2024-2025学年 第1学期(秋)

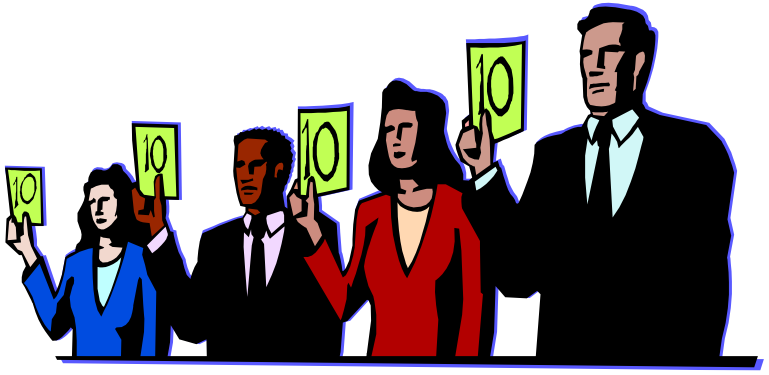
数据挖掘

集成学习

2024 年 12月

“三个臭皮匠（凑裨将），顶过诸葛亮”

投票法则



集成学习思想

- 在机器学习中，直接建立一个高性能的分类器是很困难的。
- 但是，如果能找到一系列性能相对较弱的分类器，并把它们集成起来的话，也许就能得到更好的分类器。

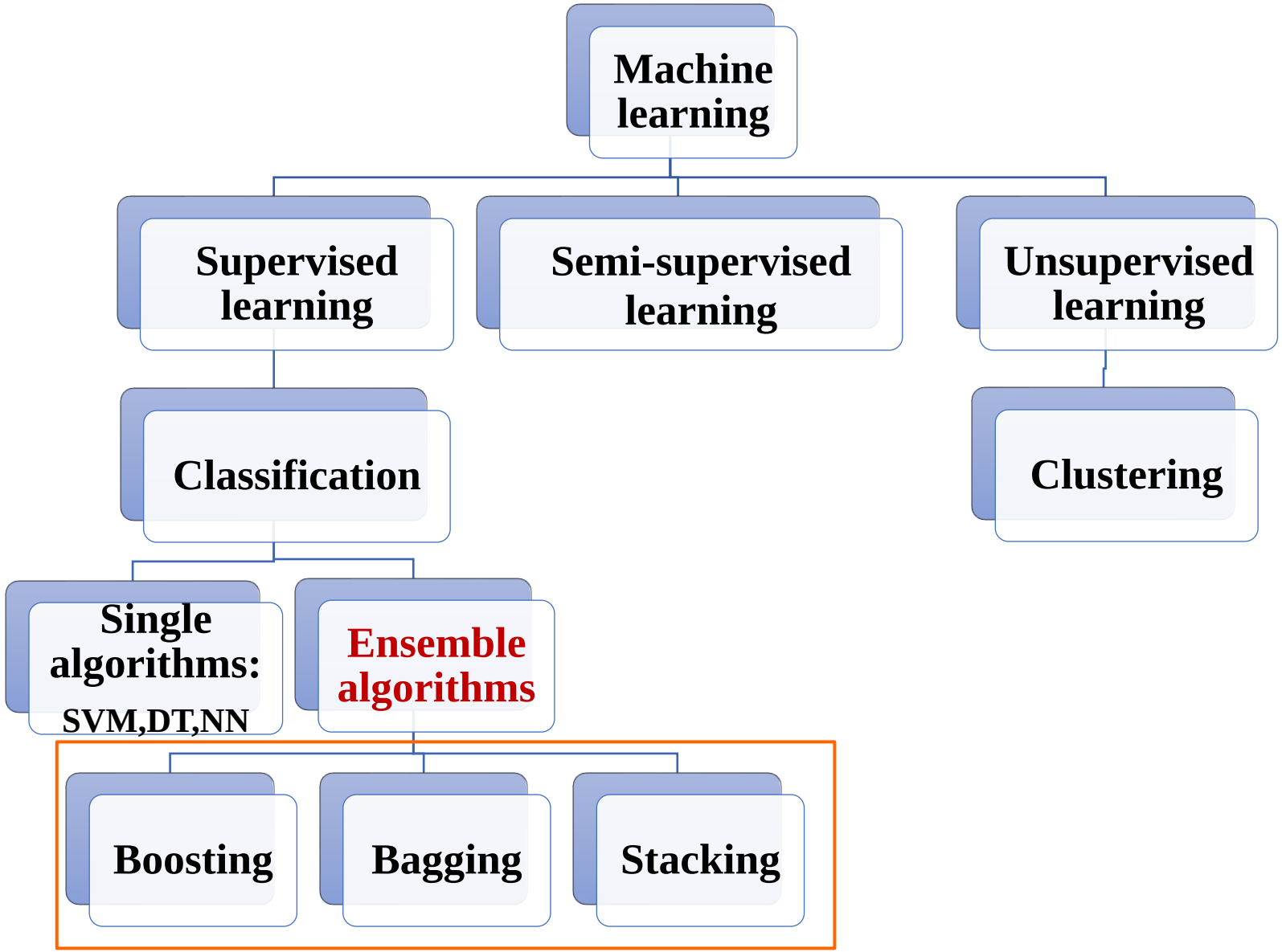
南京大学
NANJING UNIVERSITY

目录

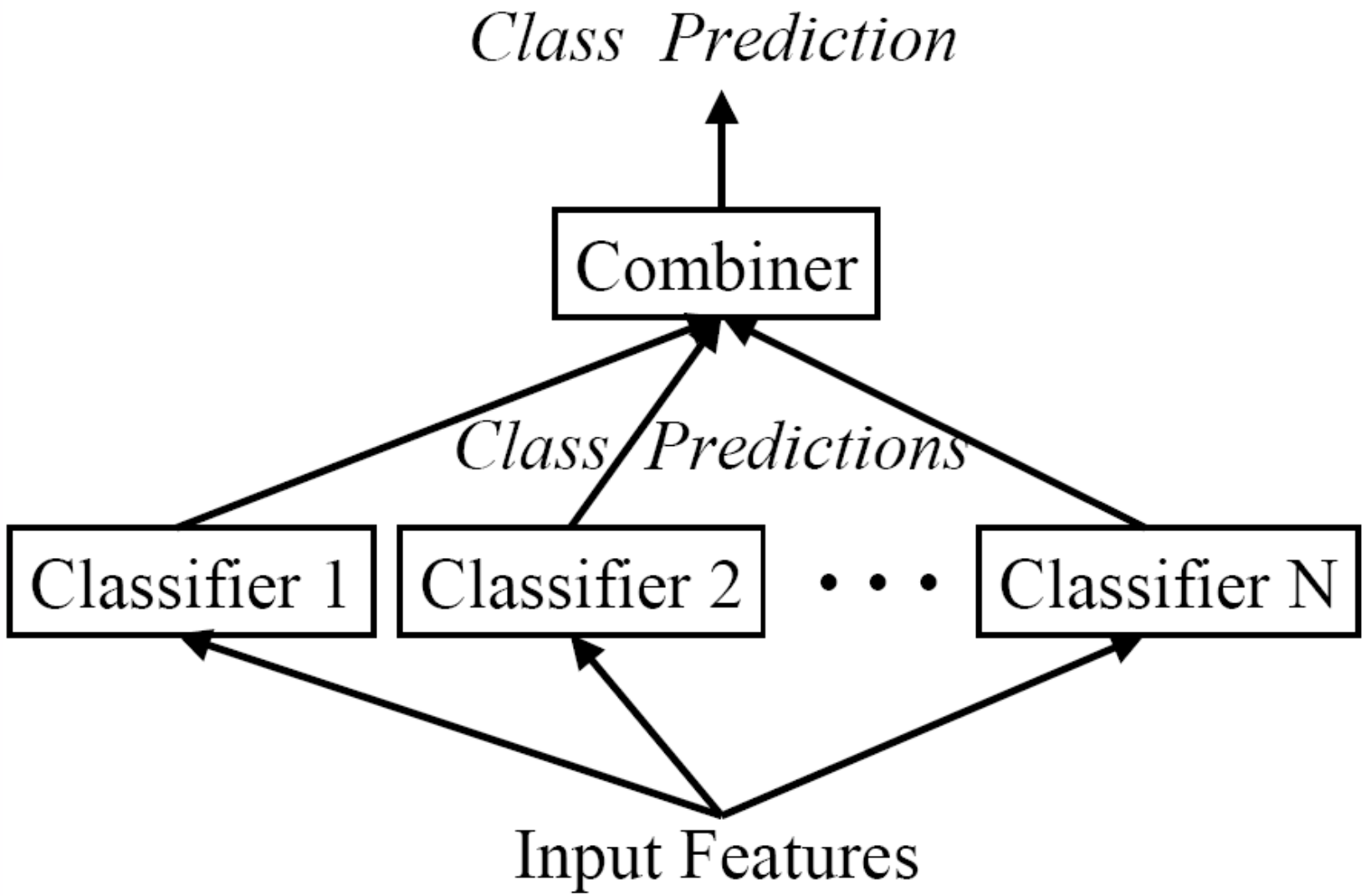
集成学习简介

- **定义**
  - **集成学习（Ensemble learning）方法通过组合多种学习算法来获得比单独使用任何一种算法更好的预测性能。.**
- **动机**
  - **提高单分类器的性能**

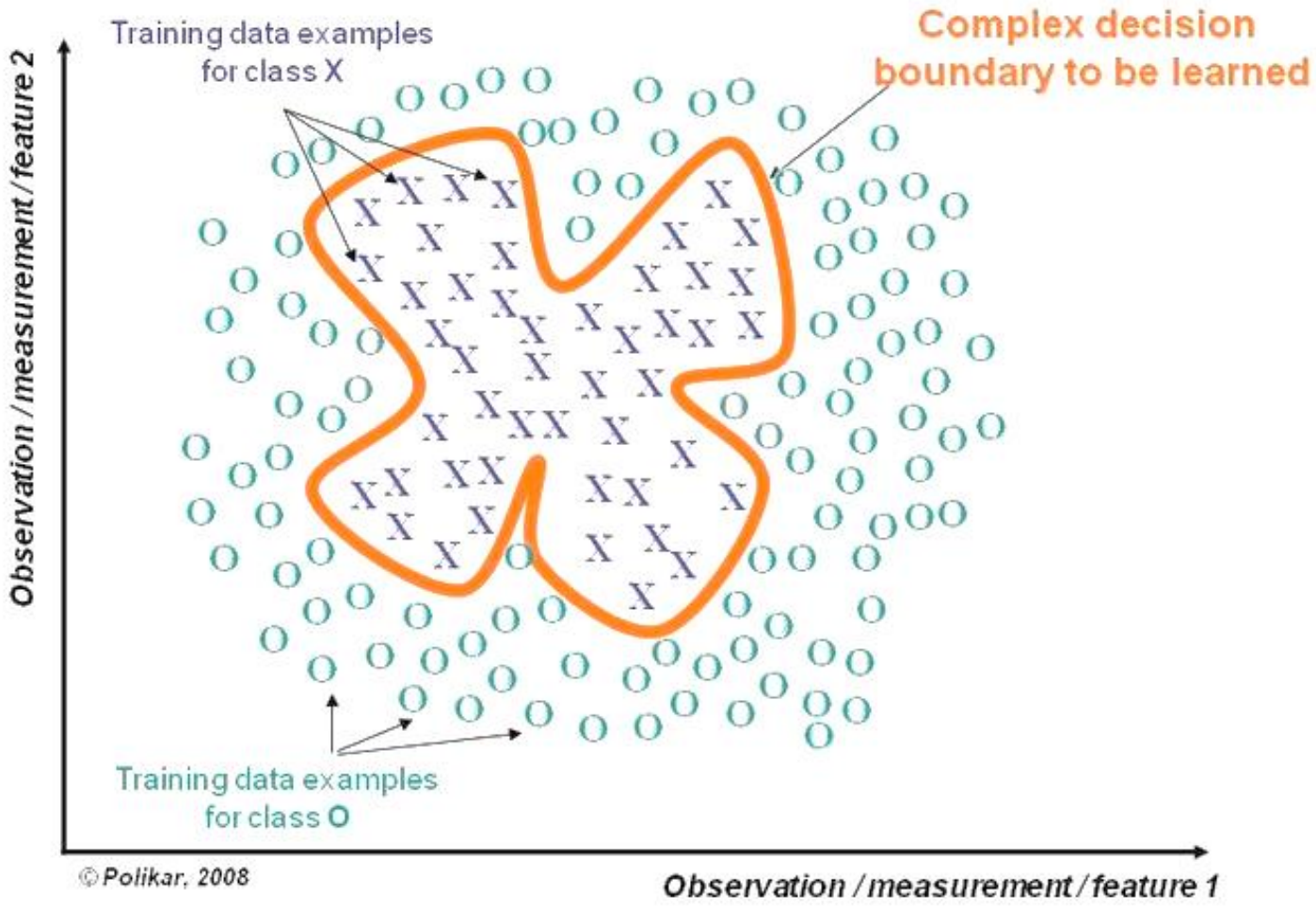
集成学习简介


Machine learning
Supervised learning
Semi-supervised learning
Unsupervised learning
Classification
Clustering
Single algorithms:
SVM,DT,NN
Ensemble algorithms
Boosting
Bagging
Stacking

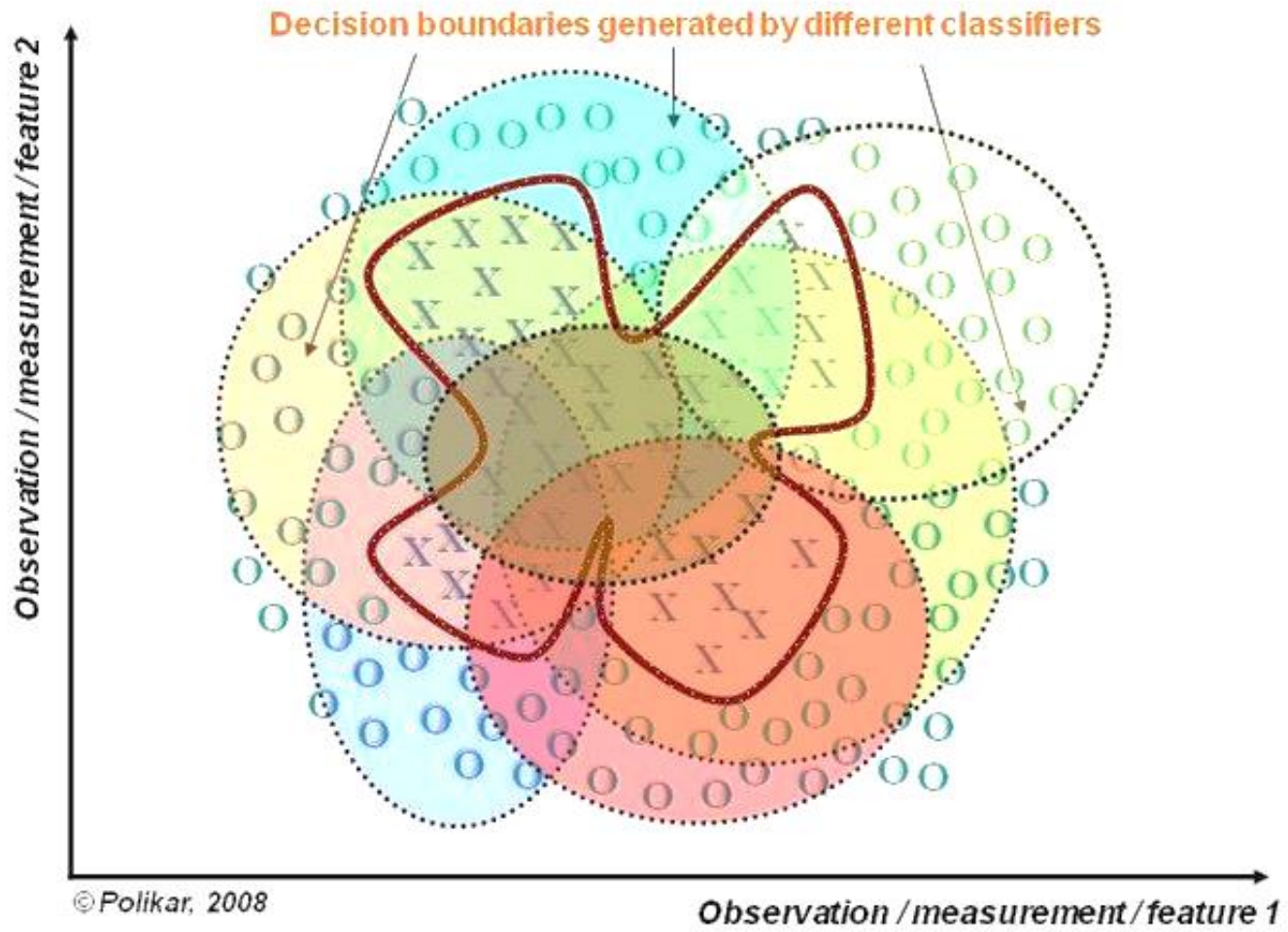
分类器组合


Class Prediction
Combiner
Class Predictions
Classifier 1
Classifier 2
• • •
Classifier N
Input Features

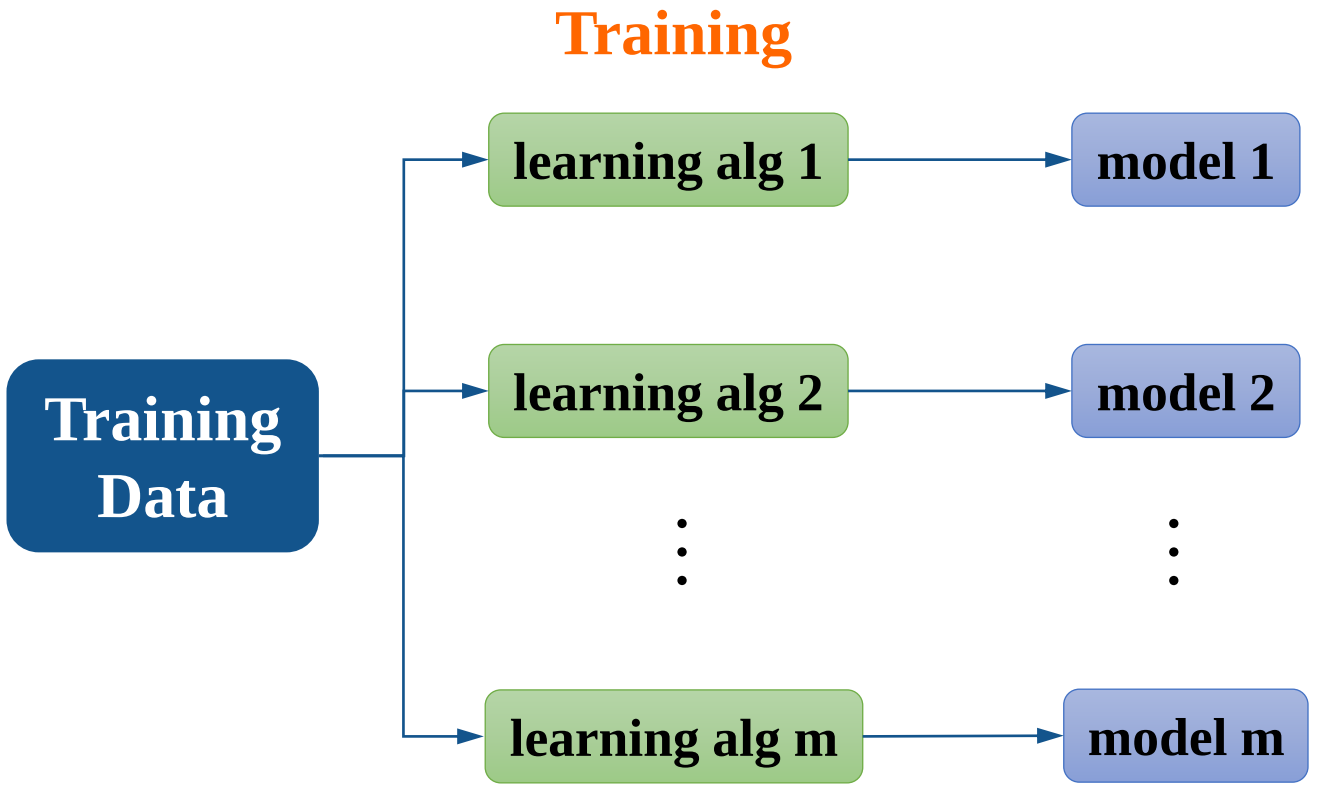
分而治之


Training data examples
for class X
Complex decision
boundary to be learned
Training data examples
for class O
Observation / measurement / feature 2
Observation / measurement / feature 1
© Polikar, 2008

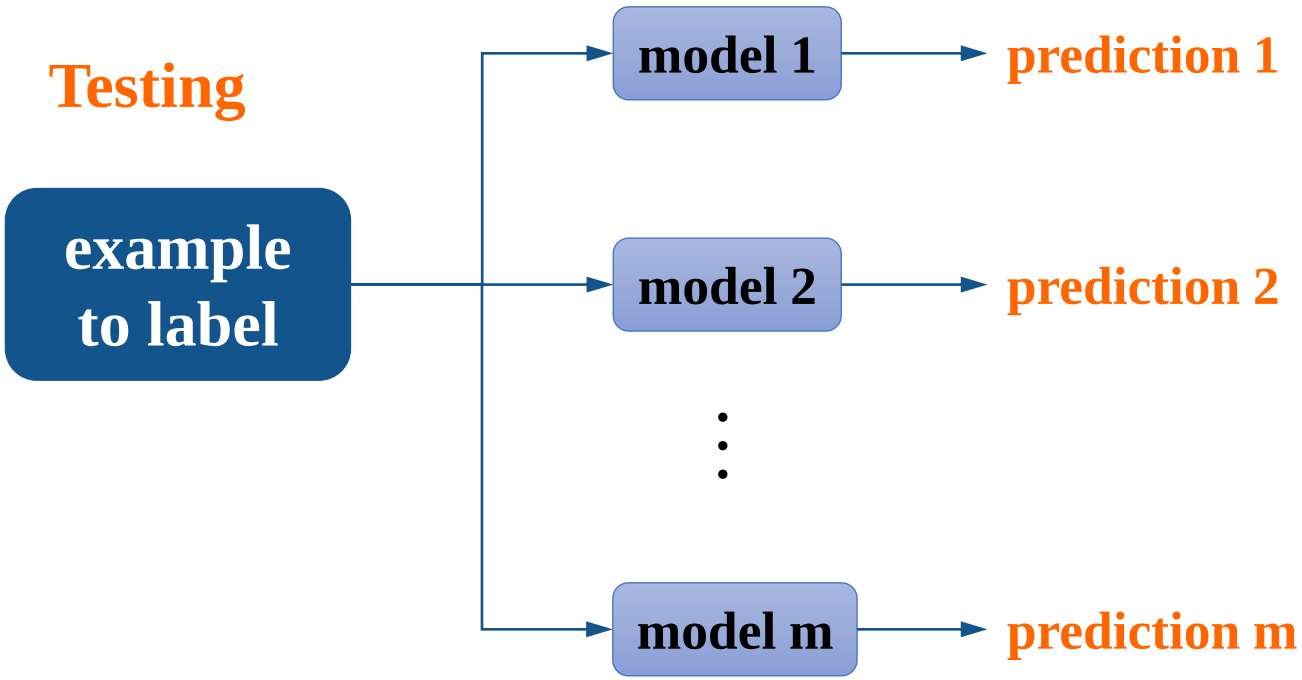
分而治之


Decision boundaries generated by different classifiers
Observation / measurement / feature 2
Observation / measurement / feature 1
© Polikar, 2008

训练阶段


Training
Training Data
learning alg 1
learning alg 2
learning alg m
model 1
model 2
model m

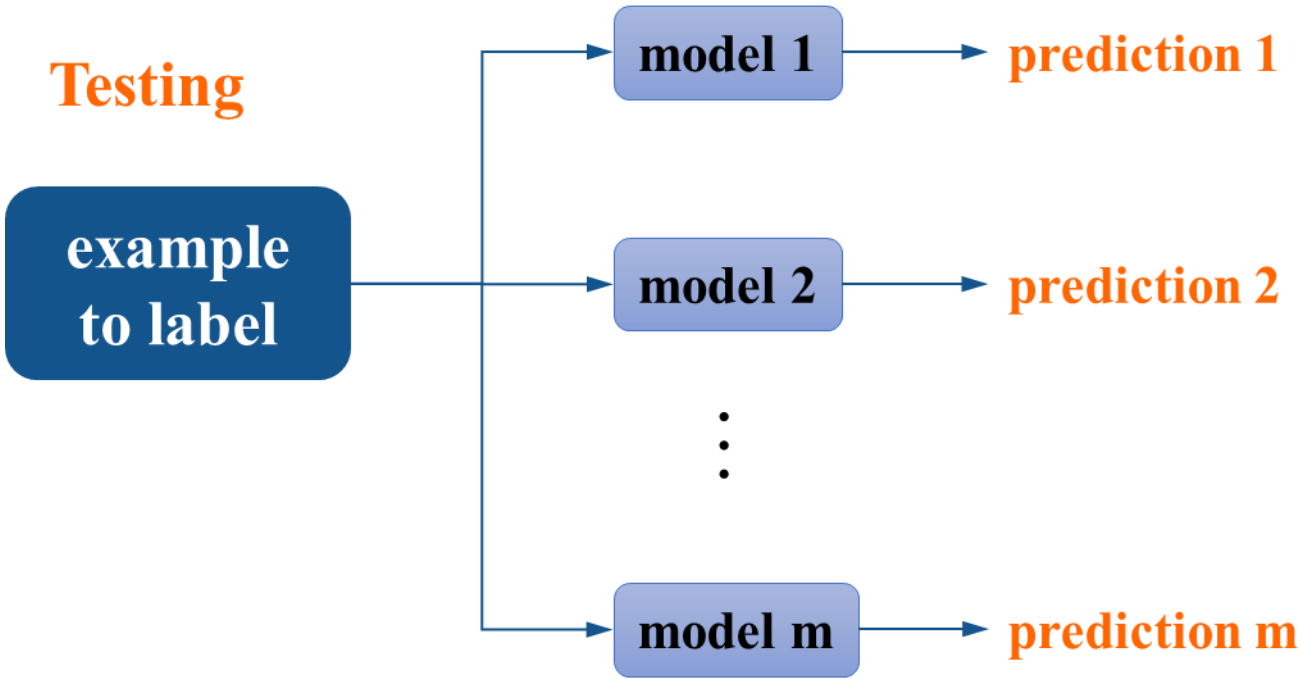
测试阶段



讨论：如何组合多个分类器呢?

