第8章 集成学习

李政伟

中国矿业大学 计算机科学与技术学院

8.1 个体与集成

■**集成学习**(ensemble learning)：通过构建并结合多个学习器来完成学习任务，有时候也被称为**多分类器系统**(multi-classifier system)、**基于委员会的学习**（committee voting method）。

■**一般结构**：先产生一组“个体学习器”（individual learner），再用某种策略将它们结合起来。

■**个体学习器**：通常由现有的学习算法从训练数据产生，如C4.5决策树算法、BP神经网络算法等。

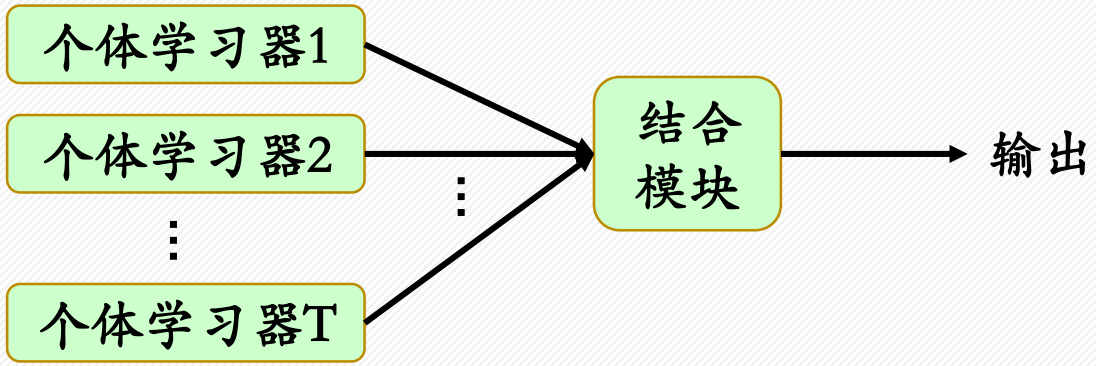


图8.1 集成学习示意图

8.1 个体与集成

同质(homogeneous)集成

- 集成中只包含同种类型的个体学习器。
- 个体学习器：又称为基学习器(base learner)。
- 对应学习算法：基学习算法(base learning algorithm)。

异质 (heterogeneous)集成

- 集成中包含不同类型的个体学习器。
- 个体学习器：由不同的学习算法生成，不再有基学习算法。
- 个体学习器：常称为组件学习器(component learner)。

Why Ensemble?

集成的泛化性能通常显著优于单个学习器的泛化性能

一个观察：

误差（曲线越低越好）

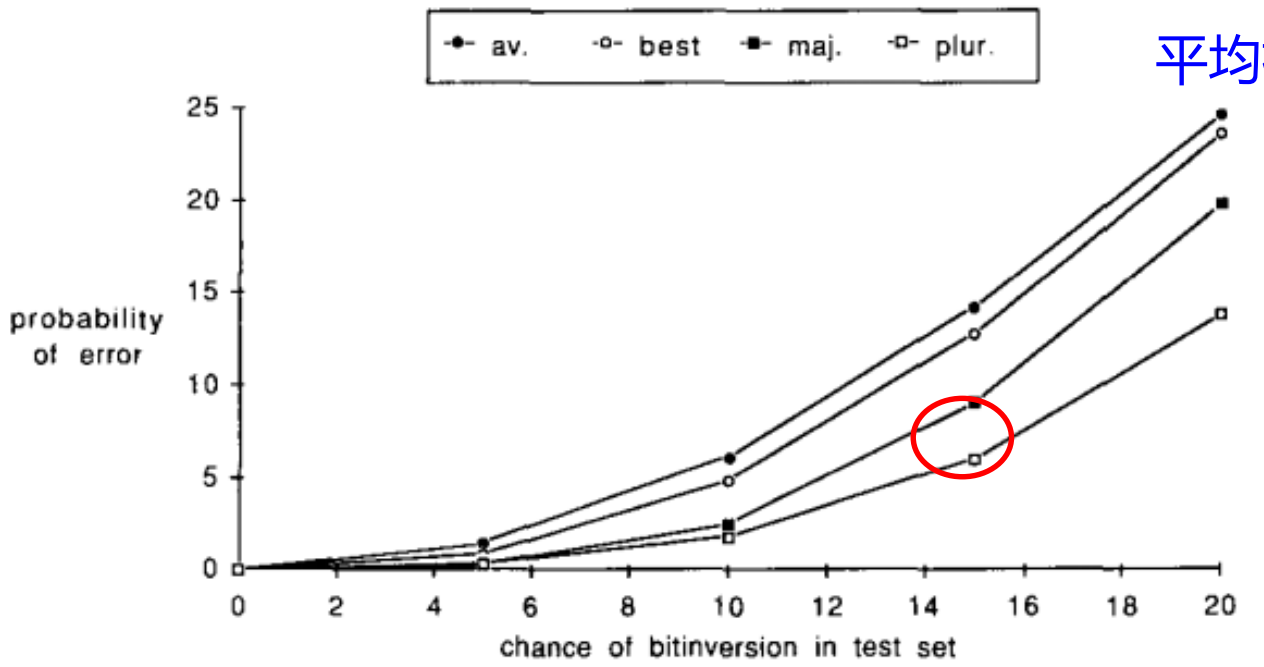


Fig. 4. Performance versus noise level in the test set is shown for individual and for consensus decisions. Data displayed shows the average and the best network, as well as collective decisions using majority and plurality for seven networks trained on individual training sets.

平均神经网络的错误率

最优神经网络的错误率

两种简单的神经网络集成算法的错误率

[Hansen & Salamon, TPAMI90]

如何得到好的集成？

	测例1	测例2	测例3
h_1	√	√	×
h_2	×	√	√
h_3	√	×	√
集成	√	√	√

(a)集成提升性能

	测例1	测例2	测例3
h_1	√	√	×
h_2	√	√	×
h_3	√	√	×
集成	√	√	×

(b)集成不起作用

	测例1	测例2	测例3
h_1	√	×	×
h_2	×	√	×
h_3	×	×	√
集成	×	×	×

(c)集成起负作用

集成的个体学习器应“好而不同”

现实各类机器学习、数据挖掘应用中，广泛使用集成学习技术

想获胜，用集成！

8.1 个体与集成

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，对每个基分类器h_i

$$P\big(h_i(\boldsymbol{x}) \neq f(\boldsymbol{x})\big) = \epsilon \tag{8.1}$$

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，则集成分类就正确:

$$H(\boldsymbol{x}) = sign\Big(\sum_{i=1}^{T} h_i(\boldsymbol{x})\Big) \tag{8.2}$$

假设基分类器的错误率相互独立，则集成的错误率为

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

$$\leq \exp\Big(-\frac{1}{2} T (1-2\epsilon)^2\Big) \tag{8.3}$$

很多成功的集成学习方法

- **序列化方法：个体学习器间存在强依赖关系**
  - **AdaBoost** [Freund & Schapire, JCSS97]
  - GradientBoost [Friedman, AnnStat01]
  - LPBoost [Demiriz, Bennett, Shawe-Taylor, MLJ06]
  - … …
- **并行化方法：个体学习器间不存在强依赖关系**
  - **Bagging** [Breiman, MLJ96]
  - **Random Forest** [Breiman, MLJ01]
  - Random Subspace [Ho, TPAMI98]
  - … …

8.2 Boosting

Boosting：一族可将弱学习器提升为强学习器的算法。

工作机制：先从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本分布进行调整，使得先前基学习器分错的训练样本在后续受到更多关注，然后基于调整后的样本分布来训练下一个基学习器；如此重复进行，直至基学习器达到事先指定的T个，最终将这T个基学习器进行加权结合。

$$h_1(x) \in \{-1,1\}$$
$$h_2(x) \in \{-1,1\}$$
$$\vdots$$
$$h_T(x) \in \{-1,1\}$$

$$H_T(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Weak classifiers

Strong classifier

Adaboost的提出

- 1990年，Schapire最先构造出一种多项式级的算法，即最初的Boost算法;
- 1993年，Drunker和Schapire第一次将神经网络作为弱学习器，应用Boosting算法解决OCR问题;
- 1995年，Freund和Schapire提出了Adaboost(Adaptive Boosting)算法，效率和原来Boosting算法一样，但是不需要任何关于弱学习器性能的先验知识，可以非常容易地应用到实际问题中。

