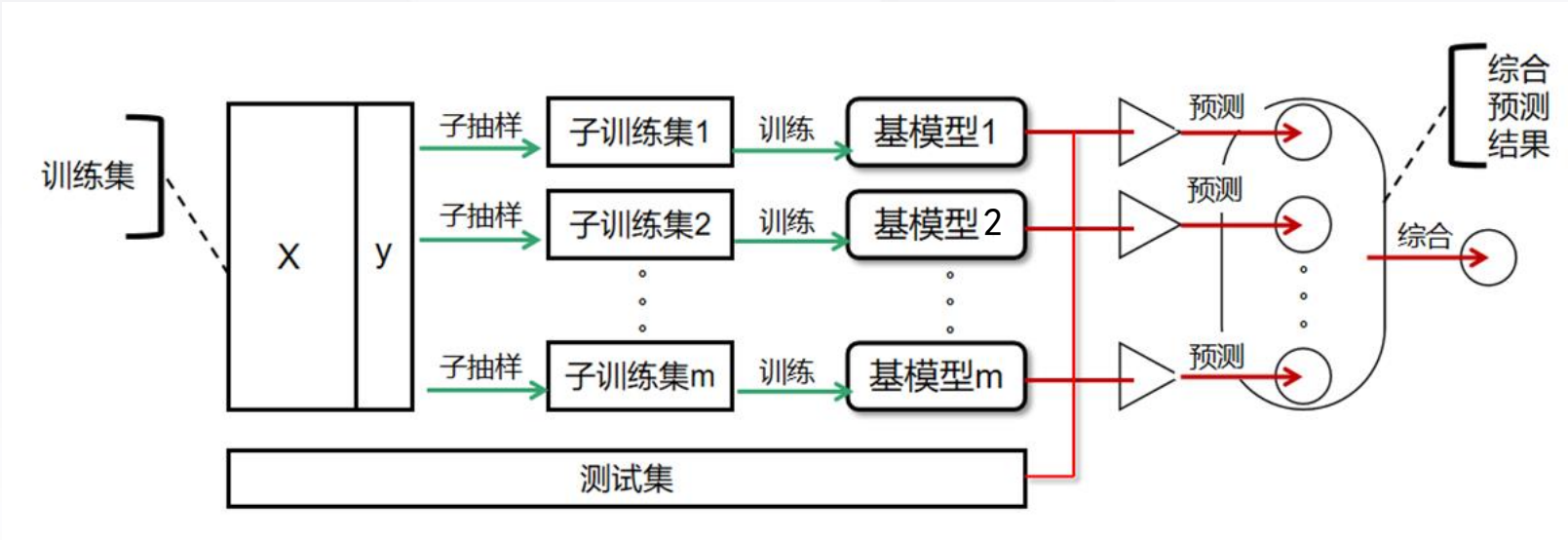
- 上面分析有一个关键假设：基分类器的误差相互独立；
- 现实任务中，个体分类器（基分类器）是为解决同一个问题训练出来的，显然不可能互相独立；
- 事实上，个体分类器的“准确性”和“多样性”本身就存在冲突。一般地，准确性很高之后，要增加多样性就需牺牲准确性。
- 如何产生并结合“好而不同”的个体分类器是集成学习研究的核心。

4.5 分类器集成的基本原理

4.5.2 个体与集成

根据个体学习器间依赖关系，目前集成学习方法大致可分为两类：

① 并行化方法：个体学习器间不存在强依赖关系、可并行生成的方法。代表算法：Bagging和随机森林。



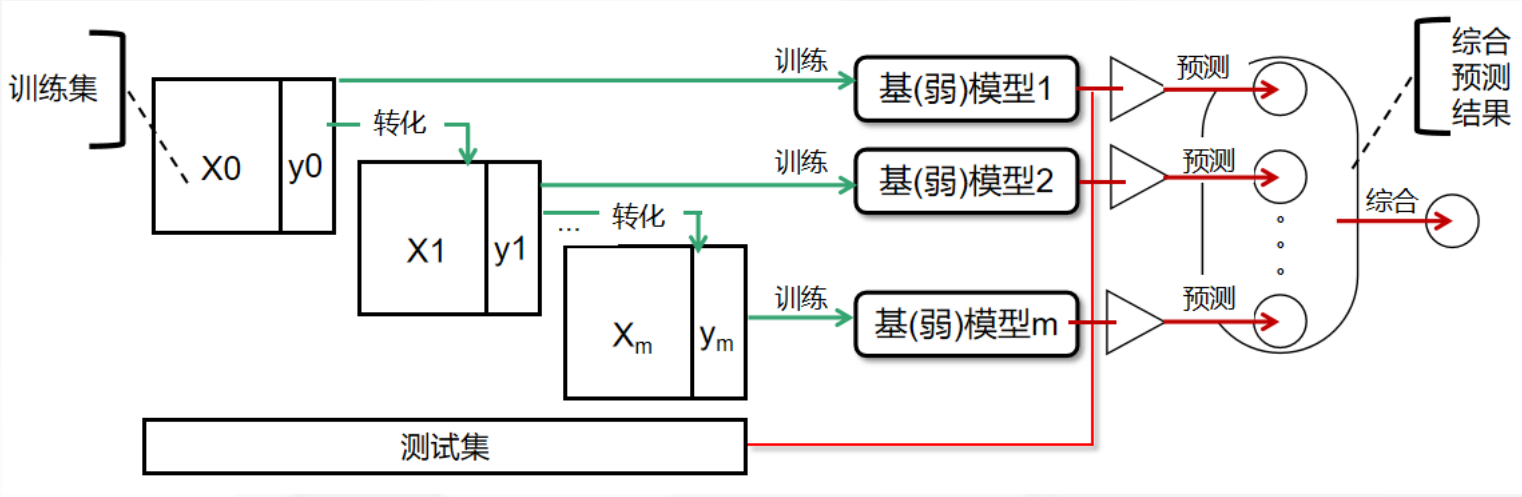
基本思想：对训练集进行子抽样，组成每个基模型所需要的子训练集，对所有基模型预测的结果进行综合产生最终的预测结果。通过不同模型的训练数据集的独立性来提高不同模型之间的独立性。