如何组合多个分类器?

- 平均
- 投票
  - 多数投票：Bagging
  - 带权重的多数投票：Adaboost
- 学习组合器
  - Stacking
  - RegionBoost



集成学习优势示例

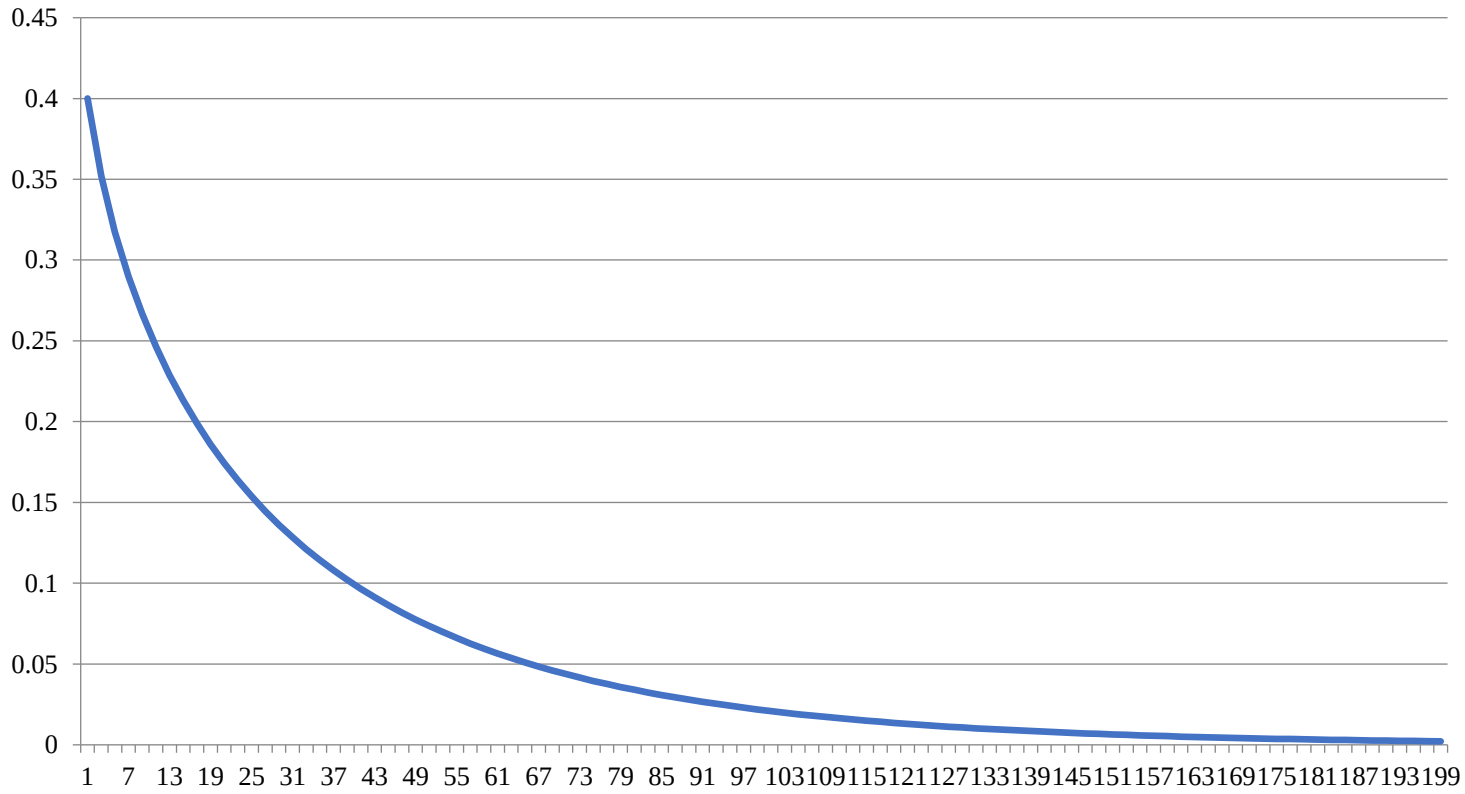
假如每个分类器的错误率为40%

model 1	model 2	model 3	prob
C	C	C	.6*.6*.6=0.216
C	C	I	.6*.6*.4=0.144
C	I	C	.6*.4*.6=0.144
C	I	I	.6*.4*.4=0.096
I	C	C	.4*.6*.6=0.144
I	C	I	.4*.6*.4=0.096
I	I	C	.4*.4*.6=0.096
I	I	I	.4*.4*.4=0.064

0.096 +0.096 +0.096 +0.064 = 35% error!

集成学习优势示例

$$p(error) = \sum_{i=(m+1)/2}^{m} \binom{m}{i} r^i (1-r)^{m-i}$$

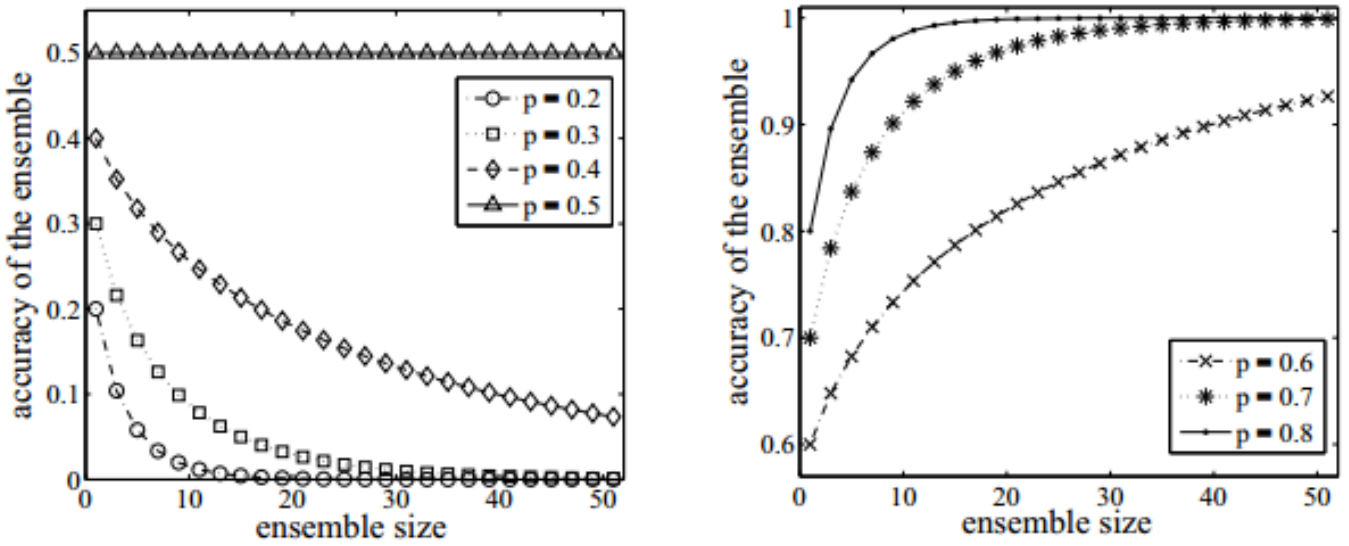


r = 0.4

集成学习中单分类器的条件

- 通过集成学习提高分类器的整体泛化能力是有条件的：
  - 分类器之间应该具有差异性
  - 分类器的精度，每个个体分类器的分类精度必须大于0.5

$$p(error) = \sum_{i=(m+1)/2}^{m} \binom{m}{i} r^i (1-r)^{m-i}$$



南京大学
NANJING UNIVERSITY

目录

集成学习算法

1. Bagging
2. Boosting
3. Stacking

Bagging：有放回的重采样

Bootstrap Aggregating

Step 1

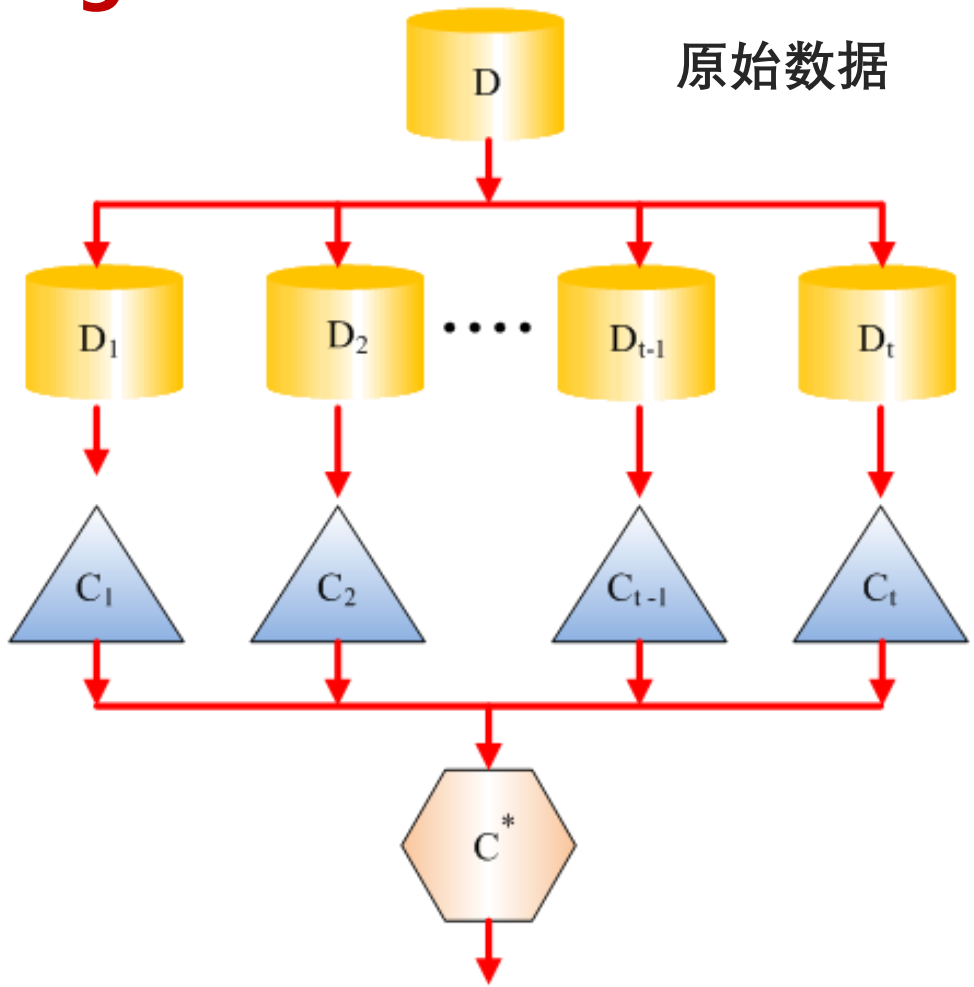
选取多个样本集合

Step 2

建立多个分类器（决策树）

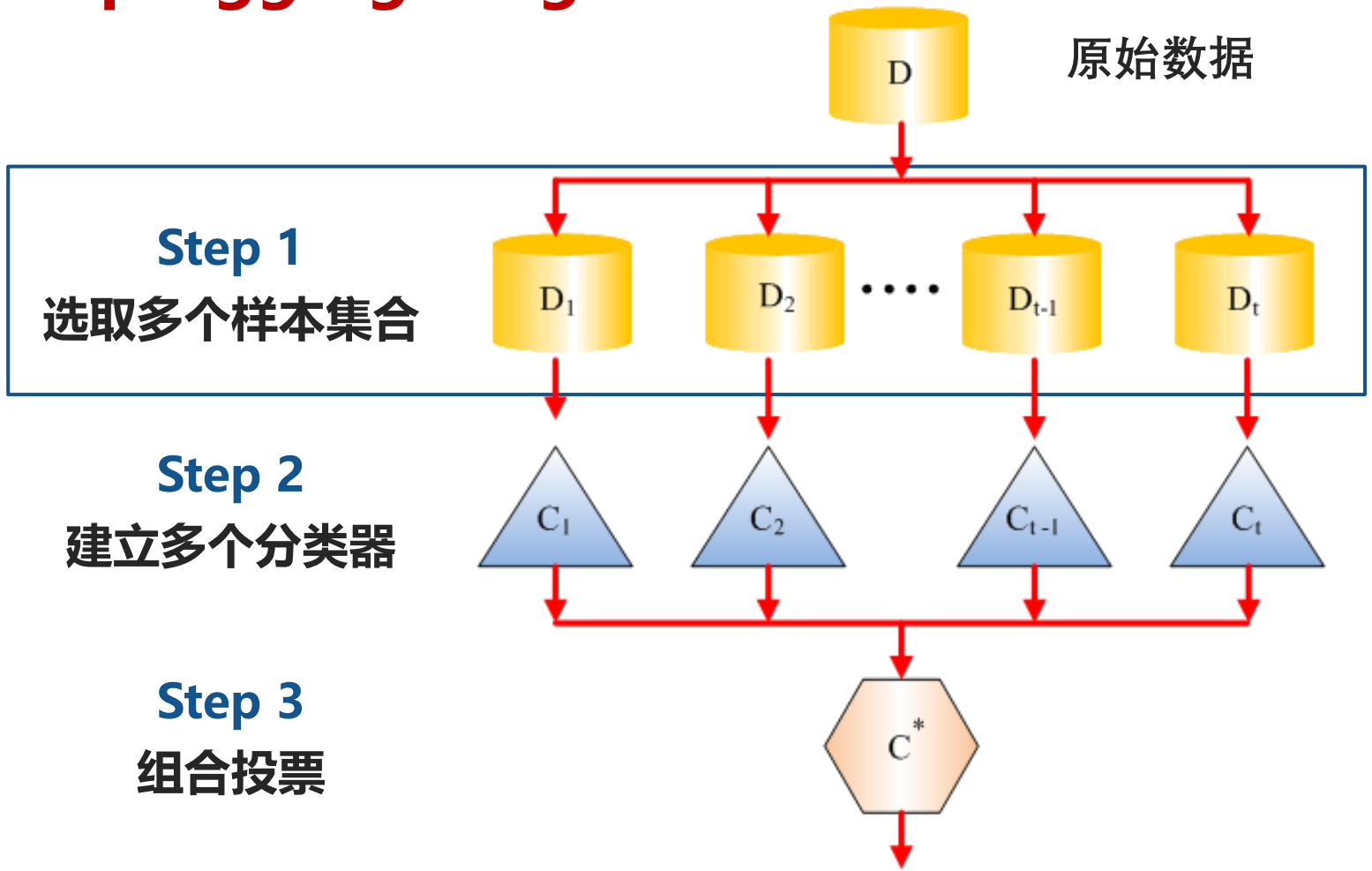
Step 3

组合投票



Bagging：有放回的重采样

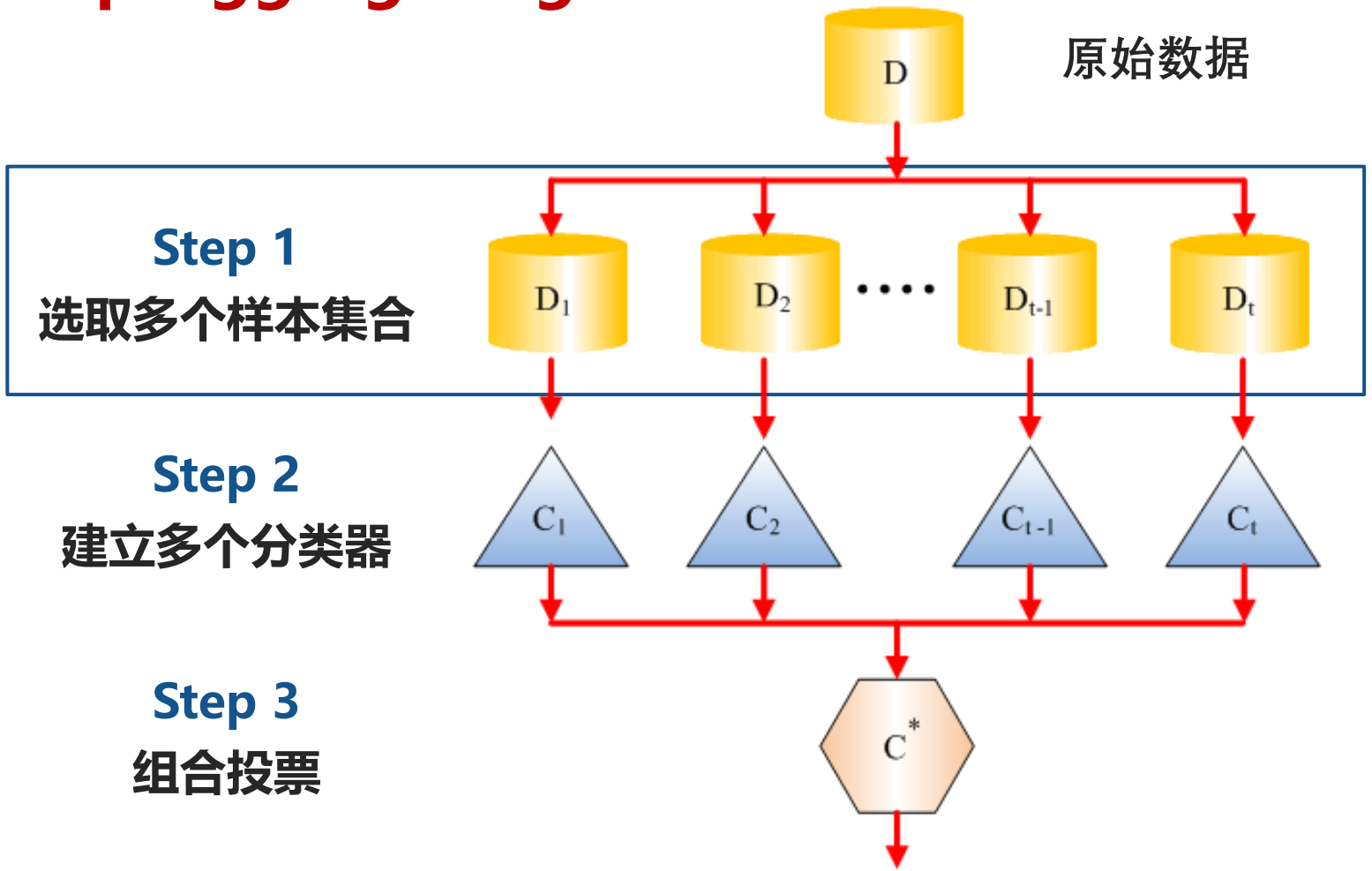
Bootstrap Aggregating



讨论：样本集合如何选取?

Bagging：有放回的重采样

Bootstrap Aggregating

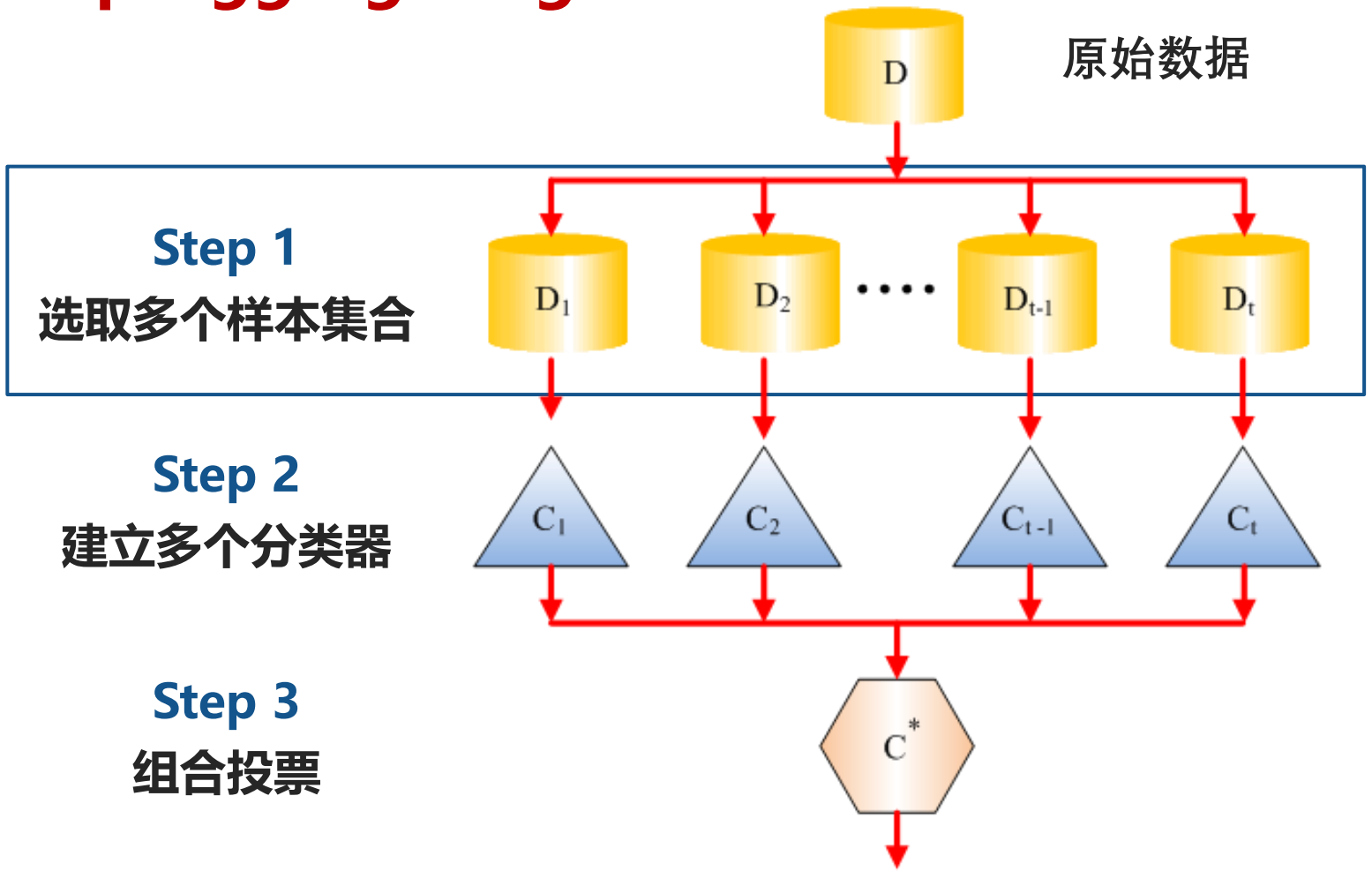


讨论：样本集合如何选取?

如果训练集规模足够大，平均分成t份即可

Bagging：有放回的重采样

Bootstrap Aggregating

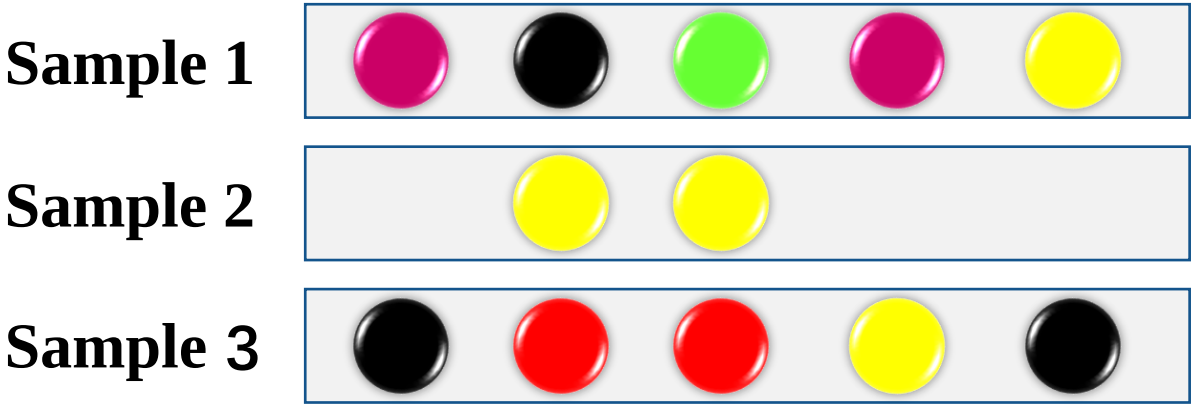


讨论：样本集合如何选取?