Adaboost基本概念

AdaBoost

Adaptive Boosting

A learning algorithm

Building a strong classifier based on a lot of weaker ones

Adaboost基本概念

- **关键**：如何解决好两个问题。

1. **每一轮如何改变训练数据的权值或概率分布?**
   - **措施**：**提高**那些被前一轮弱分类器**错误分类**样本的**权值**，**降低**那些被**正确分类**样本的**权值**
2. **如何将弱分类器组合成一个强分类器?**
   - **措施**：加权多数表决，**加大分类误差率小的弱分类器**的**权值**，使其在表决中起较大的作用，**减小分类误差率大的弱分类器**的**权值**，使其在表决中起较小的作用。

AdaBoost算法

- **输入**：训练数据集$D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\dots,(\boldsymbol{x}_m,y_m)\}$，$\boldsymbol{x}_i\in\mathcal{X}\subseteq\mathbb{R}^d$，$y_i\in\mathcal{Y}=\{-1,+1\}$
- **输出**：最终分类器$H(\boldsymbol{x})$

(1) 设定弱分类器数目为T，令$t=1$，使用均匀分布初始化训练样本集的权重分布，令m维向量$\mathcal{D}_1$表示第1次需要更新的样本权重，D_1的初始值设为

$$\mathcal{D}_t=(\mathcal{D}_{t1},\mathcal{D}_{t2},\dots,\mathcal{D}_{tm})^T=(\frac{1}{m},\frac{1}{m},\dots,\frac{1}{m})^T$$

(2) 使用权重分布为$\mathcal{D}_t$的训练样本集D学习得到第t个弱学习器h_t

(3) 计算h_t在训练样本集D上的分类错误率ϵ_t

$$\epsilon_t=\sum_{i=1}^{m}\mathcal{D}_{ti}\mathbb{I}(h_t(\boldsymbol{x}_i\neq y_i))$$

如果$\epsilon_t>0.5$，则舍弃。

AdaBoost算法

(4) 确定弱学习器h_t的组合权重α_t。由于弱学习器h_t的权重与其分类性能相关，分类错误率ϵ_t越小的h_t，则其权重α_t应该越大，有

$$\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$$

(5)依据弱学习器h_t对训练样本集D的分类错误率ϵ_t，更新样本权重

$$\mathcal{D}_{t+1,i} = \frac{\mathcal{D}_{t,i}\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t}$$

其中，$Z_t = \sum_{i=1}^{N}\mathcal{D}_{t,i}\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))$ 为归一化因子，保证更新后权重向量为概率分布。

(6) 若$t < T$，则令$t = t + 1$，并返回步骤(2)，否则，执行 步骤(7)；

(7) 对于T个分类器$h_1, h_2, \ldots, h_T$，分别将每个按权重α_t进行组合，得最终分类器。

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

AdaBoost算法推导

AdaBoost——加性模型(additive model)

基学习器的线性组合

$$H(x)=\sum_{t=1}^{T}\alpha_t h_t(\boldsymbol{x}) \tag{8.4}$$

最小化指数损失函数

$$\ell_{exp}(H|\mathcal{D})=E_{x\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}] \tag{8.5}$$

若$H(x)$能令指数损失函数最小化，则式(8.5)对$H(x)$的偏导

$$\frac{\partial \ell_{exp}}{\partial H(\boldsymbol{x})}=-e^{-H(\boldsymbol{x})}P(f(\boldsymbol{x})=1|\boldsymbol{x})+e^{H(\boldsymbol{x})}P(f(\boldsymbol{x})=-1|\boldsymbol{x}) \tag{8.6}$$

令式(8.6)为零可解得

$$H(x)=\frac{1}{2}\ln\frac{P(f(\boldsymbol{x})=1|\boldsymbol{x})}{P(f(\boldsymbol{x})=-1|\boldsymbol{x})} \tag{8.7}$$

AdaBoost算法说明

因此，

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

$$= \begin{cases} 1, P(f(\boldsymbol{x})=1|\boldsymbol{x}) > P(f(\boldsymbol{x})=-1|\boldsymbol{x}) \\ -1, P(f(\boldsymbol{x})=1|\boldsymbol{x}) < P(f(\boldsymbol{x})=-1|\boldsymbol{x}) \end{cases} \tag{8.8}$$

$$= \arg\max_{y\in\{-1,1\}} P(f(\boldsymbol{x})=y|\boldsymbol{x})$$

这意味着$\operatorname{sign}(H(x))$达到了**贝叶斯最优错误率**。换言之，若指数损失函数最小化，则**分类错误率也将最小化**。

AdaBoost算法说明

$$\ell_{exp}(\alpha_t h_t|\mathcal{D}_t) = E_{x\sim\mathcal{D}_t}[e^{-f(x)\alpha_t h_t(x)}]$$

$$= E_{x\sim\mathcal{D}_t}[e^{-\alpha_t}\mathbb{I}\big(f(\boldsymbol{x}) = h_t(\boldsymbol{x})\big) + e^{\alpha_t}\mathbb{I}(f(\boldsymbol{x}) \neq h_t(\boldsymbol{x}))]$$

$$= e^{-\alpha_t}P_{x\sim\mathcal{D}_t}\big(f(\boldsymbol{x}) = h_t(\boldsymbol{x})\big) + e^{\alpha_t}P_{x\sim\mathcal{D}_t}(f(\boldsymbol{x}) \neq h_t(\boldsymbol{x})) \quad (8.9)$$

$$= e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\,\epsilon_t$$

其中 $\epsilon_t = P_{x\sim\mathcal{D}_t}(f(\boldsymbol{x}) \neq h_t(\boldsymbol{x}))$ **考虑指数损失函数的导数**

$$\frac{\partial\ell_{exp}(\alpha_t h_t|\mathcal{D}_t)}{\partial\alpha_t} = -e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\,\epsilon_t \quad (8.10)$$

令式(8.10)为零可解得

$$\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t} \quad (8.11)$$

AdaBoost算法说明

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

$$= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}e^{-f(\boldsymbol{x})h_t(\boldsymbol{x})}] \quad (8.12)$$

注意到 $f^2(\boldsymbol{x}) = h_t^2(\boldsymbol{x}) = 1$ 式(8.12)可使用$e^{-f(x)h_t(x)}$的泰勒展式近似为

$$\ell_{exp}(H_{t-1}+h_t|\mathcal{D}) \approx \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}(1-f(x)h_t(\boldsymbol{x})+\frac{f^2(\boldsymbol{x})h_t^2(\boldsymbol{x})}{2})]$$

$$= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}(1-f(x)h_t(\boldsymbol{x})+\frac{1}{2})] \quad (8.13)$$

AdaBoost算法说明

$$h_t(\boldsymbol{x}) = \arg\min_h \ell_{exp}(H_{t-1} + h|\mathcal{D})$$

$$= \arg\min_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}(1 - f(\boldsymbol{x})h(\boldsymbol{x}) + \frac{1}{2})]$$

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

$$= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{x\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]} f(\boldsymbol{x})h(\boldsymbol{x})] \quad (8.14)$$

因$\mathbb{E}_{x\sim\mathcal{D}}[e^{-f(\boldsymbol{x})(H_{t-1}(\boldsymbol{x}))}]$是常数，令

$$\mathcal{D}_t(\boldsymbol{x}) = \frac{\mathcal{D}(\boldsymbol{x})e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{x\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]} \quad (8.15)$$

则根据数学期望的定义，等价于令

$$h_t(\boldsymbol{x}) = \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]} f(\boldsymbol{x})h(\boldsymbol{x})] \quad (8.16)$$

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

AdaBoost算法说明

由$f(\boldsymbol{x}),h(\boldsymbol{x}) \in \{-1,+1\}$,有 $f(\boldsymbol{x})h(\boldsymbol{x}) = 1 - 2\mathbb{I}(f(\boldsymbol{x}) \neq h(\boldsymbol{x}))$ (8.17)

则理想的基学习器

$$h_t(\boldsymbol{x}) = \arg\min_{\boldsymbol{h}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}[\mathbb{I}(f(\boldsymbol{x}) \neq h(\boldsymbol{x})] \quad (8.18)$$

考虑到$\mathcal{D}_t$和$\mathcal{D}_{t+1}$的关系，有

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

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

$$= \frac{\mathcal{D}(\boldsymbol{x})e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}]} \quad (8.19)$$

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

Adaboost示例

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1

均匀初始化 $\mathcal{D}_1 = (\mathcal{D}_{1,1}, \mathcal{D}_{1,2}, \dots, \mathcal{D}_{1,10})$ $\mathcal{D}_{1,i} = 0.1, \quad i = 1,2,\dots,10$

对于迭代过程t=1

a、在权值分布为$\mathcal{D}_1$的数据集上，阈值取**2.5**，分类误差率最小，基分类器h_1：

$$h_1(x) = \begin{cases} 1, & x < 2.5 \\ -1, & x \geq 2.5 \end{cases}$$

b、 $h_1(x)$的误差率：$\epsilon_1 = P(h_1(x_i) \neq y_i) = 0.1 + 0.1 + 0.1 = 0.3$

c、 $h_1(x)$的系数：$\alpha_1 = \frac{1}{2}\log\frac{1-\epsilon_1}{\epsilon_1} = 0.4236$

