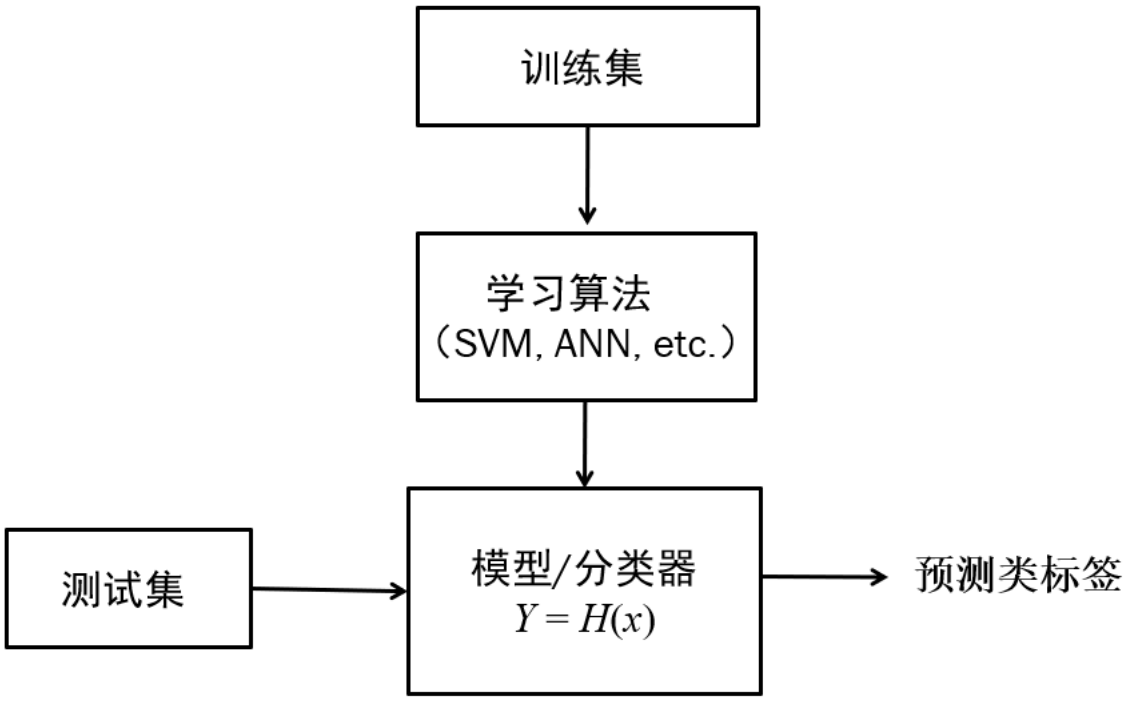
分类

👉 监督学习,VS,无监督学习

- 无监督学习(关联规则挖掘，聚类分析)
  - 无**label**
  - 挖掘**潜在的数据内部模式**
- 监督学习(分类，预测)
  - 数据有label
  - 通过label学习数据中的模式(模型)
  - 利用学习的模式对新数据进行分类预测

