（请大家预习）

第8章
模型选择与集成学习
Model Selection and Ensemble Learning

张 燕 明
ymzhang@nlpr.ia.ac.cn
people.ucas.ac.cn/~ymzhang

模式分析与学习课题组(PAL)
多模态人工智能系统实验室 中科院自动化所

助教：杨 奇 (yangqi2021@ia.ac.cn)
张 涛 (zhangtao2021@ia.ac.cn)

内容提要

- 引言
- 模型选择的相关定理和原则
- 模型选择的评价标准
- 分类器设计中的重采样技术
- 集成学习
  - Bagging, Adaboost, …

8.1 引言

- **机器学习模型**
  - **具有“不同类别”的含义：**比如“两个或三个成分的高斯混合模型之集合”、支持向量机、神经网络、…。
  - 具有**“类别固定条件下的参数可变”的含义：**高斯混合模型中待学习的参数。
- **模型选择**
  - 估计不同模型的性能，从中选择最好的模型。
- **模型评估**
  - 从选定的模型中，估计新样本的预测值（泛化误差）

8.1 引言

- **机器学习方法**
  - 任务：从已知数据（训练数据）中学习决策函数、规则、知识，用于对新数据（测试数据）的预测和分析
  - 学习方法 = 模型 + 评价 + 优化
  - **模型：**
    - **为学习方法选择一种模型，就意味选择一个特定的函数集合，**该集合$\mathcal{F} = \{f\}$称为学习方法的假设空间(hypothesis space)。如线性模型：$\mathcal{F} = \{f | f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b\}$
    - 所学模型只能在假设空间中。

8.1 引言

- **机器学习方法**
  - **评价：**
    - 评价函数（亦称目标函数、损失函数、能量函数）用来判断模型输出与数据真值之间的匹配程度。

回归：$L(y, f(\mathbf{x})) = \|y - f(\mathbf{x})\|^2$　　分类：$L(y, f(\mathbf{x})) = \begin{cases} 1, & y \neq f(\mathbf{x}) \\ 0, & y = f(\mathbf{x}) \end{cases}$

  - **优化：**
    - 一个搜索算法，在假设空间$\mathcal{F}$中找到使评价函数得分最高（好）的模型。

$$f^* = \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i))$$

8.1 引言

- **机器学习方法**

模型	评价函数	优化算法
—近邻法	—准确率/错误率	—组合优化
—支持向量机	—召回率	—贪心搜索
—超平面分类器	—平方误差	—连续优化
—朴素贝叶斯分类	—后验概率	—无/有约束优化
—逻辑斯蒂回归	—似然	—梯度下降
—神经网络	—信息增益	—共轭梯度
—决策树	—KL距离	—线性规划
—图模型	—利润	—二次规划
—贝叶斯网络	—成本/效用	—半正定规划
—命名规则	— …	

8.1 引言

- **模型的误差**

  对给定训练数据集$\boldsymbol{T} = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\}$，假设学习得到模型 $f(\mathbf{x})$

  – 训练/经验误差 (training/empirical error)

$$R_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i))$$

  – 测试/泛化/期望误差 (test/generalization/expectation error)

$$R_{\exp}(f) = E_{(\mathbf{X},Y)\sim p(\mathbf{x},y)}\left(L(Y, f(\mathbf{X}))\right)$$

训练误差≠测试误差

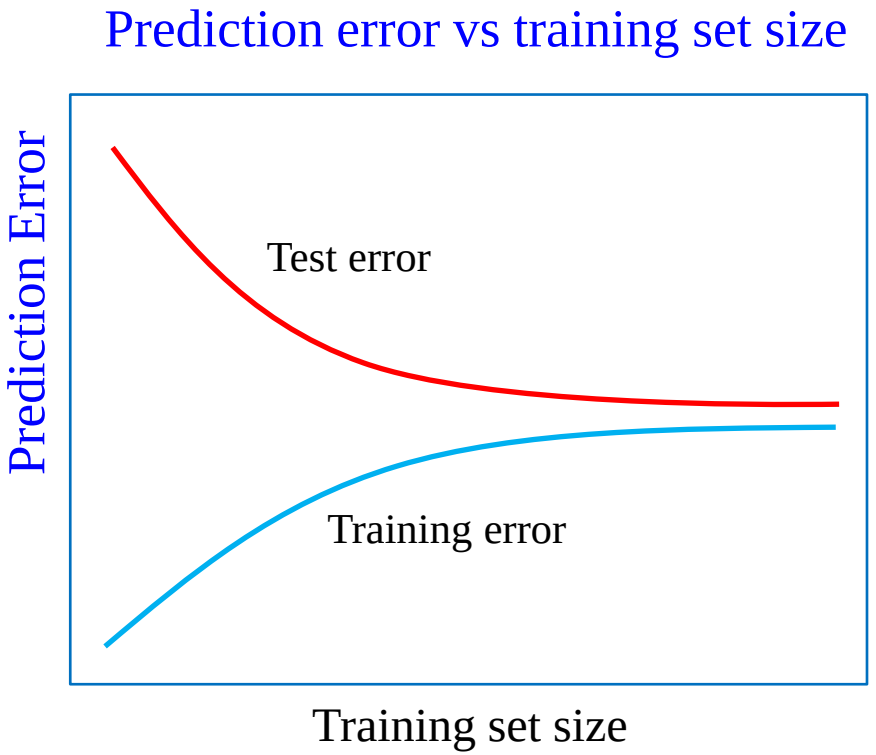
8.1 引言

- **训练数据的多少对泛化性能的影响**

给定学习方法

✓ 训练数据越多，得到的模型泛化性能越好。

✓ 训练样本增加时，训练误差与测试误差间的gap会减小。



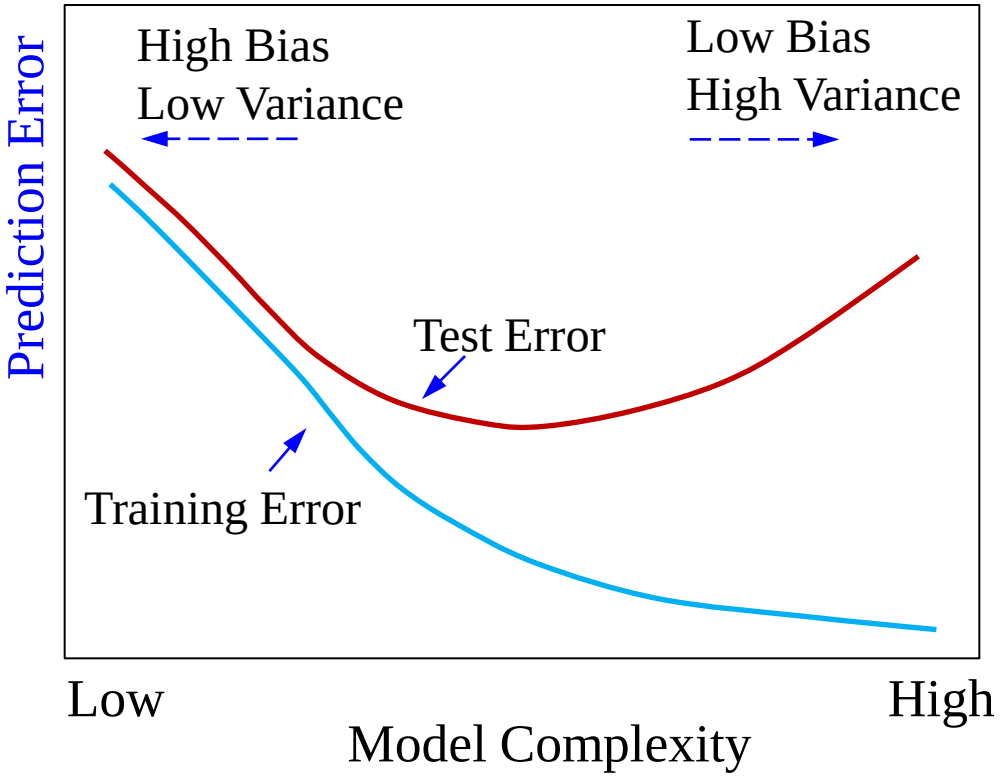
8.1 引言

- **模型复杂度的影响**

给定训练样本集

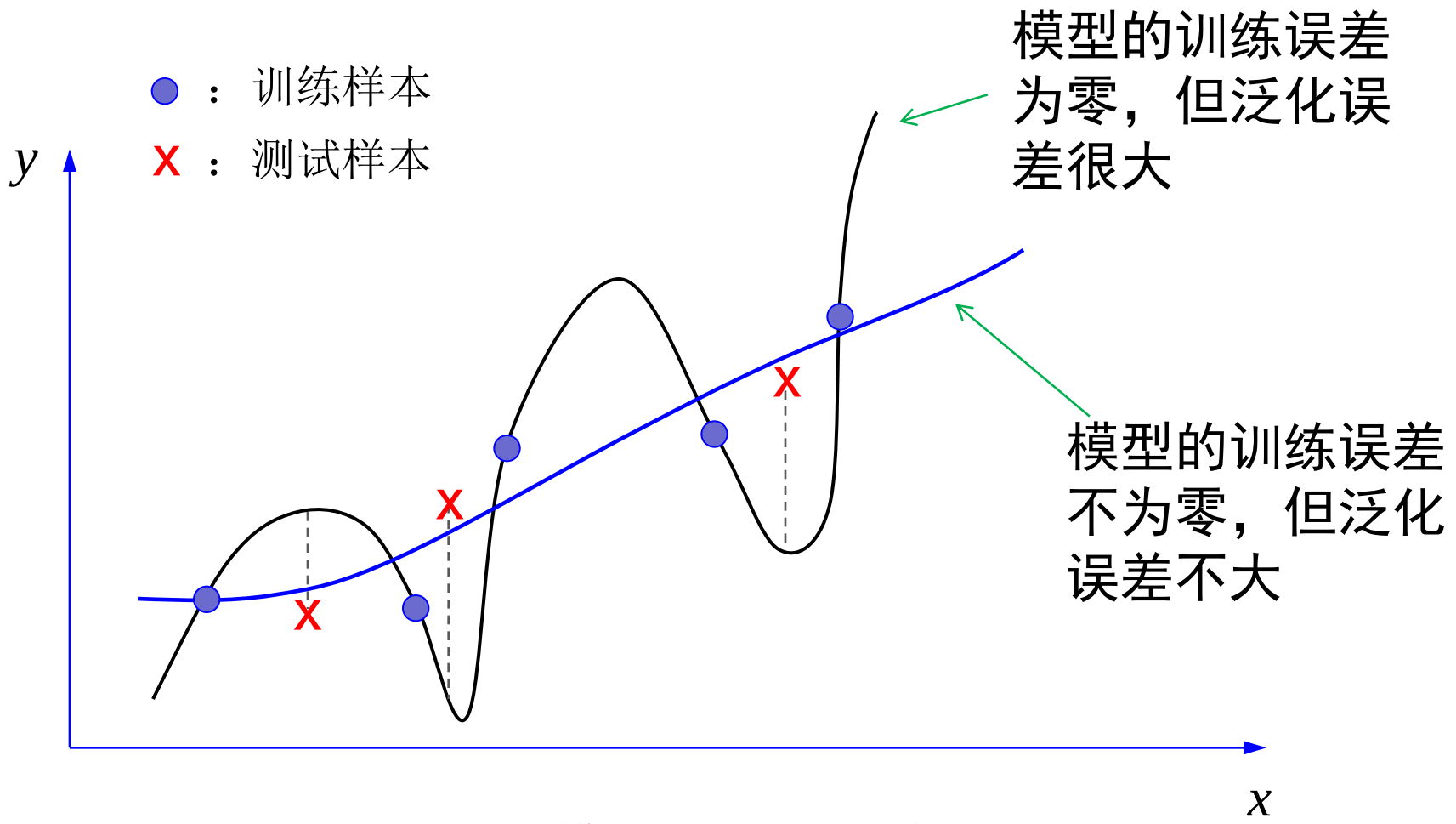
- ✓ 训练误差随模型复杂度的增加而单调减小
- ✓ 但测试误差通常随模型复杂度的增加先减小后增加
- ✓ 过拟合/过学习：复杂模型能将训练错误率降到极低，但测试错误率却很高

训练误差不是“泛化误差的一种好的估计”



8.1 引言

- **模型复杂度的影响**



使用复杂模型可能导致过拟合

8.2 模型选择原则

- **Occam剃刀原理 (Occam's Razor)**
  - 由14世纪逻辑学家**Occam提出。**
  - 如无必要，勿增实体——即简单有效原理。
    - 切勿浪费较多东西去做“用较少的东西也可以做好的事情”。
  - 剔除所有累赘：
    - 简约而不简单。
  - 对机器学习：
    - 设计者不应该选用比“必要”更加复杂的模型。
    - 在相同性能下，我们更倾向于选择简单的、参数少的模型。