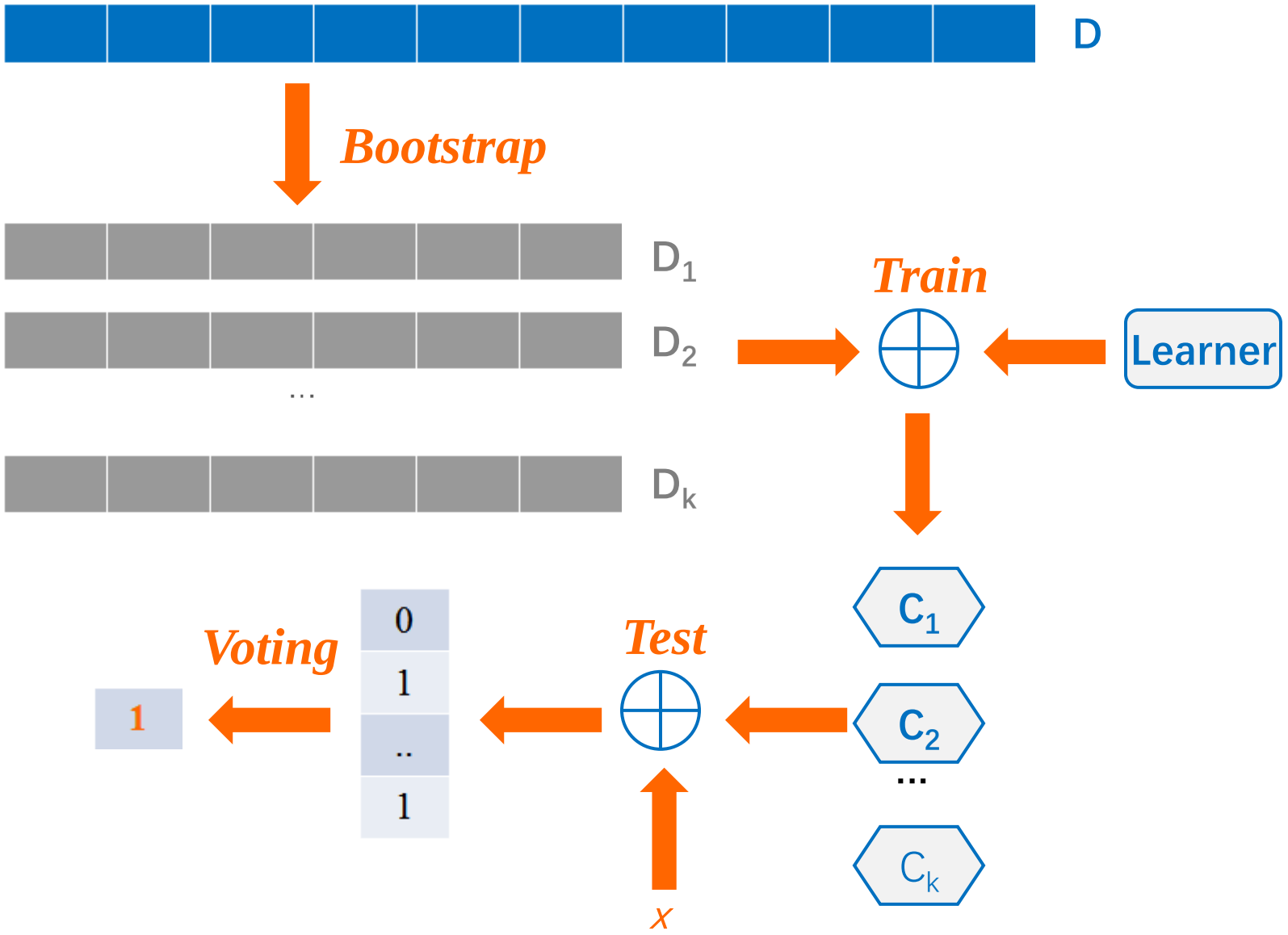
如果训练集规模足够大，平均分成t份即可

如果训练集规模不大，怎么办?

Bagging：有放回的重采样

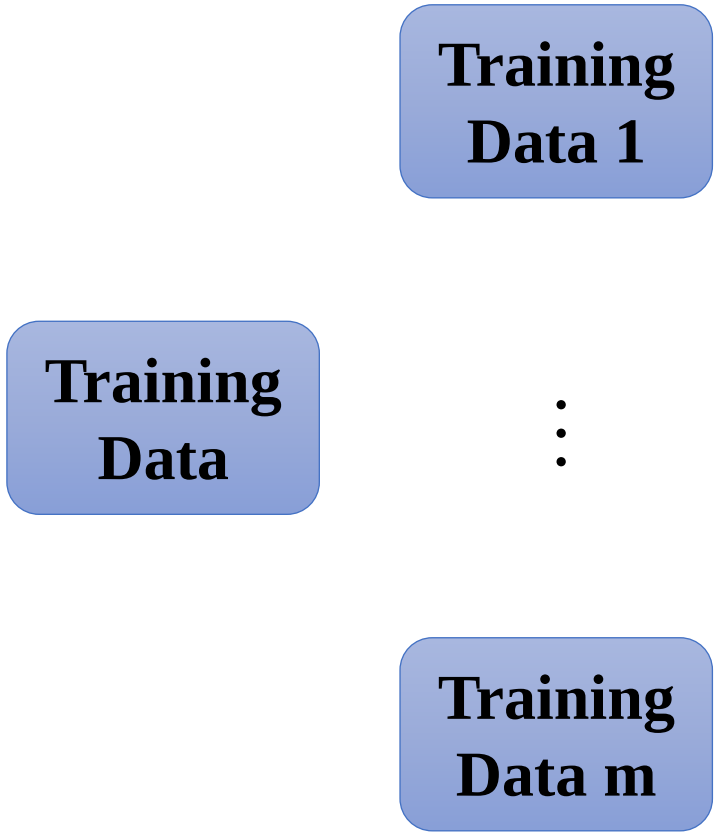


Bagging-Bootstrap采样法



Bagging-Bootstrap采样法优点

原始数据中，仅仅约63%的样本被采样到



Bagging-Bootstrap采样法优点

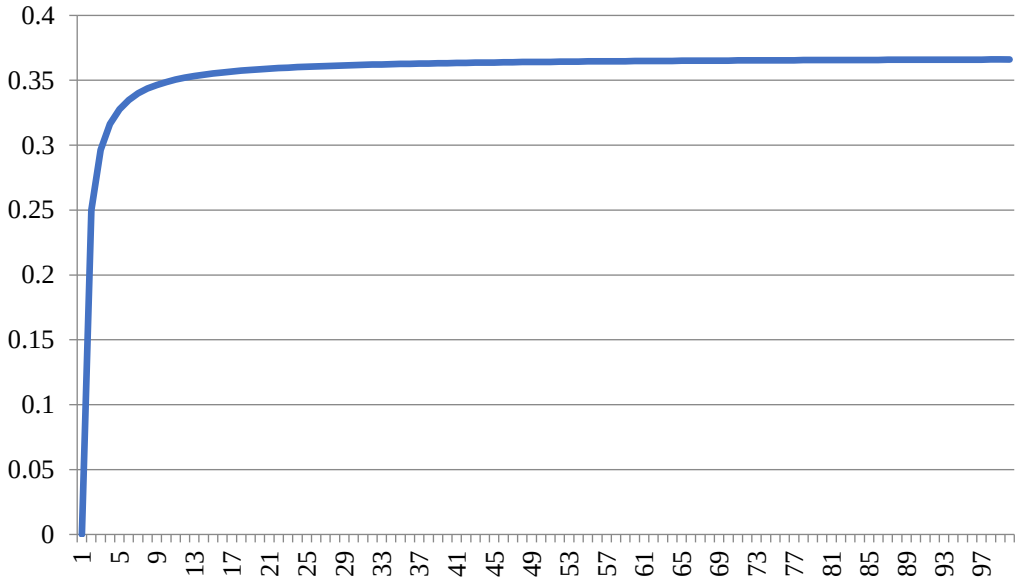
原始数据中，仅仅约63%的样本被采样到

Training Data 1

Training Data

Training Data m

$$P_{not\_choose} = \left(1 - \frac{1}{n}\right)^{n} \rightarrow \frac{1}{e}$$



Bagging-Bootstrap采样法优点

Training Data 1

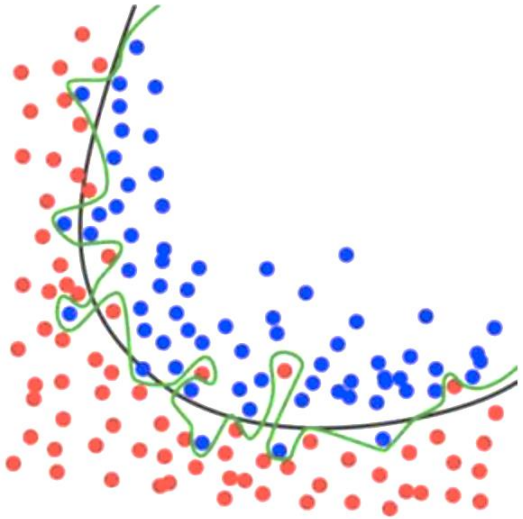
Training Data

Training Data m

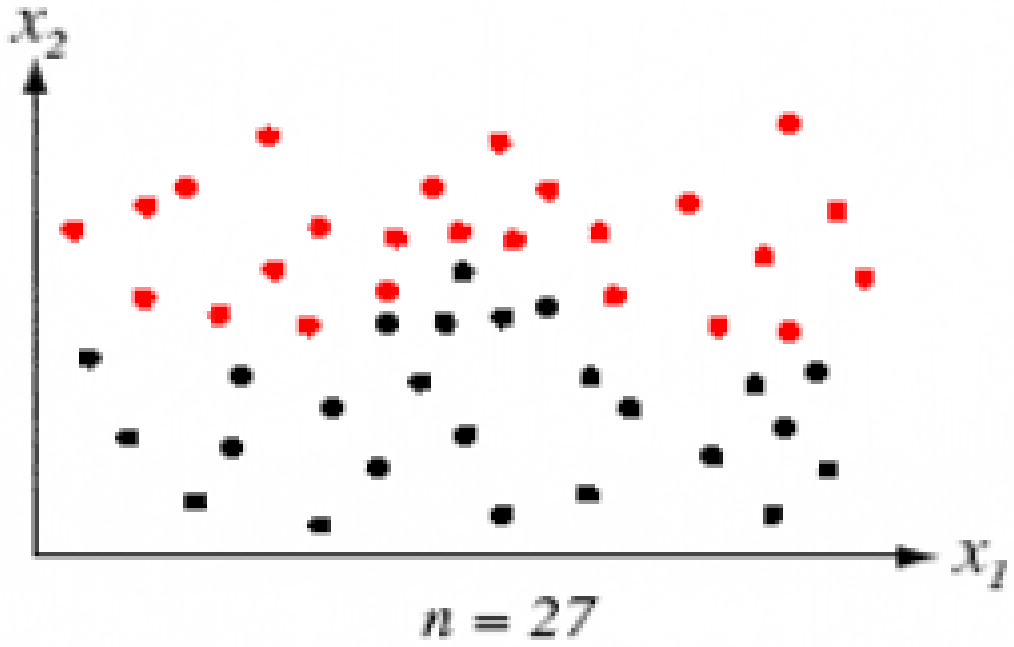
原始数据中，仅仅约63%的样本被采样到

也就是说样本训练集中，约37%的训练集将被忽略掉

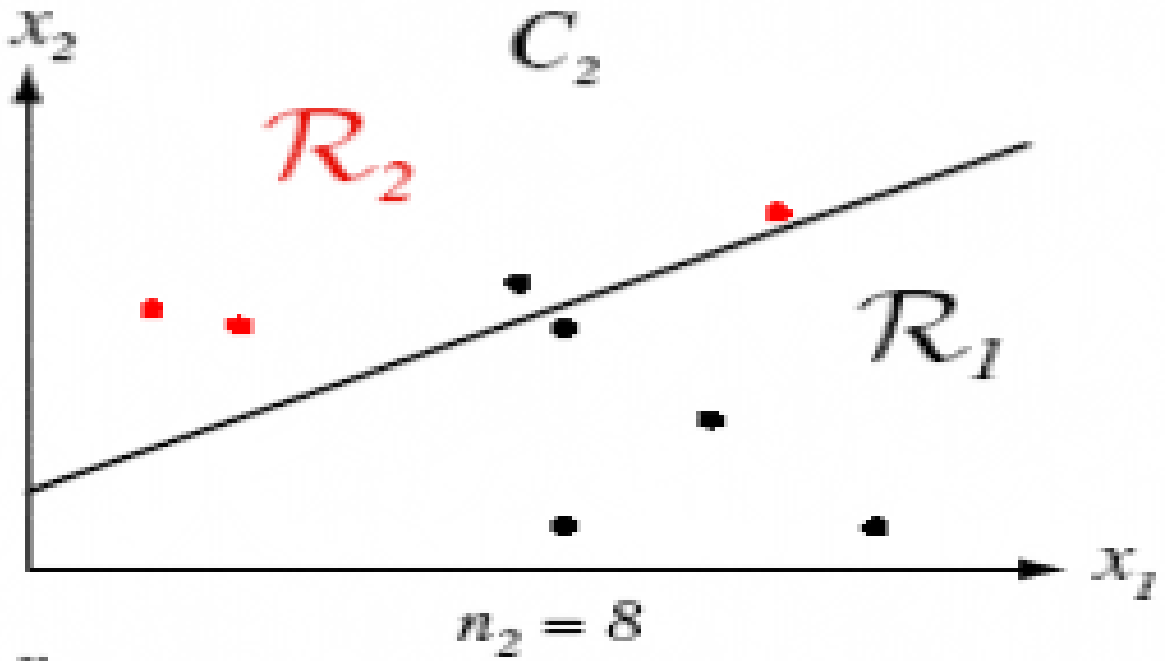
- 减少过拟合
- 降低方差
- 降低噪音数据的影响



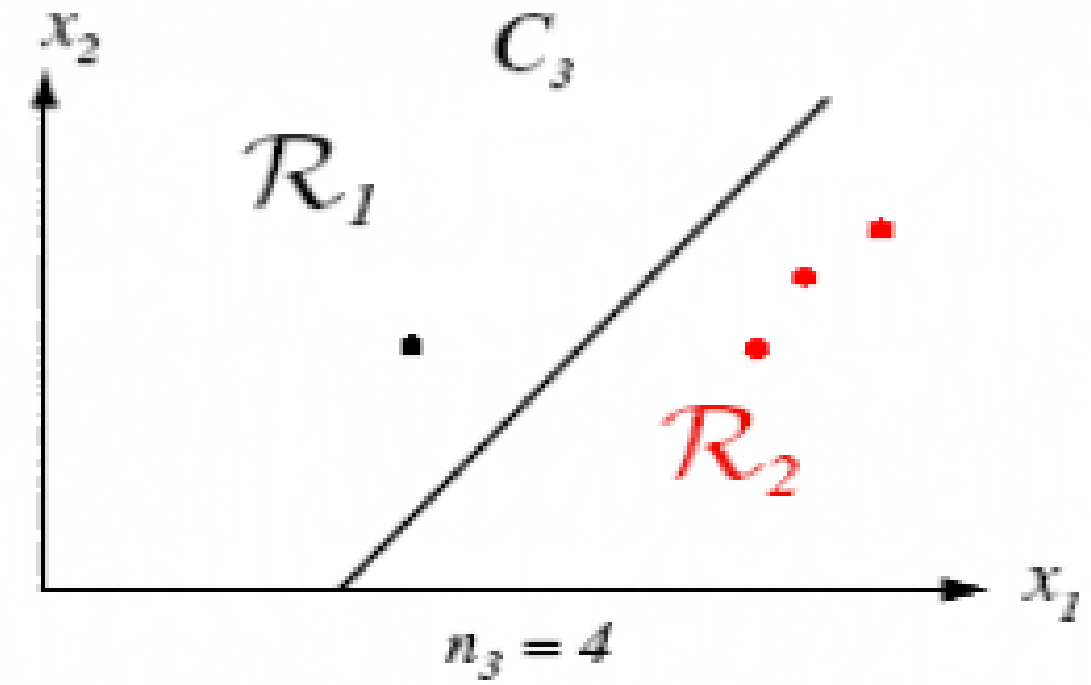
Bagging示例



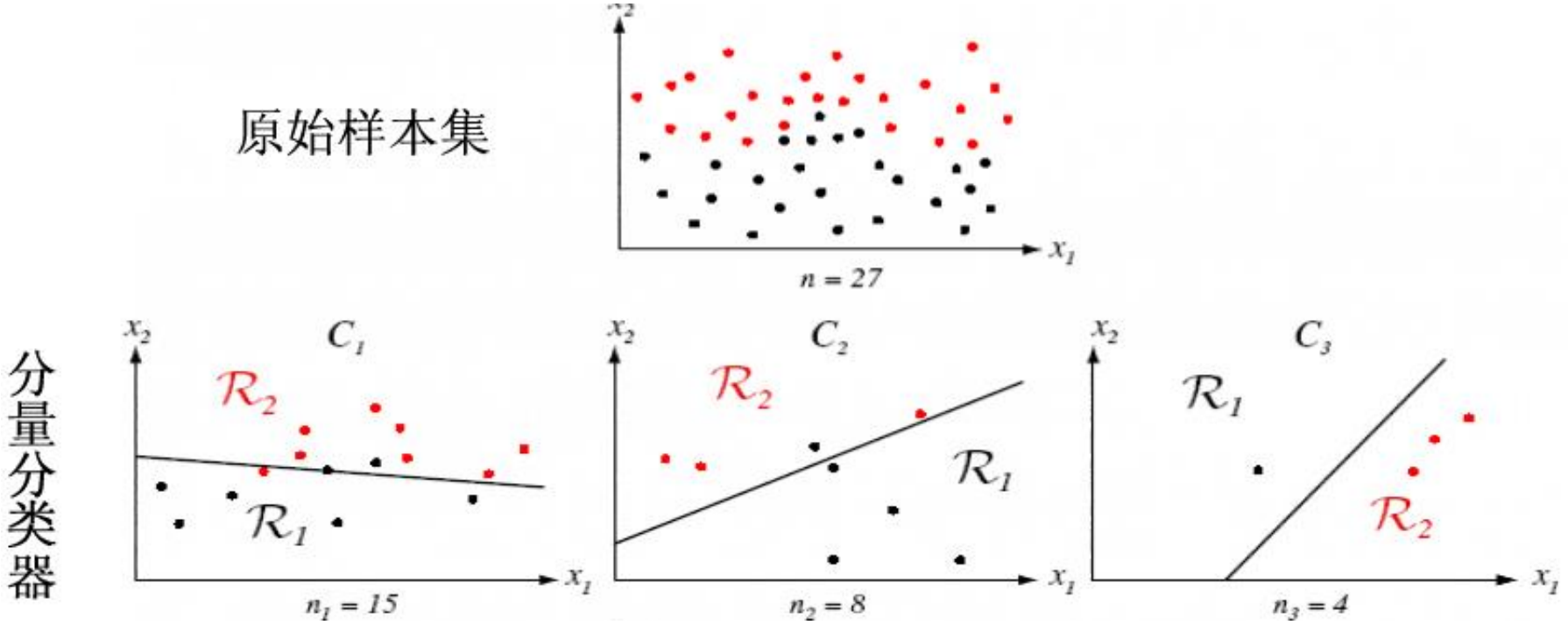
Bagging示例



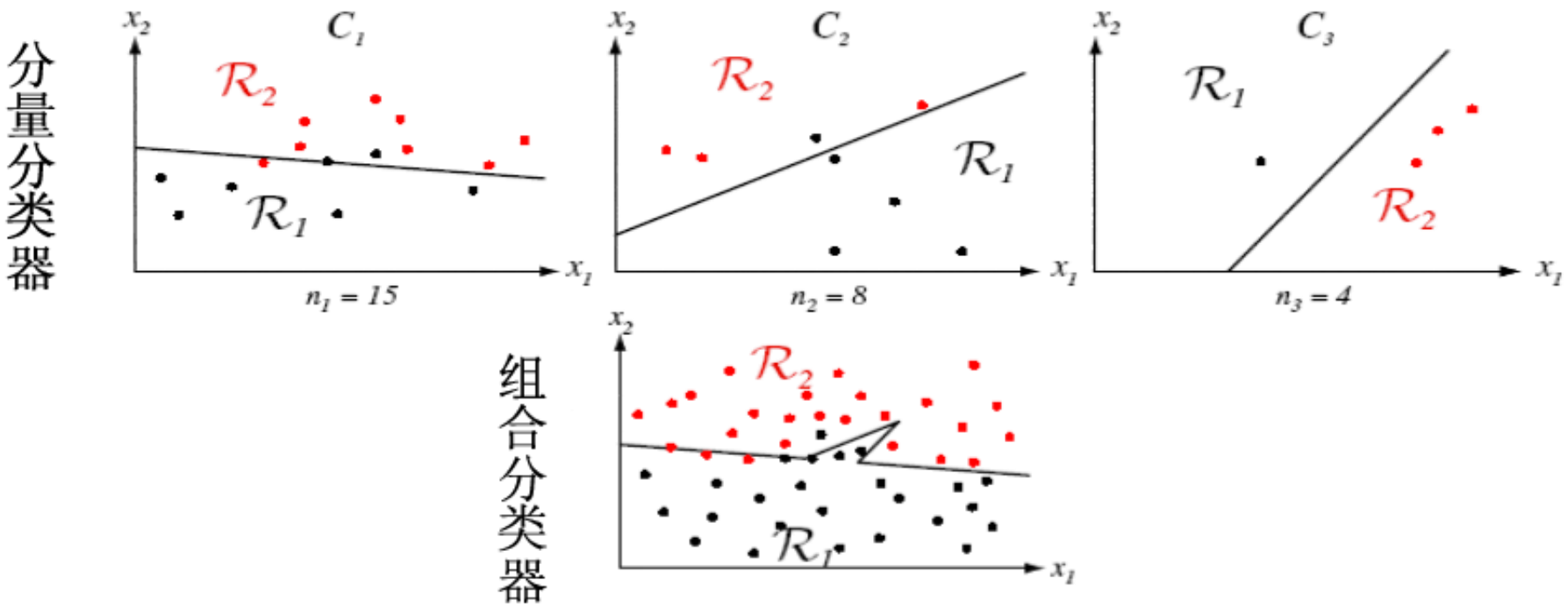
Bagging示例



Bagging示例


原始样本集
x_2
x_1
$n = 27$
分量分类器
C_1
$\mathcal{R}_2$
$\mathcal{R}_1$
$n_1 = 15$
C_2
$\mathcal{R}_2$
$\mathcal{R}_1$
$n_2 = 8$
C_3
$\mathcal{R}_1$
$\mathcal{R}_2$
$n_3 = 4$

Bagging示例

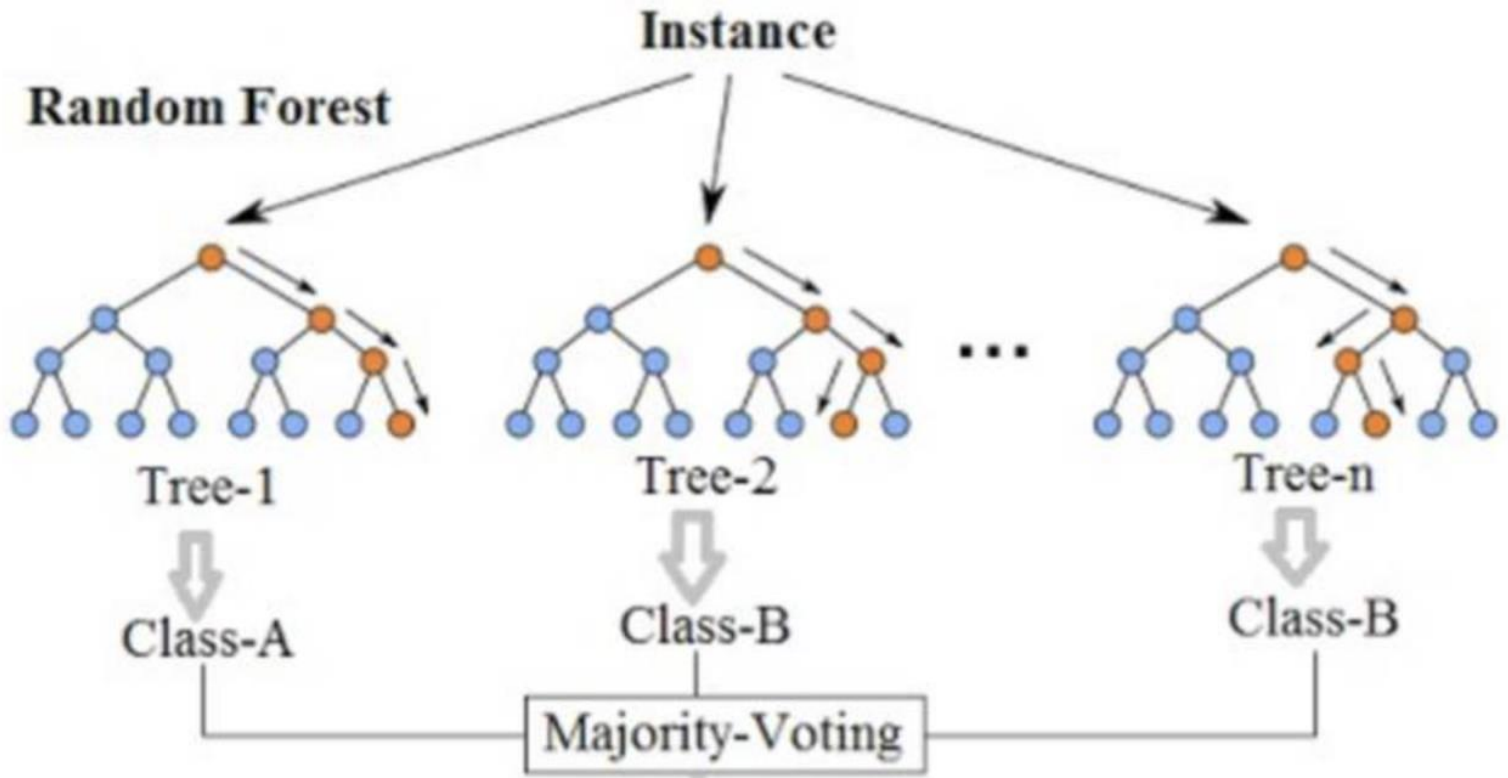

分量分类器
x_2
C_1
$\mathcal{R}_2$
$\mathcal{R}_1$
x_1
$n_1 = 15$
x_2
C_2
$\mathcal{R}_2$
$\mathcal{R}_1$
x_1
$n_2 = 8$
x_2
C_3
$\mathcal{R}_1$
$\mathcal{R}_2$
x_1
$n_3 = 4$
组合分类器
x_2
$\mathcal{R}_2$
$\mathcal{R}_1$
x_1