4.5 分类器集成的基本原理

4.5.2 个体与集成

根据个体学习器间依赖关系，目前集成学习方法大致可分为两类：

② 序列化方法：个体学习器间存在强依赖关系、必须串行生成的方法。代表算法：Boosting算法；该类方法除了训练第一个学习器之外，其他的学习器学习都需要依赖于前面生成的学习结果。



基本思想：训练过程为阶梯状，基模型按次序一一进行训练，基模型的训练集按照某种策略每次都进行一定的转化。对所有基模型预测的结果进行线性综合产生最终的预测结果。

4.5 分类器集成的基本原理

4.5.3 集成学习法特点

a) 将多个分类方法聚集在一起，以提高分类的准确率。这些分类算法可以是不同的算法，也可以是相同的算法。

b) 集成学习法由训练数据构建一组基分类器，然后通过对每个基分类器的预测进行投票来进行分类；

c) 严格来说，集成学习并不算是一种分类器，而是一种分类器结合的方法。

d) 通常一个集成分类器的分类性能会好于单个分类器；

第四章 非线性分类器

CONTENTS PAGE

4.6 随机森林

4.6.1 Bagging

4.6.2 随机森林

4.6.3 小结

4.6 随机森林

4.6.1 Bagging

- Bootstrap Aggregating缩写；

- 基本思想：

  - 在原始训练集上进行有放回的随机采样，得到若干个比较小的训练集；利用这些训练集，训练若干个模型；最后通过投票的方法进行模型集成。
  - 利用不同模型的训练数据集的独立性来提高不同模型之间的独立性。

4.6 随机森林

4.6.1 Bagging

➢ 算法基本流程

步骤1：训练样本生成过程：自助采样法（bootstrap sampling）

给定包含n个样本的数据集，采用有放回的采样方式。

a) 每次随机取出一个样本放入采样集，再把样本放回到初始数据集，这样下次采样时该样本仍有可能被选中；

b) 经过n次随机采样操作，即可得到含n个样本的子采样集；初始训练集中有的样本在采样集中多次出现，有的则没有出现；

c) 按照这种方式可以采样出T个包含n个样本的子采样集。

4.6 随机森林

4.6.1 Bagging

➢ 算法基本流程

思考：为什么采用BootStrap抽样？

a) 如果不进行随机抽样，每个基学习器的训练集都一样，则最终训练出的每个基分类器的分类结果也一样；

b) 为什么要有放回地抽样？如果不是有放回的抽样，那么每个基学习器的训练样本都不同，没有交集。此时每个基学习器都是“有偏的”、“片面的”，每个基学习器训练出来都有很大的差异；而Bagging最后分类取决于多个基学习器的投票表决。

4.6 随机森林

步骤2：基于每个采样集训练出对应模型，再把这些模型进行集成。若为分类，则利用投票方法；若为回归预测，则可用多个弱学习器得到的回归结果进行算术平均作为最终的模型输出。

- Bagging算法流程如下：

输入：样本集$D=\{(x_1,y_1),(x_2,y_2),.....,(x_n,y_n)\}$

弱分类器算法，弱分类器数量为T

输出：最终的强分类器 $f(x)$

①对于t=1，2，…，T：

a) 子采样集生成：D_t。

b) 用子采样集D_t训练第 t个弱分类器$G_t(x)$

②对T个弱分类器采用投票方法，投出最多票数的类别或者类别之一为最终类别。

4.6 随机森林

4.6.2 随机森林

- 随机森林（Random Forests）是通过集成学习的方法将多棵决策树集成的一种算法，该分类器最早由Leo Breiman和Adele Cutler提出，并被注册成了商标。

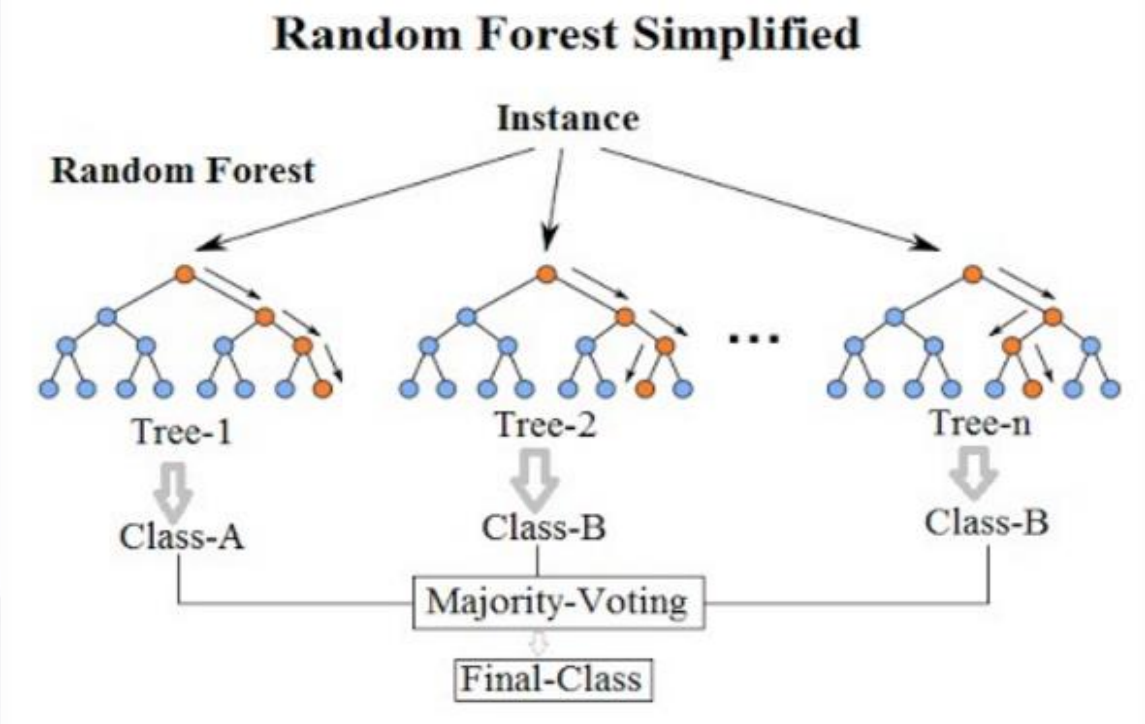
- 随机森林是一种特殊的Bagging 方式, 将决策树用作Bagging中的模型。并且对Bagging进行了改进：不但对样本进行采样，而且也对属性（特征）进行采样。