8.2 模型选择原则

- **Occam剃刀原理的应用--正则化(regularization)**
  - 基本思想：对模型复杂性进行限制或惩罚
  - 结构风险最小化：

$$\min_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$$

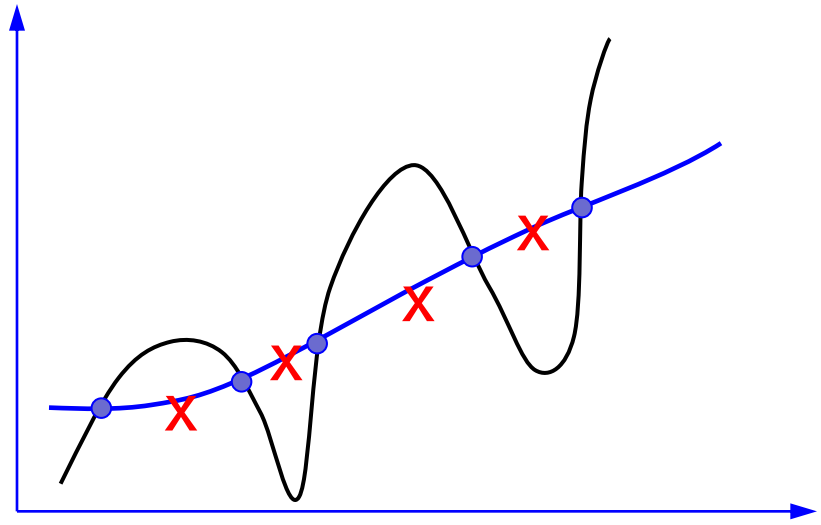
✓ $J(f)$是定义在$\mathcal{F}$上的泛函，模型f越复杂，$J(f)$越大
✓ 例如：假设$\mathcal{F}$为线性模型的集合， $J(f) = \|\mathbf{w}\|^2$

  - 一种广泛使用的模型选择方法，有效缓解过拟合

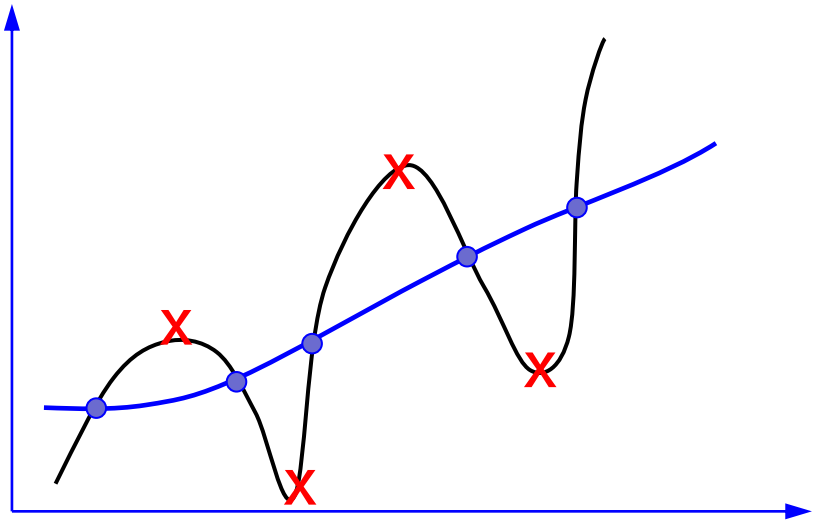
8.2 模型选择原则

- **Occam剃刀原理**
  - **Occam剃刀原理仅仅是一种经验假设或归纳偏好(inductive bias)，并非一定成立**

● ：训练样本
X ：测试样本



Occam剃刀有效



Occam剃刀失效

8.2 模型选择原则

- **没有免费的午餐定理 (No Free Lunch, NFL)**
  - 1995年，David H. Wolpert和William G. Macready 提出NFL定理：
    - 对“寻找代价函数极值”的算法，**在平均到所有可能的代价函数上时，其表现都恰好相同**。
  - **对于整个函数集(类)而言，不存在万能的最佳算法**。所有算法在整个函数集的平均表现度量是一样的。
    - 特别地，如果算法A在一些代价函数上优于算法B，那么存在一些其它函数，使B优于A。

严格表述及证明，见周志华《机器学习》(第一章第四节)

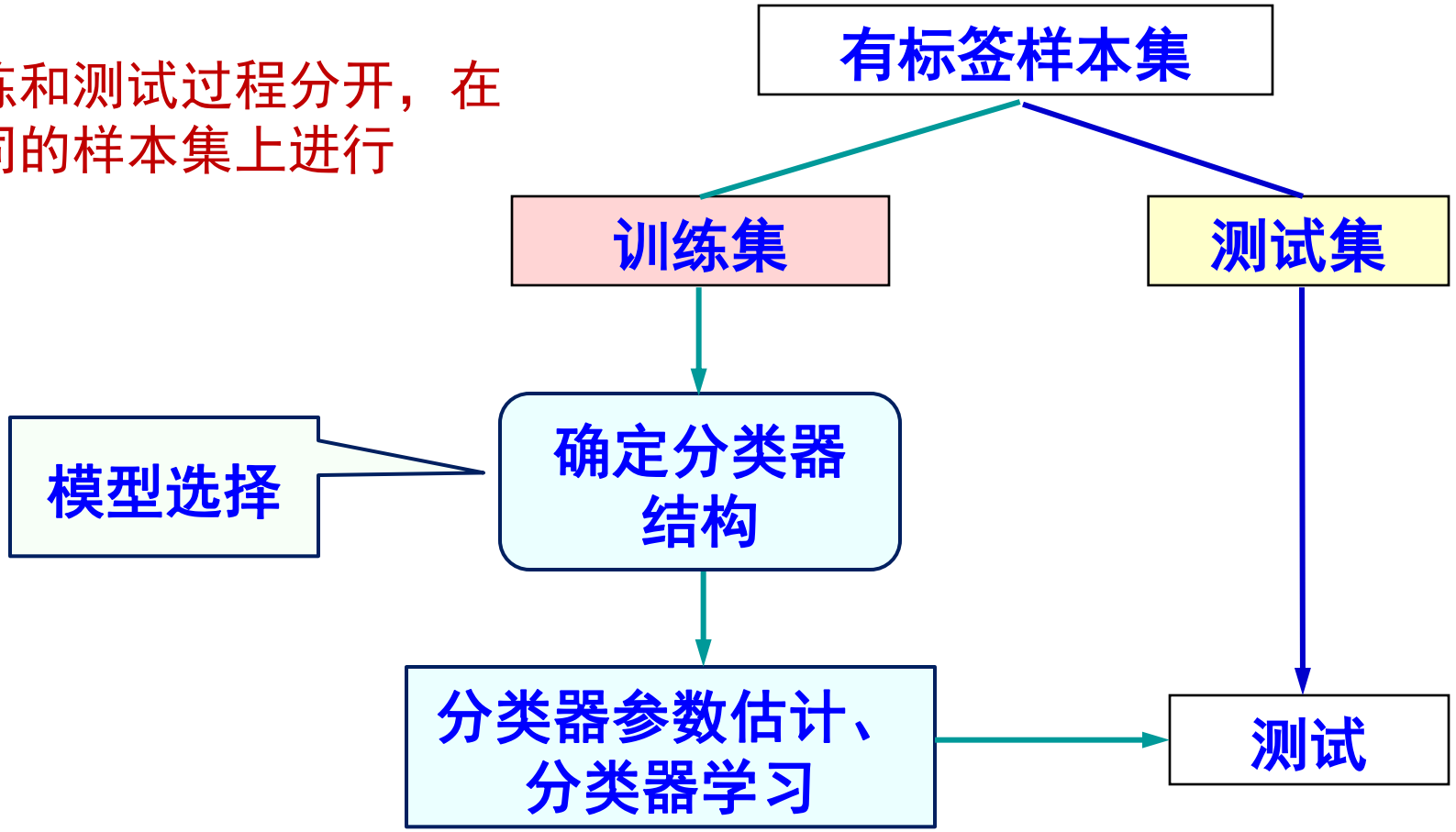
8.2 模型选择原则

- **没有免费的午餐定理**
  - **对机器学习的启示：**
    - 机器学习算法需要引入**与问题领域有关的假设**。
    - 不存在一个与具体应用无关的、普遍适用的“最优学习算法”。
    - 在无假设前提下，没有理由偏爱某一学习算法而轻视另一个。
    - 提升学习器性能，必须在另一些指标上付出相应的代价：
      - 深入理解研究对象，掌握先验知识、数据分布；
      - 引入大量训练数据量；
      - 改善代价准则。

8.3 模型评价与模型选择

- **机器学习方法的评价过程**

训练和测试过程分开，在不同的样本集上进行

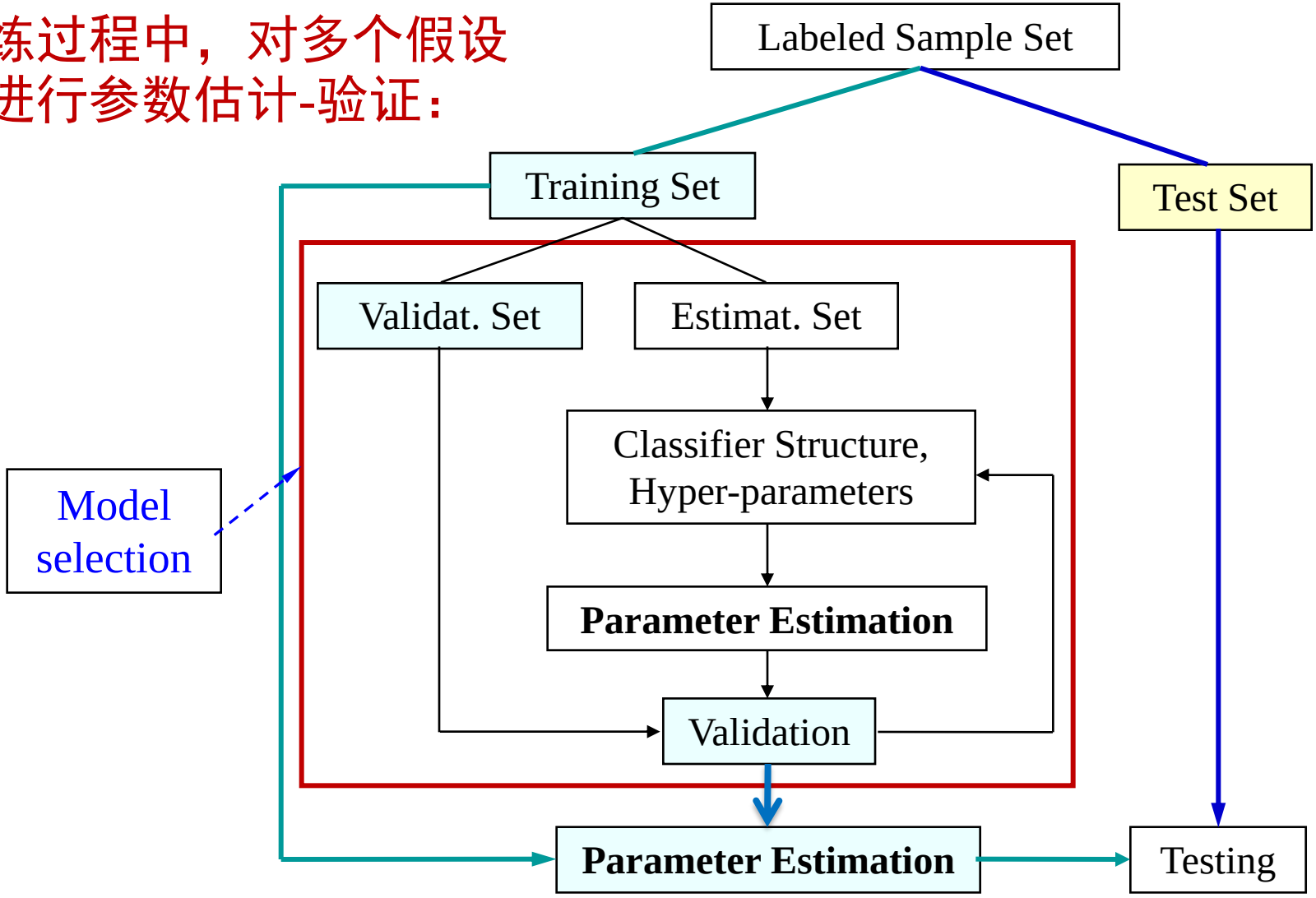


8.3 模型评价与模型选择

- **样本的划分**
  - **保持方法 (Holdout)：一部分用于训练，一部分用于测试**
  - **自助法 (Bootstrap)：有放回地随机抽取*n*个样本**
  - **交叉验证 (cross-validation)**
    - 将数据平分为 k 个子集，用 k-1个子集进行训练，余下的部分用于验证，并计算验证误差。重复这一过程 k 次，得到 k 次结果的平均。
    - **刀切法（留一法）**：每次从样本集中删除一个或者多个样本，用剩余的样本做为**“刀切样本”**进行训练
    - 交叉验证是目前最常用的一种**模型选择和评估**方法 **(特别是对模型的参数进行选择)**。