$f_1(x) = 0.4236 h_1(x)$

Adaboost示例（续）

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
D_1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

确定阈值的取值及误差率

- 当阈值取2.5时，误差率为0.3。即 x<2.5 时取 1，x>2.5 时取 -1，则数据6、7、8分错，误差率为0.3（简单理解：10个里面3个错的 ）
- 当阈值取5.5时，误差率最低为0.4。即 x<5.5 时取1，x>5.5 时取 -1，则数据3、4、5、6、7、8分错，错误率为0.6>0.5，故反过来，令 x>5.5 取 1，x<5.5 时取 -1，则数据0、1、2、9分错，误差率为0.4
- 当阈值取8.5时，误差率为0.3。即 x<8.5 时取1，x>8.5 时取 -1，则数据3、4、5分错，错误率为0.3
- 因此，阈值取2.5 或8.5时，误差率等值，所以可以任选一个作为基本分类器。这里选2.5为例。

Adaboost示例（续）

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
$h_1(x)$	1	1	1	-1	-1	-1	-1	-1	-1	-1
分类结果	对	对	对	对	对	对	错	错	错	对

d、更新训练数据的权值分布

$$(\mathcal{D}_{2,1}, \mathcal{D}_{2,2}, \ldots, \mathcal{D}_{2,10})$$

$$\mathcal{D}_{2,i} = \frac{\mathcal{D}_1(x)}{Z_1} \times \begin{cases} \exp(-\alpha_1) & if \ \ h_t(x) = f(x) \\ \exp(\alpha_1) & if \ \ h_t(x) \neq f(x) \end{cases}$$

$$\mathcal{D}_2 = (0.07143, 0.07143, 0.07143, 0.07143, 0.07143, 0.07143, 0.16666, 0.16666, 0.16666, 0.07143)$$

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
$\mathcal{D}_1$	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
$\mathcal{D}_2$	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.16666	0.16666	0.16666	0.07143

Adaboost示例（续）

y	1	1	1	-1	-1	-1	1	1	1	-1
$h_1(x)$	1	1	1	-1	-1	-1	-1	-1	-1	-1
结果	对	对	对	对	对	对	错	错	错	对
$\mathcal{D}_1$	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

$$Z_1 = \sum_{i=1}^{n} D_{1i}\exp(-y_i\alpha_1 h_1(x_i))$$

$$= \sum_{i=1}^{3} 0.1 \times \exp(-[0.4263 \times 1 \times 1]) + \sum_{i=4}^{4-6,10} 0.1 \times \exp(-[0.4263 \times (-1) \times (-1)])$$

$$+ \sum_{i=7}^{9} 0.1 \times \exp(-[0.4263 \times 1 \times (-1)])$$

$$\approx 0.3928 + 0.4582 + 0.0655 = 0.9165$$

$$\mathcal{D}_{2,i} = \frac{\mathcal{D}_{1,i}}{Z_1}\exp\left(-y_i\alpha_1 h_1(x_i)\right) = \begin{cases} \frac{0.1}{0.9165}\exp(-[0.4263 \times 1 \times 1) \\ \frac{0.1}{0.9165}\exp(-[0.4263 \times (-1) \times (-1)]) \\ \frac{0.1}{0.9165}\exp(-[0.4263 \times 1 \times (-1)]) \end{cases} \approx \begin{cases} 0.07143 & i = 1,2,3 \\ 0.07143 & i = 4,5,6,10 \\ 0.16666 & i = 7,8,9 \end{cases}$$

$\mathcal{D}_2$	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.16666	0.16666	0.16666	0.07143

Adaboost示例（续）

对迭代过程$t=2$

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1

a、在权值分布$\mathcal{D}_2$上，阈值v=8.5时，分类误差率最低

$$h_2(x)=\begin{cases}1, & x<8.5\\ -1, & x\geq 8.5\end{cases}$$

b、误差率 $\epsilon_2=0.07143+0.07143+0.07143=0.21429$

c、计算

$$\alpha_2=\frac{1}{2}\log\frac{1-\epsilon_2}{\epsilon_2}=0.6496$$

$$f_2(x)=0.6496h_2(x)$$

d、更新权值分布

$$\mathcal{D}_3=(0.04546,0.04546,0.04546,0.16667,0.16667,0.16667,0.10606,0.10606,0.10606,0.04546)$$

Adaboost示例（续）

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
D_2	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.16666	0.16666	0.16666	0.07143

确定阈值的取值及误差率

- 当阈值取2.5时，误差率为0.49998。即 x<2.5 时取 1，x>2.5 时取 -1，则数据6、7、8分错，误差率为0.16666*3=0.49998
- 当阈值取5.5时，误差率最低为0.28572。即 x<5.5 时取1，x>5.5 时取 -1，则数据3、4、5、6、7、8分错，错误率为0.07143*3+0.16666*3=0.71427>0.5，故反过来，令 x>5.5 取 1，x<5.5 时取 -1，则数据0、1、2、9分错，误差率为0.07143*4=0.28572
- 当阈值取8.5时，误差率为0.21429。即 x<8.5 时取1，x>8.5 时取 -1，则数据3、4、5分错，错误率为0.07143*3=0.21429

Adaboost示例（续）

$$Z_2 = \sum_{i=1}^{n} D_{1i}\exp(-y_i\alpha_1 h_1(x_i))$$

$$= \sum_{i=1}^{3} 0.07143 \times \exp(-[0.64963 \times 1 \times 1]) + \sum_{i=4}^{4-6} 0.07143 \times \exp(-[0.64963 \times (-1) \times 1])$$

$$+ \sum_{i=7}^{9} 0.16666 \times \exp(-[0.64963 \times 1 \times 1]) + \sum_{i=1010} 0.07143 \times \exp(-[0.64963 \times (-1) \times (-1)])$$

$$\approx 0.11191 + 0.41033 + 0.26111 + 0.03730 = 0.82065$$

$$\mathcal{D}_{3,i} = \frac{\mathcal{D}_{2,i}}{Z_2}\exp\left(-y_i\alpha_2 h_2(x_i)\right) = \begin{cases} \frac{0.07143}{0.82065}\exp(-[0.64963 \times 1 \times 1) \\ \frac{0.07143}{0.82065}\exp(-[0.64963 \times (-1) \times 1]) \\ \frac{0.16666}{0.82065}\exp(-[0.64963 \times 1 \times 1)]) \\ \frac{0.07143}{0.82065}\exp(-[0.64963 \times (-1) \times (-1)]) \end{cases} \approx \begin{cases} 0.04546 & i = 1,2,3 \\ 0.16667 & i = 4,5,6 \\ 0.10606 & i = 7,8,9 \\ 0.04546 & i = 10 \end{cases}$$

$\mathcal{D}_2$	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.16666	0.16666	0.16666	0.07143
$\mathcal{D}_3$	0.04546	0.04546	0.04546	0.16667	0.16667	0.16667	0.10606	0.10606	0.10606	0.04546

Adaboost示例（续）

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
$\mathcal{D}_1$	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
$h_1(x)$	1	1	1	-1	-1	-1	-1	-1	-1	-1
分类结果	对	对	对	对	对	对	错	错	错	对
$\mathcal{D}_2$	0.07143	0.07143	0.07143	0.07143	0.07143	0.07143	0.16666	0.16666	0.16666	0.07143
$h_2(x)$	1	1	1	1	1	1	1	1	1	-1
分类结果	对	对	对	错	错	错	对	对	对	对
$\mathcal{D}_3$	0.04546	0.04546	0.04546	0.16667	0.16667	0.16667	0.10606	0.10606	0.10606	0.04546

Adaboost示例（续）

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1

对迭代过程t=3

a、在权值分布$\mathcal{D}_3$上，阈值v=5.5时，分类误差率最低

$$h_3(x)=\begin{cases}-1, & x<5.5\\ 1, & x\geq 5.5\end{cases}$$

b、误差率

$$\epsilon_3=0.1820$$

c、计算

$$\alpha_3=\frac{1}{2}\log\frac{1-\epsilon_3}{\epsilon_3}=0.7519$$

$$f_3(x)=0.7519h_3(x)$$

d、更新权值分布

$$\mathcal{D}_4=(0.1250,0.1250,0.1250,0.1019,0.1019,0.1019,0.0648,0.0648,0.0648,0.1250)$$