- 随机森林方法结合 Breimans 的“Bootstrap aggregating”想法和 Ho 的“random subspace method”建造决策树集合。

4.6 随机森林

4.6.2 随机森林

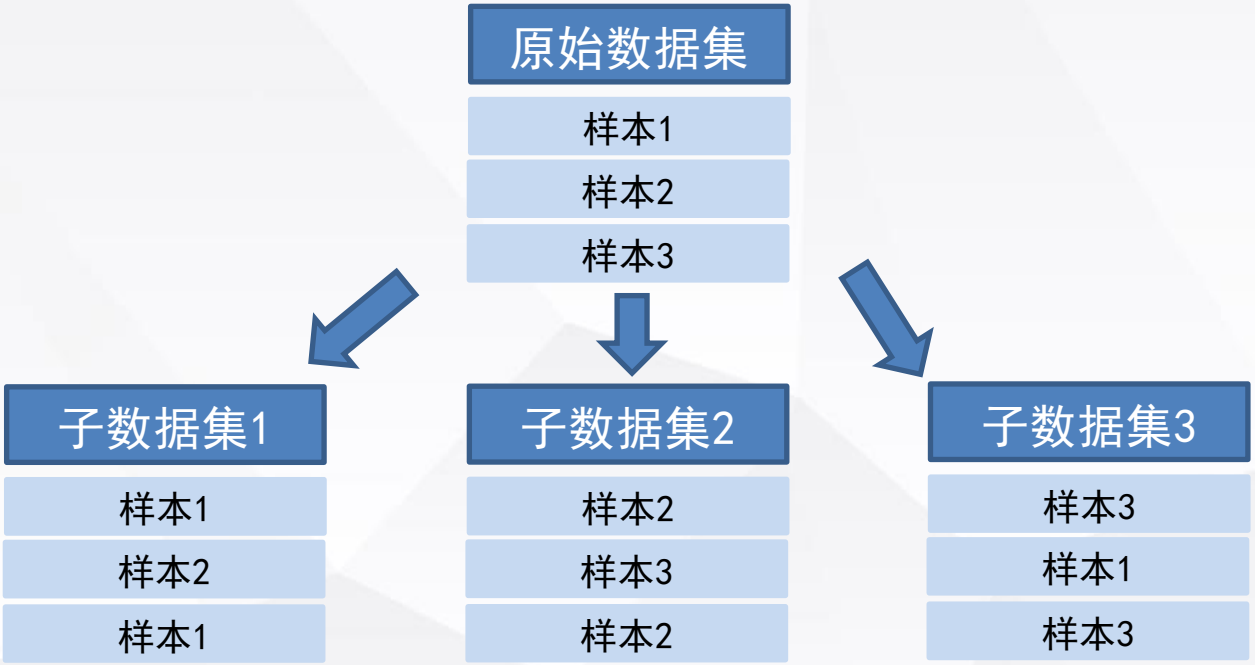
每一棵决策树都是一个分类器，对于每一个输入样本，每棵决策树会有1个分类结果。而随机森林集成了所有的分类投票结果，将投票次数最多的类别指定为最终的输出。



4.6 随机森林

4.6.2 随机森林：一般创建步骤

① **构建子数据集：**样本容量为N的样本集，有放回的抽取N次，每次抽取1个，最终形成了N个样本（样本有可能重复）组成的子数据集（即bootstrap取样）。这N个样本用来训练一棵决策树。