8.3 模型评价与模型选择

在训练过程中，对多个假设模型进行参数估计-验证：



8.3 模型评价与模型选择

- **基于交叉验证的模型（超）参数选择**
  - 给出参数的候选集合
  - 对每一个参数，将训练数据平分为 k 个子集，用 k-1个子集进行训练，余下一个子集用于验证，并计算验证误差。
  - 根据验证误差，选择统计性能最优的那个参数作为模型的参数，对未来的样本，模型将使用该参数。

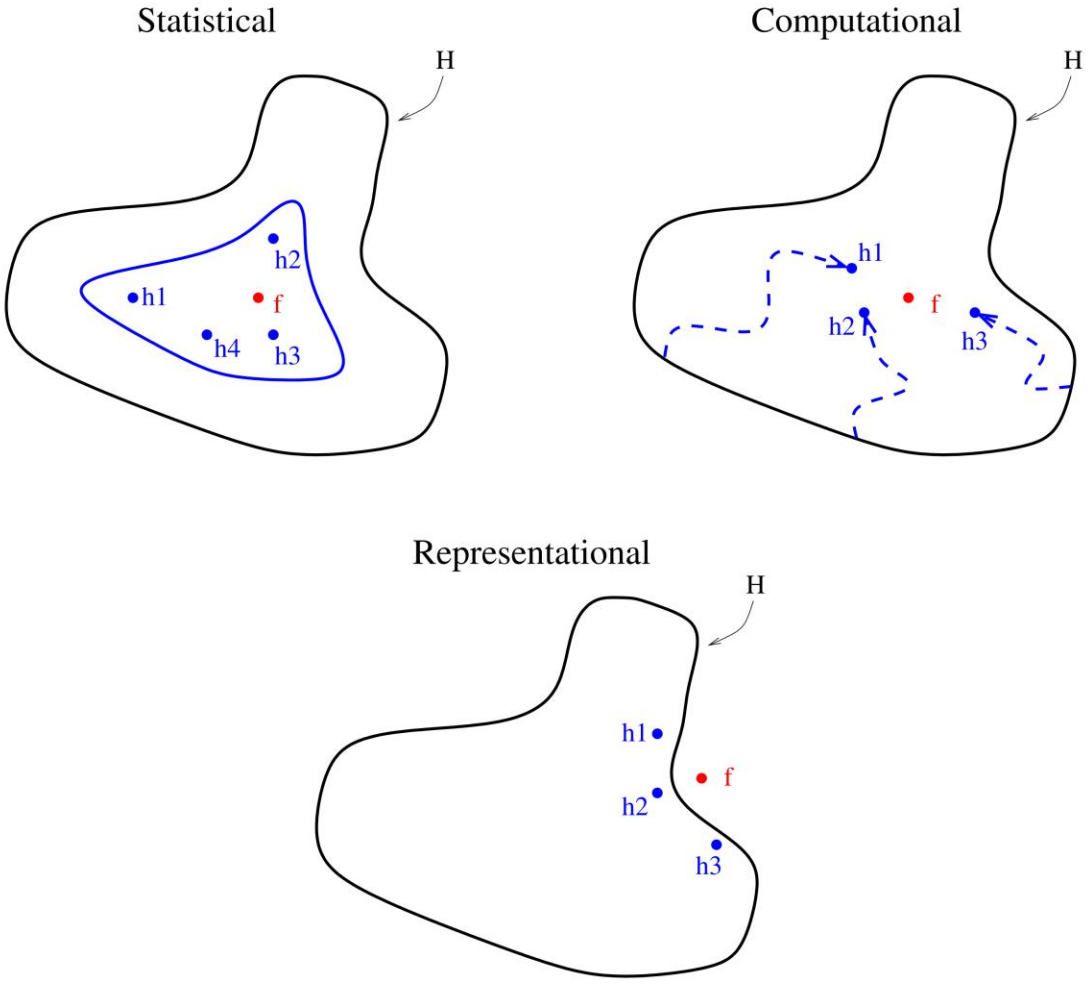
8.4 分类器集成

- **目标：**
  - **将若干单个分类器集成起来**，共同完成最终的分类任务，期望取得比单个分类器更好的性能。
- **统计上：**
  - 对于一般的学习任务，**要搜索的假设空间通常十分巨大**。但训练**样本个数却不足够**用于精确地学习到目标假设。
  - 即使学习到可满足训练集的假设，其泛化能力不一定优秀。因此，输出一个能够很好地拟合训练集的单个假设同样会面临风险。
  - **策略：**
    - **将多个假设集成起来能够降低这种风险（误差抵消）。**

8.4 分类器集成

- **计算上：**
  - 分类器模型训练通常面临高计算复杂度。比如，最优人工神经网络和最优决策树。
  - 策略：尽管单个假设可以很简单，也并非最优，但**多假设的集成可使最终结果更接近实际的目标函数值，降低风险（误差抵消）** 。
- **表示上：**
  - 任务的真实假设可能并不在学习器的假设空间之中。
  - 假设空间开放性：
    - 即不封闭在某一特定类型或特定参数
    - 将其设置为一系列假设的集成，有可能表示出不在假设空间中的目标假设。

8.4 分类器集成



T. G. Dietterich. Ensemble Methods in Machine Learning. MCS, 2000.

8.4 分类器集成

- **集成学习的有效条件**
  - **每个单一的学习器错误率都应当低于0.5**，否则集成的结果反而会提高错误率。
  - **进行集成学习的每个分类器还应当各不相同。**
    - 如果每个分类器分类结果相差不大，则集成后的分类器整体和单个分类器做出的决策实际上没有显著差异，性能得不到提高。

8.4 分类器集成

- **集成学习的常用技术手段**
  - **通过处理训练数据**（bagging, boosting)，比如，对训练样本进行随机分组，对错分样本进行加权。
  - **通过处理特征**，比如，每次只选择一部分特征来训练分类器
  - **通过处理类别标号**，比如，对多类问题，一对一策略、一对多策略。
  - **通过改进学习方法**，比如，变更学习参数(如多核学习)、模型结构(如神经网络结构) 等

8.4 分类器集成

- **分类器集成算法分类**
  - 按基本分类器类型是否相同
    - **异态集成**—基分类器类型不同
      - 叠加法：将基学习器分布在多个层次上。第一层学习器按照某种规则对分类进行预测，然后第一层的预测结果作为第二层的输入，…。
    - **同态集成**—基分类器类型相同
      - 基分类器多采用神经网络、二叉树、K—近邻

8.4 分类器集成

- **分类器集成算法分类**
  - **按训练数据处理方式**
    - **Bagging**
    - **Random subspace（随机子空间）**
    - **Boosting/Adaboost**
    - **随机森林（讲了决策树之后可展开）**

8.5 层叠泛化

- **Stacked Generalization**
  - 采用多层结构。第一层的学习器配置不同的学习算法，由训练数据集生成。第一层的输出作为第二层的输入。**第二层学习层称为“元学习器”**。
  - 基本过程
    - 给定训练数据 $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$
    - 第一层：学习器 $L_1, L_2, \ldots, L_T$
    - 第二层：学习器 L

8.5 层叠泛化

Stacked Generalization

1 Input data set $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$

2 for $t = 1, \ldots, T$

3 $h_t = L_t(D)$ // 学习T个不同的基分类器

4 end

5 $D' = \varnothing$ // 准备生成一个新的数据集合

6 for $i = 1, \ldots, n$,

7 for $t = 1, \ldots, T$

8 $z_{it} = h_t(\mathbf{x}_i)$

9 end

10 $D' = D' \cup \{ ([z_{i1}, z_{i2}, \ldots, z_{iT}], y_i) \}$

11 end

12 $h = L(D')$ // 采用收集到的数据，训练h

13 return $H(\mathbf{x}) = h (h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_T(\mathbf{x}))$

8.6 样本装袋

- **Bagging**
  - 训练一组基分类器，每个基分类器通过一个 bootstrap 训练样本集来训练。
  - 一个bootstrap训练样本集通过有放回地随机从一个给定的数据中抽样得到。

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n}\to\frac{1}{e}\approx 0.368$$

  - 获得基分类器之后，bagging通过投票进行统计，被投票最多的类为预测类。

Bagging

1 Input data set $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$

2 Given basic learner L, number of learners T

3 for $t = 1, \ldots, T$

4 $D_t = \text{bootstrap}(D)$ // 从D中生成一个bootstrap集

5 $h_t = L(D_t)$ // 学习基分类器

6 end

7 return h_t , $t = 1, \ldots, T$

最终的分类器（即投票最大者）：

$$H(\mathbf{x}) = \arg\max_{y \in Y} \sum_{i=1}^{T} I(y == h_t(\mathbf{x}))$$

取遍所有的标签

真值函数

8.7 随机子空间

- **Random Subspace**
  - 随机子空间通常也被称为**属性装袋** (attribute bagging)
  - 随机子空间的基分类器通常由**线性分类器、支持向量机**等组成。

8.7 随机子空间

- **随机子空间算法**
  - 训练
    - 令 n 为训练样本的个数，D 为训练数据的特征维数
    - 令 d 为子特征个数，$d < D$, 每次可以不同
    - 令 T 为所有分类器的个数
    - for $t = 1, \ldots, T$
      - 从 D 个特征中随机选择 d 个特征来构建训练集合，然后，学习一个分类器
    - end
  - 测试
    - 对新样本，通过多数投票法则来预测其类别

8.8 Adaboost

- **提升方法 (Boosting)**
  - Boosting是一种提高给定学习算法准确度的方法，是一种常用的统计学习方法，应用广泛且有效。
  - 在分类问题中，它通过改变训练样本的权重，学习多个分类器，并将这些分类器进行组合，提高分类性能。
  - 基本思路
    - 对于一个复杂任务，将多个专家的判断进行适当的综合，要比其中任何一个专家单独的判断要好。
    - 这就是“三个臭皮匠顶个诸葛亮”。
  - 提升方法中最具代表性的算法是**Adaboost（Adaptive boosting)**。

8.8 Adaboost

- **发展历程**
  - 其思想起源于L. G. Valiant于1983年提出的PAC (Probably Approximately Correct) 学习模型。
  - 1988年，Kearns和Valiant提出强可学习和弱可学习。
    - 在PAC学习框架中，一个概念（一个类），如果存在一个多项式的学习算法能够学习它，并且正确率很高，则称这个概念是**强可学习的**；
    - 一个概念，如果存在一个多项式的学习算法能够学习它，学习正确率仅比随机猜测略好，即该概念是**弱可学习的**。
  - 但留下一个问题：**强可学习与弱可学习是否等价**。
    - 如果等价，则可以将弱可学习通过“提升”变成强可学习，这样可以避免直接寻找强可学习算法。

- **发展历程 (续)**
  - 1990年，Schapire通过**构造性方法对“等价性”问题作出了肯定的回答**。证明多个弱分类器可以集成为一个强分类器。这就形成了集成学习的理论基础。
  - 1991年，Freund提出了boost-by-majority 策略，其亮点思想是：增加**“对难学习部分进行学习的可能性”**，迫使学习者提出新的假设，使其在“难学习部分”少犯错误。但该算法需要知道弱学习算法学习正确率的下限，这一点在实际应用中难以回答。
  - 1995年，Freund和Schapire提出Adaboost算法，避免这这些难点，还将其推广至分类问题和回归问题
  - 之后，有诸多改进：Gentle Adaboost, Real AdaBoost, Logit AdaBoost， ……

8.8 Adaboost

- **方法**
  - **核心思想：**给定训练集，寻找比较粗糙的分类规则（弱分类器）要比寻找精确的分类规则要简单得多。从弱学习算法出发，反复学习，得到一系列弱分类器；然后组合这些弱分类器，构成一个强分类器。
  - **策略：**改变训练数据的概率（权重）分布，针对不同的训练数据的分布，调用弱学习算法来学习一系列分类器。
  - **两个基本问题：**
    - 在每轮训练中，如何改变训练数据的权值或分布？
    - 如何将一系列的弱分类器组合成一个强分类器？

8.8 Adaboost

- **方法**
  - **关于第一个问题：**
    - Adaboost的做法是：提高被前一轮弱分类器分错的样本的权重，降低被正确分类的样本的权重。于是，错分的样本将在下一轮弱分类器中得到更多关注，分类问题被一系列弱分类器“分而治之”。
  - **关于第二个问题：**
    - Adaboost的做法是：采用加权投票的方法。具体地，按照弱分类器的分类错误率对其加权，错误率较小的弱分类器获得较大的权重，使其在表决中起更大作用。
  - Adaboost的巧妙之处在于将这些想法融合于一个算法之中！