最终分类器

$$H(x)=\text{sign}(f_3(x))=\text{sign}[0.4236h_1(x)+0.6496h_2(x)+0.7519h_3(x)]$$

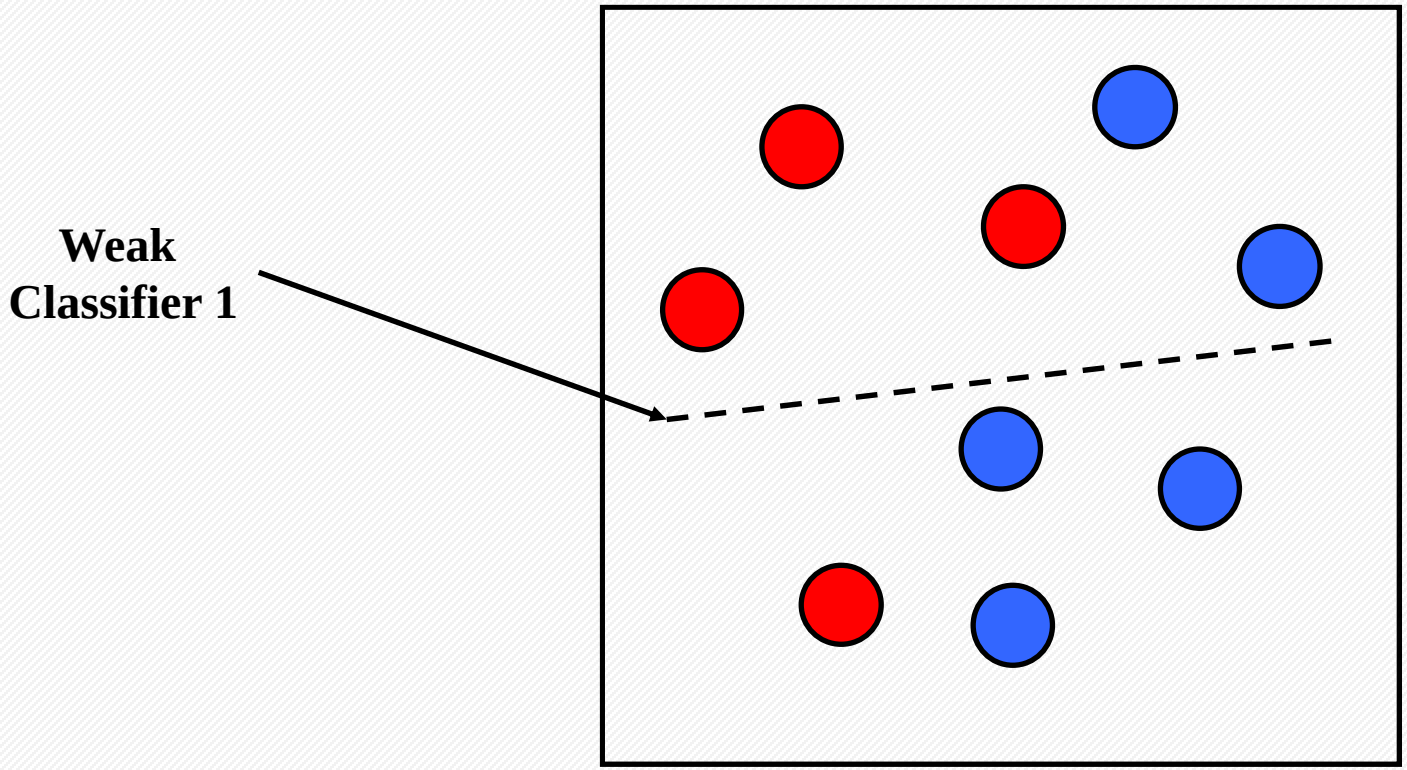
Adaboost示例（续）

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1
$\mathcal{D}_3$	0.04546	0.04546	0.04546	0.16667	0.16667	0.16667	0.10606	0.10606	0.10606	0.04546

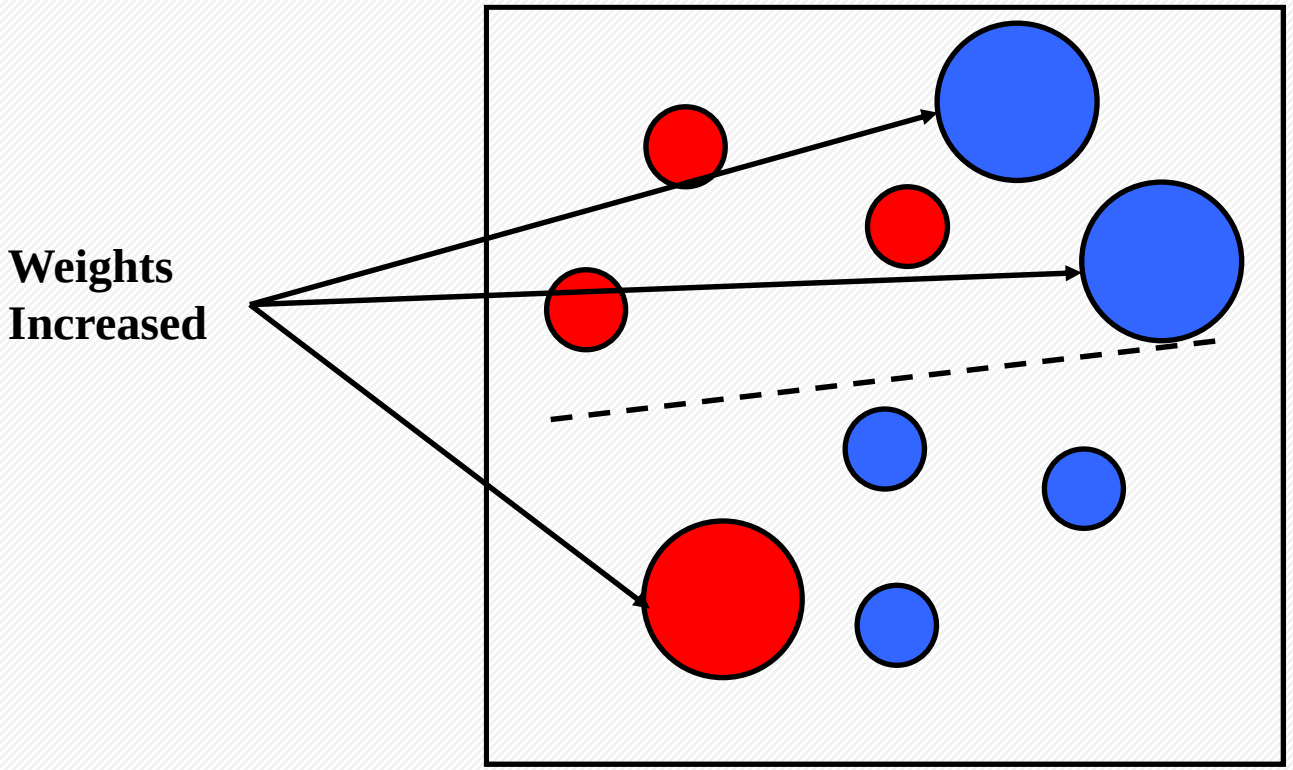
确定阈值的取值及误差率

- 当阈值取2.5时，误差率为0.31818。即 x<2.5 时取 1，x>2.5 时取 -1，则数据6、7、8分错，误差率为0.10606*3=0.31818
- 当阈值取5.5时，误差率最低为0.18184。即 x<5.5 时取1，x>5.5 时取 -1，则数据3、4、5、6、7、8分错，错误率为0.16667*3+0.10606*3=0.81819>0.5，故反过来，令 x>5.5 取 1，x<5.5 时取 -1，则数据0、1、2、9分错，误差率为0.04546*4=0.18184
- 当阈值取8.5时，误差率为0.13638。即 x<8.5 时取1，x>8.5 时取 -1，则数据3、4、5分错，错误率为0.04546*3=0.13638（取过，不列入考虑范围）
- 由上面可知，阈值取8.5时，误差率最小，但8.5取过了，所以取5.5。