Bagging：随机森林

- 基本思想
  - 随机森林就是通过集成学习的思想将多棵树集成的一种算法，它的基本单元是决策树（如CART）。
  - 随机森林的名称中有两个关键词，一个是“随机”，一个就是“森林”。“森林”很好理解，一棵叫做树，那么成百上千棵就可以叫做森林了，其实这也是随机森林的主要思想--集成思想的体现。“随机”的包括随机选取训练样本集和随机选取分裂属性集。

Bagging：随机森林

- 基本思想
  - **行采样：** 随机抽样训练集
  - **列采样：** 随机选择节点上的一部分特征，其中选择一个最优的特征来做决策树的左右子树划分



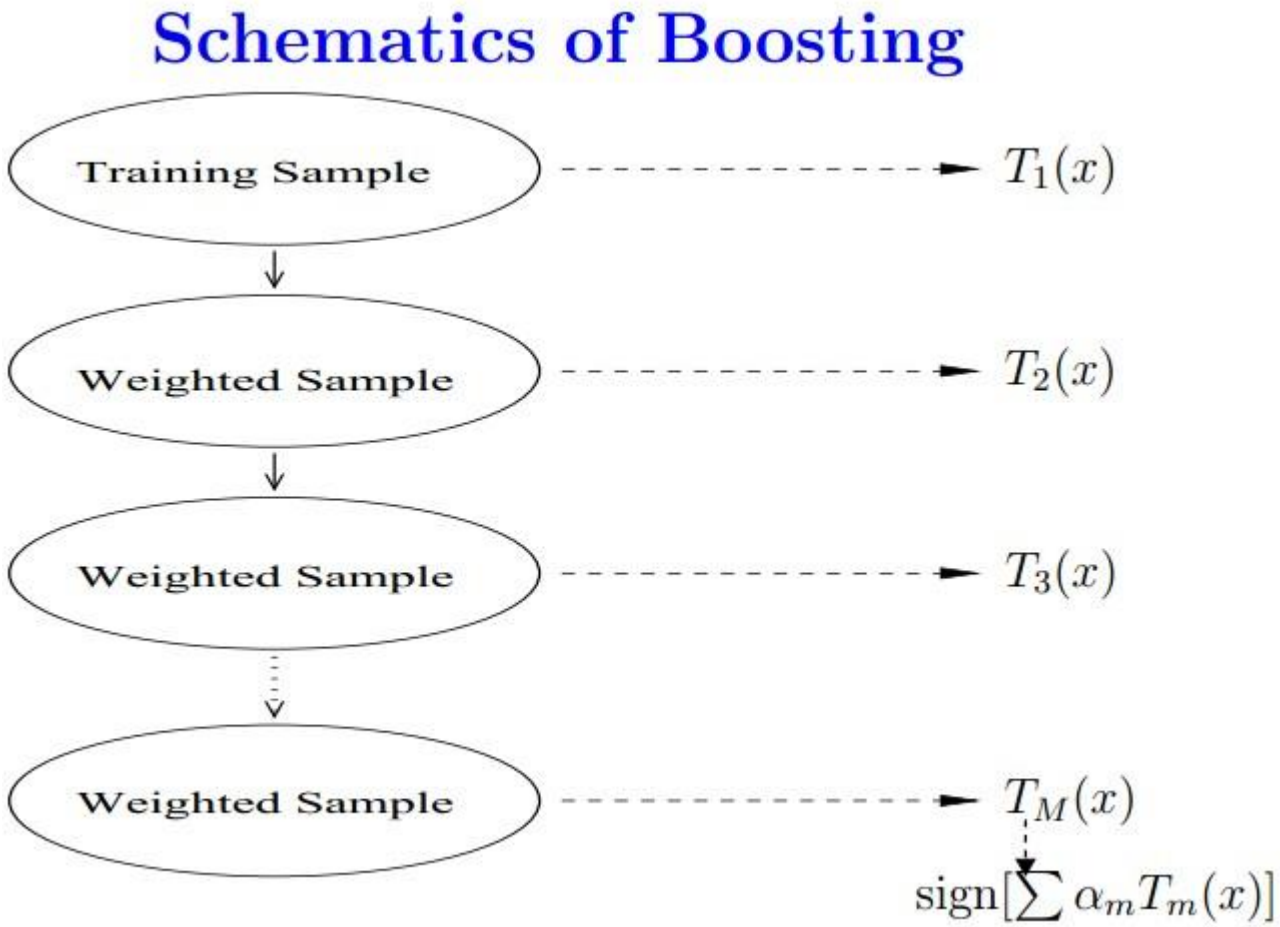
Bagging：随机森林

- **优点：**
  - 两个随机性的引入，使得随机森林不容易陷入过拟合；
  - 两个随机性的引入，使得随机森林具有很好的抗噪声能力；
  - 对数据集的适应能力强：既能处理离散型数据，也能处理连续型数据，数据集无需规范化且能够有效地运行在大数据集上；
  - 能够处理具有高维特征的输入样本，而且不需要降维；
  - 对于缺省值问题也能够获得很好得结果。

集成学习算法

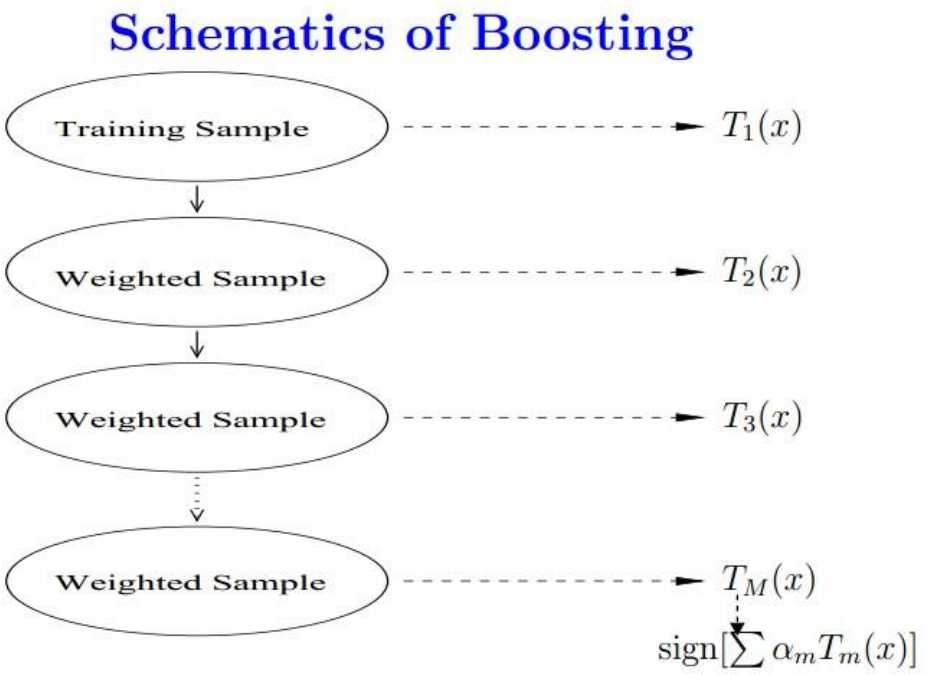
1. Bagging
2. Boosting
3. Stacking

Boosting提出


Schematics of Boosting
Training Sample
Weighted Sample
Weighted Sample
Weighted Sample
$T_1(x)$
$T_2(x)$
$T_3(x)$
$T_M(x)$
$\text{sign}[\sum \alpha_m T_m(x)]$

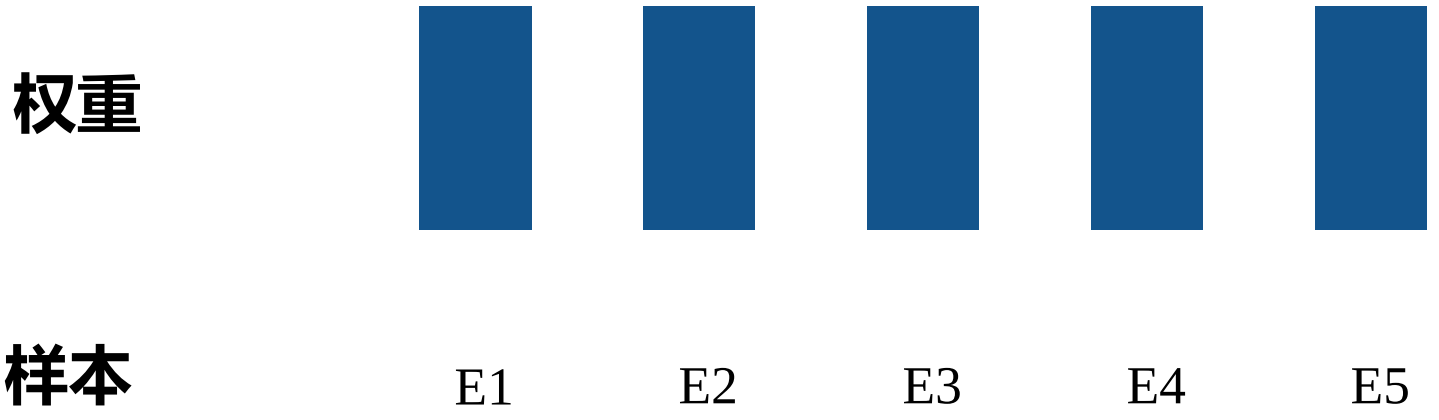
Boosting提出

- 提升方法是一个迭代的过程
  - 通过改变样本分布，使得分类器聚焦在那些很难分的样本上，对那些容易错分的数据加强学习，增加错分数据的权重
  - 这样错分的数据再下一轮的迭代就有更大的作用（对错分数据进行惩罚）