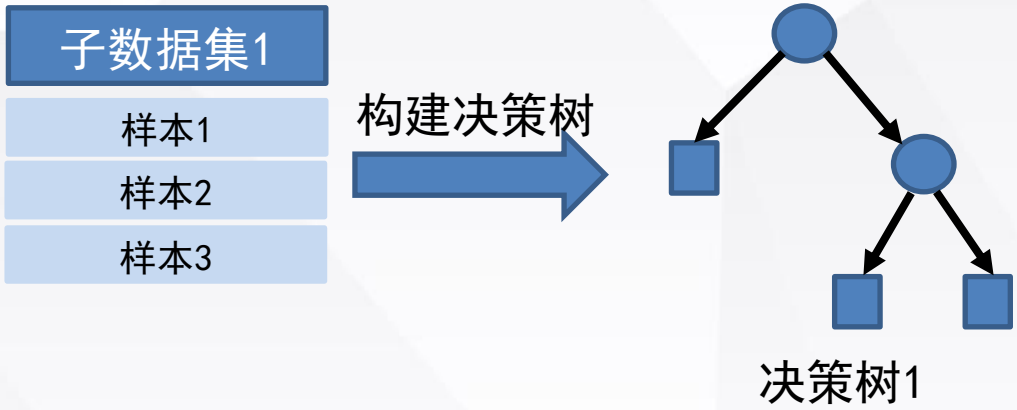


子数据集的数据量与原始数据集相同，不同子数据集中样本可以重复，同一子数据集中样本也可以重复。

4.6 随机森林

4.6.2 随机森林：一般创建步骤

② 利用子数据集构建决策树。

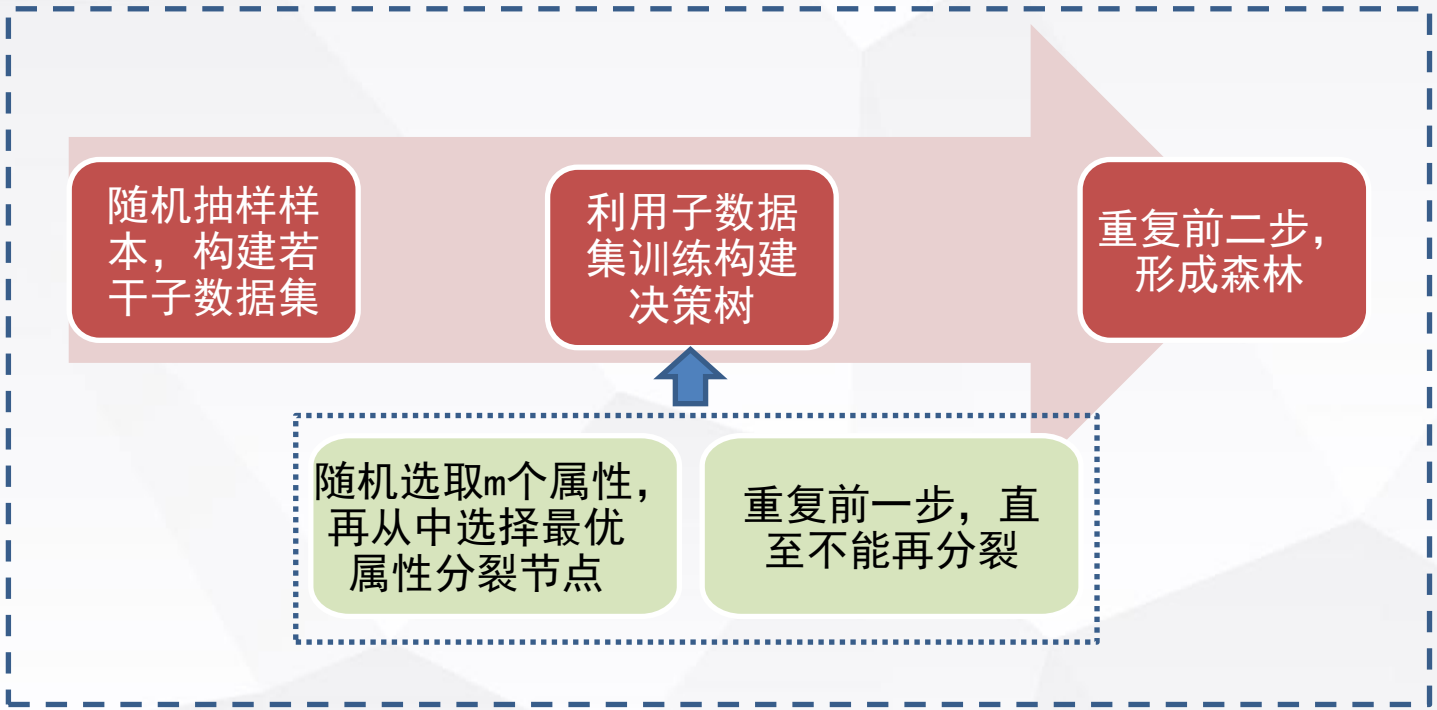


a) 当每个样本有M个属性，在决策树每个节点需要分裂时，随机从这M个属性中选取出m个属性，满足条件$m<<M$。然后从这m个属性中采用某种策略（例如信息增益）来选择1个最优属性作为该节点的分裂属性，进行分裂操作。