8.8 Adaboost

- **Adaboost算法**
  - 给定一个两类分类训练数据集

$$T = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$$

其中，每个样本点由实例和标记组成。实例(即样本) $\mathbf{x}_i \in R^d$，标记 $y_i \in \{-1,+1\}$。记 X 为样本空间，Y 为标记集合。

Adaboost从训练数据中学习一系列弱分类器或者基分类器，并将这些弱分类器线性地组合成一个强分类器。

8.8 Adaboost

- **Adaboost算法**
  - 输入训练数据集：$x_i \in R^d, y_i \in \{-1, +1\}$

$$T=\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$$

  - 输入弱学习算法
  - (1) 初始化训练数据的权值分布

$$D_1=\{w_{11}, w_{12}, \ldots, w_{1n}\}, \quad w_{1i}=1/n, \qquad i = 1, \ldots, n$$

  - (2) 对 $m = 1, 2, \ldots, M$

    (2a) 使用具有权值分布 D_m 的训练数据，学习弱分类器

$$G_m(\mathbf{x}): \boldsymbol{X} \rightarrow \{-1, +1\}$$

Adaboost算法(续)

(2b) 计算 $G_m(\mathbf{x})$ 在训练数据集上的加权分类错误率：

$$e_m = P(G_m(\mathbf{x}_i) \neq y_i) = \sum_{i=1}^{n} w_{mi} \underline{I(G_m(\mathbf{x}_i) \neq y_i)}$$

真值函数

(2c) 计算 $G_m(\mathbf{x})$ 的贡献系数：

$$\alpha_m = \frac{1}{2} \ln \frac{1 - e_m}{e_m}$$

α_m **表示** $G_m(\mathbf{x})$ 在最终分类器中的重要性。当 $e_m \leq 0.5$ 时，$\alpha_m \geq 0$。同时，$\boldsymbol{\alpha_m}$ **将随着** $\boldsymbol{e_m}$ **的减小而增大**。

所以，分类误差率越小的弱分类器在最终分类器中的作用越大。

Adaboost算法(续)

(2d) 更新训练数据集的权重分布：

$$D_{m+1}=\{w_{m+1,1}, w_{m+1,2}, \cdots, w_{m+1,n}\}$$

具体计算如下：

$$w_{m+1,i} = \frac{w_{m,i}}{Z_m} \times \begin{cases} \exp(-\alpha_m), & \text{if } G_m(\mathbf{x}_i) = y_i \\ \exp(\alpha_m), & \text{if } G_m(\mathbf{x}_i) \neq y_i \end{cases}$$

若正确分类，减少权重；否则，增加权重

$$= \frac{w_{m,i}}{Z_m} \times \exp(-\alpha_m y_i G_m(\mathbf{x}_i))$$

其中，Z_m 是归一化因子，它使 D_{m+1} 成为一个概率分布：

$$Z_m = \sum_{i=1}^{n} w_{mi} \exp(-\alpha_m y_i G_m(\mathbf{x}_i))$$

Adaboost算法(续)

– (3) 构建弱分类器的线性组合：

$$f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m G_m(\mathbf{x})$$

对于两类分类问题，得到最终的分类器：

$$G(\mathbf{x}) = \text{sign}\left(f(\mathbf{x}) \right) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(\mathbf{x}) \right)$$

8.8 Adaboost

- **对Adaboost算法的说明**
  - 步骤(1)：假设训练数据集具有均匀分布的权重，保证第一步能在原始数据上学习到弱分类器。
  - 步骤(2)：Adaboost 反复学习多个弱分类器，并顺序执行以下操作：
    - (a) 使用当前加权分布 D_m 训练数据，学习弱分类器 $G_m(\mathbf{x})$。

- **对Adaboost算法的说明(续)**
  - 步骤(2)：
    - (b) 计算弱分类器$G_m(x)$在加权训练数据集上的分类错误率：

$$e_m = P(G_m(\mathbf{x}_i) \neq y_i) = \sum_{G_m(\mathbf{x}_i) \neq y_i} w_{mi}$$

      - 其中，w_{mi} 为第 m 轮中第 i 个实例的权值，且 $\sum_{i=1}^{n} w_{mi} = 1$。
      - $G_m(x)$ 在加权训练数据集上的分类错误率是被其错分样本的权值之和。
      - 由此可以看出“数据权值分布 D_m 与弱分类器 $G_m(\mathbf{x})$ 的分类错误率之间的关系”。

- **对Adaboost算法的说明(续)**
  - 步骤(2)：
    - (c) 计算弱分类器 $G_m(\mathbf{x})$ 的系数：

$$\alpha_m = \frac{1}{2}\ln\frac{1-e_m}{e_m}$$

      - α_m 表示 $G_m(\mathbf{x})$ 在最终分类器中的重要性。α_m是 e_m 的**单调递减函数**。
      - 当 $e_m \le 0.5$ 时，$\alpha_m \ge 0$。同时，α_m 将随着 e_m 的减小而增大。所以分类错误率越小的弱分类器在最终分类器中的作用越大。

- **对Adaboost算法的说明(续)**
  - 步骤(2)：
    - (d) 更新训练数据的权值分布，为下一轮作准备：

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \times \begin{cases} \exp(-\alpha_m), & \text{if } G_m(\mathbf{x}_i) = y_i \\ \exp(\alpha_m), & \text{if } G_m(\mathbf{x}_i) \neq y_i \end{cases}$$

被弱分类 $G_m(\mathbf{x})$ 误分的样本的权重得以扩大。相对于正确分类的样本，扩大 $\exp(2\alpha_m) = (1\text{-}e_m)/e_m$倍。因此，误分样本在下一轮学习中起的作用会更大。

不断改变训练样本的权值，使其在弱分类器中起不同的作用，这是Adaboost的一个特点。

- **对Adaboost算法的说明(续)**
  - 步骤(3)：

$$f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m G_m(\mathbf{x})$$

线性组合 $f(\mathbf{x})$ 实现了对M个弱分类器的加权表决。系数 α_m 表示弱分类器 $G_m(\mathbf{x})$ 的重要性。注意，所有 α_m 之和并不为 1。

- $f(\mathbf{x})$ 的符号决定了实例(即样本) $\mathbf{x}$ 的类别， $f(\mathbf{x})$ 的绝对值表示分类的置信度。

利用弱分类器的线性组合构建最终的分类器，也是AdaBoost的另一个特点。

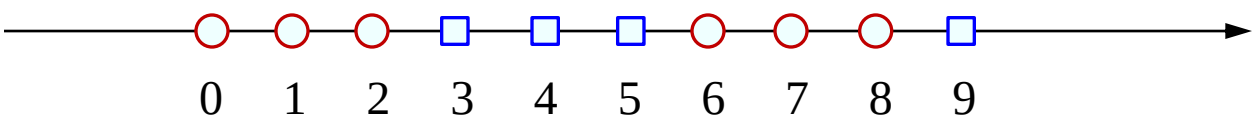
8.8 Adaboost

sign(*x*-*v*)

- 例子：给定如表所示训练数据。假设弱分类器由“如果 $x < v$，则 x 属于第一类；如果 $x > v$，则 x 属于第二类”产生，阈值v使该分类器在训练数据集上分类误差率最低。 试用Adaboost算法学习一个强分类器。

序号	1	2	3	4	5	6	7	8	9	10
x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1

上表中：共10个一维空间的样本，x表示样本，y表示标签。



解： 初始化数据权值分布：

$$D_1 = (w_{1,1}, w_{1,2}, \cdots, w_{1,10}), \quad w_{1i} = 0.1, \quad i = 1, 2, \cdots, 10$$

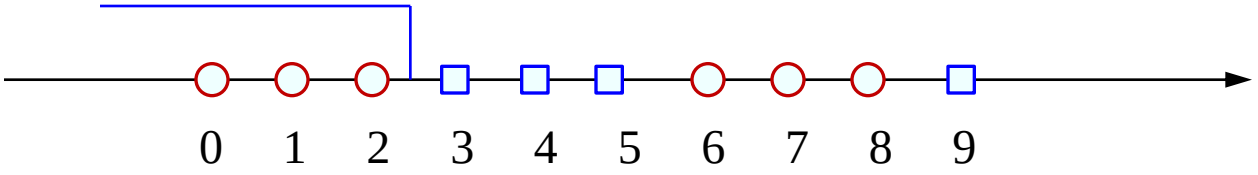
对m=1:

(a) 在权值分布为 D_1 的训练数据上，阈值 v 取2.5时分类误差率最低 (取不同的阈值，然后计算加权错误率，选择最小者)，故弱分类器 $G_1(x)$ 为：

$$G_1(x) = \begin{cases} 1, & x < 2.5 \\ -1, & x > 2.5 \end{cases}$$

(b) $G_1(x)$在训练数据集上的误差率:

$$e_1 = P(G_1(x_i) \neq y_i) = 0.3$$



$$D_1 = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)$$

解(续)：

对m=1:

(c) 计算 $G_1(x)$ 的系数：$\alpha_1 = \frac{1}{2}\log\frac{1-e_1}{e_1} = 0.4236$

(d) 更新训练数据的权值分布：

$$D_2 = (w_{21}, \cdots, w_{2i}, \cdots, w_{210})$$

$$w_{2i} = \frac{w_{1i}}{Z_1}\exp(-\alpha_1 y_i G_1(x_i)), \quad i = 1, 2, \cdots, 10$$

$$D_2 = (0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.0715, 0.1666, 0.1666, 0.1666, 0.0715)$$

(e) 获得 $f_1(x) = 0.4236\, G_1(x)$。

此时，分类器 $\text{sign}[f_1(x)]$ 在训练集上有3个误分样本 $\{x_7, x_8, x_9\}$

解(续)：

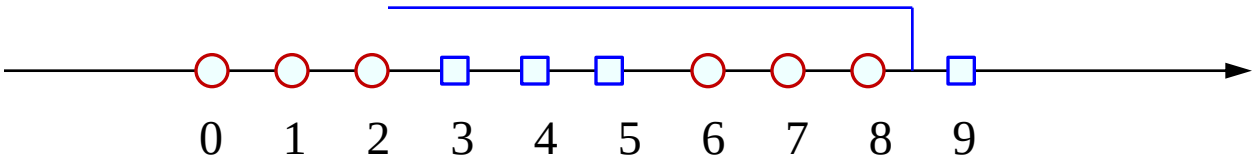
对m=2:

(a) 在权值分布为 D_2 的训练数据上，阈值 v 取8.5时分类误差率最低 (取不同的阈值，遍历，然后计算加权错误率，选择最小者)，故弱分类器 $G_2(x)$ 为：

$$G_2(x)=\begin{cases}1, & x<8.5\\ -1, & x>8.5\end{cases}$$

(b) $G_2(x)$ 在训练数据集上的误差率:

$$e_2=P(G_2(x_i)\neq y_i)=0.2143$$



$$D_2=(0.0715,0.0715,0.0715,0.0715,0.0715,0.0715,0.1666,0.1666,0.1666,0.0715)$$

解(续)：

对m=2:

(c) 计算 $G_2(x)$ 的系数：$\alpha_2 = \frac{1}{2}\log\frac{1-e_2}{e_2} = 0.6496$

(d) 更新训练数据的权值分布：

$$D_3 = (w_{31}, \cdots, w_{3i}, \cdots, w_{310})$$

$$w_{3i} = \frac{w_{2i}}{Z_2}\exp(-\alpha_2 y_i G_2(x_i)), \quad i = 1, 2, \cdots, 10$$

$$D_3 = (0.0455, 0.0455, 0.0455, 0.1667, 0.1667, 0.1667, 0.1060, 0.1060, 0.1060, 0.0455)$$

(e) 获得 $f_2(x) = 0.4236\, G_1(x) + 0.6496\, G_2(x)$

此时，分类器 sign[$f_2(x)$] 在训练集上有3个误分样本$\{x_4, x_5, x_6\}$

解(续)：

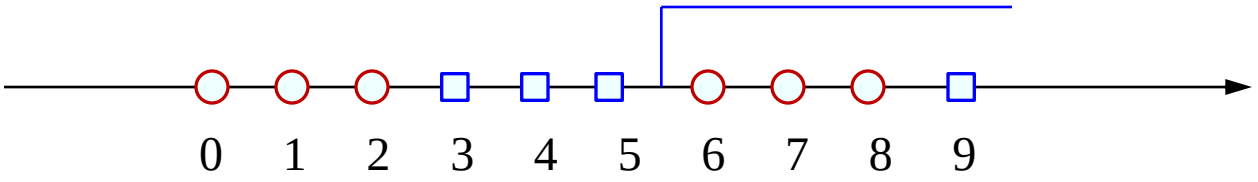
对m=3:

(a) 在权值分布为 D_3 的训练数据上，阈值 v 取5.5时分类误差率最低，故弱分类器 $G_3(x)$ 为：

$$G_3(x)=\begin{cases}1, & x>5.5\\ -1, & x<5.5\end{cases}$$

(b) $G_3(x)$在训练数据集上的误差率:

$$e_3=P(G_3(x_i)\neq y_i)=0.1820$$



$$D_3=(0.0455,0.0455,0.0455,0.1667,0.1667,0.1667,0.1060,0.1060,0.1060,0.0455)$$

解(续)：

对m=3:

(c) 计算 $G_3(x)$的系数： $\alpha_3 = \frac{1}{2}\log\frac{1-e_3}{e_3} = 0.7514$

(d) 更新训练数据的权值分布：

$$D_4 = (w_{41}, \cdots, w_{4i}, \cdots, w_{410})$$

$$w_{4i} = \frac{w_{3i}}{Z_3}\exp(-\alpha_3 y_i G_3(x_i)), \quad i = 1, 2, \cdots, 10$$

$$D_4 = (0.125, 0.125, 0.125, 0.102, 0.102, 0.102, 0.065, 0.065, 0.065, 0.125)$$

(e) 获得 $f_3(x) = 0.4236\, G_1(x) + 0.6496\, G_2(x) + 0.7514\, G_3(x)$

此时，分类器 $\text{sign}[f_3(x)]$ 在训练数据集误分样本个数为0。

获得最终分类器： $\text{sign}[f_3(x)]$

8.9 Adaboost的理论解释

- **关于Adaboost算法的理论解释，有如下三个定理：**
  - 定理1：最终分类器关于训练误差的界
  - 定理2：最终分类器关于训练误差的界 (更具体)
  - 定理3：Adaboost与前向分步算法(加法模型)之间的关系

李航著：统计学习方法(第8章), 清华大学出版社, 2012

8.9 Adaboost的理论解释

$$f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m G_m(\mathbf{x})$$

$$G(\mathbf{x}) = \text{sign}\left(f(\mathbf{x}) \right)$$

- **定理1：Adaboost算法最终分类器的训练误差界为：**

$$\frac{1}{n}\sum_{i=1}^{n} I(G(\mathbf{x}_i) \neq y_i) \leq \frac{1}{n}\sum_{i=1}^{n} \exp\left(-y_i f(\mathbf{x}_i)\right) = \prod_{m=1}^{M} Z_m$$

训练误差：
错误个数除以总数

上界：
归一化因子的乘积

Adaboost最基本的性质是它能在学习过程中不断地减少训练误差。

- **定理1证明：第一部分**

  - 当$G(\mathbf{x}_i) \neq y_i$，　$y_i f(\mathbf{x}_i) < 0$，此时$\exp(-y_i f(\mathbf{x}_i)) \geq 1$。综合起来有：

$$\frac{1}{n}\sum_{i=1}^{n} I(G(\mathbf{x}_i) \neq y_i) \leq \frac{1}{n}\sum_{i=1}^{n} \exp\left(-y_i f(\mathbf{x}_i)\right)$$

- **定理1证明：第二部分**

注意到样本权重的更新方式：

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \times \exp(-\alpha_m y_i G_m(\mathbf{x}_i))$$

所以，有：$w_{m+1,i} Z_m = w_{mi} \exp(-\alpha_m y_i G_m(\mathbf{x}_i)),\ i = 1,..,n$

$$\frac{1}{n}\sum_{i=1}^{n}\exp\left(-y_i f(\mathbf{x}_i)\right)$$

$$=\frac{1}{n}\sum_{i=1}^{n}\exp\left(-\sum_{m=1}^{M}\alpha_m y_i G_m(\mathbf{x}_i)\right)$$

$$=\sum_{i=1}^{n}w_{1i}\prod_{m=1}^{M}\exp\left(-\alpha_m y_i G_m(\mathbf{x}_i)\right)\quad \because w_{1i}=\frac{1}{n}$$

$$=\sum_{i=1}^{n}w_{1i}\exp\left(-\alpha_1 y_i G_1(\mathbf{x}_i)\right)\prod_{m=2}^{M}\exp\left(-\alpha_m y_i G_m(\mathbf{x}_i)\right)$$

$$=\sum_{i=1}^{n}Z_1 w_{2i}\prod_{m=2}^{M}\exp\left(-\alpha_m y_i G_m(\mathbf{x}_i)\right)$$

$$=Z_1\sum_{i=1}^{n}w_{2i}\exp\left(-\alpha_1 y_i G_1(\mathbf{x}_i)\right)\prod_{m=3}^{M}\exp\left(-\alpha_m y_i G_m(\mathbf{x}_i)\right)$$

$$=Z_1\sum_{i=1}^{n}Z_2 w_{3i}\prod_{m=3}^{M}\exp\left(-\alpha_m y_i G_m(\mathbf{x}_i)\right)$$

$$=Z_1 Z_2\sum_{i=1}^{n}w_{3i}\prod_{m=3}^{M}\exp\left(-\alpha_m y_i G_m(\mathbf{x}_i)\right)$$

$$=Z_1 Z_2\cdots Z_{M-1}\sum_{i=1}^{n}w_{Mi}\exp\left(-\alpha_M y_i G_M(\mathbf{x}_i)\right)$$

$$=\prod_{m=1}^{M}Z_m$$

（后续见右上）

证毕

8.9 Adaboost的理论解释

- **定理2：Adaboost的训练误差界可进一步写为:**

$$\prod_{m=1}^{M} Z_m = \prod_{m=1}^{M}\left(2\sqrt{e_m(1-e_m)}\right) = \prod_{m=1}^{M}\left(\sqrt{1-4\gamma_m^2}\right) \le \exp\left(-2\sum_{m=1}^{M}\gamma_m^2\right)$$

其中，$\gamma_m = 0.5 - e_m$。

- **定理2证明:**

$$Z_m = \sum_{i=1}^{n} w_{mi} \exp(-\alpha_m y_i G_m(\mathbf{x}_i))$$

$$= \sum_{y_i = G_m(\mathbf{x}_i)} w_{mi} \, \mathrm{e}^{-\alpha_m} + \sum_{y_i \neq G_m(\mathbf{x}_i)} w_{mi} \, \mathrm{e}^{\alpha_m}$$

$$= (1 - e_m)\mathrm{e}^{-\alpha_m} + e_m e^{\alpha_m}$$

$$= 2\sqrt{e_m(1-e_m)} \quad = \quad \sqrt{1-4\gamma_m^2} \qquad \because \alpha_m = \frac{1}{2}\ln\frac{1-e_m}{e_m}$$

对于不等式： $\prod_{m=1}^{M}\left(\sqrt{1-4\gamma_m^2}\right) \leq \exp\left(-2\sum_{m=1}^{M}\gamma_m^2\right)$

可先由 e^x 和 $\sqrt{1-x}$ 在 x=0 的泰勒展开式推出不等式：

$$\sqrt{1-4\gamma_m^2} \leq \exp\left(-2\gamma_m^2\right)$$

8.9 Adaboost的理论解释

- **推论：如果存在一个$\gamma>0$，对所有弱分类器$G_m(\mathbf{x})$有$\gamma_m \geq \gamma$，则：**

$$\frac{1}{n}\sum_{i=1}^{n} I(G(\mathbf{x}_i) \neq y_i) \leq \exp\left(-2\sum_{m=1}^{M}\gamma_m^2\right) \leq \exp\left(-2M\gamma^2\right)$$

这表明 Adaboost的训练误差随弱分类器的增多而以指数速度下降。(γ 总是可以大于0的)

要注意这是算法的一个性质。实际中Adaboost 并不需要知道这个下界 γ 。这正是Freund和Schapire设计 Adaboost 时所考虑的。因此，与早期的方法不同， Adaboost具有自适应性，即它能适应弱分类器各自的训练误差。**这正是其名称的由来。**

8.9 Adaboost的理论解释

- **从加法模型的角度对Adaboost的解释**
  - 定理3：Adaboost算法是一种学习模型为加法模型、损失函数为指数函数、学习方法为前向分步算法的两类分类学习方法。
- **加法模型：**

$$f(\mathbf{x}) = \sum_{m=1}^{M} \beta_m \, b(\mathbf{x}; \gamma_m)$$

其中，$b(\mathbf{x}; \gamma_m)$ 为基函数，γ_m 为基函数的参数，β_m 为基函数的权重。

8.9 Adaboost的理论解释

- **对加法模型的学习**
  - 给定训练数据及损失函数 $L(y, f(\mathbf{x}))$，加法模型 $f(\mathbf{x})$ 可以通过最小化经验风险（即所有训练样本引起的损失之和）来学习：

$$\min_{\{\beta_m,\gamma_m\}} \sum_{i=1}^{n} L\left(y_i, \sum_{m=1}^{M} \beta_m b(\mathbf{x}_i;\gamma_m)\right)$$

这是一个复杂的优化问题，期望通过最优化技术一次性求出来通常很困难。取决于基函数的形式。

8.9 Adaboost的理论解释

- **对加法模型的学习**
  - 前向分步算法 (forward stagewise algorithm)
  - **求解该问题的基本思路：**
    - 因为学习模型是加法模型，每一步只需学习一个基函数及其参数，逐步逼近优化目标函数（见前一页），那么就可以简化优化的复杂度。具体地，每步只优化如下损失函数：

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{n} L\left(y_i,\ f_{m-1}(\mathbf{x}_i) + \beta b(\mathbf{x}_i;\gamma)\right)$$

8.9 Adaboost的理论解释

- **前向分步算法**
  - 输入训练数据集：$T=\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$
  - 给定损失函数 $L(y, f(\mathbf{x}))$，基函数 $b(\mathbf{x}; \gamma)$；
  - (1) 初始化 $f_0(\mathrm{x}) = 0$
  - (2) 对 $m = 1, 2, \ldots, M$：
    - (2a) 极小化当前损失函数，得到参数β_m，γ_m：

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{n} L\left(y_i, f_{m-1}(\mathbf{x}_i) + \beta b(\mathbf{x}_i; \gamma)\right)$$

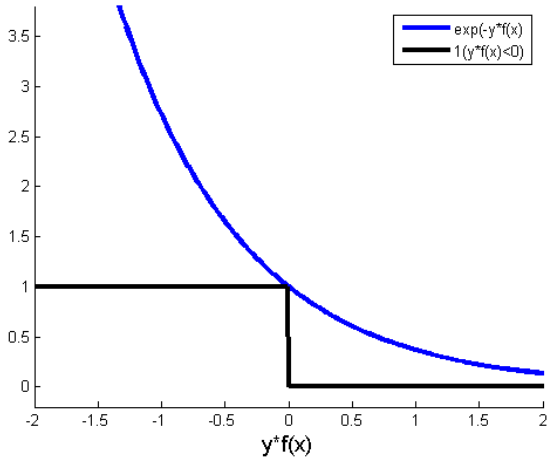
    - (2b) 更新：$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \beta_m b(\mathbf{x}; \gamma_m)$
  - (3) **获得最后模型：** $f(\mathbf{x}) = f_M(\mathbf{x}) = \sum_{m=1}^{M} \beta_m b(\mathbf{x}; \gamma_m)$

8.9 Adaboost的理论解释

- 定理3：AdaBoost是前向分步算法的特例，且学习模型是由基分类器组成的加法模型，损失函数为指数函数。

我们只需要证明当前向分步算法的损失函数为指数函数时，该算法就变为Adaboost，此定理即可获证。

具体的指数函数为： $L(y, f(x)) = \exp(-y f(x))$。



- **定理3的证明**
  - 假设经过 m-1轮迭代，前向分步算法已经得到 $f_{m-1}(\mathbf{x})$：

$$f_{m-1}(\mathbf{x}) = \alpha_1 G_1(\mathbf{x}) + \alpha_2 G_2(\mathbf{x}) + \ldots + \alpha_{m-1} G_{m-1}(\mathbf{x})$$

  - 进一步，在第m轮前向分步算法通过求解如下化优模型得到α_m，$G_m(\mathbf{x})$：

$$\left(\alpha_m, G_m(\mathbf{x})\right) = \arg\min_{\alpha, G(\mathbf{x})} \sum_{i=1}^{n} \exp\left(-y_i\left(f_{m-1}(\mathbf{x}_i) + \alpha G(\mathbf{x}_i)\right)\right)$$

令 $\bar{w}_{mi} = \exp\left(-y_i f_{m-1}(\mathbf{x}_i)\right)$ ← 与 α 和 $G(\mathbf{x})$无关

  - 因此，上述优化问题可表示为：

$$\left(\alpha_m, G_m(\mathbf{x})\right) = \arg\min_{\alpha, G(\mathbf{x})} \sum_{i=1}^{n} \bar{w}_{mi} \exp\left(-y_i \alpha G(\mathbf{x}_i)\right)$$

- **定理3的证明(续)**
  - 进一步只需证明：

求解如下问题获得的最优解 α^*_m 和 $G^*_m(\mathbf{x})$ 就是AdaBoost算法所得到的 α_m 和 $G_m(\mathbf{x})$：

$$\left(\alpha_m, G_m(\mathbf{x})\right) = \arg\min_{\alpha, G(\mathbf{x})} \sum_{i=1}^{n} \bar{w}_{mi} \exp\left(-y_i \alpha G(\mathbf{x}_i)\right)$$