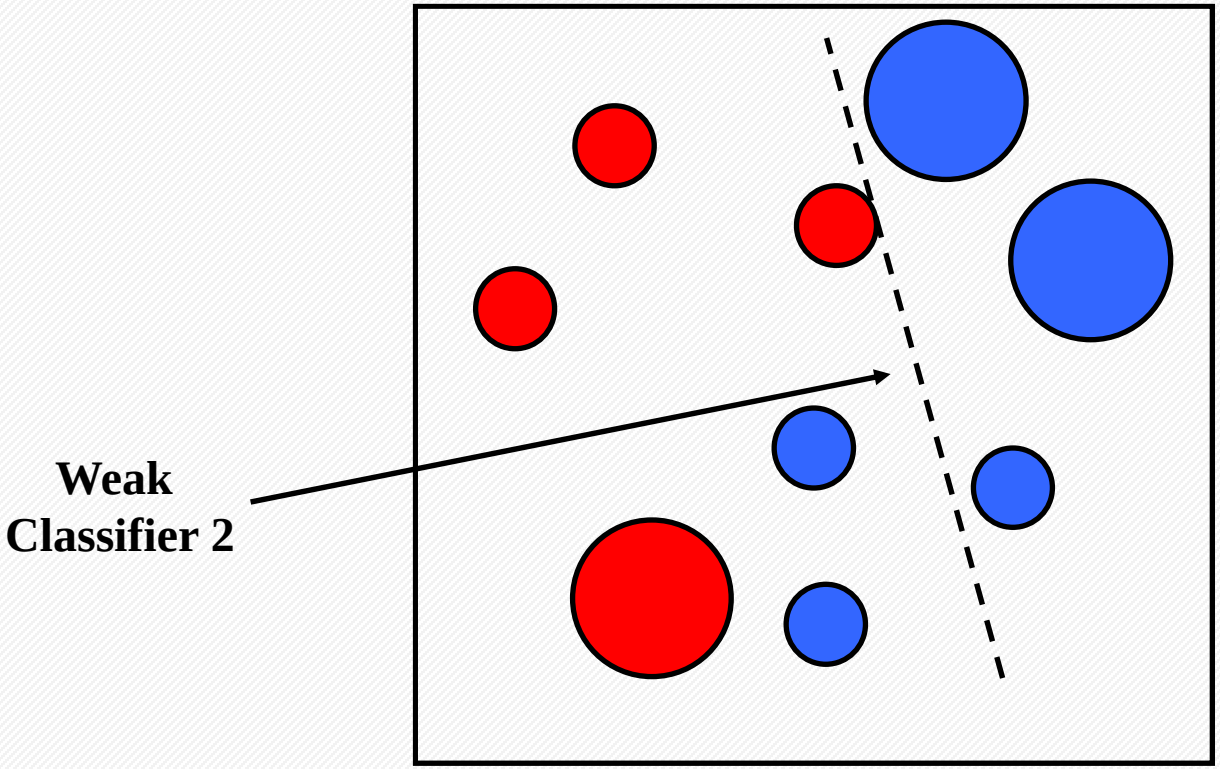
Boosting illustration



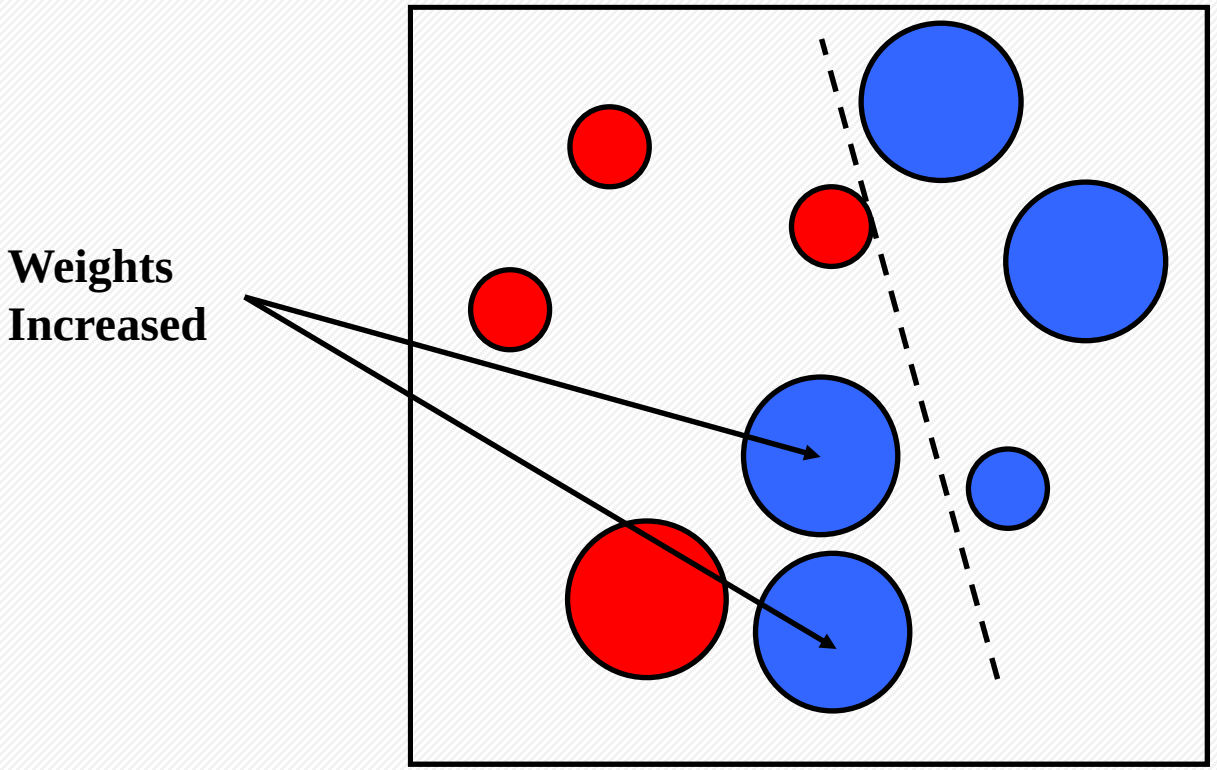
Boosting illustration



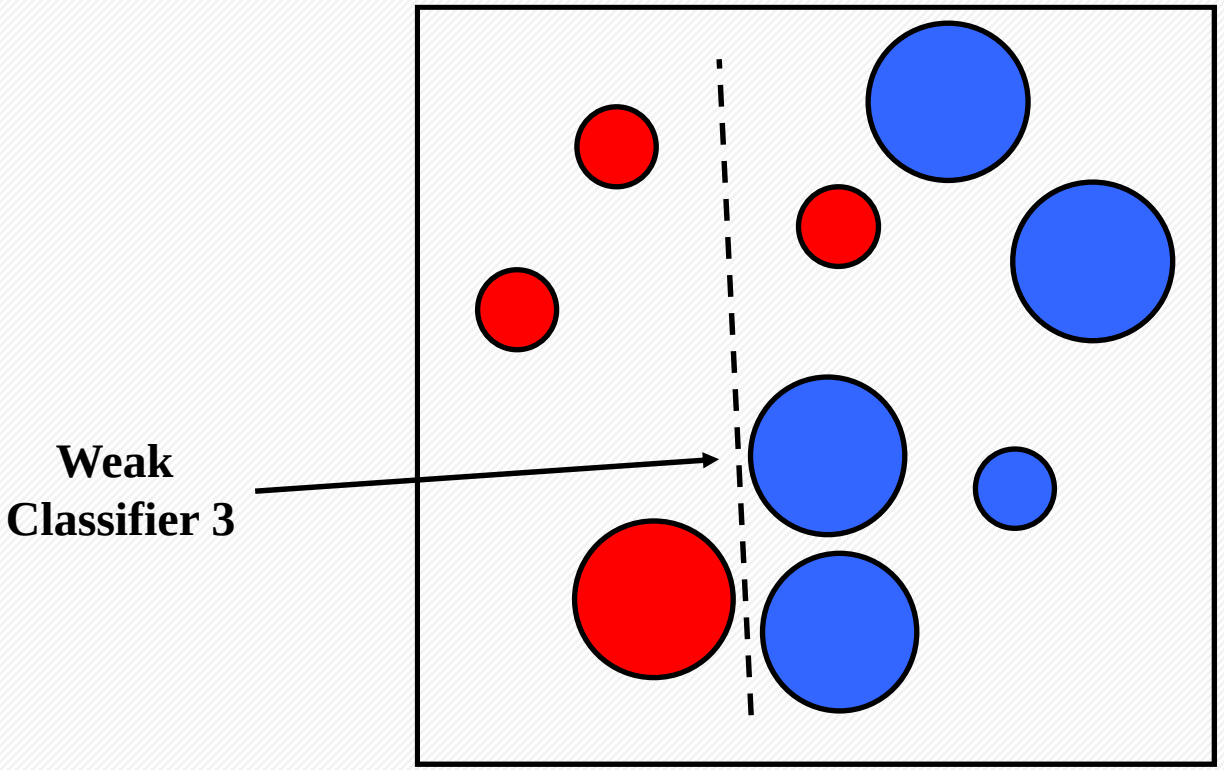
Boosting illustration



Boosting illustration

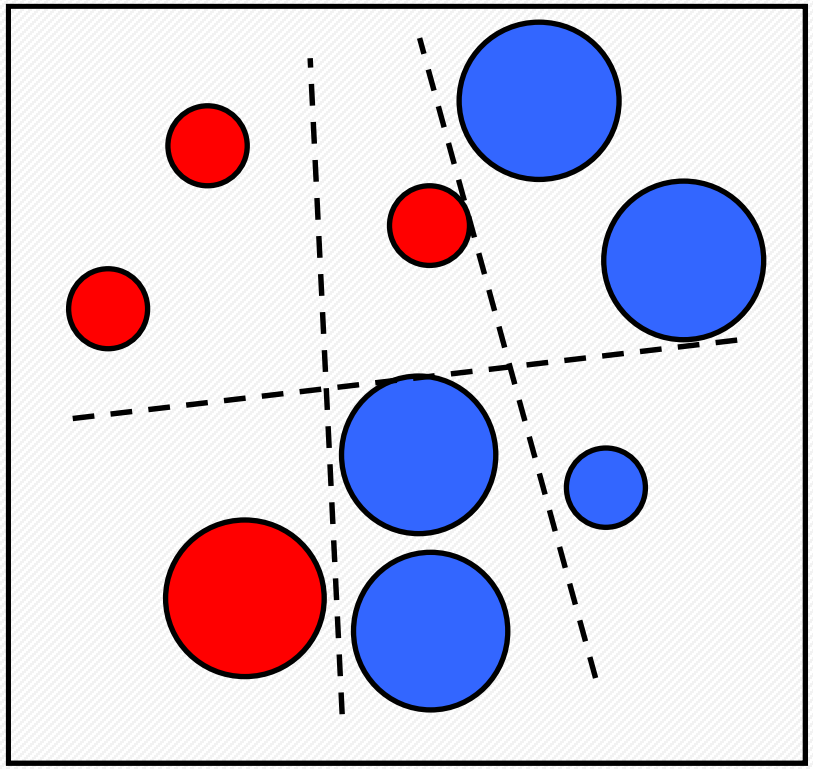


Boosting illustration

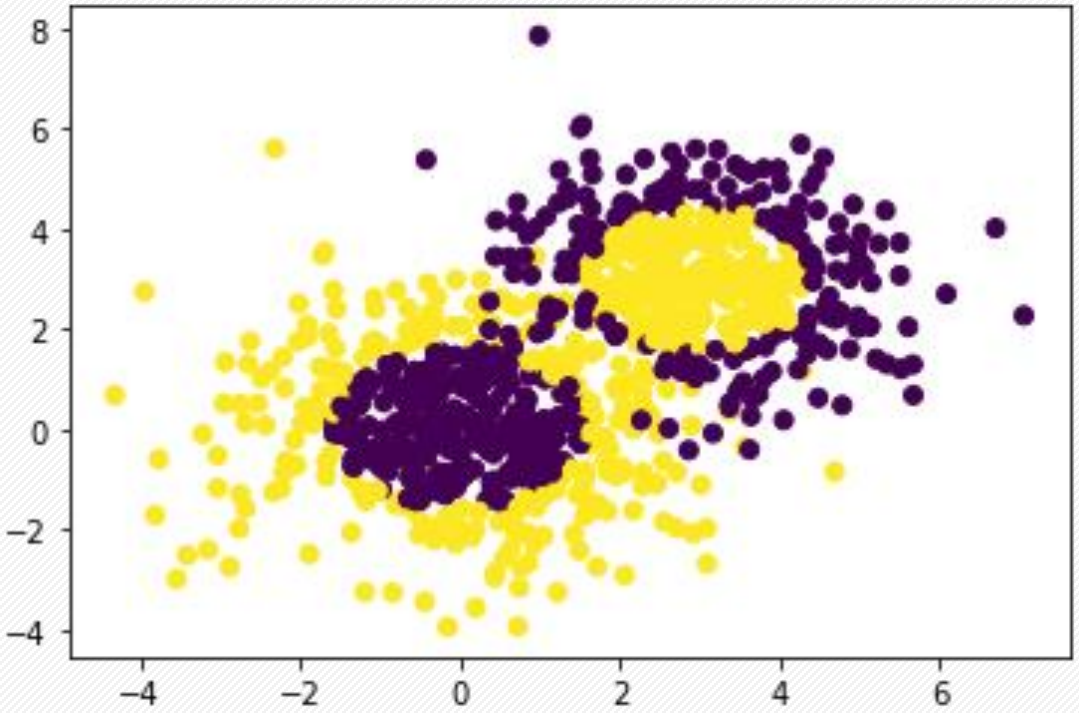


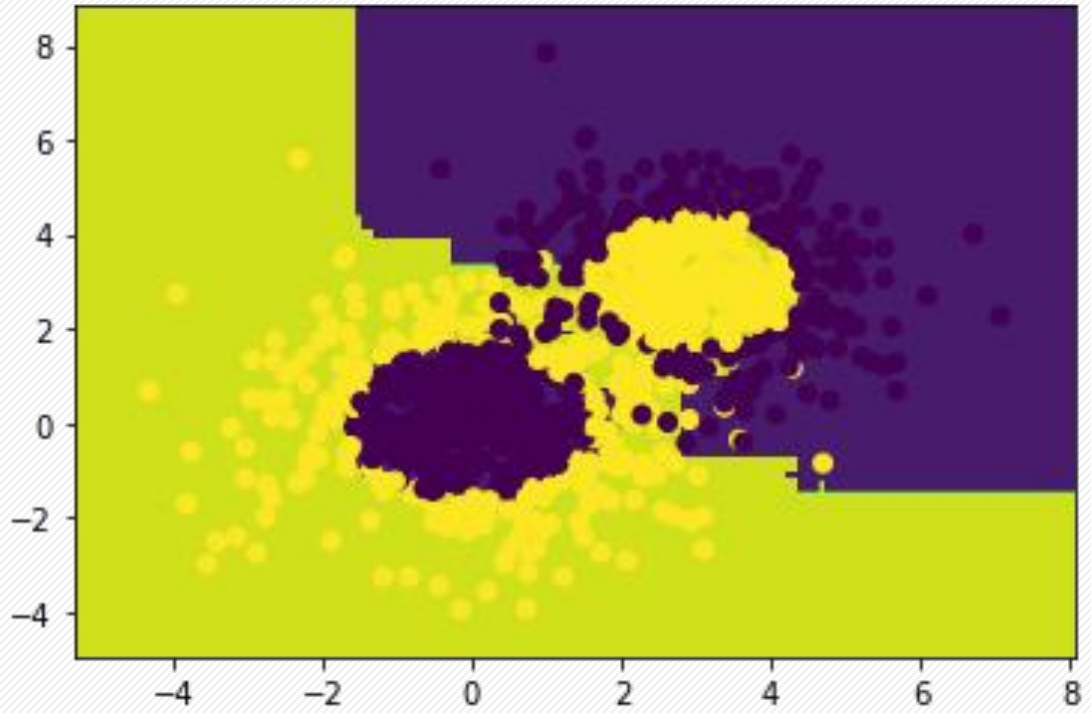
Boosting illustration

Final classifier is a combination of weak classifiers



例8.2





8.3 Bagging与随机森林

8.3.1 Bagging