👉 监督学习框架

监督学习的基本框架



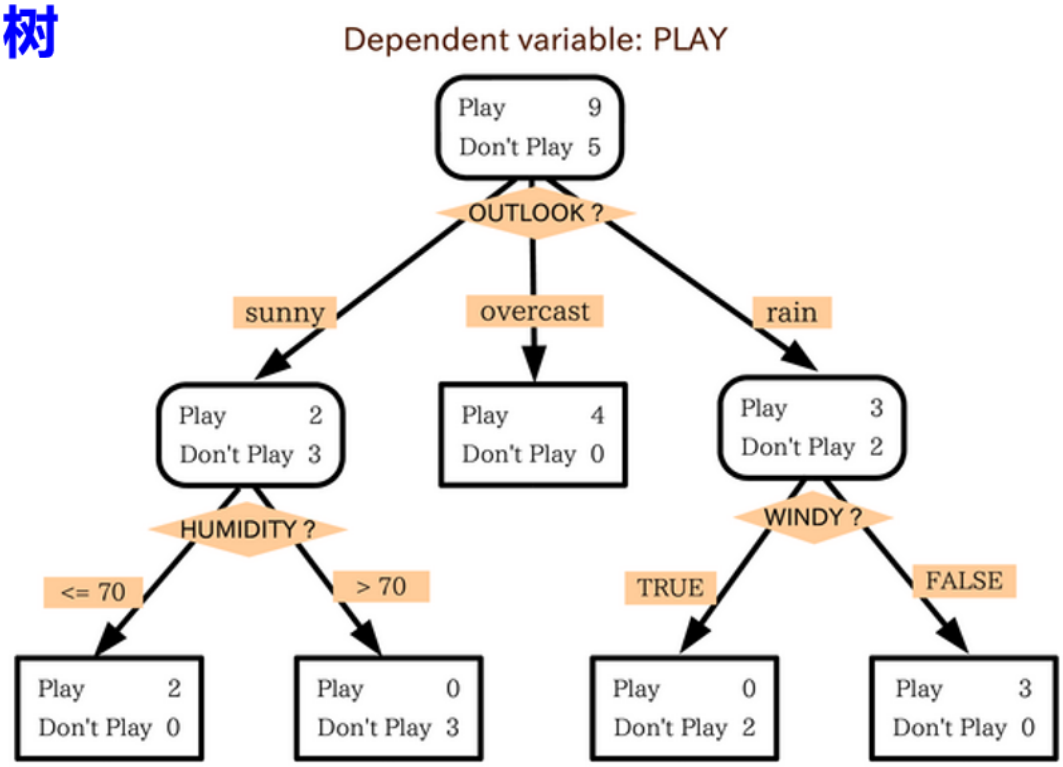
👉 模型分类

- 生成模型(Generative Model)
  - 从数据中学习**原始的真实数据生成模型。**常见的方法是**学习数据的联合分布。**常见的模型——**朴素贝叶斯方法，隐马尔可夫模型**
  - 特点
    - 当容量大时，生成模型容易接近真实模型
    - 能处理具有**隐含变量**的情况
- 判别模型(Discriminative Model)
  - 从数据中学习**不同类概念的区别**从而分类。如KNN,SVM,ANN,DT

决策树

Independent variables				Dep. var
OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	FALSE	Don't Play
sunny	80	90	TRUE	Don't Play
overcast	83	78	FALSE	Play
rain	70	96	FALSE	Play
rain	68	80	FALSE	Play
rain	65	70	TRUE	Don't Play
overcast	64	65	TRUE	Play
sunny	72	95	FALSE	Don't Play
sunny	69	70	FALSE	Play
rain	75	80	FALSE	Play
sunny	75	70	TRUE	Play
overcast	72	90	TRUE	Play
overcast	81	75	FALSE	Play
rain	71	80	TRUE	Don't Play

树



DT算法思想

- 自顶向下的分治方式构造决策树
- 使用**Nominal属性**的数据(如果是**Numeric属性**，则需要进行离散化)，**递归选择测试属性来划分样本**
- 测试属性是根据**启发信息或是统计信息来选择**，如**信息增益**

DT算法流程

- 树以训练样本的**单个结点**开始
    - 如果样本都在同一类，则该结点为树叶，并用该类标记
    - 否则，选取**最有分类能力的属性**成为决策树的当前结点
- 根据分类属性将训练样本分为若干子集。对于每个子集，重复上述操作，递归形成决策树。**一旦一个属性出现在一个结点上，就不必再该结点的子代考虑它**
- 递归停止的条件
    - 结点的所有样本属于同一类
    - 没有剩余属性可以用来进一步划分样本
    - 对于某一分支，没有 **满足该分支条件的样本**，则以样本的多数类创建一个树叶

DT算法伪代码

算法：Generate_decision_tree。由数据分区*D*中的训练元组产生决策树。

输入：

- 数据分区*D*，训练元组和它们对应类标号的集合。
- *attribute_list*,候选属性的集合。
- *Attribute_selection_method*,一个确定“最好地”划分数据元组为个体类的分裂准则的过程。这个准则由分裂属性（*splitting_attribute*）和分裂点或划分子集组成。