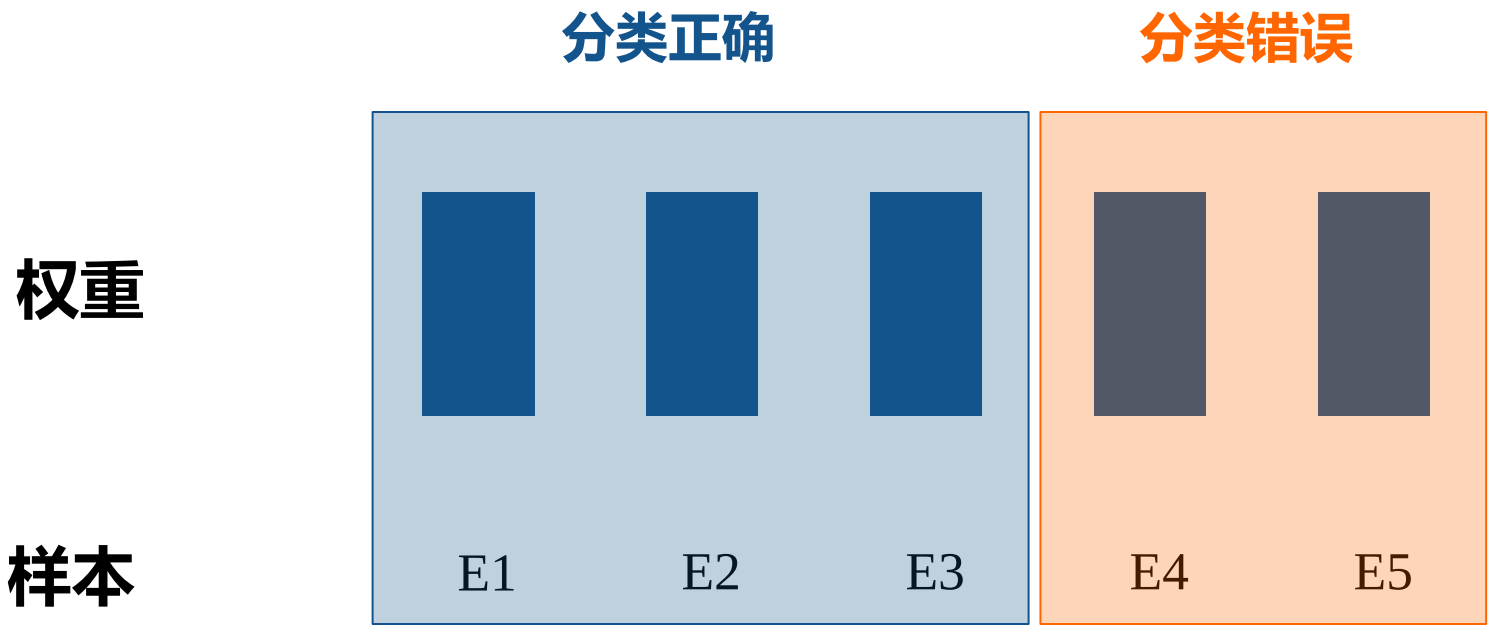
Boosting提出 —— 示例1

初始化：所有样本等权重



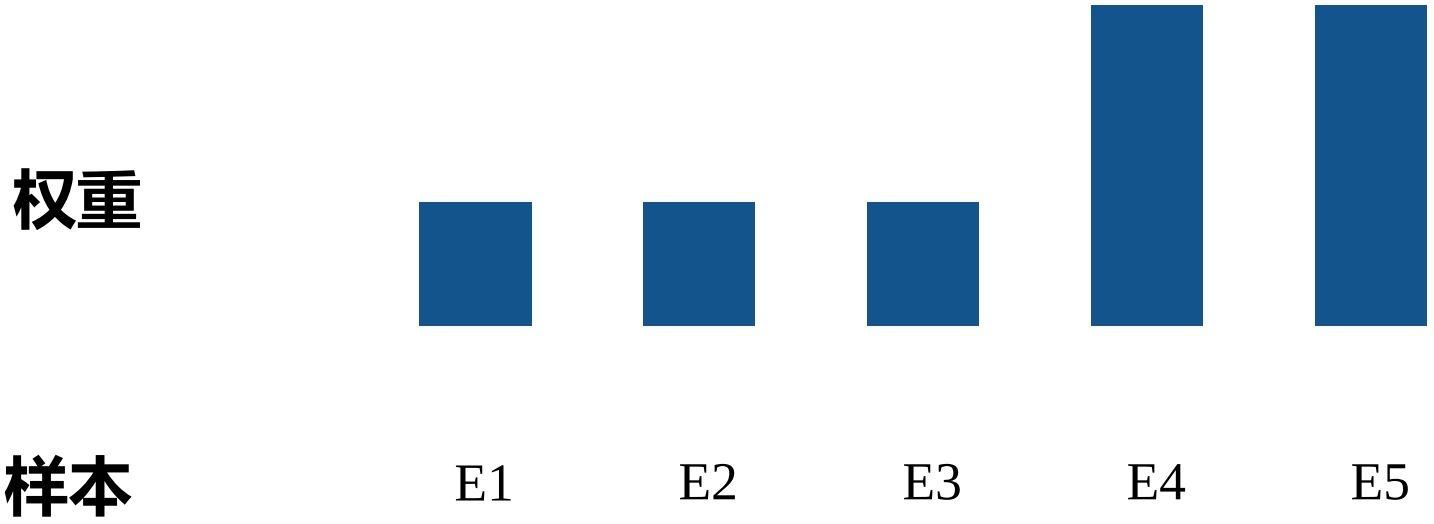
Learn a weak classifier

Boosting提出 —— 示例1



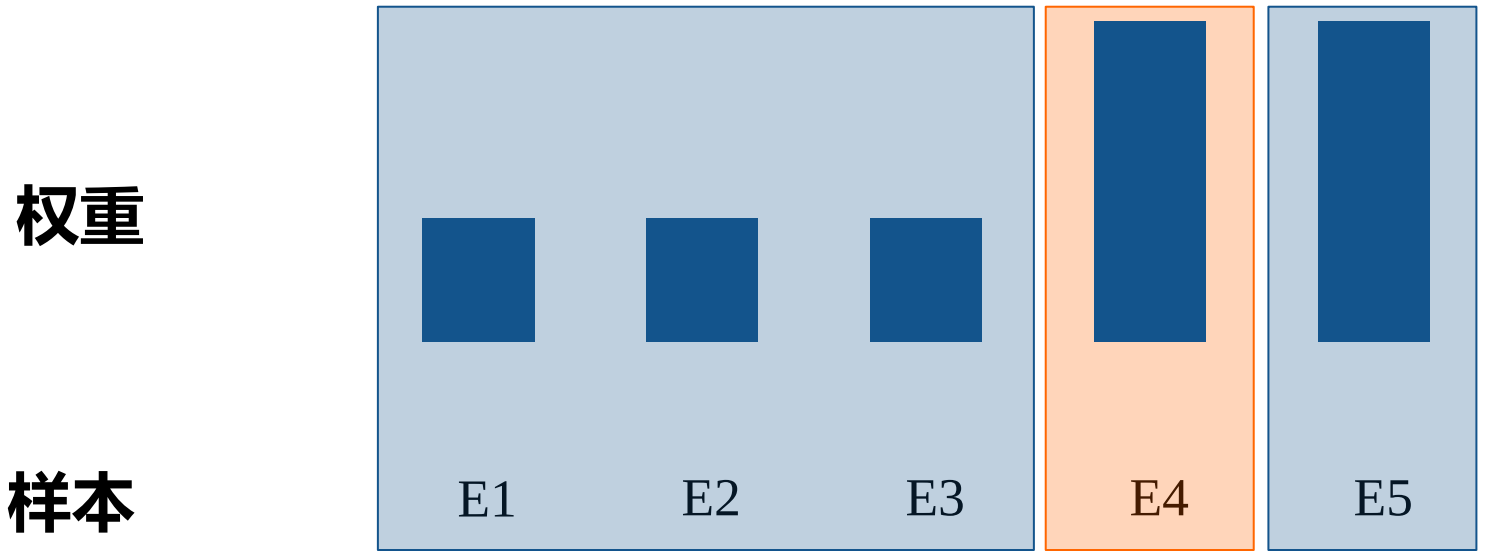
学习弱分类器

Boosting提出 —— 示例1



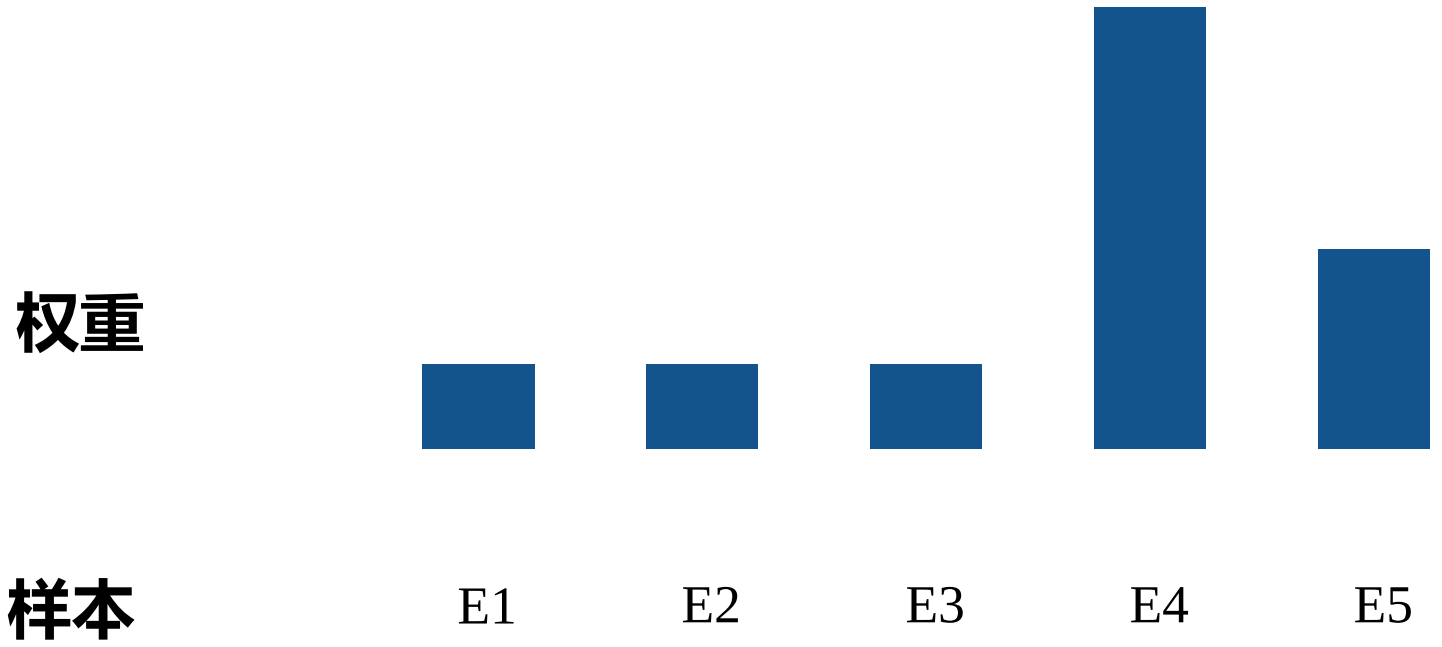
提升被错误分类样本权重

Boosting提出 —— 示例1



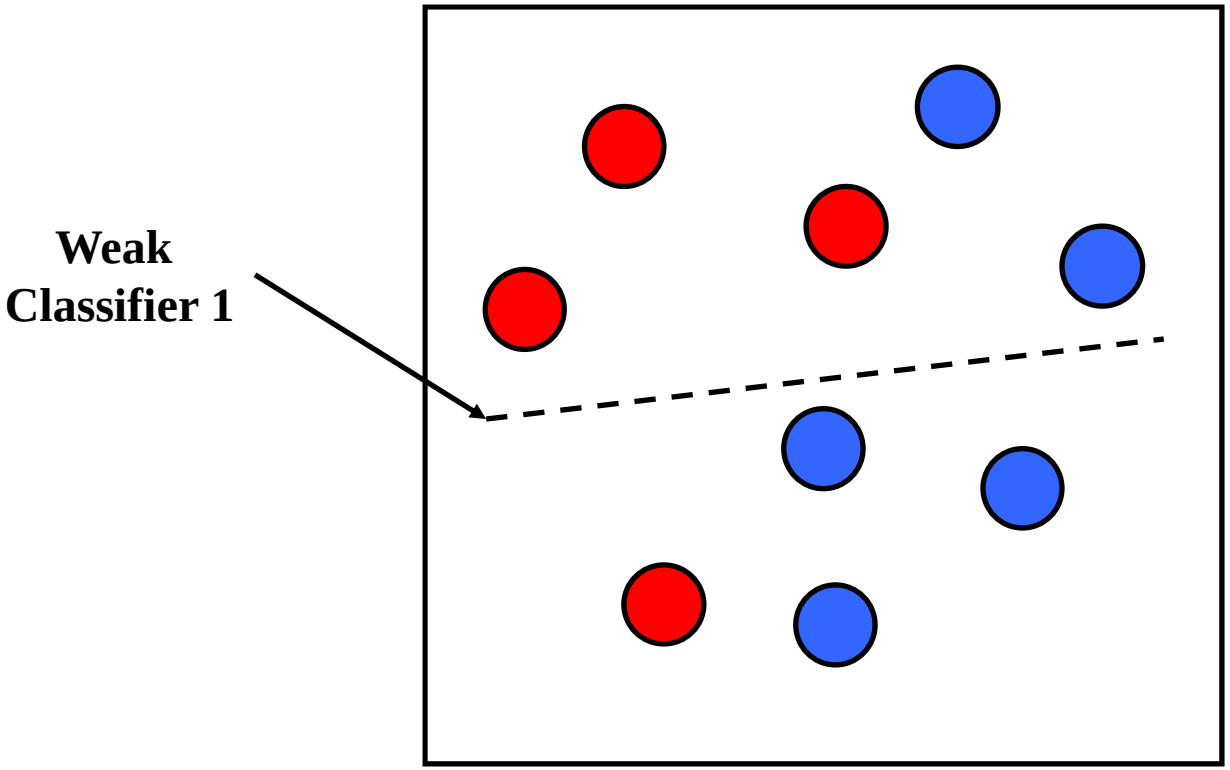
学习新的弱分类器

Boosting提出 —— 示例1

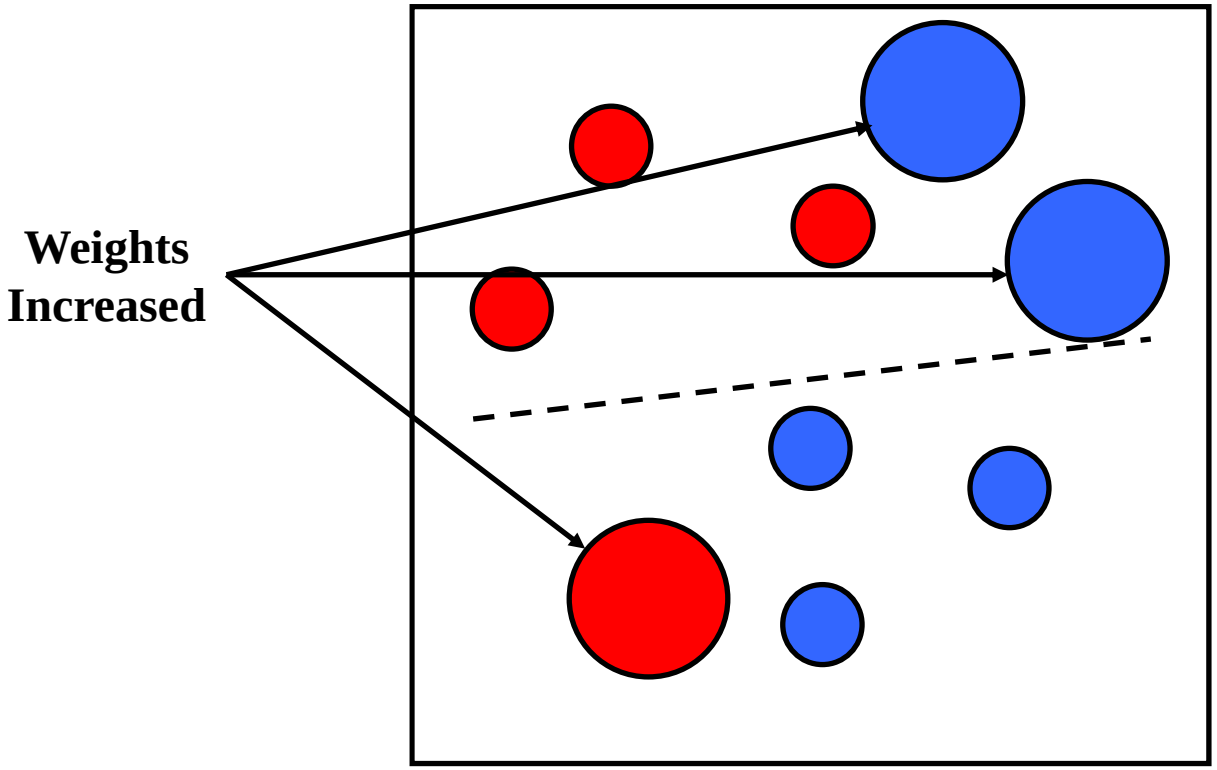


提升被错误分类样本权重，学习新的弱分类器

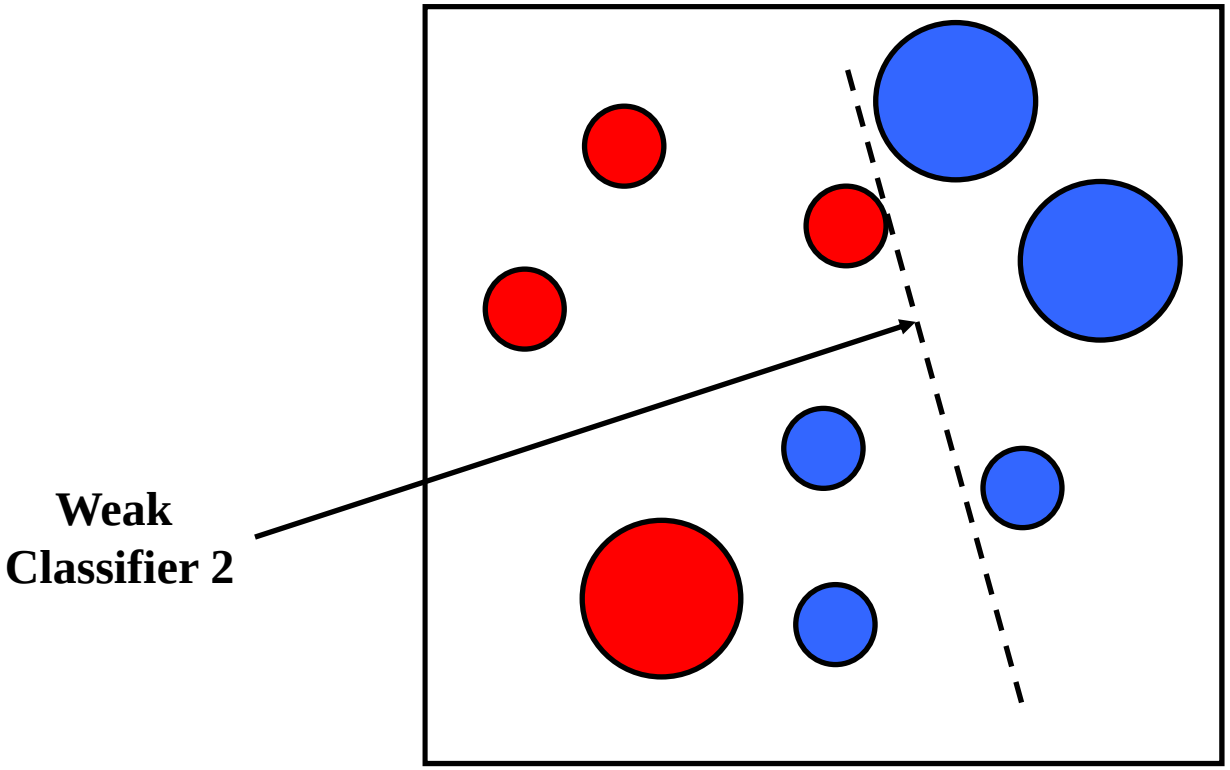
Boosting提出 —— 示例2



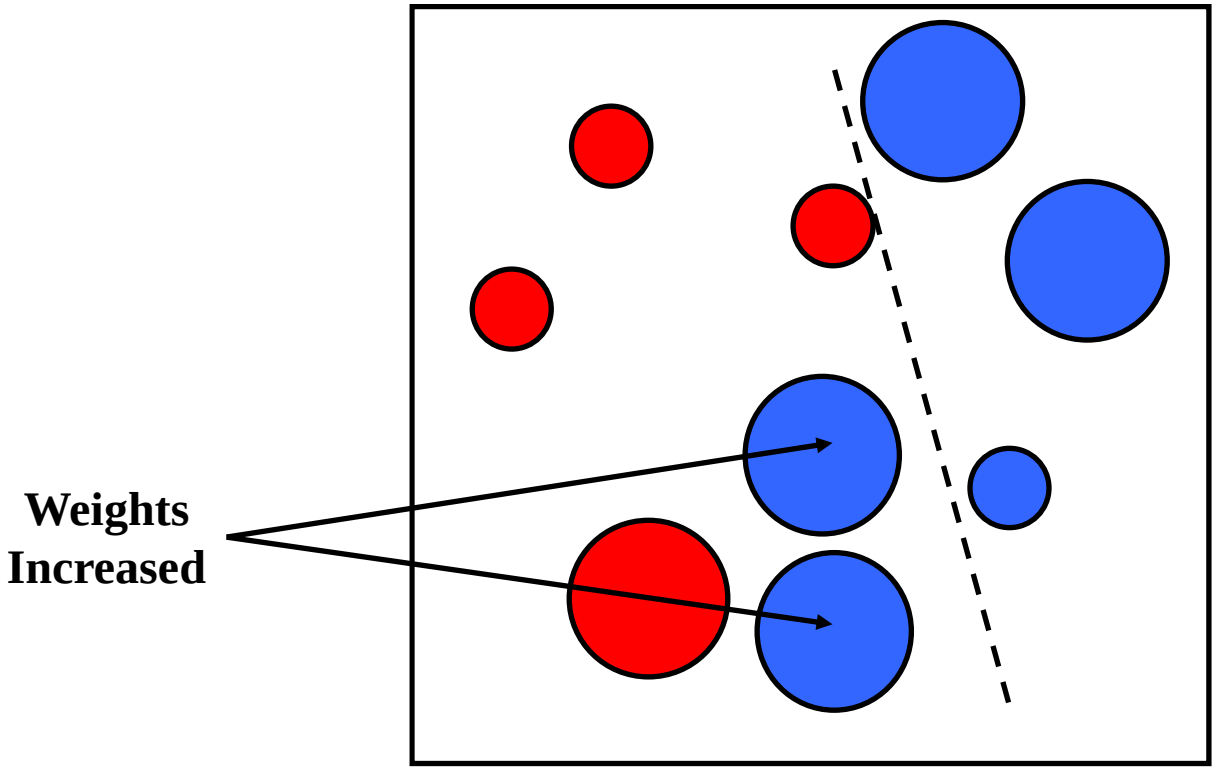
Boosting提出 —— 示例2



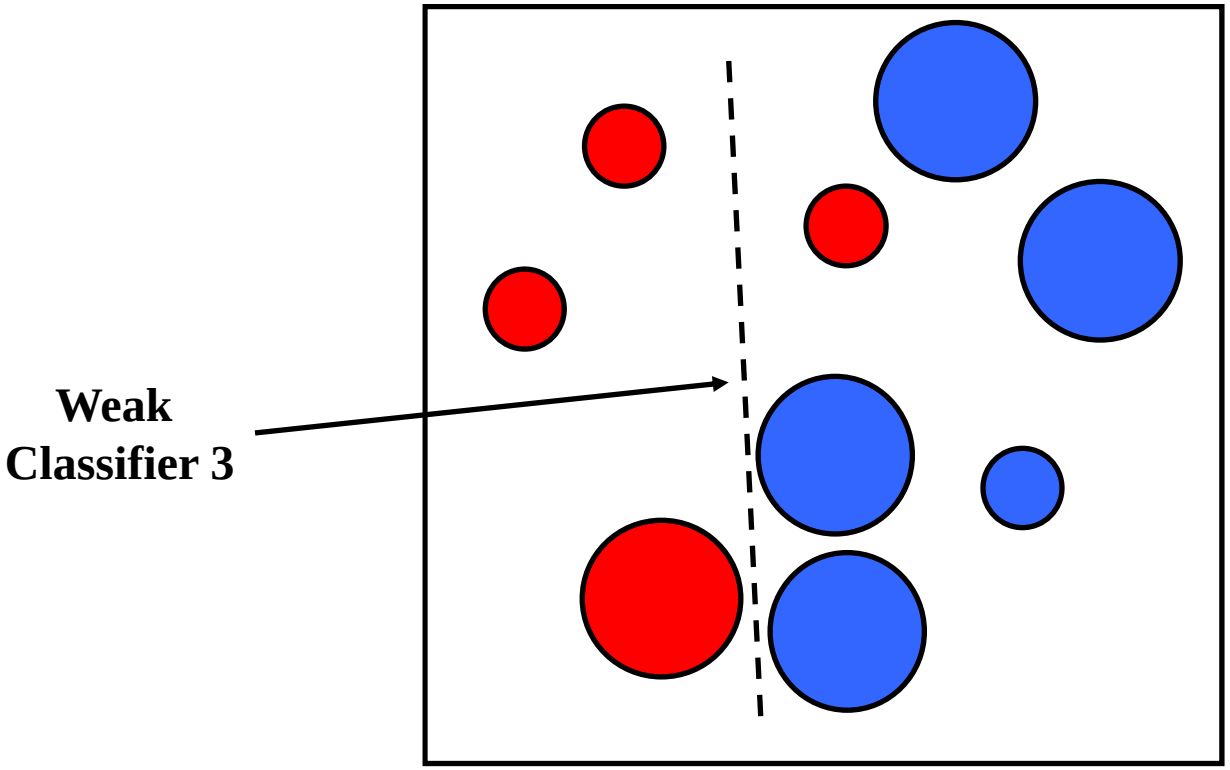
Boosting提出 —— 示例2



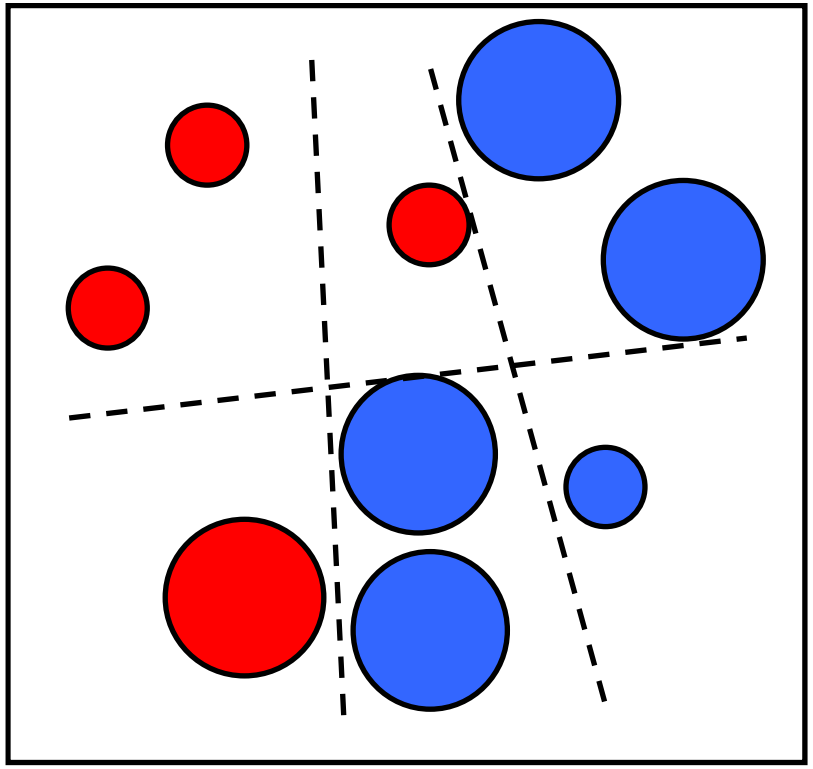
Boosting提出 —— 示例2



Boosting提出 —— 示例2



Boosting提出 —— 示例2



Final classifier is
a combination of weak classifiers

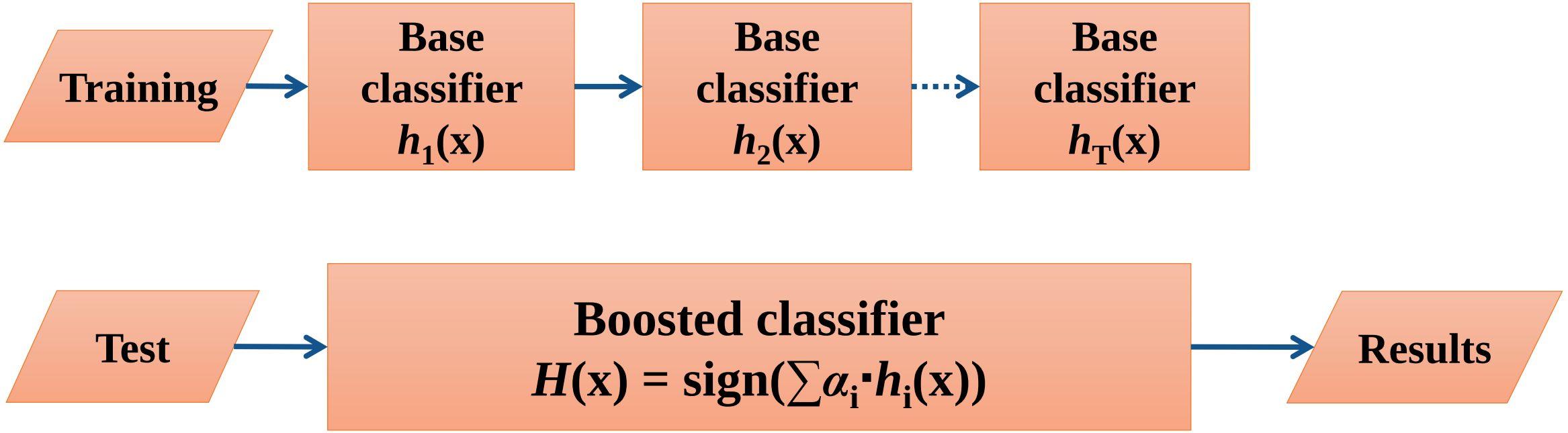
Boosting Weight

- 数据的权重有两个作用
  - 使用这些权值作为抽样分布，进行对数据的抽样
  - 分类器可以使用权值学习有利于高权重样本的分类器。把一个弱分类器提升为一个强分类器
- Boosting算法之一：AdaBoost算法

Boosting Weight

- 核心思想
- 样本的权重
  - 没有先验知识的情况下，初始的分布应为等概分布，也就是训练集如果有N个样本，每个样本的分布概率为1/N
  - 每次循环后提高错误样本的分布概率，分错样本在训练集中所占权重增大， 使得下一次循环的弱学习器能够集中力量对这些错误样本进行判断。
- 弱学习器的权重
  - 准确率越高的弱学习机权重越高
- 循环控制：损失函数达到最小
  - 在强学习器的组合中增加一个加权的弱学习器，使准确率提高，损失函数值减小。

Boosting分类过程



AdaBoost: train

for k = 1 to *iterations*:

- **classifier$_k$ = learn a weak classifier based on weights**
- **calculate weighted error for this classifier (加权分类误差)**

$$e_k = \sum_{i=1}^{n} w_i * 1[label_i \neq classifier_k(x_i)]$$

- **calculate "score" for this classifier (分类器的系数)**

$$a_k = \frac{1}{2}\log\left(\frac{1-e_i}{e_i}\right)$$

- **change the example weights (权值的更新)**

$$w_i = \frac{1}{Z} w_i \exp\left(-a_k * label_i * classifier_k(x_i)\right)$$

Adaboost例子

- 训练样本，初始时样本权重相同

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
初始权值	*w1 i*	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

- 1,2,3,7,8,9为同一类
- 4,5,6,10为同一类
- 根据x>v和x<v来分类

Adaboost例子

- 第一次迭代 $G_1(x) = \begin{cases} 1, & x < 2.5 \\ -1, & x > 2.5 \end{cases}$

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
分类器结果	$G1(x)$	1	1	1	-1	-1	-1	-1	-1	-1	-1
分类结果		对	对	对	对	对	对	错	错	错	对

- $G_1(x)$的误差率最低：$e_1 = P(G_1(x_i) \neq y_i) = \sum_{G_1(x_i) \neq y_i} w_{1i} = 0.1 + 0.1 + 0.1 = 0.3$
- $G_1(x)$的权重为：

$$\alpha_1 = \frac{1}{2} ln \frac{1-e_1}{e_1} = \frac{1}{2} ln \frac{1-0.3}{0.3} \approx 0.42365$$

- 分类函数：

$$f_1(x) = \alpha_1 G_1(x) = 0.42365 G_1(x)$$

for k = 1 to *iterations*:

- $classifier_k$ = learn a weak classifier based on weights
- calculate weighted error for this classifier(加权分类误差)

$$\varepsilon_k = \sum_{i=1}^{n} w_i * 1[label_i \neq classifier_k(x_i)]$$

- calculate "score" for this classifier(分类器的系数)

$$\alpha_k = \frac{1}{2} \log\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

- change the example weights(权值的更新)

$$w_i = \frac{1}{Z} w_i \exp(-\alpha_k * label_i * classifier_k(x_i))$$

Adaboost例子

- 第一次迭代

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
初始权值	*w1i*	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

$$G_1(x) = \begin{cases} 1, & x < 2.5 \\ -1, & x > 2.5 \end{cases} \qquad \alpha_1 = \frac{1}{2} ln \frac{1-e_1}{e_1} = \frac{1}{2} ln \frac{1-0.3}{0.3} \approx 0.42365$$

$$\begin{aligned} Z_1 &= \sum_{i=1}^{n} w_{1i} exp(-y_i \alpha_1 G_1(x_i)) \\ &= \sum_{i=1}^{3} 0.1 \times exp(-[1 \times 0.4263 \times 1]) \\ &\quad + \sum_{i=4}^{4-6,10} 0.1 \times exp(-[(-1) \times 0.4263 \times (-1)]) \\ &\quad + \sum_{i=7}^{9} 0.1 \times exp(-[1 \times 0.4263 \times (-1)]) \\ &\approx 0.3928 + 0.4582 + 0.0655 \\ &= 0.9165 \end{aligned}$$

$$\begin{aligned} w_{2i} &= \frac{w_{1i}}{Z_1} exp(-y_i \alpha_1 G_1(x_i)) \\ &= \begin{cases} \frac{0.1}{0.9165} exp(-[1 \times 0.4236 \times 1]), & i = 1,2,3 \\ \frac{0.1}{0.9165} exp(-[(-1) \times 0.4236 \times (-1)]), & i = 4,5,6,10 \\ \frac{0.1}{0.9165} exp(-[1 \times 0.4236 \times (-1)]), & i = 7,8,9 \end{cases} \\ &\approx \begin{cases} 0.07143 & i = 1,2,3 \\ 0.07143 & i = 4,5,6,10 \\ 0.16666 & i = 7,8,9 \end{cases} \end{aligned}$$

for k = 1 to *iterations*:

- $classifier_k$ = learn a weak classifier based on weights
- calculate weighted error for this classifier(加权分类误差)

$$\varepsilon_k = \sum_{i=1}^{n} w_i * 1[label_i \neq classifier_k(x_i)]$$

- calculate "score" for this classifier(分类器的系数)

$$\alpha_k = \frac{1}{2} \log\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

- change the example weights(权值的更新)

$$w_i = \frac{1}{Z} w_i \exp(-\alpha_k * label_i * classifier_k(x_i))$$

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
初始权值1	*w1iw1i*	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
更新权值2	*w2iw2i*	0.0714	0.0714	0.0714	0.0714	0.0714	0.0714	0.1667	0.1667	0.1667	0.0714

Adaboost例子

- 第二次迭代 $G_2(x) = \begin{cases} 1, & x < 8.5 \\ -1, & x > 8.5 \end{cases}$

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
权值分布	*w2i*	0.0714	0.0714	0.0714	0.0714	0.0714	0.0714	0.1667	0.1667	0.1667	0.0714
分类器结果	*G2(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

- $G_2(x)$的误差率最低：

$$e_2 = P(G_2(x_i) \neq y_i) = \sum_{G_2(x_i) \neq y_i} w_{2i} = 0.07143 + 0.07143 + 0.07143 = 0.21429$$

- $G_2(x)$的权重为：

$$\alpha_2 = \frac{1}{2} ln \frac{1-e_2}{e_2} = \frac{1}{2} ln \frac{1-0.21429}{0.21429} \approx 0.64963$$

- 分类函数：

$$f_2(x) = \alpha_2 G_2(x) = 0.64963 G_2(x)$$

for k = 1 to *iterations*:

- $classifier_k$ = learn a weak classifier based on weights
- calculate weighted error for this classifier(加权分类误差)

$$\varepsilon_k = \sum_{i=1}^{n} w_i * 1[label_i \neq classifier_k(x_i)]$$

- calculate "score" for this classifier(分类器的系数)

$$\alpha_k = \frac{1}{2} \log\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

- change the example weights(权值的更新)

$$w_i = \frac{1}{Z} w_i \exp(-\alpha_k * label_i * classifier_k(x_i))$$

Adaboost例子

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

$$\alpha_2 = \frac{1}{2} ln \frac{1-e_2}{e_2} = \frac{1}{2} ln \frac{1-0.21429}{0.21429} \approx 0.64963$$

$$\begin{aligned} Z_2 &= \sum_{i=1}^{n} w_{2i} exp(-y_i \alpha_2 G_2(x_i)) \\ &= \sum_{i=1}^{3} 0.07143 \times exp(-[1 \times 0.64963 \times 1]) \\ &\quad + \sum_{i=4}^{6} 0.07143 \times exp(-[(-1) \times 0.64963 \times 1]) \\ &\quad + \sum_{i=7}^{9} 0.16666 \times exp(-[1 \times 0.64963 \times 1]) \\ &\quad + \sum_{i=10}^{10} 0.07143 \times exp(-[(-1) \times 0.64963 \times (-1)] \\ &\approx 0.11191 + 0.41033 + 0.26111 + 0.03730 \\ &= 0.82065 \end{aligned}$$

for k = 1 to *iterations*:

- classifier$_k$ = learn a weak classifier based on weights
- calculate weighted error for this classifier(加权分类误差)

$$\varepsilon_k = \sum_{i=1}^{n} w_i * 1[label_i \neq classifier_k(x_i)]$$

- calculate "score" for this classifier(分类器的系数)

$$\alpha_k = \frac{1}{2} \log\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

- change the example weights(权值的更新)

$$w_i = \frac{1}{Z} w_i \exp(-\alpha_k * label_i * classifier_k(x_i))$$

$$w_{3i} = \frac{w_{2i}}{Z_2} exp(-y_i \alpha_2 G_2(x))$$

$$= \begin{cases} \frac{0.07143}{0.82065} exp(-[1 \times 0.64963 \times 1]) \approx 0.04546, & i = 1,2,3 \\ \frac{0.07143}{0.82065} exp(-[(-1) \times 0.64963 \times 1]) \approx 0.16667, & i = 4,5,6 \\ \frac{0.16666}{0.82065} exp(-[1 \times 0.64963 \times 1]) \approx 0.10606, & i = 7,8,9 \\ \frac{0.07143}{0.82065} exp(-[(-1) \times 0.64963 \times (-1)]) \approx 0.04546, & i = 10 \end{cases}$$

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
初始权值1	*w1 i*	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
权值2	*w2 i*	0.0714	0.0714	0.0714	0.0714	0.0714	0.0714	0.1667	0.1667	0.1667	0.0714
更新权值3	$w_{3i}	0.0455	0.0455	0.0455	0.1667	0.1667	0.1667	0.1061	0.1061	0.1061	0.0455

Adaboost例子

- 第三次迭代 $G_3(x) = \begin{cases} -1, & x < 5.5 \\ 1, & x > 5.5 \end{cases}$

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
权值分布	$w3i$	0.0455	0.0455	0.0455	0.1667	0.1667	0.1667	0.1061	0.1061	0.1061	0.0455
分类器结果	$G3(x)$	-1	-1	-1	-1	-1	-1	1	1	1	1
分类结果		错	错	错	对	对	对	对	对	对	错

- G3(x)的误差率最低：

$$e_3 = P(G_3(x_i) \neq y_i) = \sum_{G_3(x_i) \neq y_i} w_{3i} = 0.04546 + 0.04546 + 0.04546 + 04546 = 0.18184$$

- G3(x)的权重为：

$$\alpha_3 = \frac{1}{2} ln \frac{1-e_3}{e_3} = \frac{1}{2} ln \frac{1-0.18188}{0.18184} \approx 0.75197$$

- 分类函数：

$$f_3(x) = \alpha_3 G_3(x) = 0.75197 G_3(x)$$

for k = 1 to *iterations*:

- $classifier_k$ = learn a weak classifier based on weights
- calculate weighted error for this classifier(加权分类误差)

$$\varepsilon_k = \sum_{i=1}^{n} w_i * 1[label_i \neq classifier_k(x_i)]$$

- calculate "score" for this classifier(分类器的系数)

$$\alpha_k = \frac{1}{2} \log\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

- change the example weights(权值的更新)

$$w_i = \frac{1}{Z} w_i \exp(-\alpha_k * label_i * classifier_k(x_i))$$

Adaboost例子

- 第三次迭代

序号	i	1	2	3	4	5	6	7	8	9	10
数据	x	0	1	2	3	4	5	6	7	8	9
类别标签	y	1	1	1	-1	-1	-1	1	1	1	-1
初始权值1	$w1i$	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
权值2	$w2i$	0.0714	0.0714	0.0714	0.0714	0.0714	0.0714	0.1667	0.1667	0.1667	0.0714
权值3	$w3i$	0.0455	0.0455	0.0455	0.1667	0.1667	0.1667	0.1061	0.1061	0.1061	0.0455
更新权值4	$w4i$	0.125	0.125	0.125	0.1019	0.1019	0.1019	0.0648	0.0648	0.0648	0.125

- 最终分类器：

$$Gm(x)=sign(0.42365G1(x)+0.64963+G2(x)+0.75197G3(x))$$

AdaBoost: train 回顾算法

for k = 1 to *iterations*:

- **classifier$_k$ = learn a weak classifier based on weights**
- **calculate weighted error for this classifier (加权分类误差)**

$$e_k = \sum_{i=1}^{n} w_i * 1[label_i \neq classifier_k(x_i)]$$

- **Calculate "score" for this classifier (分类器的系数)**

$$a_k = \frac{1}{2}\log\left(\frac{1-e_i}{e_i}\right)$$

为什么?