其中，$\bar{w}_{mi} = \exp\left(-y_i f_{m-1}(\mathbf{x}_i)\right)$

因此，当前任务是如何求解上述优化问题。

- **定理3的证明(续)**

对目标函数作如下变换：

$$\sum_{i=1}^{n} \bar{w}_{mi} \exp\left(-y_i \alpha G(\mathbf{x}_i)\right)$$

$$= \sum_{y_i = G(\mathbf{x}_i)} \bar{w}_{mi} \, \mathrm{e}^{-\alpha} + \sum_{y_i \neq G(\mathbf{x}_i)} \bar{w}_{mi} \, \mathrm{e}^{\alpha}$$

$$= \mathrm{e}^{-\alpha} \sum_{i=1}^{n} \bar{w}_{mi} - \mathrm{e}^{-\alpha} \sum_{i=1}^{n} \bar{w}_{mi} I(y_i \neq G(\mathbf{x}_i)) + \mathrm{e}^{\alpha} \sum_{i=1}^{n} \bar{w}_{mi} I(y_i \neq G(\mathbf{x}_i))$$

$$= \mathrm{e}^{-\alpha} \sum_{i=1}^{n} \bar{w}_{mi} + (\mathrm{e}^{\alpha} - \mathrm{e}^{-\alpha}) \sum_{i=1}^{n} \bar{w}_{mi} I(y_i \neq G(\mathbf{x}_i))$$

- **定理3的证明(续)**
  - 首先求$G^*_m(\mathbf{x})$：

目标函数：
$$\mathrm{e}^{-\alpha}\sum_{i=1}^{n}\bar{w}_{mi}+(\mathrm{e}^{\alpha}-\mathrm{e}^{-\alpha})\sum_{i=1}^{n}\bar{w}_{mi}I(y_i\neq G(\mathbf{x}_i))$$

与$G(\mathbf{x})$无关　$\alpha>0$时，恒大于零

因此有如下等价转换：

$$G_m^*(\mathbf{x})=\arg\min_{G(\mathbf{x})}\sum_{i=1}^{n}\bar{w}_{mi}\exp\left(-y_i\alpha G(\mathbf{x}_i)\right)$$

$$\Updownarrow$$

$$G_m^*(\mathbf{x})=\arg\min_{G(\mathbf{x})}\sum_{i=1}^{n}\bar{w}_{mi}I\left(y_i\neq G(\mathbf{x}_i)\right)$$

这正是Adaboost第m步时所构造的弱分类器。

- **定理3的证明(续)**
  - 其次求 α^*_m：

目标函数：
$$\mathrm{e}^{-\alpha}\sum_{i=1}^{n}\bar{w}_{mi}+(\mathrm{e}^{\alpha}-\mathrm{e}^{-\alpha})\sum_{i=1}^{n}\bar{w}_{mi}I(y_i\neq G(\mathbf{x}_i))$$

对 α 求导数，并令其等于0，可得：
$$\alpha^*_m=\frac{1}{2}\ln\frac{1-e_m}{e_m}$$

其中，e_m是$G(\mathbf{x})$的加权分类错误率：
$$e_m=\frac{\sum_{i=1}^{n}\bar{w}_{mi}I(y_i\neq G(\mathbf{x}_i))}{\sum_{i=1}^{n}\bar{w}_{mi}}=\sum_{i=1}^{n}w_{mi}I(y_i\neq G(\mathbf{x}_i))$$

这里的 α^*_m 与Adaboost 算法的第 2(c) 步的 α_m 完全一致。

- **定理3的证明(续)**
  - 关于样本权重的更新，有如下两式

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \alpha_m G_m(\mathbf{x}), \quad \bar{w}_{mi} = \exp\left(-y_i f_{m-1}(\mathbf{x}_i)\right)$$

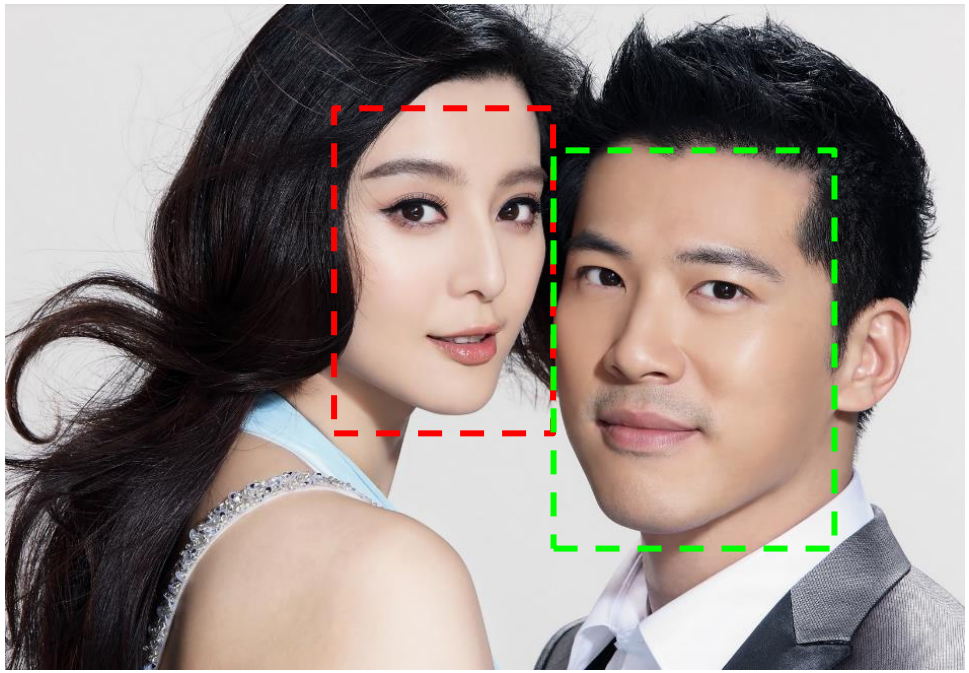
可得：
$$\begin{aligned}
\bar{w}_{mi} &= \exp\left(-y_i f_{m-1}(\mathbf{x}_i)\right) \\
&= \exp\left(-y_i \left(f_{m-2}(\mathbf{x}) + \alpha_{m-1} G_{m-1}(\mathbf{x})\right)\right) \\
&= \exp\left(-y_i \left(f_{m-2}(\mathbf{x})\right)\right) \cdot \exp\left(-y_i \alpha_{m-1} G_{m-1}(\mathbf{x})\right) \\
&= \bar{w}_{m-1i} \exp\left(-y_i \alpha_{m-1} G_{m-1}(\mathbf{x})\right)
\end{aligned}$$

这与 Adaboost 算法的第2(d)步样本权值更新形式完全相同，只差一个归一化因子(不影响α和$G(\mathbf{x})$的求解)，因而等价。

证毕.

8.10 基于Adaboost的人脸检测

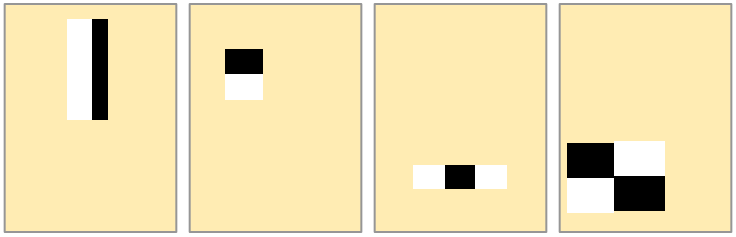
- **人脸检测**



Paul Viola, Michael Jones. Rapid Object Detection Using a Boosted Cascade of Simple Features, CVPR, Vol.1, pp. 551-558, 2001

8.10 基于Adaboost的人脸检测

- **Haar特征—矩形特征**
  - 一个Haar特征由一个矩形滤波器组成
  - 一个Haar特征的响应值为白色滤波器响应值减去灰色滤波器响应值（即白色区域的像素灰度之和减去黑色区域的像素灰度之和）



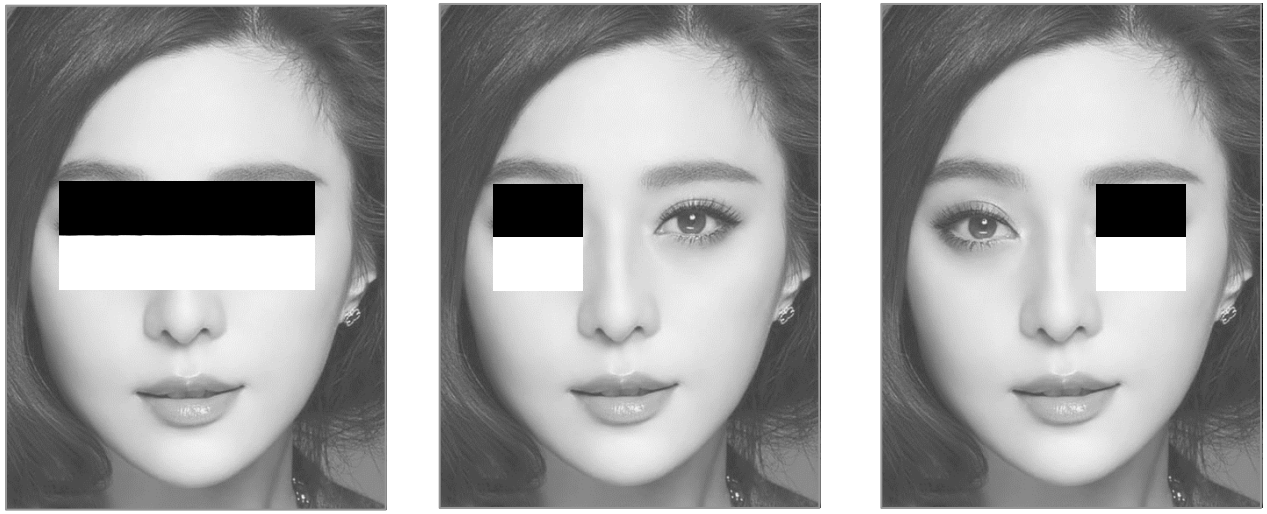
Haar(*x*, y, *w*, *h*, *type*)