b) 决策树形成过程中每个节点均按照步骤a)进行分裂，直到不能再分裂为止。注意整个决策树形成过程中没有进行剪枝。

4.6.2 随机森林：一般创建步骤

③ 按照步骤1-2建立大量的决策树，构成随机森林。



随机森林构建流程

4.6.2 随机森林：一般创建步骤

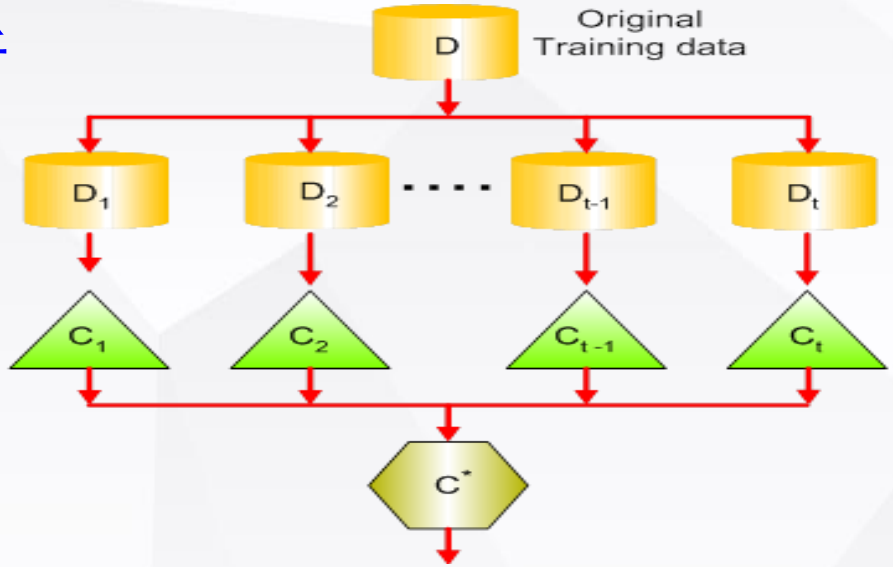
- **对新样本数据进行分类：**

通过对子决策树的分类判别结果进行投票，得到随机森林的判别结果。

例如：假设随机森林中包含3棵子决策树，其中2棵子决策树对该新样本的分类结果为A类，1棵子决策树对该新样本的分类结果为B类，则随机森林对该新样本的分类结果为A类。

4.6 随机森林

4.6.2 随机森林：分类效果分析



影响随机森林分类效果的因素包括：

① 森林中任意两棵树的相关性：相关性越大，错误率越大；

② 森林中每棵树的分类能力：每棵树分类能力越强，整个森林的错误率越低；

③ 减少特征m的个数，树的相关性和分类能力都会相应下降；增大m，两者也会随之增大。所以随机森林的一个关键问题就是如何选择最优的m，这也是随机森林唯一的参数。

4.6.3 小结

随机森林的优点

① 准确率高；

② 能够运行在大数据集上；

③ 能够处理具有高维特征的样本输入，不需要降维（随机选取m个属性构建决策树）；

④ 能评估各个特征在分类问题上的重要性；

⑤ 对于缺省值问题，也能够获得好的结果。即具有良好抗噪能力。

4.6.3 小结

随机森林问题分析：

① 随机森林对于分类工作表现得很好，但是对于回归问题则表现一般，因为它无法给出一个精确连续的预测值。

② 随机森林包含大量随机产生的决策树，构建过程难以有效控制，只能通过尝试不同的参数来调整，可解释性差。

第四章 非线性分类器

CONTENTS PAGE

4.7 Boosting方法

4.7.1 PAC可学习理论

4.7.2 Boosting方法

4.7.3 AdaBoost

4.7.4 集成策略讨论

4.7 Boosting方法

4.7.1 PAC可学习理论

- 最初的Boosting算法产生于计算学习理论中的PAC学习。
- 计算学习理论中的PAC（Probably Approximately Correct）学习

  - 如果存在一种算法能够以很大的概率，学到一个很高的精确度，并且在多项式时间内完成（时间是输入变量、概率、和精确度的多项式函数），则该模型是PAC可学习的。
  - PAC学习的本意是对各种模型进行分类，找出哪些模型是PAC可学习的。
  - 实际上，由于需要模型对任意高的概率和精确度都可学习，只有很少的模型是属于PAC可学习范畴的“强”可学习模型。“弱”可学习模型只能保证以任意高的概率学到比随机猜好一点（在多项式时间内完成）。

4.7 Boosting方法

4.7.2 Boosting方法

➢ 发展历史

- 在实践中，获得一个弱学习算法容易，而获得强学习算法要困难得多。
- 1984年，Valiant 和 Kearns 首次提出了 PAC 学习模型中弱学习算法和强学习算法的等价性问题，即任意给定仅比随机猜测略好的弱学习算法，是否可以将其提升为强学习算法？
- 1990 年，Schapire 证明了：“弱”可学习模型与“强”可学习模型这两个概念是等价的。说明，任何弱学习算法都可以被组合成一个强学习算法。这个证明就是最初的Boosting算法。

4.7 Boosting方法

4.7.2 Boosting方法

➢ 概述

- Boosting是一簇可将弱学习算法转化成强学习算法的方法。
- Boosting按照一定的顺序来训练不同的基模型，每个模型都针对前序模型的错误进行专门训练，从而增加不同基模型之间的差异性。
- Boosting是一种非常强大的集成方法，只要基模型的准确率比随机猜测高，就可以通过集成方法来显著地提高集成模型的准确率。

4.7 Boosting方法

4.7.2 Boosting方法

- 工作机制

① 首先，从初始训练集训练出一个弱分类器1，根据弱分类器1的表现对训练样本分布进行调整，使得之前弱分类器1识别错误的训练样本在后面得到更多的重视。

② 然后，基于调整后的训练集来训练弱分类器2，如此重复进行，直到弱分类器数达到事先指定的数目T；

③ 最终，将这T个弱分类器通过集成策略进行整合，得到最终的强分类器。

4.7 Boosting方法

4.7.2 Boosting方法

- 特点分析

① Boosting系列算法可以提高任意给定学习算法准确度；

② **训练过程为阶梯状**，弱分类器按次序一一进行训练，弱分类器的训练集按照某种策略每次都进行一定的转化；

③ Boosting中所有的**弱分类器可以是不同类的分类器**。

- 关键问题

Boosting方法需要解决两个问题：

① 如何调整训练集用于训练弱分类器？

② 如何将训练得到的各个弱分类器联合起来形成强分类器？

4.7 Boosting方法

4.7.3 AdaBoost方法

AdaBoost（Adaptive Boosting，自适应增强）算法是Boosting方法的一种，针对Boosting中的两个关键问题分别采用以下策略：

① 针对关键问题1，使用加权后选取的训练数据代替随机选取的训练样本。提高那些被前一轮弱分类器错误分类样本的权值，并降低那些被正确分类的样本的权值。经过该操作，后一轮的弱分类器会更关注那些没有被正确分类的样本。

② 针对关键问题2，将弱分类器联合起来，使用加权投票机制代替平均投票机制。让分类效果好的弱分类器具有较大的权重，而分类效果差的分类器具有较小的权重。

4.7 Boosting方法

4.7.3 AdaBoost方法

➢ AdaBoost算法原理

考虑一个两分类问题，数据x是一个n维向量，$y \in \{-1,+1\}$代表类别。现在有m个训练样本，用$(x_1,y_1),(x_2,y_2),...,(x_m,y_m)$来表示。在每一轮迭代中，通过调用弱学习算法，产生一个新的弱分类器。

定义$D_t(i)$代表第t轮迭代过程中第i个样本点的权重。因为最初的m个训练样本是固定的，所以需要在每一轮的迭代中，调整每一个样本点的权重来产生一个新的样本点分布，作为下一轮迭代的输入。

4.7.3 AdaBoost方法

➢ AdaBoost算法原理

步骤1：初始化样本集中所有样本点的权重

$$D_1(i)=\frac{1}{m},i=1,2,\cdots,m$$

每一个训练样本最开始时都被赋予相同的权值

步骤2：训练弱分类器

a） 调用弱学习算法，产生一个基分类器h_t。

b） 对于样本集中的每一个点，把基分类器h_t的分类结果与样本的实际类别y进行比较，当 $h_t(\boldsymbol{x}_i)\neq y_i$ ，判定为错分样本。

c） 计算该基分类器h_t的错误率ε_t；

d） 更新样本集中所有样本点权重，产生新的$D_{t+1}(i)$ 分布。用于下次迭代。

4.7 Boosting方法

A. 每一轮迭代产生的基分类器应用于样本集分类所产生的错误率ε_t：

$$\varepsilon_t = \sum_{i=1}^{m} D_t(i) I(h_t(\boldsymbol{x}_i) \neq y_i) / \sum_{i=1}^{m} D_t(i)$$