- 给定含m个样本的数据集，先随机取出一个样本放入采样集中，再把该样本放回初始数据集。使得下次采样时该样本仍有可能被选中，经过m次随机采样操作，得到包含m个样本的采样集。
- 新数据集和原数据集的大小相等。
- 采样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，再将这些基学习器进行结合。
- Bagging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：训练数据集$D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_N,y_N)\}$，
基学习算法$\mathfrak{L}$；　　训练轮数T

过程：for t=1, 2, …, T do

$h_t=\mathfrak{L}(\mathcal{D},\mathcal{D}_{bs})$

end for

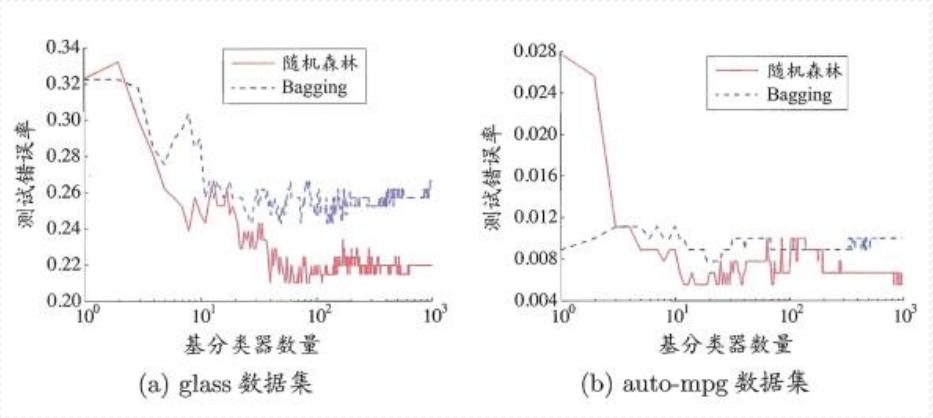
输出：$H(x)=\underset{y\in\mathcal{Y}}{\arg\max}\sum_{t=1}^{T}\mathbb{I}(h_t(x)=y)$