输出：一棵决策树。

方法：

```
(1)  创建一个结点N;
(2)  if D中的元组都在同一类C中then
(3)    返回N作为叶结点,以类C标记;
(4) if attribut_list为空then
(5)    返回N作为叶结点，标记为D中的多数类;                          //多数表决
(6) 使用Attribute_selection_method(D, attribute_list),找出“最好的”splitting_criterion;
(7) 用splitting_criterion标记结点N;
(8)  if splitting_attribute是离散值的,并且允许多路划分then        //不限于二叉树
(9)     attribute_list←attribute_list-splitting_attribute; // 删除分裂属性
(10) for splitting_criterion的每个输出j
     //划分元组并对每个分区产生子树
(11)    设Dj是D中满足输出j的数据元组的集合;                       // 一个分区
(12)    if Dj为空then
(13)     加一个树叶到结点N,标记为D中的多数类;
(14)    else加一个由Generate_decision_tree(Dj,attribute_list)返回的结点到N;
     endfor
(15) 返回N;
```

属性选择度量——如何选择最优分类能力的属性

信息增益IG——ID3

见Chapter2 维归约

信息增益(ID3)

- 选择具有最高信息增益的属性
- 令 p_i 为D中的任一元组属于类 C_i概率, 估计为 $|C_{i,D}|/|D|$
- D中元组分类需要的信息熵(entropy)：

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- (利用 A 分裂D 为v个部分后)分类D 需要的信息为:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- 以属性A分枝得到的信息增益

$$Gain(A) = Info(D) - Info_A(D)$$

👉 缺点

- 信息增益倾向于有**大量不同取值的属性**——即划分更细，更纯
    - 会出现极端的情况，即划分的每个子集都只有一个样本，此时每个类的信息熵为0
- 用信息增益率来解决这一问题

信息增益率——C4.5

- 引入了,$SplitInfo_A(D)$,

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times log_2(\frac{|D_j|}{|D|})$$

- 信息增益率的公式

$$Gain_Ratio(A) = \frac{Gain(A)}{SplitInfo_A(D)}$$

👉 选择最大增益率的属性为分裂属性

Gini指标——CART

👉 Gini指标**度量数据元组的不纯度**

👉 定义

$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$ 数据集D基于属性A分裂为子集D_1，D_2 $gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$ $\Delta gini(A) = gini(D) - gini_A(D)$

👉 以不纯度最小的即,$gini_A(D)$,最小的A来作为分裂属性 也就是,$\Delta gini(A)$,最大的(不纯度减小最大的)

过拟合

👉 监督学习普遍存在的问题

👉 过拟合的原因

- 训练样本只是真实模型下的一个抽样集
- 模型泛化能力不强

👉 解决的策略

- 增加样本集，去除噪声
- 降低模型复杂度
- Train-Validation-Test
- 加入正则项等

树剪枝

👉 设定决策树的最大高度来限制树的生长

👉 设定每个结点必须包含的**最少记录数**，当结点中记录的个数小于这个数值时要停止分割

👉 剪枝的策略

- 先剪枝——提前终止树的构造
    - 选择一个合适的阈值很难
- 后剪枝——从完全生长的树中剪去树枝
    - 计算量代价大。但对于小样本的情况，后剪枝的方法较为优秀

从DT提取规则

👉 根据每一条路径可以创建一个规则，以**IF-THEN**形式表示

- 路径上的每个**属性-值**对形成规则前件 即IF部分
- 叶结点包含类预测，形成规则后件 即THEN部分

KNN

KNN算法思想

- 根据**距离函数**计算待分类样本和每个训练样本的距离(**作为相似度**)
- 选择与待分类样本距离最小的K个样本作为X的K个最近邻
- 以X的K个最近邻中的**大多数所属的类别**作为X的类别

KNN优缺

- 优点
    - 简单，易于理解，无需训练

    - 准确度一般较高
    - 特别适合**多标签问题(对象有多个类别的标签)**，KNN比SVM表现的好
- 缺点
    - 懒惰算法，对测试样本分类时计算量大，内存开销大
    - 当样本不平衡时，效果不好
    - 可解释性较差，无法给出决策树那样的规则