（误分类样本的权值之和）

其中$I(.)$为指示函数，当括号里的表达式成立时I为1，否则为0。

同时，计算基分类器h_t的权重系数α_t：（用于后续加权投票集成）

$$\alpha_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$$

当$\varepsilon_t = 1/2$时，$\alpha_t = 0$。根据Boosting思想，基分类器h_t的错误率应小于50%，因此α_t是大于0的值。ε_t越小，则表示基分类器h_t的分类性能越好，权重α_t越大。

4.7 Boosting方法

B. 更新样本集中所有样本点的权重，产生新的 D_{t+1} 分布：

更新原则：减小分类正确的样本点的权重，增大分类错误的样本点的权重。此时，在下一轮迭代中，上一轮被分错的样本点将被重视。

$$D_{t+1}(i)=\begin{cases}D_t(i)\exp\left(-\alpha_t\right), h_t(\boldsymbol{x}_i)=y_i\\ D_t(i)\exp\left(\alpha_t\right), h_t(\boldsymbol{x}_i)\neq y_i\end{cases}$$

又因

$$y_i h_t(\boldsymbol{x}_i)=\begin{cases}+1, h_t(\boldsymbol{x}_i)=y_i & \text{分类结果与实际类别一致，则为+1；}\\ -1, h_t(\boldsymbol{x}_i)\neq y_i & \text{分类结果与实际类别不一致，则为−1；}\end{cases}$$

样本点权重公式可进一步化简为：

$$D_{t+1}(i)=D_t(i)\exp\left[-\alpha_t y_i h_t(\boldsymbol{x}_i)\right]$$

4.7.3 AdaBoost方法

➢ AdaBoost算法原理

步骤3：在全部的T轮迭代完成之后，定义最终的强分类器为H，则把迭代过程中的所有基分类器h_t按照一定的规则组合起来

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

组合规则：加大分类误差率小的弱分类器的权重，使其在最终的分类函数中起着较大的决定作用，而降低分类误差率大的弱分类器的权重，使其在最终的分类函数中起着较小的决定作用。

即对于一个样本点x，其分类结果就是$H(x)$。

4.7.3 AdaBoost方法

➢ AdaBoost算法的另一种解释

实际上，AdaBoost算法也可以看作一种分步（Stage-Wise）优化的加性模型，其损失函数定义为：

$$\mathcal{L}(H) = \exp(-yH(\boldsymbol{x}))$$

$$= \exp\left(-y\sum_{t=1}^{T}\alpha_t h_t(\boldsymbol{x})\right)$$

$y \in \{+1, -1\}$是样本$\boldsymbol{x}$对应的真实标签，$H(\boldsymbol{x})$为组合后的分类器

假设经过t-1次迭代，得到：

$$H_{t-1}(\boldsymbol{x}) = \sum_{i=1}^{t-1}\alpha_i h_i(\boldsymbol{x})$$

则第t次迭代的目标是找到α_t和$h_t(\boldsymbol{x})$使得下面的损失函数最小

$$\mathcal{L}(\alpha_t, h_t(\boldsymbol{x})) = \sum_{i=1}^{m}\exp\left(-y_i\left(H_{t-1}(\boldsymbol{x}_i) + \alpha_t h_t(\boldsymbol{x}_i)\right)\right)$$

m表示样本数量

4.7.3 AdaBoost方法

$$\mathcal{L}(\alpha_t, h_t(\boldsymbol{x})) = \sum_{i=1}^{m} \exp\left(-y_i \left(H_{t-1}(\boldsymbol{x}_i) + \alpha_t h_t(\boldsymbol{x}_i)\right)\right)$$

- AdaBoost算法的另一种解释

令 $D_t(i) = \exp\left(-y_i H_{t-1}(\boldsymbol{x}_i)\right)$ ，则损失函数可写为

$$\begin{aligned}\mathcal{L}(\alpha_t, h_t(\boldsymbol{x})) &= \sum_{i=1}^{m} D_t(i) \exp\left(-y_i \alpha_t h_t(\boldsymbol{x}_i)\right) \\ &= \sum_{y_i = h_t(\boldsymbol{x}_i)} D_t(i) \exp\left(-\alpha_t\right) + \sum_{y_i \neq h_t(\boldsymbol{x}_i)} D_t(i) \exp\left(\alpha_t\right) \\ &\propto \left(1-\varepsilon_t\right) \exp\left(-\alpha_t\right) + \varepsilon_t \exp\left(\alpha_t\right)\end{aligned}$$

其中 ε_t 是基分类器 $h_t(\boldsymbol{x})$ 的错误率

$$\varepsilon_t = \frac{\sum_{y_i \neq h_t(\boldsymbol{x}_i)} D_t(i)}{\sum_{i=1}^{m} D_t(i)}$$

4.7.3 AdaBoost方法

➢ AdaBoost算法的另一种解释

$$\begin{aligned}\mathcal{L}(\alpha_t, h_t(\boldsymbol{x})) &= \sum_{i=1}^{m} D_t(i)\exp\left(-y_i\alpha_t h_t(\boldsymbol{x}_i)\right) \\ &= \sum_{y_i=h_t(\boldsymbol{x}_i)} D_t(i)\exp(-\alpha_t) + \sum_{y_i\neq h_t(\boldsymbol{x}_i)} D_t(i)\exp(\alpha_t) \\ &\propto (1-\varepsilon_t)\exp(-\alpha_t) + \varepsilon_t\exp(\alpha_t)\end{aligned}$$