8.3.2 随机森林(Random Forest,RF)

Bagging的一个扩展变体。RF在以**决策树**为基础学习器构建Bagging集成的基础上，进一步在决策树的训练过程中引入**随机属性选择**。RF对基决策树的每个结点，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，然后再从这个子集中选择一个最优属性用于划分。一般情况下，参数控制了随机性的引入程度。推荐值$k = \log_2 d$。

优点：随机森林简单、容易实现、计算开销小。

被誉为“**代表集成学习技术水平的方法**”。



(a) glass 数据集　　(b) auto-mpg 数据集

8.4 结合策略

学习器结合可能会从三个方面带来好处[Dietterich, 2000]：

（1）**统计**：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，此时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**；

（2）**计算**：学习算法往往会陷入局部极小，有的局部极小点所对应的泛化性能可能很糟糕，而通过多次运行之后进行结合，可**降低陷入糟糕局部极小点的风险**；

（3）**表示**：某些学习任务的真实假设可能不在当前学习算法所考虑的假设空间中，此时若使用单学习器则肯定无效，而通过结合多个学习器，由于相应的**假设空间有所扩大**，有可能学得更好的近似。

8.4 结合策略

常用结合方法

- 平均法
  - 简单平均法
  - 加权平均法
- 投票法
  - 绝对多数投票法
  - 相对多数投票法
  - 加权投票法
- 学习法

8.4.1 平均法

- **简单平均法（simple averaging）**

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

- **加权平均法（weighted averaging）**

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

$$\omega_i \geq 0, \text{且} \sum_{i=1}^{T} \omega_i = 1$$

其中，加权平均法的权重一般从训练数据中学习得到。但由于噪声干扰和过拟合问题，加权平均法也未必优于简单平均法[Xu et al, 1992; Ho et al, 1994;Kittler et al., 1998]。

推荐：个体学习器性能相差较大时，宜用加权平均法，否则宜用简单平均法。

8.4.2 投票法

将h_i在样本x上的预测输出表示为一个N维向量$(h_i^1(x), h_i^2(x), \ldots, h_i^N(x))$，其中$h_i^j(x)$是$h_i$在类别标记$c_j$上的输出。

- 绝对多数投票法（majority voting）

$$H(\boldsymbol{x}) = \begin{cases} c_j, & if \sum_{i=1}^{T} h_i^j(\boldsymbol{x}) > 0.5 \sum_{k=1}^{N} \sum_{i=1}^{T} h_i^k(\boldsymbol{x}) \\ reject, & otherwise \end{cases}$$

- 相对多数投票法（plurality voting）

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

- 加权投票法（weighted voting）

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

8.4.3 学习法

Stacking 算法

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
初级学习算法 $\mathfrak{L}_1, \mathfrak{L}_2, \ldots, \mathfrak{L}_T$;
次级学习算法 $\mathfrak{L}$.

过程：

```
1: for t = 1, 2, ..., T do
2:     h_t = L_t(D);
3: end for
```

使用初级学习算法产生初级学习器

```
4: D' = ∅;
5: for i = 1, 2, ..., m do
6:     for t = 1, 2, ..., T do
7:         z_it = h_t(x_i);
8:     end for
9:     D' = D' ∪ ((z_i1, z_i2, ..., z_iT), y_i);
10: end for
```

生成次级训练集

```
11: h' = L(D');
```

输出： $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x}))$

8.5 多样性

8.5.1 误差-分歧分解

假定用个体学习器$h_1, h_2, \ldots, h_T$通过加权平均法结合产生集成来完成回归学习任务$f: R^d \mapsto R$。

$H(\boldsymbol{x}) = \sum_{i=1}^{T} \omega_i h_i(\boldsymbol{x})$，其中$\omega_i \geq 0$是学习器权重，$\sum_{i=1}^{T} \omega_i = 1$

1、分歧

个体学习器h_i的分歧

$$A(h_i|\boldsymbol{x}) = (h_i(\boldsymbol{x}) - H(\boldsymbol{x}))^2$$

集成的分歧

$$\bar{A}(h|\boldsymbol{x}) = \sum_{i=1}^{T} \omega_i A(h_i|\boldsymbol{x})$$
$$= \sum_{i=1}^{T} \omega_i (h_i(\boldsymbol{x}) - H(\boldsymbol{x}))^2$$

8.5.1 误差-分歧分解

2、个体学习器h_i和集成H的平方误差

个体学习器h_i的平方误差：

$$\mathrm{E}(h_i|\boldsymbol{x}) = (f(\boldsymbol{x}) - h_i(\boldsymbol{x}))^2$$

集成H的平方误差：

$$\mathrm{E}(H|\boldsymbol{x}) = (f(\boldsymbol{x}) - H(\boldsymbol{x}))^2$$

个体学习器误差的加权均值：

$$\overline{E}(h|\boldsymbol{x}) = \sum_{i=1}^{T} \omega_i \mathrm{E}(h_i|\boldsymbol{x})$$

8.5.1 误差-分歧分解

4、误差-分歧分解（error-ambiguity decomposition）

step 1： 将平方误差带入集成的分歧中，得：

$$\bar{A}(h|\boldsymbol{x}) = \sum_{i=1}^{T} \omega_i \mathrm{E}(h_i|\boldsymbol{x}) - \mathrm{E}(H|\boldsymbol{x})$$
$$= \bar{E}(h|\boldsymbol{x}) - \mathrm{E}(H|\boldsymbol{x})$$

step 2： 令 $p(x)$表示样本概率密度，在全样本上有（对step 1式两边对x积分）：

$$\sum_{i=1}^{T} \omega_i \int A(h_i|\boldsymbol{x})p(x)dx = \sum_{i=1}^{T} \omega_i \int E(h_i|\boldsymbol{x})p(x)dx - \int \mathrm{E}(H|\boldsymbol{x})p(x)dx$$

step 3：个体学习器h_i在全样本上的泛化误差：

$$\mathrm{E}(h_i) = E_i = \int \mathrm{E}(h_i|\boldsymbol{x})p(x)dx$$

8.5.1 误差-分歧分解

个体学习器h_i泛化误差的加权均值：

$$\bar{E} = \sum_{i=1}^{T} \omega_i E_i$$

个体学习器h_i在全样本上的分歧项：

$$A(h_i) = A_i = \int A(h_i|\boldsymbol{x})p(x)dx$$

个体学习器h_i的加权分歧值：

$$\bar{A} = \sum_{i=1}^{T} \omega_i A_i$$

集成的泛化误差：

$$\mathrm{E}(H) = E = \int E(H|\boldsymbol{x})p(x)dx$$

集成的泛化误差E(误差-分歧分解)：

$$E = \bar{E} - \bar{A}$$

上面这个分析首先由[Krogh and Vedelsby, 1995]给出，称为误差-分歧分解。

现实任务中很难直接对该式进行优化

8.5.2 多样性度量

给定数据集$D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\ldots,(\boldsymbol{x}_m,y_m)\}$，对于二分类任务，$y_i \in$ {-1，1}，分类器h_i与h_j的预测结果列联表（contingency table）为：