- **change the example weights (权值的更新)**

$$w_i = \frac{1}{Z} w_i \exp\left(-a_k * label_i * classifier_k(x_i)\right)$$

如何确定α

- 集成学习模型 $f(x)=\sum_{m=1}^{M}\alpha_m G_m(x)$
- 指数损失函数 $L(y,f(x))=\exp[-yf(x)]$
- 公式推导

1、已知 $f_m(x)=f_{m-1}(x)+\alpha_m G_m(x)$

2、公式代入

将 $f_m(x)=f_{m-1}(x)+\alpha_m G_m(x)$ 代入损失函数得：

$$L(y,f(x))=\sum_{i=1}^{N}\exp[-y_i(f_{m-1}(x)+\alpha_m G_m(x))]$$

如何确定α

$$L(y, f(x)) = \sum_{i=1}^{N} [\exp[-y_i (f_{m-1}(x)][\exp(-y_i \alpha_m G_m(x))]]$$

- 3、部分替换

由于 y_i 和 $f_{m-1}(x)$ 已知，可以令 $\overline{w}_{mi} = \exp[-y_i f_{m-1}(x_i)]$ ，于是

$$L(y, f(x)) = \sum_{i=1}^{N} \overline{w}_{mi} \exp(-y_i \alpha_m G_m(x))$$

- 4、公式展开

$$\begin{aligned}
L(y, f(x)) &= \sum_{i=1}^{N} \overline{w}_{mi} \exp(-y_i \alpha_m G_m(x)) \\
&= \sum_{y_i = G_m(x_i)} \overline{w}_{mi} e^{-\alpha} + \sum_{y_i \neq G_m(x_i)} \overline{w}_{mi} e^{\alpha} \\
&= \sum_{y_i = G_m(x_i)} \overline{w}_{mi} e^{-\alpha} + \sum_{y_i \neq G_m(x_i)} \overline{w}_{mi} e^{-\alpha} - \sum_{y_i \neq G_m(x_i)} \overline{w}_{mi} e^{-\alpha} + \sum_{y_i \neq G_m(x_i)} \overline{w}_{mi} e^{\alpha} \\
&= e^{-\alpha} \sum_{i=1}^{N} \overline{w}_{mi} + (e^{\alpha} - e^{-\alpha}) \sum_{y_i \neq G_m(x_i)} \overline{w}_{mi} \\
&= e^{-\alpha} \sum_{i=1}^{N} \overline{w}_{mi} + (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} \overline{w}_{mi} I(y_i \neq G_m(x_i))
\end{aligned}$$

如何确定α

- 5、求导

$$e^{-\alpha}\sum_{i=1}^{N}\overline{w}_{mi}+(e^{\alpha}-e^{-\alpha})\sum_{i=1}^{N}\overline{w}_{mi}I(y_i\neq G_m(x_i))$$

把上式对 α 求导，再令导函数为 0，得：

$$-e^{-\alpha}\sum_{i=1}^{N}\overline{w}_{mi}+(e^{\alpha}+e^{-\alpha})\sum_{i=1}^{N}\overline{w}_{mi}I(y_i\neq G_m(x_i))=0$$

上式两边同时除以 $\sum_{i=1}^{N}\overline{w}_{mi}$，得：

$$-e^{-\alpha}+(e^{\alpha}+e^{-\alpha})\frac{\sum_{i=1}^{N}\overline{w}_{mi}I(y_i\neq G_m(x_i))}{\sum_{i=1}^{N}\overline{w}_{mi}}=0$$

- 6、部分替换

令 $\frac{\sum_{i=1}^{N}\overline{w}_{mi}I(y_i\neq G_m(x_i))}{\sum_{i=1}^{N}\overline{w}_{mi}}=e_m$，则有：

$$-e^{-\alpha}+(e^{\alpha}+e^{-\alpha})e_m=0$$

如何确定α

- 7、求导计算

for k = 1 to *iterations*:

- classifier$_k$ = learn a weak classifier based on weights
- calculate weighted error for this classifier(加权分类误差)

$$\varepsilon_k = \sum_{i=1}^{n} w_i * 1[label_i \neq classifier_k(x_i)]$$

- calculate "score" for this classifier(分类器的系数)

$$\alpha_k = \frac{1}{2}\log\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$

- change the example weights(权值的更新)

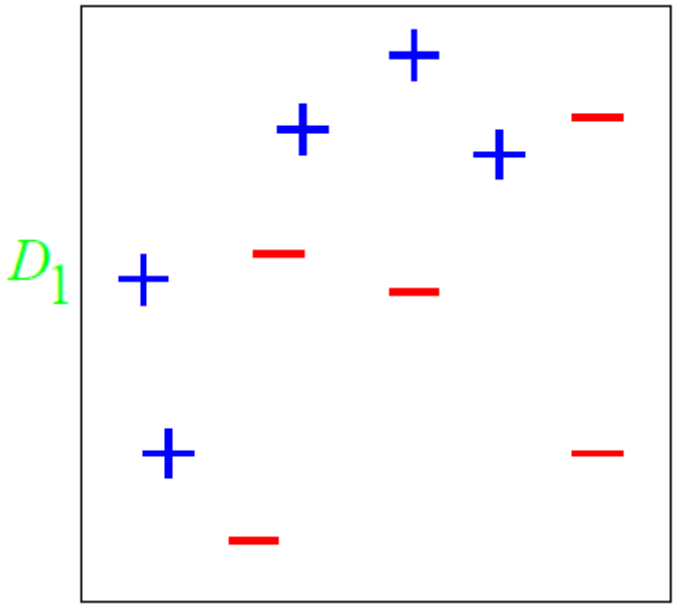
$$w_i = \frac{1}{Z} w_i \exp(-\alpha_k * label_i * classifier_k(x_i))$$

$$-e^{-\alpha} + (e^{\alpha} + e^{-\alpha})e_m = 0$$
$$(e^{\alpha} + e^{-\alpha})e_m = e^{-\alpha}$$
$$(e^{2\alpha} + 1)e_m = 1$$
$$e^{2\alpha} + 1 = \frac{1}{e_m}$$
$$e^{2\alpha} = \frac{1}{e_m} - 1 = \frac{1 - e_m}{e_m}$$
$$2\alpha = \log\frac{1 - e_m}{e_m}$$
$$\alpha = \frac{1}{2}\log\frac{1 - e_m}{e_m}$$

- 其中

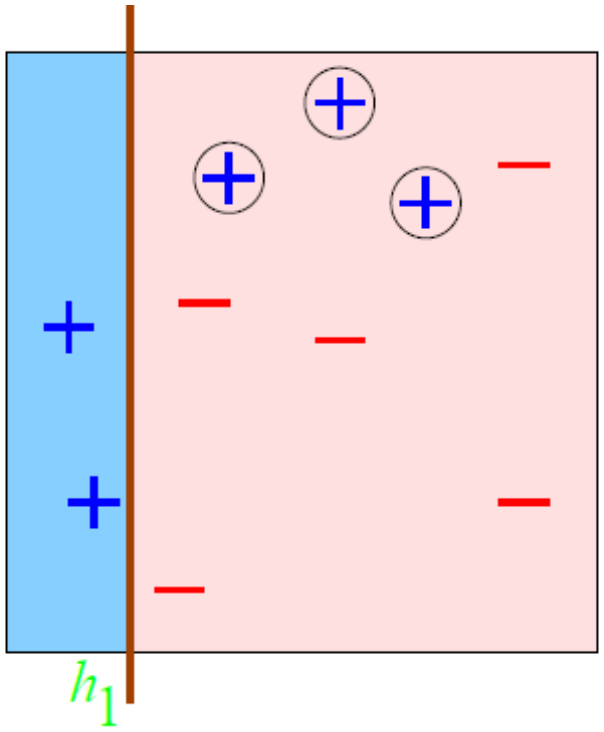
$$e_m = \frac{\sum_{i=1}^{N} \overline{w}_{mi} I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} \overline{w}_{mi}} = \sum_{i=1}^{N} w_{mi} I(y_i \neq G_m(x_i))$$

A Toy Example



A Toy Example

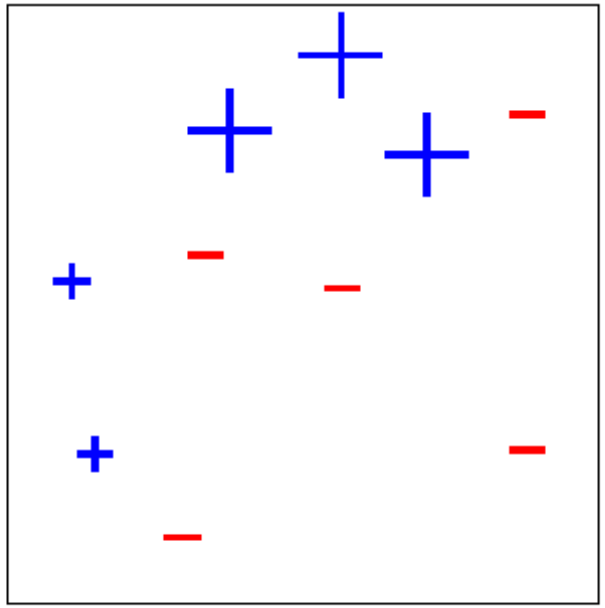
Round 1



$\varepsilon_1 = 0.30$

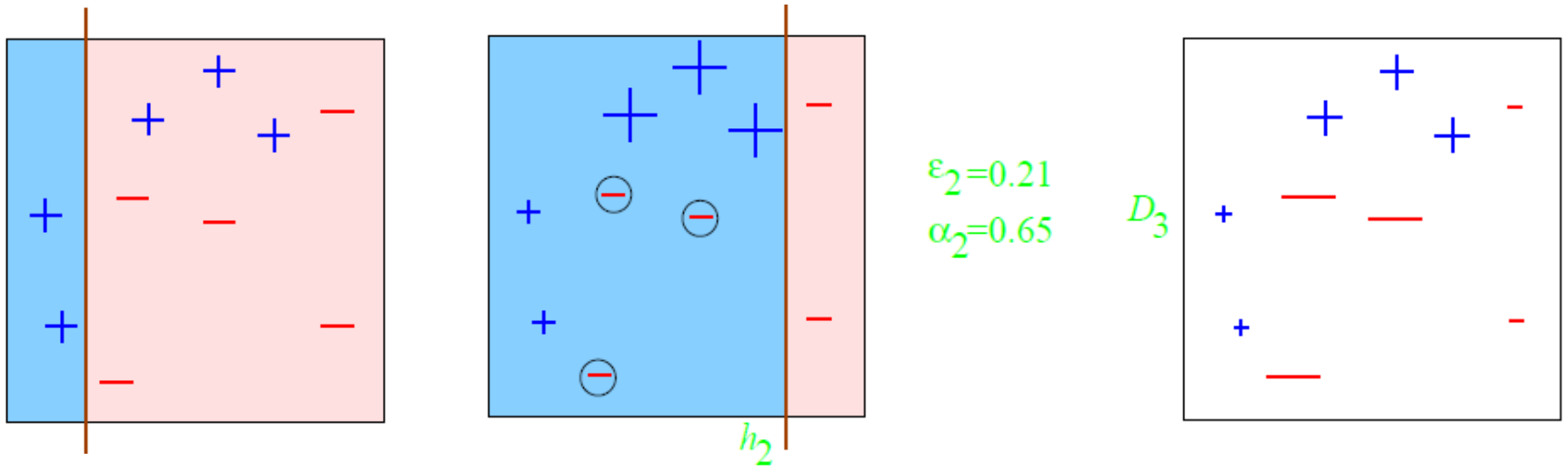
$\alpha_1 = 0.42$

D_2



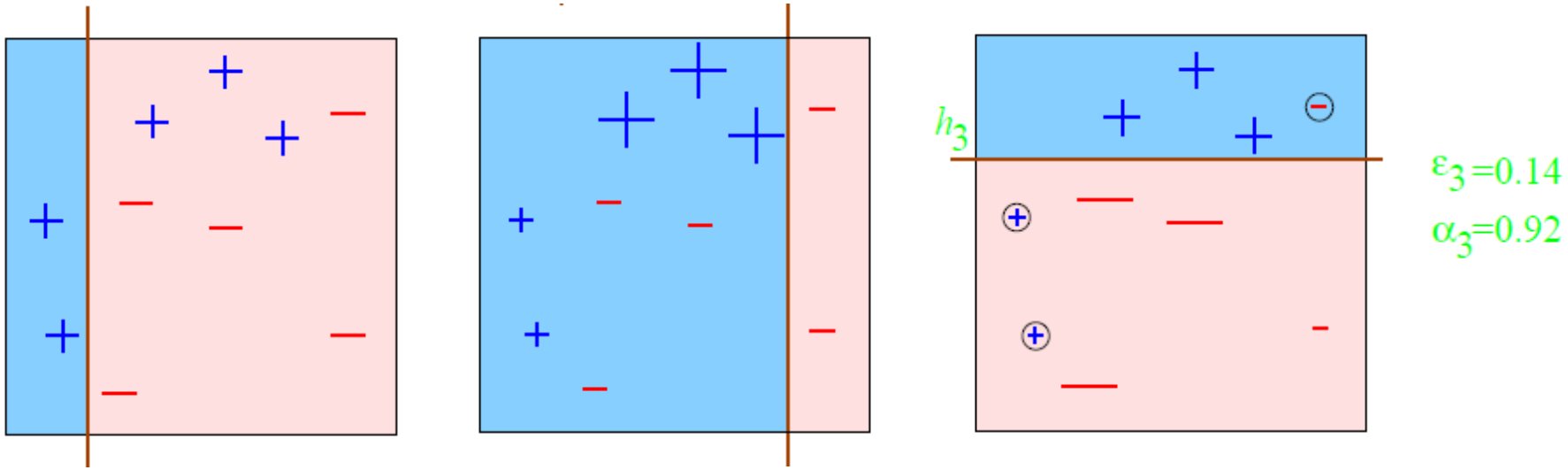
A Toy Example

Round 2



A Toy Example

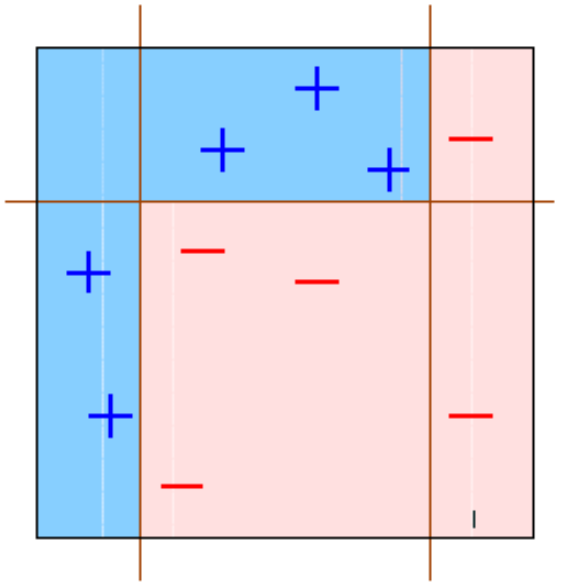
Round 3



A Toy Example

Final Hypothesis

$$H_{\text{final}} = \text{sign}\left(0.42 \; + 0.65 \; + 0.92 \; \right)$$



AdaBoost小结

- 主要优点
  - Adaboost作为分类器时，分类精度高
  - 可以使用各种回归分类模型来构建弱学习器，非常灵活。
  - 作为简单的二元分类器时，构造简单，结果可理解。
  - 不容易发生过拟合
- 主要缺点
  - 对异常样本敏感，异常样本在迭代中可能会获得较高的权重，影响最终的强学习器的预测准确性。

其他Boosting方法

- 不同的损失函数和极小化损失函数方法决定了boosting的最终效果

Name	Loss	Derivative	f^*	Algorithm
Squared error	$\frac{1}{2}(y_i - f(\mathbf{x}_i))^2$	$y_i - f(\mathbf{x}_i)$	$\mathbb{E}[y\|\mathbf{x}_i]$	L2Boosting
Absolute error	$\|y_i - f(\mathbf{x}_i)\|$	$\operatorname{sgn}(y_i - f(\mathbf{x}_i))$	$\operatorname{median}(y\|\mathbf{x}_i)$	Gradient boosting
Exponential loss	$\exp(-\tilde{y}_i f(\mathbf{x}_i))$	$-\tilde{y}_i \exp(-\tilde{y}_i f(\mathbf{x}_i))$	$\frac{1}{2}\log\frac{\pi_i}{1-\pi_i}$	AdaBoost
Logloss	$\log(1 + e^{-\tilde{y}_i f_i})$	$y_i - \pi_i$	$\frac{1}{2}\log\frac{\pi_i}{1-\pi_i}$	LogitBoost

算法对比

Items	Bagging	Boosting
采样算法	均匀取样	根据错误率来取样
各轮训练集选取	随机的	与前面各轮的学习结果有关
预测函数权重	没有权重	有权重
并行性	各个预测函数可以并行生成	只能顺序生成
准确性	没有boosting高	在大多数数据集中，高
过拟合	不会	在有些数据集中，会

Gradient Boost Decision Tree（GBDT)

- GBDT、Treelink、 GBRT (Gradient Boost Regression Tree)、Tree Net、MART (Multiple Additive Regression Tree)
  - 由多棵决策树构成，通常都是上百棵树，而且每棵树规模都较小（即树的深度会比较浅）
  - 模型预测的时候，对于输入的一个样本实例，然后会遍历每一棵决策树，每棵树都会对预测值进行调整修正，最后得到预测的结果

$$F(X) = F_0 + \beta_1 T_1(X) + \beta_2 T_2(X) + \ldots + \beta_M T_M(X)$$

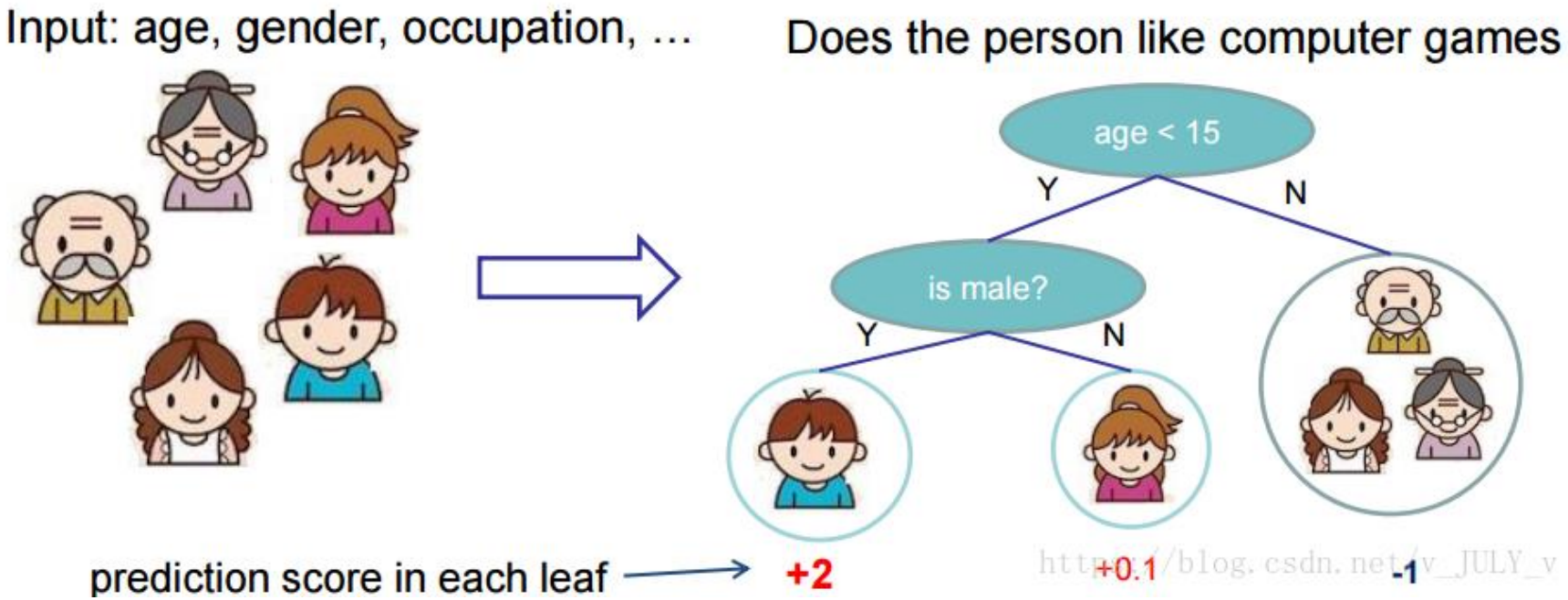
Gradient Boost Decision Tree （GBDT)

$$F(X) = F_0 + \beta_1 T_1(X) + \beta_2 T_2(X) + \dots + \beta_M T_M(X)$$

- F0是设置的初值， Ti是一棵一棵的决策树
- 对于不同的问题和选择不同的损失函数，初值的设定是不同的
- 比如回归问题并且选择高斯损失函数，那么这个初值就是训练样本的响应均值

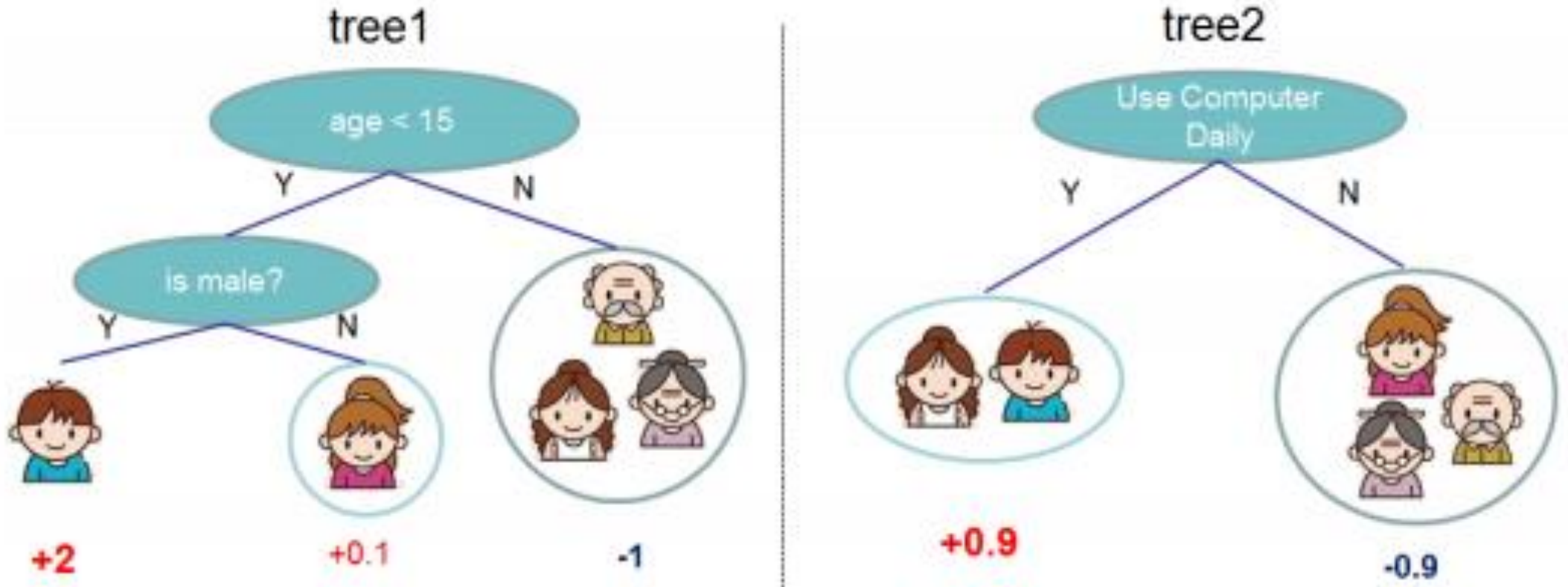
例子

- 预测一家人对电子游戏的喜好程度
- 特征
  - **年龄**：年轻和年老相比，年轻更可能喜欢电子游戏
  - **性别**：男性和女性相比，男性更喜欢电子游戏
  - **电脑使用频率**：经常用电脑更喜欢电子游戏