求上式关于α_t 的导数并令其为0，可得到 $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

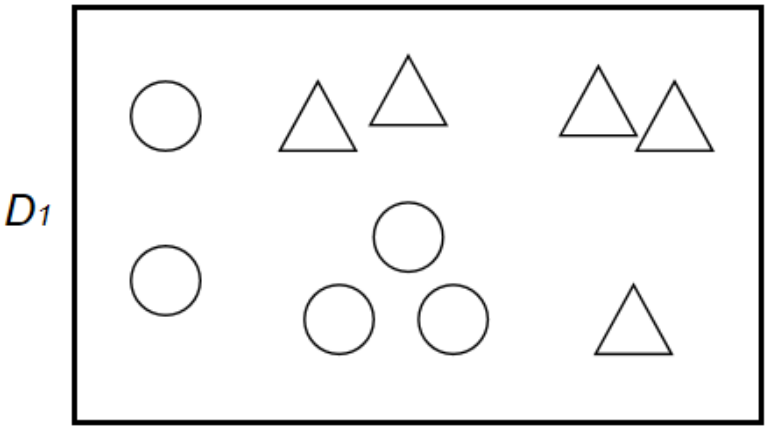
可以证明，上述推导和AdaBoost算法原理中的定义是等价的。

4.7 Boosting方法

4.7.3 AdaBoost方法

- AdaBoost算法图例分析

用一个简单的例子，来看一下Adaboost的实现过程：

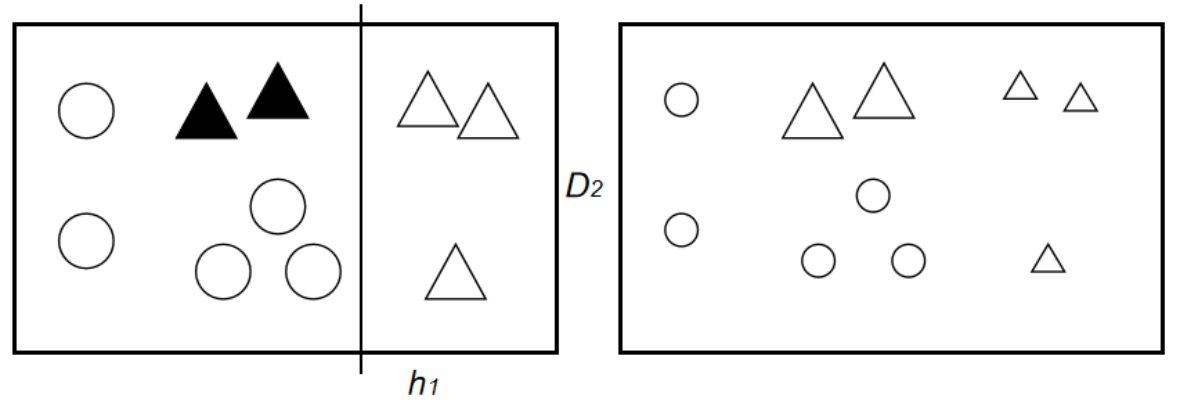


图中“△”与“○”分别表示两种类别，用水平或垂直的直线作为分类器，来进行分类。

4.7 Boosting方法

- 第一步：初始化样本集中所有样本点的权重

每个样本的初始权重$D_1(i)=1/10=0.1$，$i=1,2,\ldots,10$。



图中h_1为一个基分类器。可以看出有2个样本分类错误，故

$$\varepsilon_1 = 0.2, \alpha_1 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right) = 0.6931$$

$$D_{t+1}(i) = \begin{cases} D_t(i)\exp(-\alpha_t), h_t(\boldsymbol{x}_i) = y_i \\ D_t(i)\exp(\alpha_t), h_t(\boldsymbol{x}_i) \neq y_i \end{cases}$$

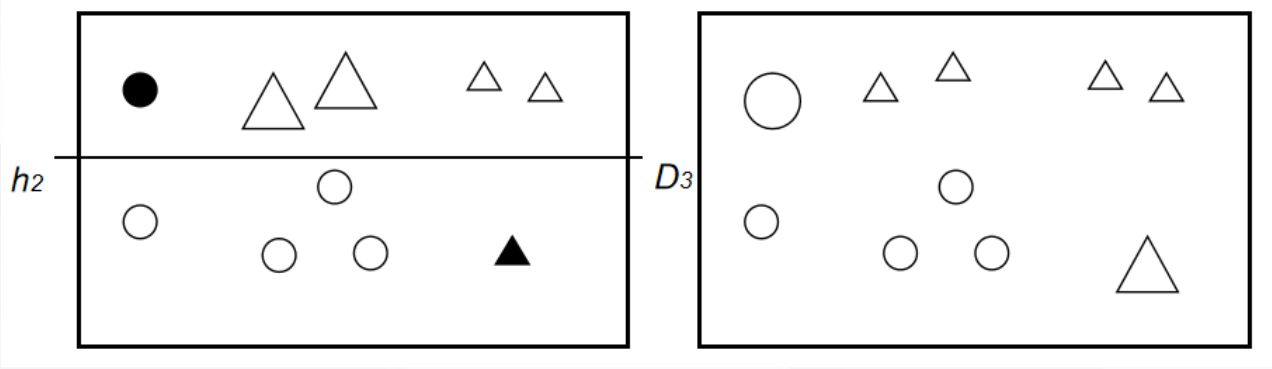
同时得到新的样本权重分布D_2：

两个被分类错误的样本，其权重更新为：$D_2 = 0.1\times e^{\alpha_1} = 0.2$

其他分类正确的样本，其权重更新为：$D_2 = 0.1\times e^{-\alpha_1} = 0.05$

4.7 Boosting方法

➢ 第二步：利用更新的样本权值，训练学习新的基分类器h_2



此时有2个样本分类错误，因此：

$$\varepsilon_t = \sum_{i=1}^{m} D_t(i) I(h_t(\boldsymbol{x}_i) \neq y_i) / \sum_{i=1}^{m} D_t(i)$$

$$\varepsilon_2 = \frac{0.0500 + 0.0500}{0.0500 \times 8 + 0.2000 \times 2} = 0.1250 \qquad \alpha_2 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_2}{\varepsilon_2}\right) = 0.9730$$