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

- 不合度量（disagreement measure）

$$dis_{ij}=\frac{b+c}{m}$$

- 相关系数（correlation coefficient）

$$\rho_{ij}=\frac{ad-bc}{\sqrt{(a+b)(a+c)(c+d)(b+d)}}$$

8.5.2 多样性度量

- Q-统计量（Q-statistic）

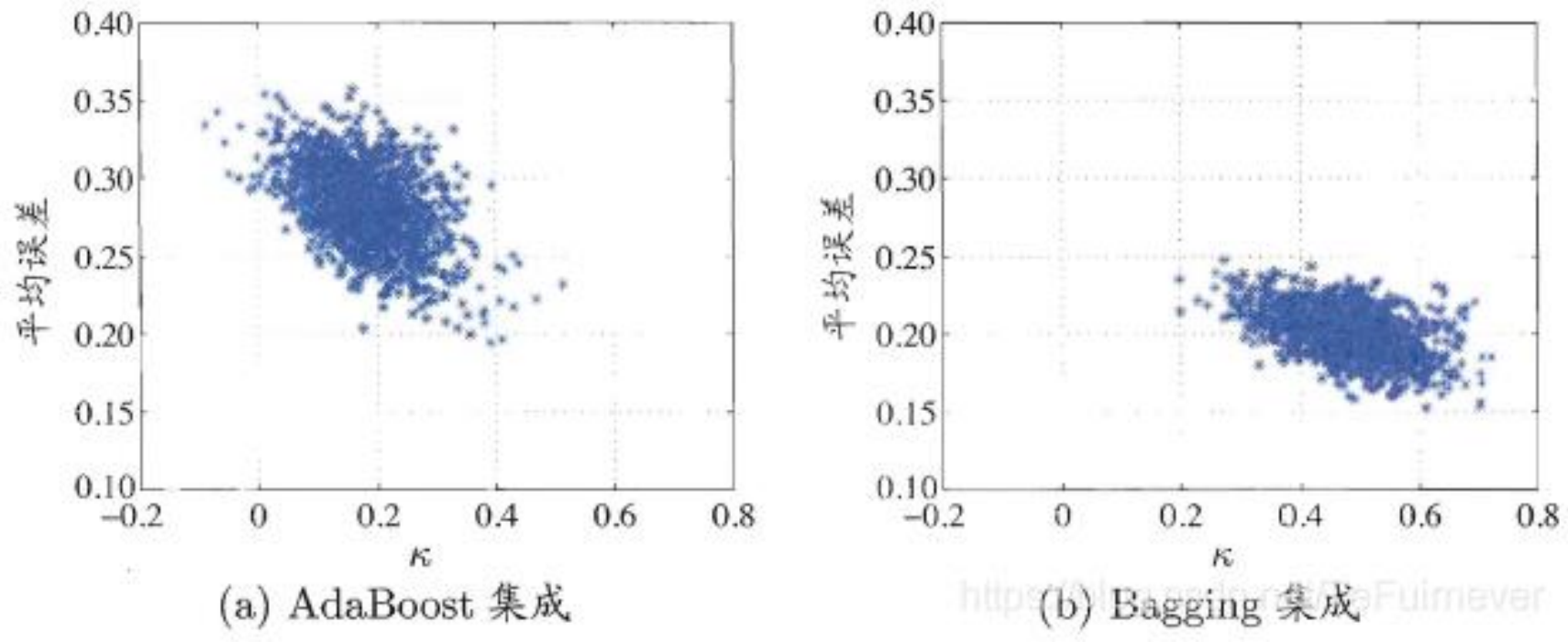
$$Q_{ij} = \frac{ad - bc}{ad + bc}$$

- k-统计量（k-statistic）

$$k = \frac{p_1 - p_2}{1 - p_2}$$

$$p_1 = \frac{a + d}{m}$$

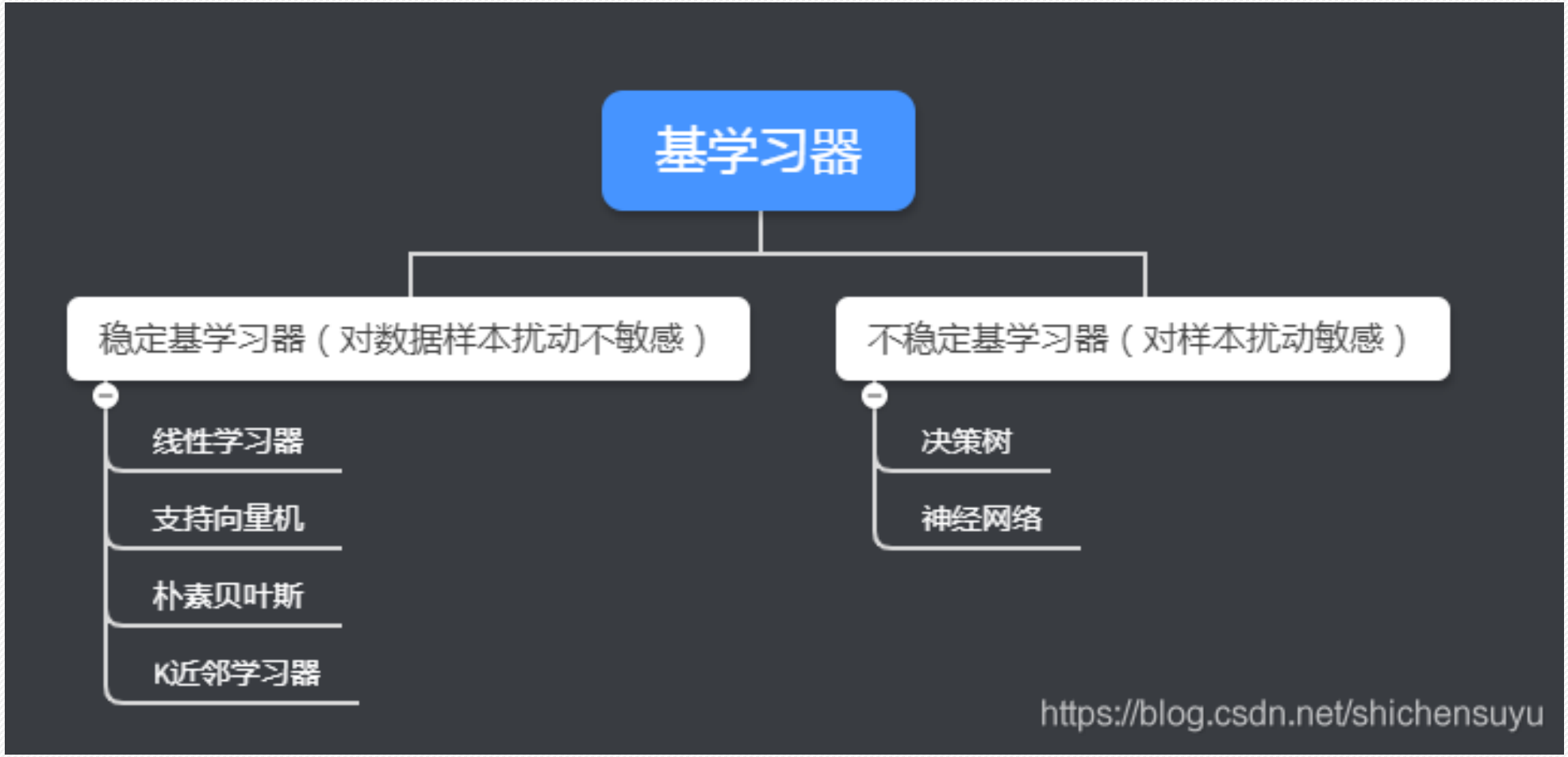
$$p_2 = \frac{(a + b)(a + c) + (c + d)(b + d)}{m^2}$$



8.5.3 多样性增强

- **数据样本扰动（适合不稳定基学习器，如决策树、神经网络等）**

通常基于采样法，如在Bagging中使用自助采样，在AdaBoost采用序列采样



8.5.3 多样性增强

- **输入属性扰动（适合包含大量冗余属性的数据）**

不同的子空间（即属性子集）提供了观察数据的不同视角。显然，从不同的属性子集训练出的个体学习器必然有所不同。

随机子空间(random subspace)算法采用输入属性扰动：从初始属性集中抽取出若干个属性子集，再基于每个属性子集训练一个基学习器。

d' 小于初始属性数 d.

$\mathcal{F}_t$ 包含 d' 个随机选取的属性，D_t 仅保留 $\mathcal{F}_t$ 中的属性.

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
基学习器数 T;
子空间属性数 d'.

过程：

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad \mathcal{F}_t = \mathrm{RS}(D, d')$
3: $\quad D_t = \mathrm{Map}_{\mathcal{F}_t}(D)$
4: $\quad h_t = \mathfrak{L}(D_t)$
5: **end for**

输出： $H(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}\left(h_t\left(\mathrm{Map}_{\mathcal{F}_t}(\boldsymbol{x})\right) = y\right)$

图 8.11 随机子空间算法

8.5.3 多样性增强

- **输出表示扰动**

对输出表示进行操纵可以增强多样性对训练样本的类标记稍作变动，如翻转法：随机改变一些训练样本的标记对输出表示进行转化，如输出调制法：将分类输出转化为回归输出后构建个体学习器将原任务拆解为多个可同时求解的子任务，如ECOC法：利用纠错输出码将多分类任务拆解为一系列二分类来训练基学习器

- **算法参数扰动**

通过随机设置不同的参数，往往可产生差别较大的个体学习器。
例如负相关法：显示地通过正则化项来强制个体神经网络使用不同的参数。使用单一学习器时通常使用交叉验证等方法确定参数值，这事实上已经使用了不同参数训练出多个学习器，只不过最终选取其中一个学习器进行使用，而集成学习则相当于把这些学习器都利用起来。

The end