矩形模板的位置和大小

矩形模板的类型

8.10 基于Adaboost的人脸检测

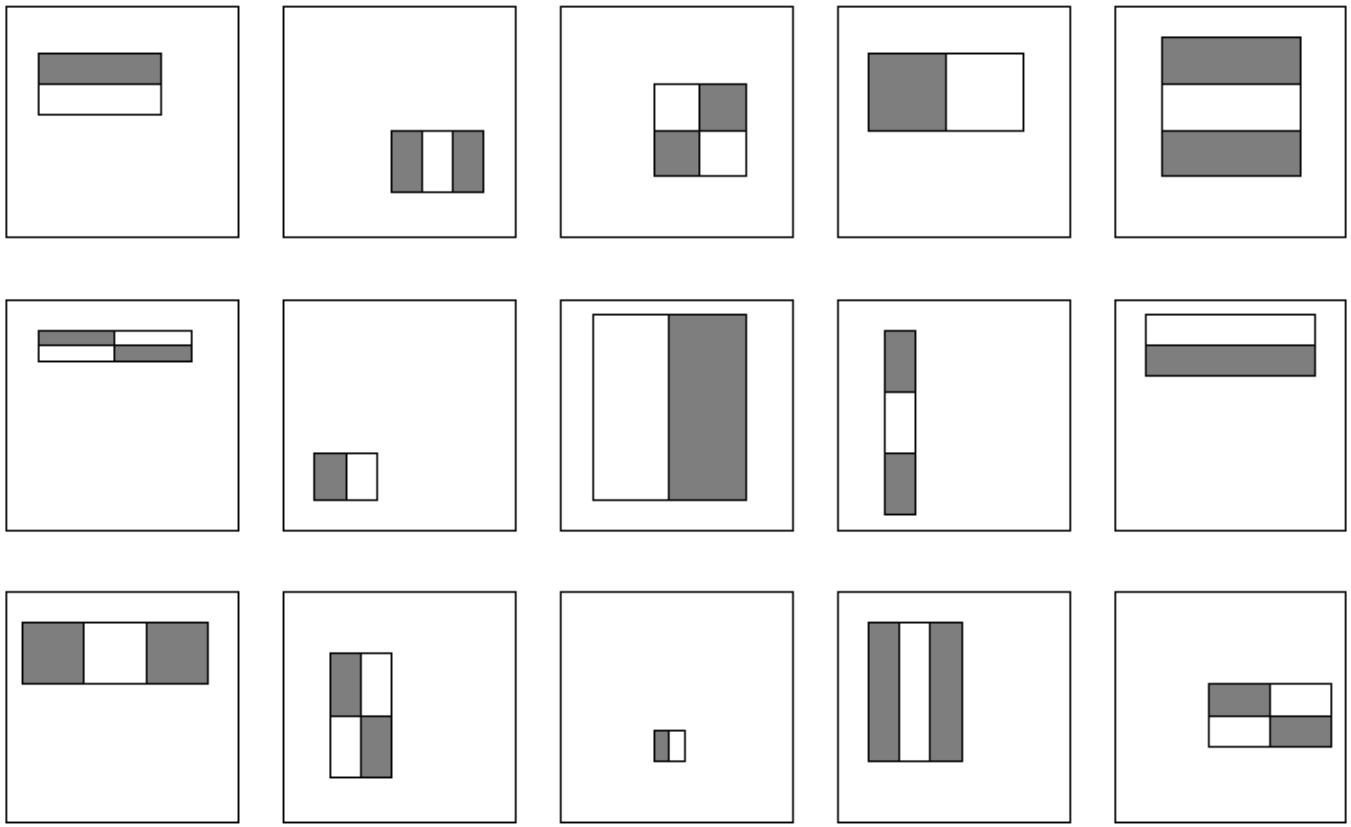
- **Haar特征—矩形特征**



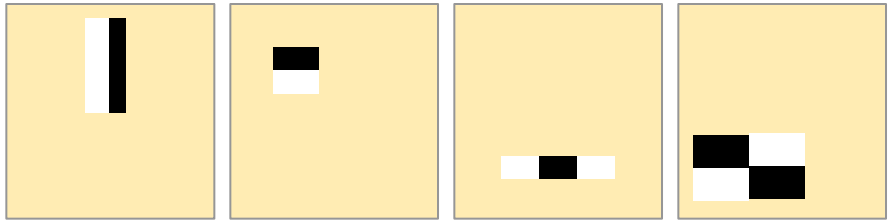
可很好地描述人脸的眼部区域的灰度分布情况

8.10 基于Adaboost的人脸检测

- **Haar特征—**能构造的矩形模板很多！



- **Haar特征—能构造的矩形模板很多！**



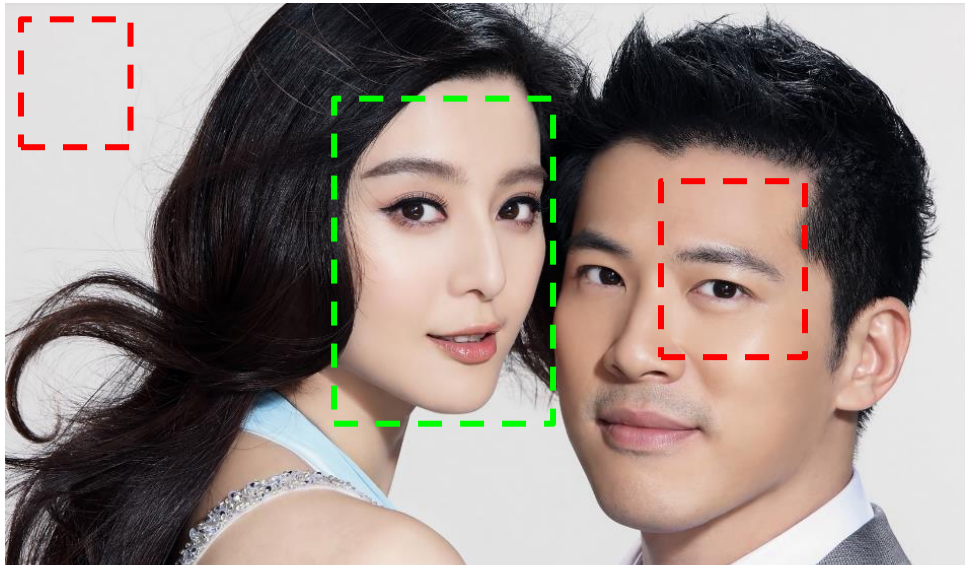
对于一个24x24的图像窗口，从一些基本的矩形模板出发，根据其比例可缩放，位置可移动，角度可旋转，可以构造出多达**几十万甚至上百万个**不同的矩形模板。

这就意味着，从一张图像，可获得一个几十万维甚至上百万维的特征向量！

我们当然可以根据训练集样本（人脸和非人脸），采用SVM来完成此任务！

8.10 基于Adaboost的人脸检测

- **Haar特征—计算量太大！**

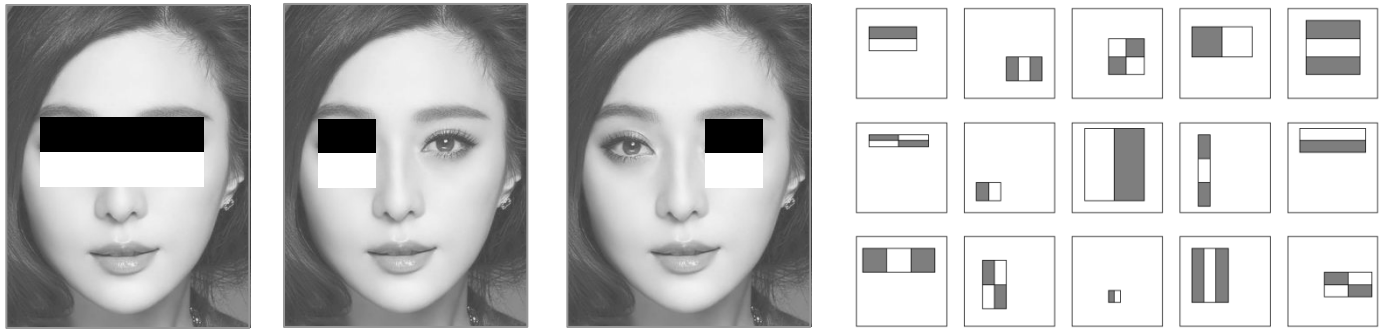


由于不知道人脸的具体尺寸，不同比例的窗口都得扫描一遍！

（做法：每次取固定大小的窗口按行按列扫描图像，每获得一个窗口，首先将其缩放到**训练图像的大小**，比如：24x24，然后获得上几十万维Haar特征，最后采用SVM进行分类）

8.10 基于Adaboost的人脸检测

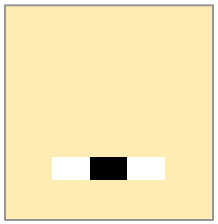
- **Haar特征—换一个角度！**



让我们来评价
每一个Haar特征对人脸的描述能力，
也就是评价其分类判别能力！

这相当于：对每一个Haar特征（一个标量）引入一个弱分器！

8.10 基于Adaboost的人脸检测



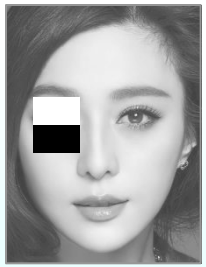
- 从每一个Haar特征构建一个弱分类器！
  - 首先，对于一个给定子窗口和矩形模板，计算其Haar特征值Haar($x, y, w, h, type$)，它是一个标量。
  - 然后，引入一个分类器：

$$G(x,y,w,h,type)=\begin{cases}1, & \text{if } \mathrm{Haar}(x,y,w,h,type)<\theta \\ 0, & \text{otherwise}\end{cases}$$

含义：通过比较Haar特征值是否超过某个阈值来判断是否为人脸

注意，阈值θ 是可变的，因此它是一个待确定的参数。

- 如何将弱分类器设计得更好，更灵活！

计算左边矩形模板的一个Haar特征，可同时得到右边一个模板的Haar特征！（到底取哪一个更适合表示人脸呢）

$$G(x,y,w,h,type)=\begin{cases}1, & \text{if} \quad p\cdot \text{Haar}(x,y,w,h,type)<p\cdot\theta \\ 0, & \text{otherwise}\end{cases}$$

进一步，用符号“f”来表示 Haar(x, y, w, h, type)，则该弱分类器可写成明确的参数形式：

$$G(\mathbf{x},f,p,\theta)=\begin{cases}1, & \text{if} \quad pf<p\theta \\ 0, & \text{otherwise}\end{cases} \qquad (p=\pm 1)$$

8.10 基于Adaboost的人脸检测

- **从训练弱分类器开始**

现在，我们有了几十万个弱分类器，每个
弱分类器有两个参数！

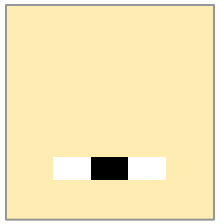
回忆一下如何构造强分类器！

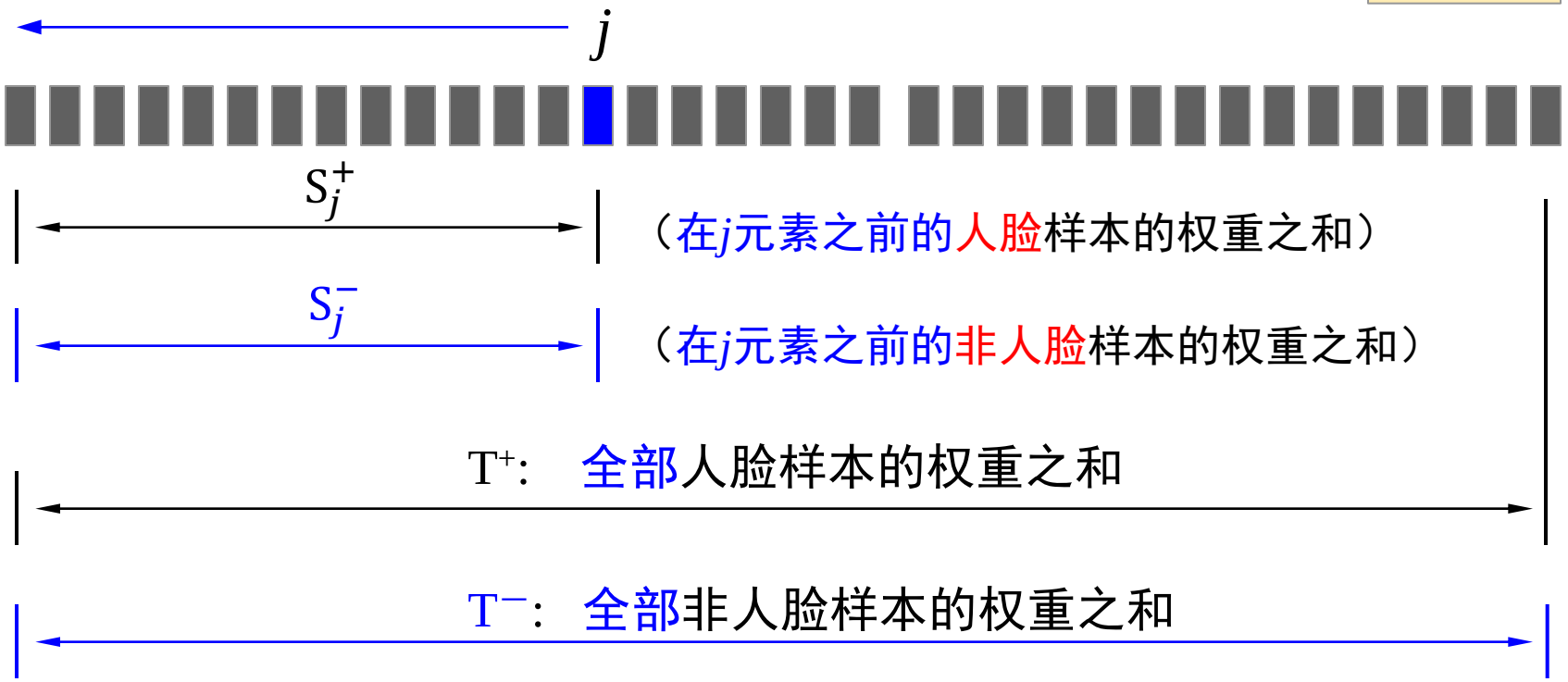
在每次迭代确定一个弱分类器时，需要找一个
错误率最小的弱分类器。
也就是说，此时要通过学习的方式来确定一个弱分类器！