例子

由两棵决策树构成

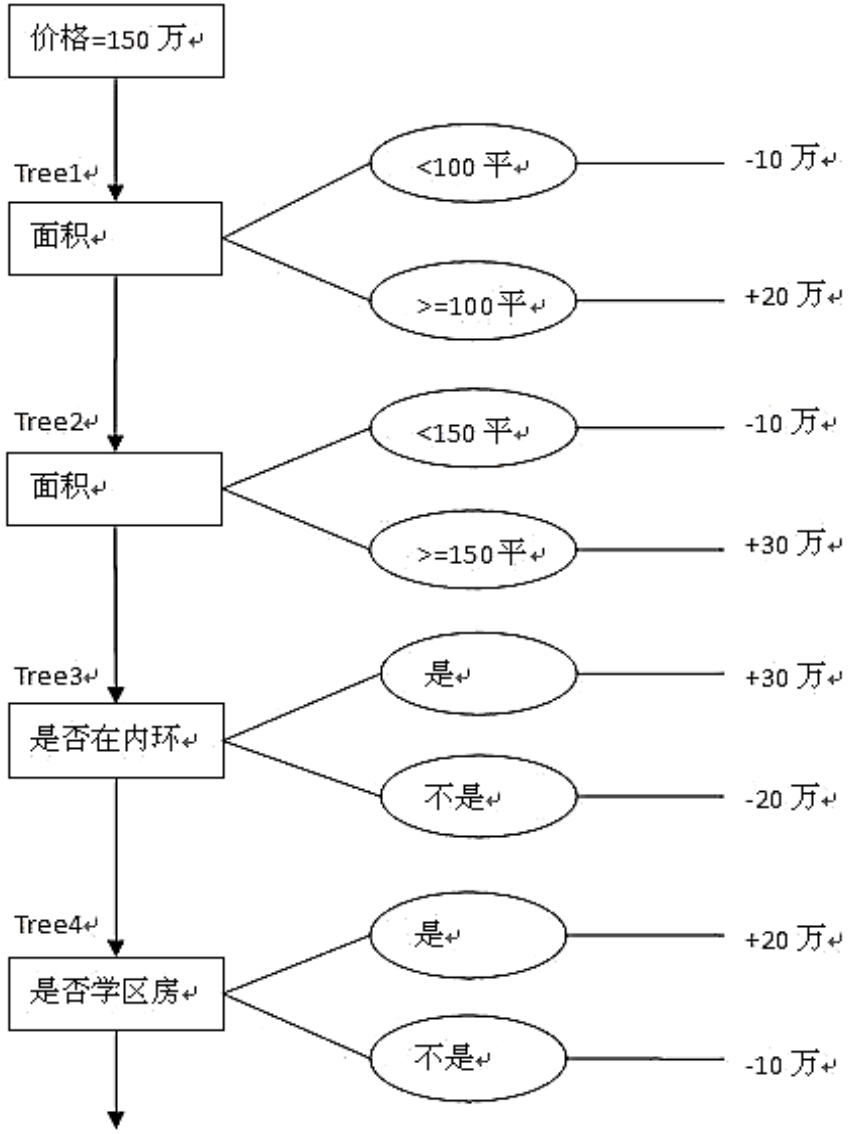


f() = 2 + 0.9= 2.9

f()= -1 - 0.9 = -1.9

例子

- 一套房子的价格
- 特征有三个
  - 房子的面积
  - 是否在内环
  - 是否学区房
- 由四棵决策树构成
  - Decision Stump
- 初值设为价格的均值150万
- 一个面积为120平的内环非学区房的价格预测值为：150+20-10+30-10= 180 万



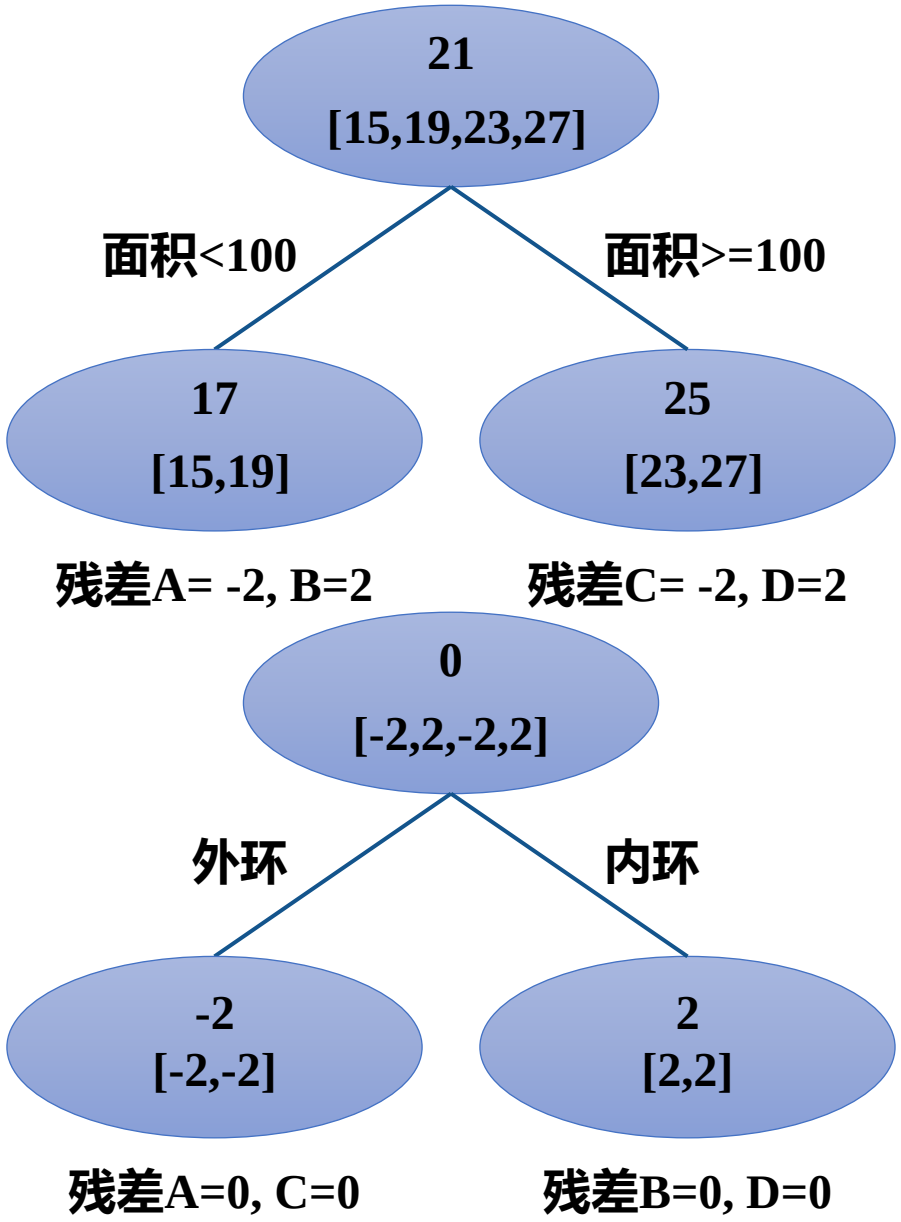
GBDT学习过程

- Boosting迭代，即通过迭代多棵树来共同决策
- GBDT是把所有树的结论累加起来做最终结论的，所以可以想到每棵树的结论并不是房价本身，而是房价的一个累加量
- 每一棵树学的是之前所有树结论和的残差，这个残差就是一个加预测值后能得真实值的累加量

GBDT学习过程 —— 例子引入

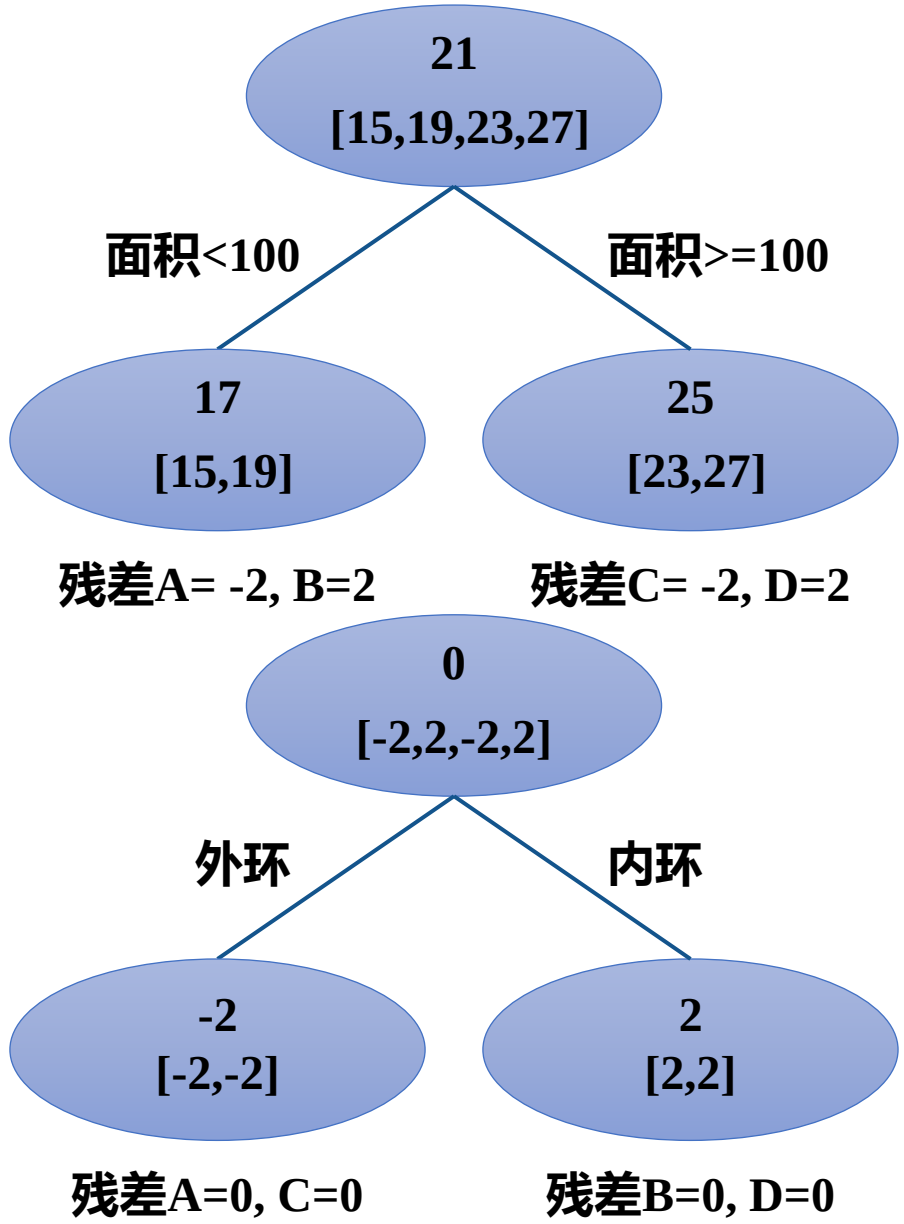
- 房子的价格

面积	是否内环	房价
92	否	15
90	是	19
120	否	23
130	是	27



GBDT学习过程 —— 例子引入

- 预测
  - 125平方米，内环：25 + 2=27



GBDT例子的几点问题

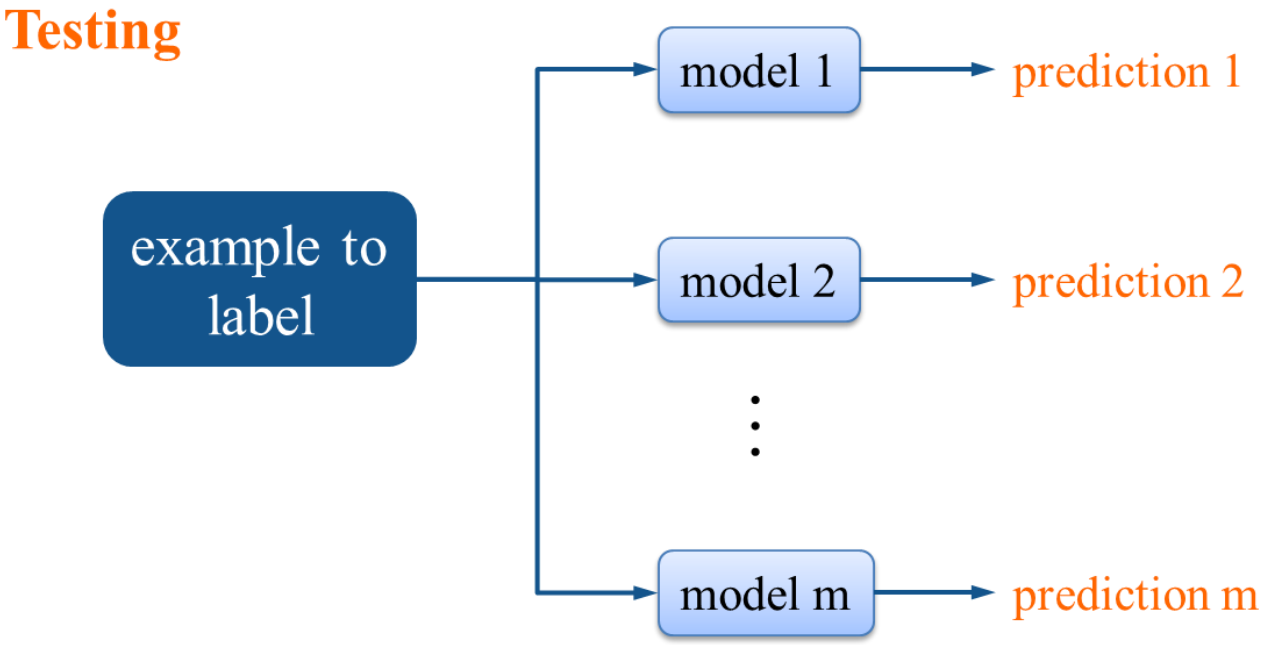
- GBDT有何优点？
  - 防止过拟合
  - 每一步的残差计算其实变相地增大了分错instance的权重，而已经分对的instance则都趋向于0
- 其他类似算法
  - XGBoost

集成学习算法

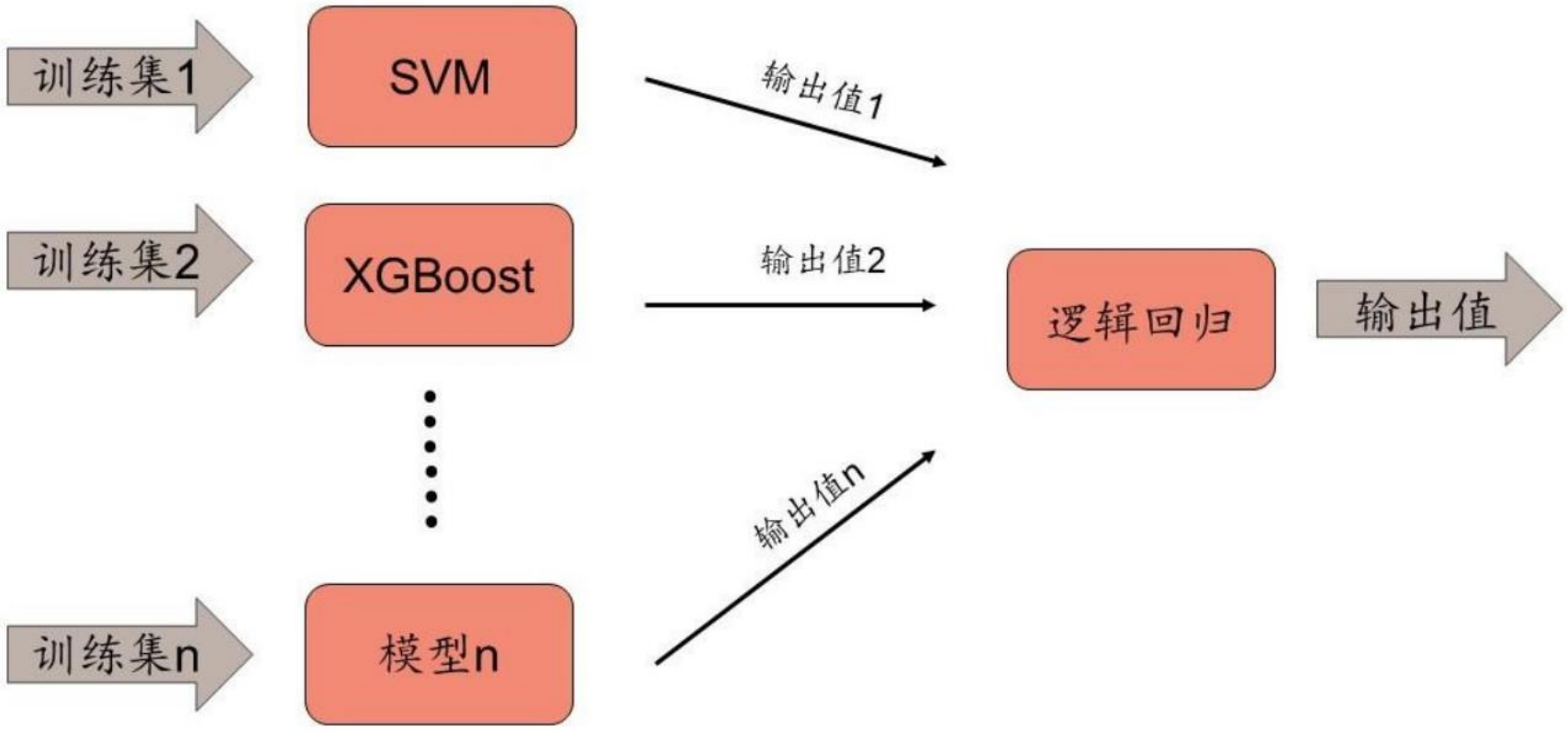
1. Bagging
2. Boosting
3. Stacking

如何组合多个分类器?

- 平均
- 投票
  - 多数投票：Bagging
  - 带权重的多数投票：Adaboost
- 学习组合器
  - Stacking
  - RegionBoost



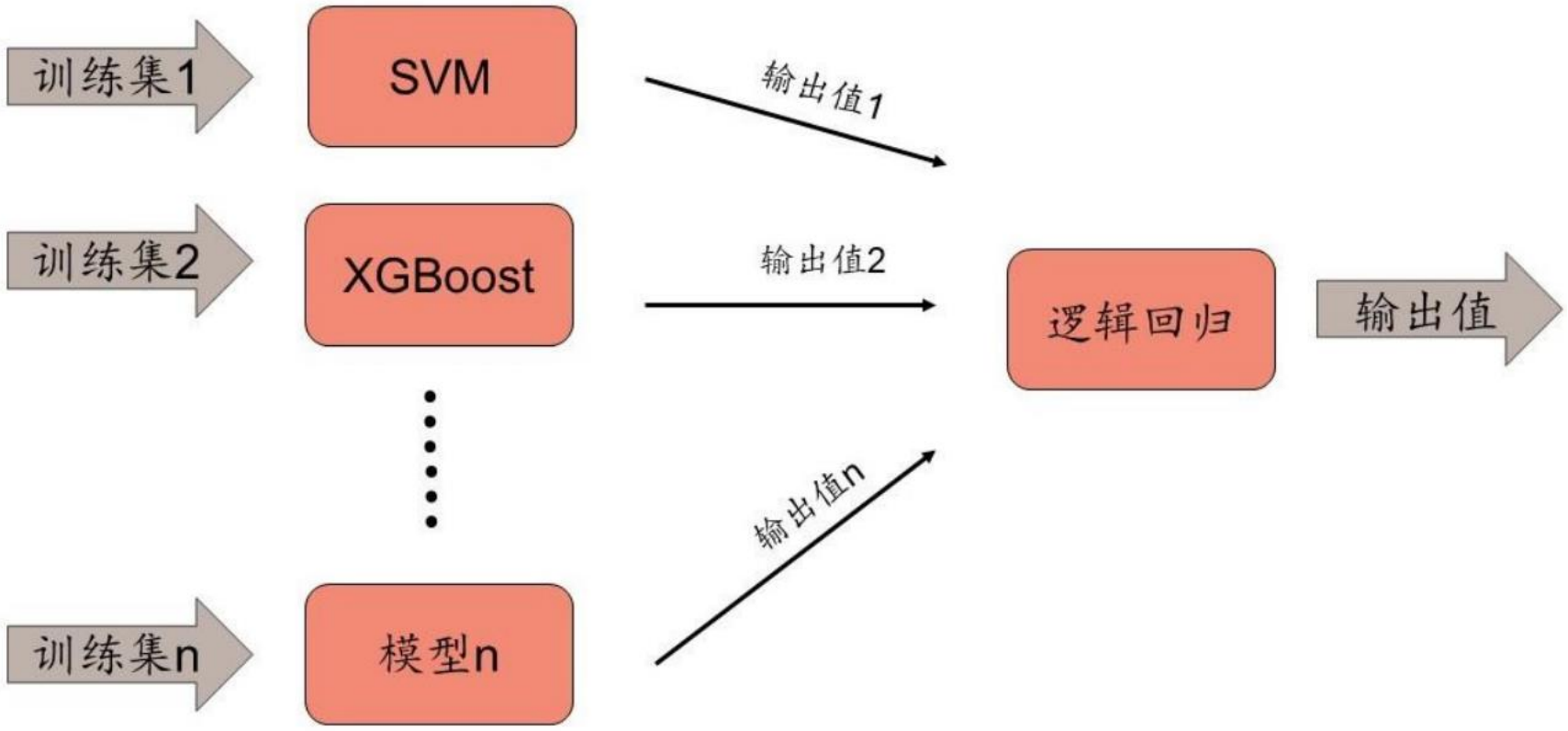
Stacking



数据 *基分类器* *元分类器*

$\{(x_1, y_1) \dots (x_n, y_n)\}$ $\{(h_1(x_i), h_2(x_i), \dots, h_k(x_i), y_i)\}$

Stacking



数据　　*基分类器*

$\{(x_1, y_1) \dots (x_n, y_n)\}$

元分类器

$\{(h_1(x_i), h_2(x_i), \dots, h_k(x_i), y_i)\}$

基分类器的结果再训练出一个元分类器

南京大学
NANJING UNIVERSITY

目录

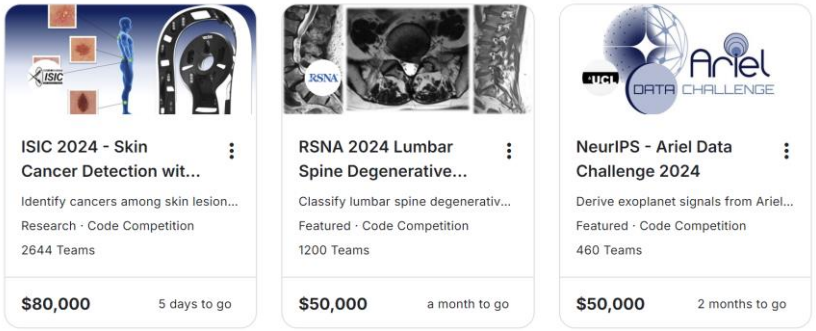
集成学习应用

- 竞赛的冠军
  - Kaggle、KDDCUP、ImageNet、天池
- 视觉领域
  - 人脸检测、目标追踪、物体识别
- 金融领域：
  - 股票选择、金融反欺诈
- 生物领域
  - 基因组功能预测、癌症预测

数据科学竞赛

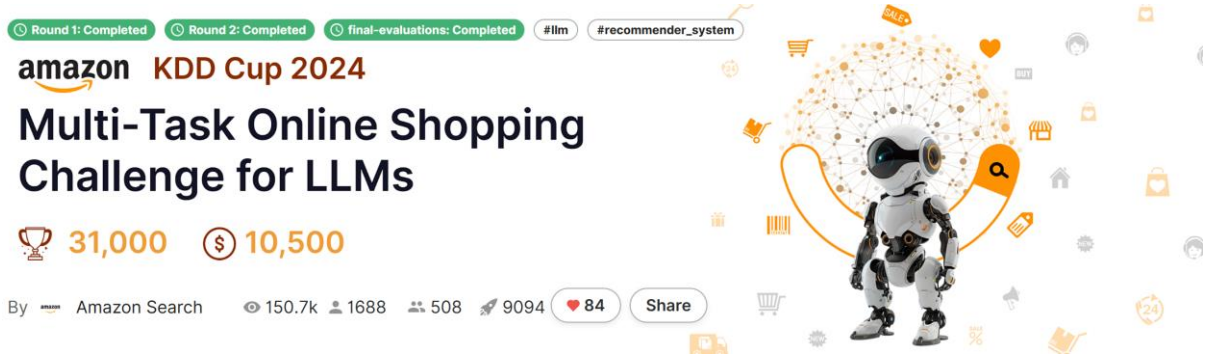
- **Kaggle数据建模与分析竞赛平台**

  https://www.kaggle.com/competitions



- **KDDCUP**

  https://kdd.org/kdd-cup



- **天池大数据竞赛**

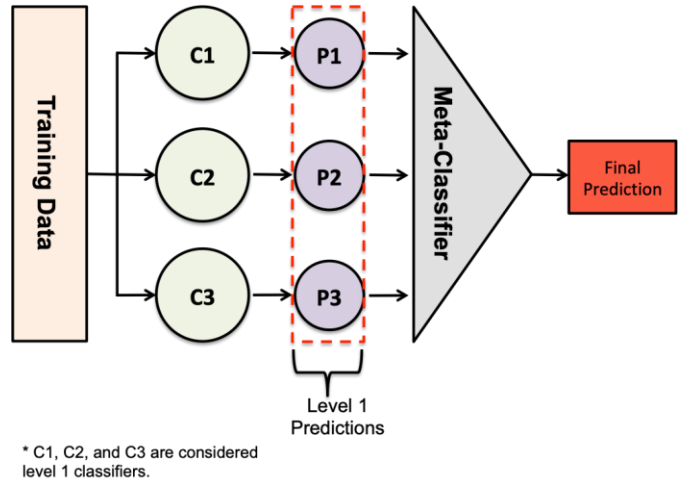
  https://tianchi.aliyun.com/competition/gameList/activeList