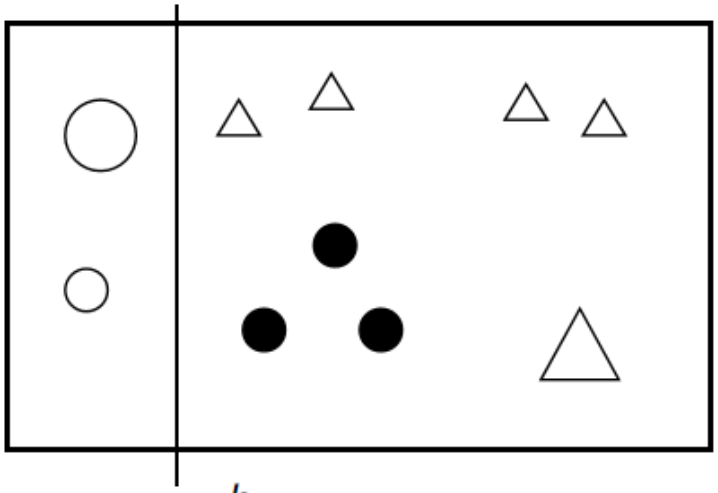
同时可以得到新的权值分布：1）对于上一轮中分类错误的样本，在这一轮中都分类正确，因此它的权重变为： $D_3 = 0.2000 \times e^{-\alpha_2} = 0.0756$

2）本轮中分类错误的样本，其权重更新为：$D_3 = 0.0500 \times e^{\alpha_2} = 0.1323$

3）其他样本，权重更新为：$D_3 = 0.0500 \times e^{-\alpha_2} = 0.0189$

4.7 Boosting方法

- 第三步：利用更新的样本权值，训练学习新的基分类器h_3



$$\varepsilon_t = \sum_{i=1}^{m} D_t(i) I(h_t(\boldsymbol{x}_i) \neq y_i) / \sum_{i=1}^{m} D_t(i)$$

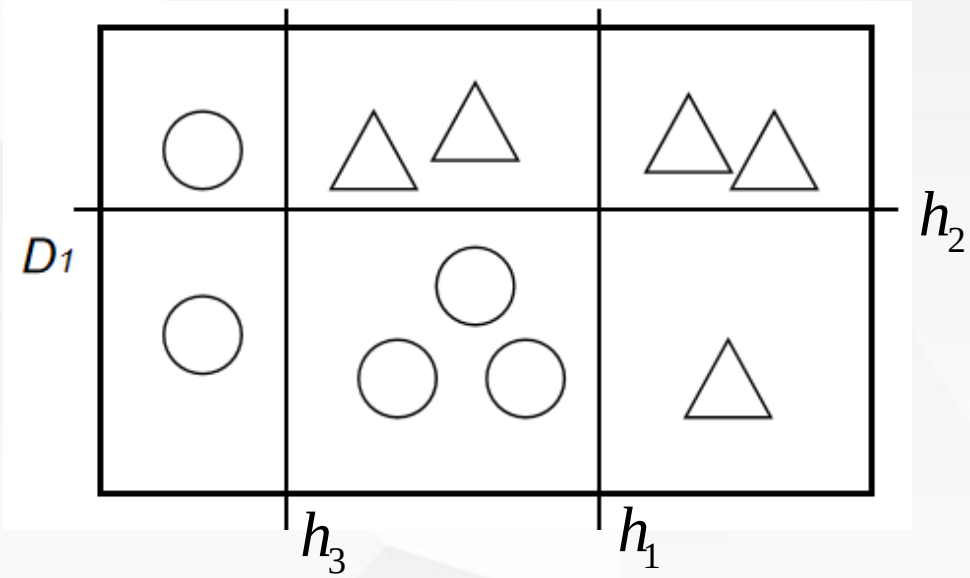
有3个样本分类错误，因此：

$$\varepsilon = 0.1071$$

$$\alpha_t = 1.0604$$

4.7 Boosting方法

- 第四步：整合所有的子分类器，可以得到最后的强分类器$H(x)$



$$H(\boldsymbol{x}) = \alpha_1 h_1(\boldsymbol{x}) + \alpha_2 h_2(\boldsymbol{x}) + \alpha_3 h_3(\boldsymbol{x})$$

$$H(\boldsymbol{x}) = 0.6931 h_1(\boldsymbol{x}) + 0.9730 h_2(\boldsymbol{x}) + 1.0604 h_3(\boldsymbol{x})$$

结论：简单的弱分类器组合起来，能得到比较好的分类效果。

4.7 Boosting方法

4.7.4 集成策略讨论

➢ 平均法

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

- 简单平均法是加权平均法的特例；
- 集成学习中的各种结果集成方法都可以看成是加权平均法的变种或特例；
- 加权平均法可认为是集成学习研究的基本出发点；
- 加权平均法未必一定优于简单平均法。

一般而言，在个体学习器相差较大时宜采用加权平均法，而在个体学习器性能相近时则采用简单平均法。

4.7 Boosting方法

4.7.4 集成策略讨论

➢ 投票法

① 绝对多数投票法

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

若某标记得票过半数，则预测为该标记；否则拒绝输出

② 相对多数投票法

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

输出为得票最多的类别，若同时有多个类别获得最高票，则从中随机选取一个输出。

③ 加权投票法

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

为每个基分类器增加权重，然后输出得票最多的类别。

4.7 Boosting方法

4.7.4 集成策略讨论

- 学习法

当训练数据很多时，一种更为强大的集成策略是使用“学习法”。即通过另一个学习器来进行结合。

学习法经典代表：Stacking

Stacking基本思路：（自学！）

集成学习小结

- 分类器集成的两大类方法
- Bagging方法
  - 随机森林算法
- Boosting方法
  - AdaBoost算法

作业

- 试分析Bagging和Boosting的区别
- 分析AdaBoost算法的优缺点