让我们开始吧！
在刚开始的时候，所有样本的权重都是相同的！

- **弱分类器训练**

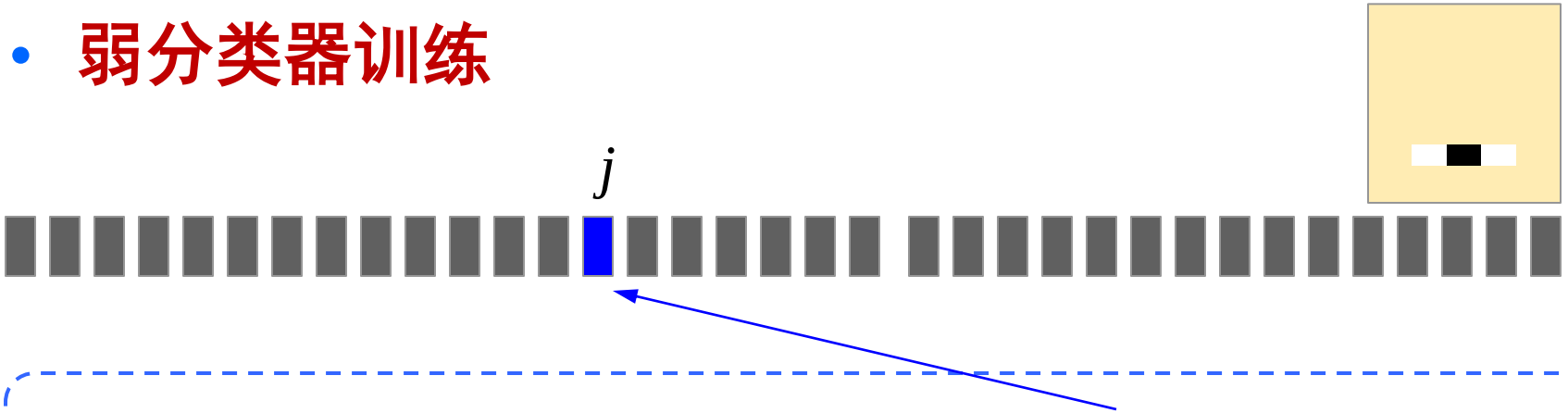
所有样本在同一个Harr特征的特征值（排序）：





逐元素扫描此排序，计算以上四个量：T^+、T^-、$\{S_j^+\}$、$\{S_j^-\}$。

- **弱分类器训练**



假设选取当前元素(第j个)的特征值作为阈值，且所得弱分类器将样本分为两类。有以下两种情形：

- ✓ 该元素之前的所有样本分为**人脸**，该元素之后(含)的所有样本分为**非人脸**；
- ✓ 该元素之前的所有样本分为**非人脸**，该元素之后(含)的所有样本分为**人脸**。

以上两种情形，哪一种更好呢？根据错误率来决定！

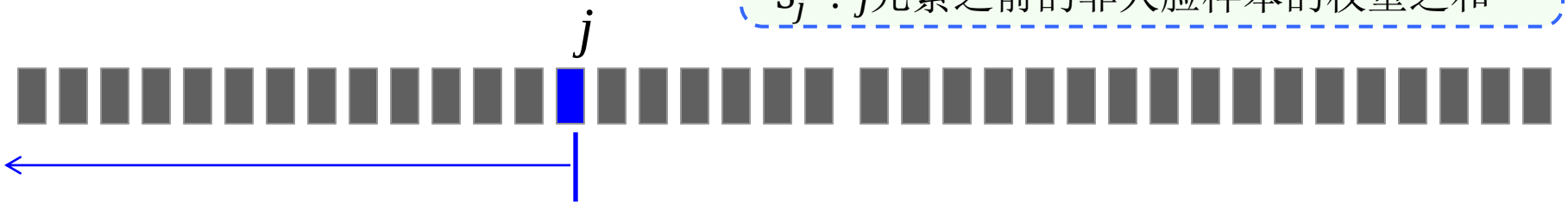
- **弱分类器误差计算**

T^+：全部人脸样本的权重之和

T^-：全部非人脸样本的权重之和

S_j^+：j元素之前的人脸样本的权重之和

S_j^-：j元素之前的非人脸样本的权重之和



如果该元素之前为人脸，之后为非人脸，则误差为：

$$S_j^- + (T^+ - S_j^+)$$

错分：之前为“非人脸”的样本权重总和

错分：之后为“人脸”样本权重总和

$$p = +1: \quad G(\mathbf{x}, f, p, \theta) = \begin{cases} 1, & \text{if } f < \theta \\ 0, & \text{otherwise} \end{cases}$$

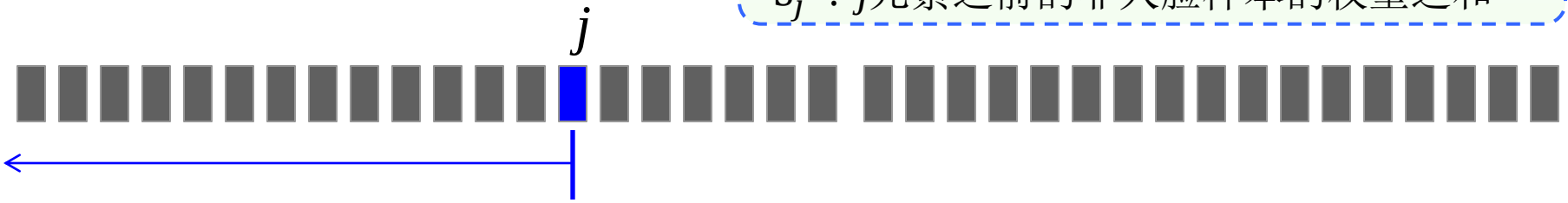
- **弱分类器误差计算**

T^+：全部人脸样本的权重之和

T^-：全部非人脸样本的权重之和

S_j^+：j元素之前的人脸样本的权重之和

S_j^-：j元素之前的非人脸样本的权重之和



如果该元素之前为非人脸，之后为人脸，则误差为：

$$S_j^+ + (T^- - S_j^-)$$

错分：之前为“人脸”的样本权重总和

错分：之后为“非人脸”样本权重总和

$$p = -1: \quad G(\mathbf{x}, f, p, \theta) = \begin{cases} 1, & \text{if } f > \theta \\ 0, & \text{otherwise} \end{cases}$$

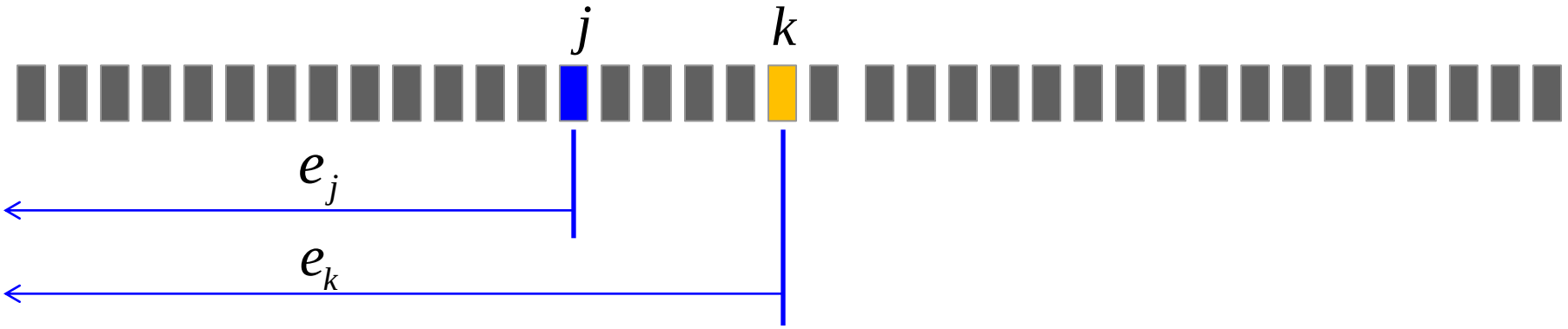
- **弱分类器误差计算**

T^+：全部人脸样本的权重之和

T^-：全部非人脸样本的权重之和

S_j^+：j元素之前的人脸样本的权重之和

S_j^-：j元素之前的非人脸样本的权重之和



现在，取两个误差中的最小者：

$$e_j = \min\left(S_j^- + (T^+ - S_j^+),\ S_j^+ + (T^- - S_j^-)\right), \quad j = 1, 2, \cdots, n$$

最后，遍历所有样本，取所有误差 $\{e_j\}_{j=1}^n$ 中的最小者，获得一个弱分类器。通过该方法，弱分类器的两个参数都确定了！

8.10 基于Adaboost的人脸检测

- 强分类器构造

现在，我们知道了如何在当前样本的权值分布下学习一个最优的弱分类器。

那么，如何构造强分类器？

按在8. 7节讲的步骤即可！即得到如下组合函数：

$$f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m G_m(\mathbf{x})$$

最后，可以得到一个强分类器（按Viola和 Jones建议）

$$H(x) = \begin{cases} 1, & f(\mathbf{x}) \geq \frac{1}{2}\sum_{i=1}^{M} \delta_i \\ 0, & \text{otherwise} \end{cases} \quad \left(\text{where } \delta_t = \log 1/\beta_t, \quad \beta_t = e_t/(1-e_t)\right)$$

8.10 基于Adaboost的人脸检测

- **获得强分类器**

$$f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m G_m(\mathbf{x})$$

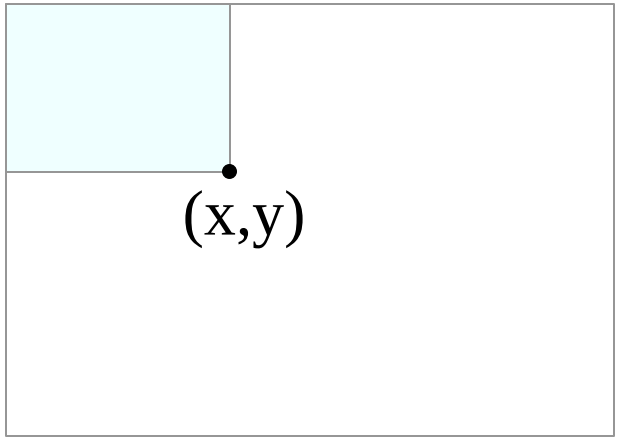
$$H(x) = \begin{cases} 1, & f(\mathbf{x}) \geq \dfrac{1}{2}\sum_{m=1}^{M} \delta_m \\ 0, & \text{otherwise} \end{cases}$$

$$\left(\delta_m = \log 1/\beta_m, \quad \beta_m = e_m/(1-e_m)\right)$$

对于一幅待检测图像，该强分类器相当于让所有弱分类器投票，再对投票结果按弱分类器的错误率加权求和，将此和进一步与平均结果比较得出最终的结果。

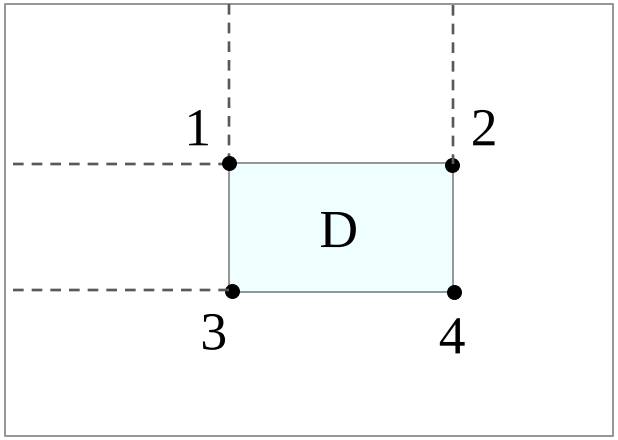
8.10 基于Adaboost的人脸检测

- **其它补充说明1**
  - 积分图：快速计算指定区域内的像素灰度的总和，因此该概念与计算Haar特征Haar(x, y, w, h, type)有关。
  - 积分图：每个点上的值对应图像左上角区域的像素和。



$$I'(x,y)=\sum_{u\le x,v\le y} I(u,v)$$

积分图的计算

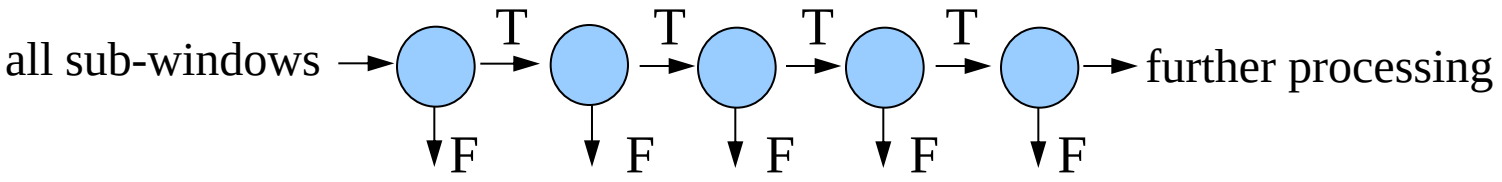


$$sum(D)=I'(4)+I'(1)-I'(2)-I'(3)$$

任意区域像素和的计算

8.10 基于Adaboost的人脸检测

- **其它补充说明2**
  - 我们已经获得了强分类器，按理说已经结束了对Adaboost的训练。为什么要建立cascade结构？
    - 这个主要是两点考虑：
      - 一是速度，级联的时候有出口能迅速过滤掉容易搞定的负样本，这样后面计算量就少很多了；
      - 二是分治，早期的特征表达能力不够强，这样级联能让后面的分类器训练时有针对性地解决当前留下的难样本



8.10 基于Adaboost的人脸检测

- **其它补充说明3**

大家应该看到背后所隐含的特征选择功能！

致谢

- PPT由向世明老师提供

Thank All of You!

(Questions?)

张燕明

ymzhang@nlpr.ia.ac.cn

people.ucas.ac.cn/~ymzhang

模式分析与学习课题组(PAL)

中科院自动化研究所· 模式识别国家重点实验室