- ……

竞赛模型集成融合

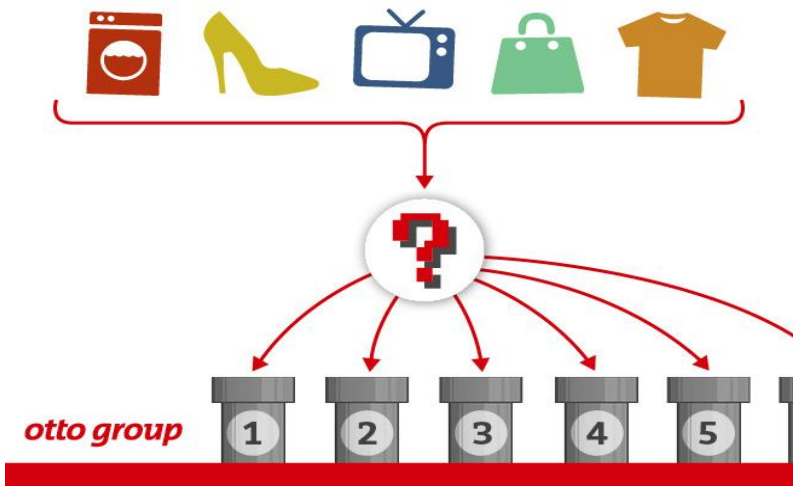
- 对提交文件进行集成
  - 只需要模型在测试集上的预测结果，而不需要重新训练一个模型
  - 投票、平均、加权平均
- 堆叠/混合集成
  - 使用大量**基分类器**，然后使用**元分类器**来融合它们的预测，旨在降低泛化误差

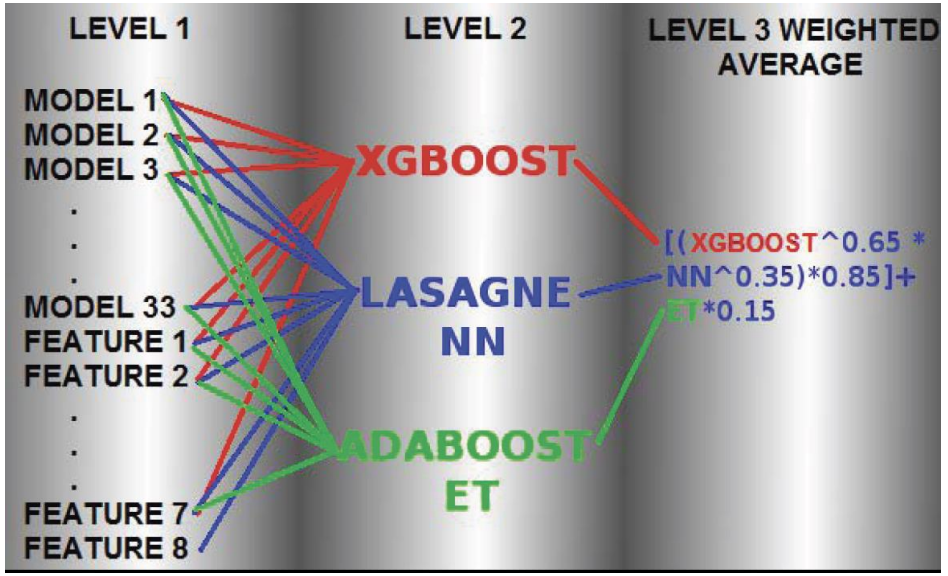
ID	Type	SKU	Name	Published	Is featured	Visibility in catalog
17	variable	MH01	Chaz Kangeroo Hoodie	1	0	visible
11	variation	MH01-L-Black	Chaz Kangeroo Hoodie-L-Black	1	0	visible
12	variation	MH01-L-Gray	Chaz Kangeroo Hoodie-L-Gray	1	0	visible
13	variation	MH01-L-Orange	Chaz Kangeroo Hoodie-L-Orange	1	0	visible
8	variation	MH01-M-Black	Chaz Kangeroo Hoodie-M-Black	1	0	visible
9	variation	MH01-M-Gray	Chaz Kangeroo Hoodie-M-Gray	1	0	visible
10	variation	MH01-M-Orange	Chaz Kangeroo Hoodie-M-Orange	1	0	visible
5	variation	MH01-S-Black	Chaz Kangeroo Hoodie-S-Black	1	0	visible
6	variation	MH01-S-Gray	Chaz Kangeroo Hoodie-S-Gray	1	0	visible
7	variation	MH01-S-Orange	Chaz Kangeroo Hoodie-S-Orange	1	0	visible
14	variation	MH01-XL-Black	Chaz Kangeroo Hoodie-XL-Black	1	0	visible
15	variation	MH01-XL-Gray	Chaz Kangeroo Hoodie-XL-Gray	1	0	visible
16	variation	MH01-XL-Orange	Chaz Kangeroo Hoodie-XL-Orange	1	0	visible
2	variation	MH01-XS-Black	Chaz Kangeroo Hoodie-XS-Black	1	0	visible
3	variation	MH01-XS-Gray	Chaz Kangeroo Hoodie-XS-Gray	1	0	visible
4	variation	MH01-XS-Orange	Chaz Kangeroo Hoodie-XS-Orange	1	0	visible



Kaggle竞赛案例

- 欧图（Otto）集团产品分类挑战赛的第一名解决方案（三级堆叠）
  - 第一级：采用33个基模型预测作为第二级的元特征，另有 8 个工程特征
  - 第二级：采用XGBoost, NN 及 Adaboost等3个元模型
  - 第三级：由二级预测的加权平均值组成。

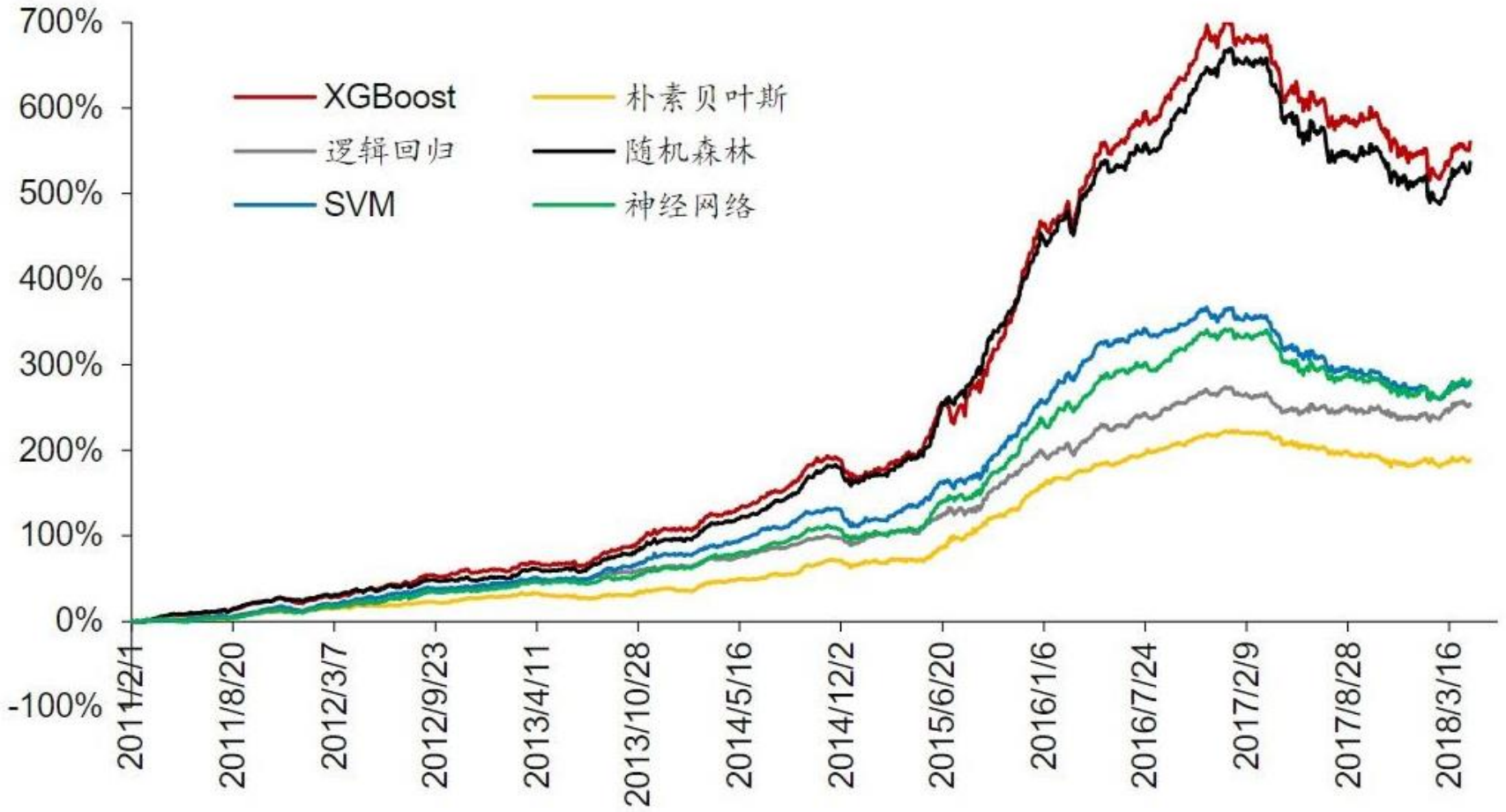




Stacking股票选择

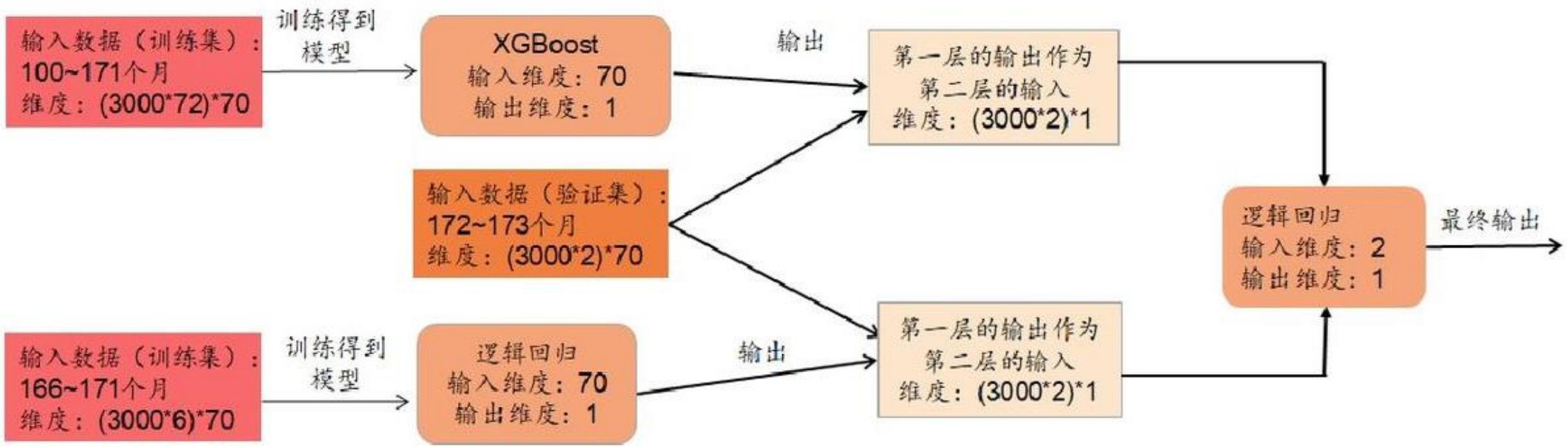
基模型的对比和选取

各机器学习模型相对中证500的超额收益（训练数据72个月）



Stacking股票选择

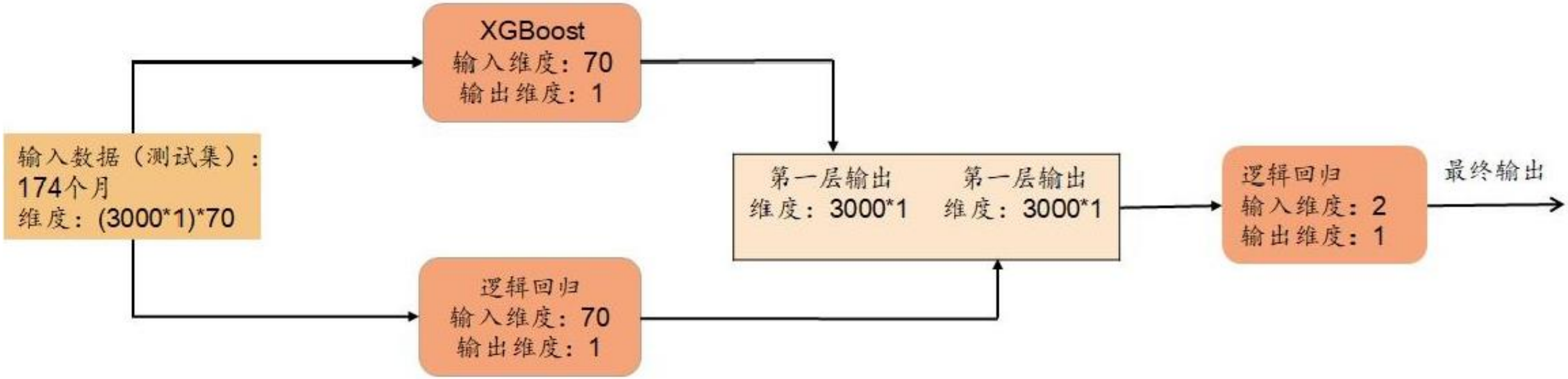
Stacking集成学习训练过程



华泰证券研究所

Stacking股票选择

Stacking集成学习测试过程

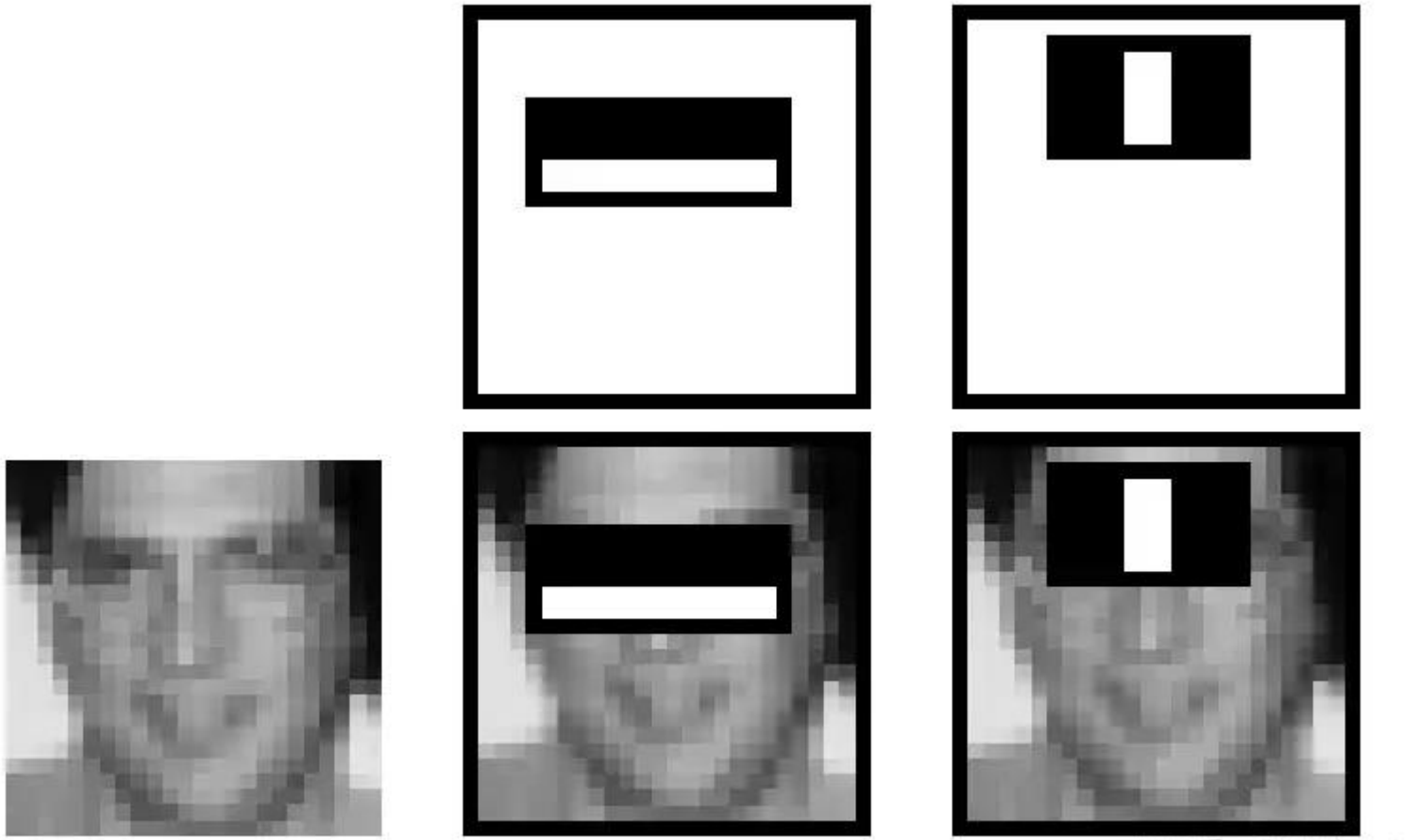


华泰证券研究所

AdaBoost人脸检测

- **特征提取**：利用Harr-like小波特征得到海量的特征数据集
- **AdaBoost迭代训练过程**
  - 获取当前**最优弱分类器**：计算出分类错误率最小的Harr特征
  - 根据分类结果**更新权重**
- 使用**AdaBoost级联分类器**去检测任意一张图片，识别人脸

Harr-like小波特征



最优弱分类器

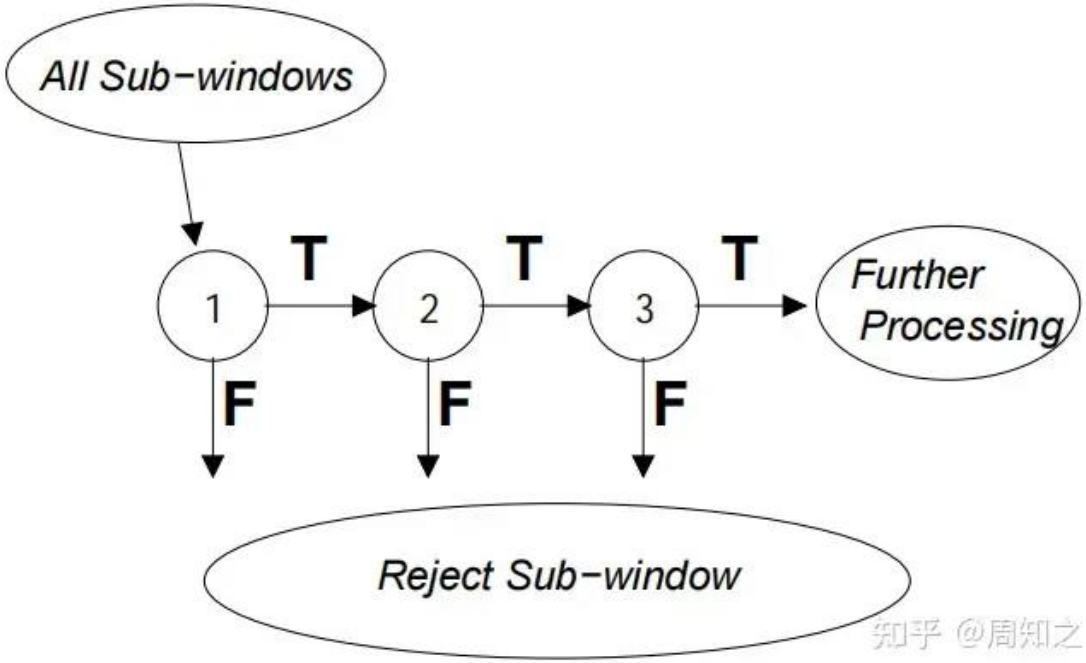
- **弱分类器：二叉树（Stump）**
  - **对于每个Harr特征,计算所有训练样本的特征值并排序**
  - **寻找合适的阈值，使该特征对所有样本的分类误差最小。**

    1. 全部人脸样本的权重和t1
    2. 全部非人脸样本的权重和t0
    3. 在当前元素之前的人脸样本的权重和s1
    4. 在当前元素之前的非人脸样本的权重和s0
    5. 当前元素的分类误差：r=min{[s1+(t0-s0)],[s0+(t1-s1)]}

- **最优弱分类器：分类误差最小的二叉树**

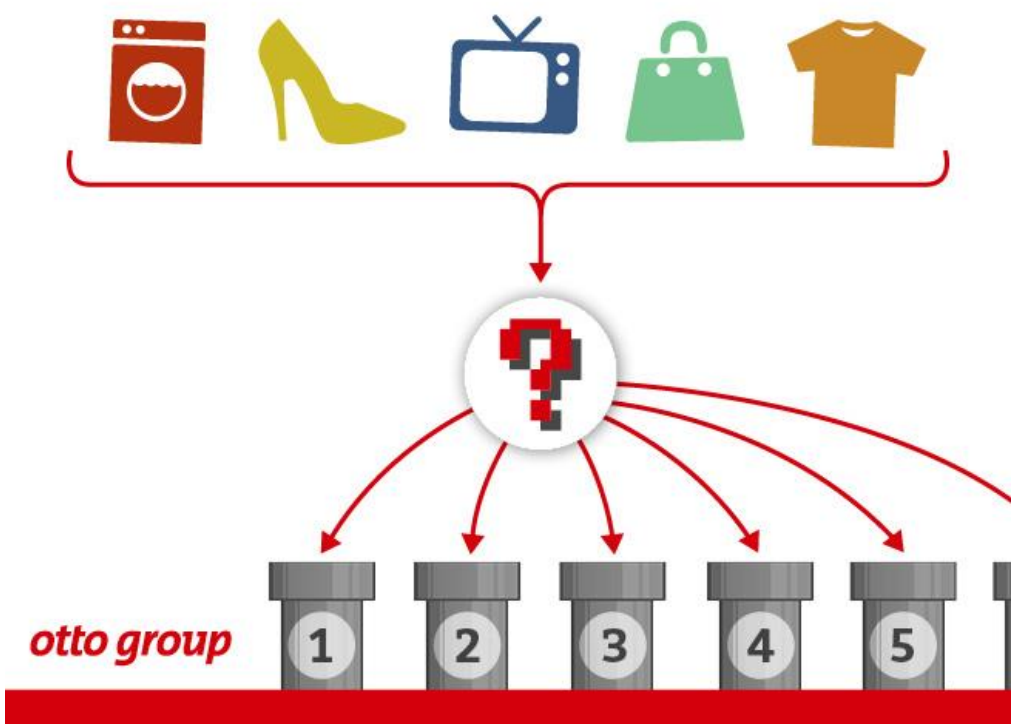
AdaBoost级联分类器

级联结构：将多个强分类器连接在一起进行操作。



作业：欧图集团产品分类

Otto Group是德国本土最大的在线零售商之一，拥有多种多样的产品线。为了提高运营效率和客户体验，准确地对产品进行分类变得至关重要。本作业将基于Otto Group提供的一组匿名化的产品数据集，要求同学们运用集成学习的方法来构建一个产品分类模型。



该数据集包含超过 200,000 种产品的 93 个特征。目标是构建一个能够区分主要产品类别的预测模型。

作业：欧图集团产品分类

作业目标

1. 理解并应用集成学习技术。
2. 对数据进行必要的清洗，包括处理缺失值、异常值等。
3. 对分类特征进行编码，如使用独热编码（One-Hot Encoding）。
4. 分析特征重要性以理解哪些因素对于产品分类最为关键。
5. 通过调整参数优化所选模型的性能。
6. 提交最终模型的预测结果。

数据集介绍及下载：https://www.kaggle.com/competitions/otto-group-product-classification-challenge/data

作业：欧图集团产品分类

具体要求：

- 数据集使用Kaggle网站提供的训练数据集（train.csv，不要修改文件名），文件路径设置为当前代码文件路径（如“./train.csv”）。
- 将下载的训练数据集分为训练集和测试集，比例自定，至少保持20%的数据用于测试。
- 计算模型的准确率（需不低于90%）等评价指标。

作业：欧图集团产品分类

上交形式

- 提交完整的代码文件（使用Jupyter Notebook格式）。
- 文件命名为“**学号-姓名-准确率.ipynb**”（如**602024710021-张三-97.8.ipynb**）。
- 不需要提交数据集，只提交Jupyter文件。

注意事项

- 确保代码具有良好的可读性，适当添加注释说明。
- 遵守学术诚信原则，严禁抄袭他人作品。