KNN常见问题

- K值设定
- 类别的判断方式
    - 投票法没有考虑近邻的距离的远近，距离更近的近邻更应该决定最终的分类，**加权投票法更恰当**
- 距离度量方式的选择
    - 当变量数变多(即高维)，欧式距离的区分能力变差
    - 值域越大的变量会在距离计算时占据主导，**应先对变量进行标准化**
- 训练样本的参考原则
    - 减少训练集的大小
    - 通过**聚类**，将聚类产生的**中心点**作为新的训练样本

朴素贝叶斯

贝叶斯定理

$$给定训练样本集X，假设H的先验概率P(H|X)有以下的条件概率公式\ P(H|X)=\frac{P(X|H)P(H)}{P(X)}$$

👉 其中$P(X \mid H)$为后验概率

👉 假设Y是类变量，X是特征向量,$X=(x_1,x_2,\cdots,x_n)$,,有

$$P(y|X)=\frac{P(X|y)P(y)}{P(X)}$$

👉 朴素贝叶斯基于一个十分重要的假设——**类条件独立假设**，X的n个分量相互独立，

👉 故$P(x_1,x_2,\cdots,x_n \mid y)$ 可以根据相互独立的假设表示为

$$P(x_1 \mid y) \times P(x_2 \mid y) \times \cdots \times P(x_n \mid y)$$

👉 故原条件概率式可表示为

$$P(y|x_1,x_2,\cdots,x_n)=\frac{P(x_1|y)P(x_2|y)\cdots P(x_n|y)P(y)}{P(x_1)P(x_2)\cdots P(x_n)}=\frac{P(y)\prod_{i=1}^{n}P(x_i|y)}{P(x_1)P(x_2)\cdots P(x_n)}$$

👉 只需要学习条件概率分布即可

👉 不需要计算,$P(X)$, 因为对于一条新数据 P(X)的值不会发生变化 在计算先验概率时不会有影响 故可以得到朴素贝叶斯分类器 👇

朴素贝叶斯分类器

👉 ,$\hat{y}=arg\max_y P(y)\prod_{i=1}^{n}P(x_i|y)$,

SVM(支持向量机)

👉 基于统计学习理论的**VC维理论**和**结构风险最小原理**

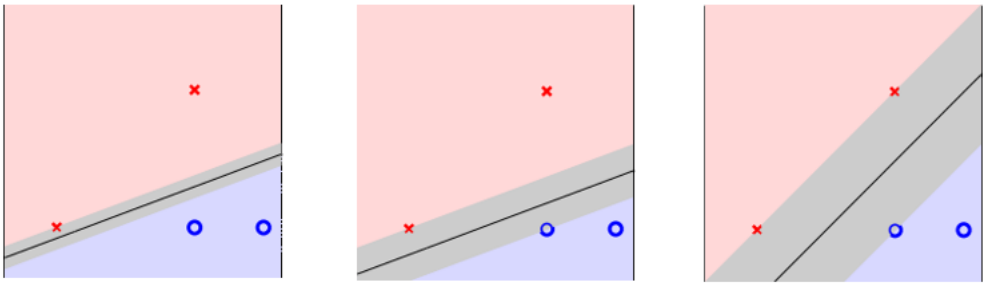
- VC维
    - 对函数类的一种度量，可认为问题的复杂程度，**VC维越高，一个问题就越复杂**
- 结构风险
    - 经验风险,$R_{emp}(w)$,
        - 分类器在样本数据上的分类结果与真实结果之间的差值
    - 置信风险,,$\Phi(h,n)$
        - 样本数量
            - 给定的样本数量越大，学习结果越有可能正确，置信风险越小
        - 分类函数的VC维
            - VC维越大，泛化能力越差
    - 寻求经验风险和置信风险的**和最小**，即结构风险最小,$R(w) \leq R_{emp}(w)+\Phi(h,n)$,

👉 线性可分和不可分

- 若一个**线性函数**能够将样本完全正确的分开，则称数据是线性可分的，否则则称非线性可分
- 线性函数
    - 在一维空间是**一个点**，在二维空间是**一条直线**，在三维空间是**一个平面**
    - 线性函数有一个统一的名称——**超平面(Hyper Plane)**

SVM线性分类器

SVM基本思想——最大间隔化



- 最大间隔化
  - 使数据能够完全的分割开并使**支持向量所在超平面**与**分类面的几何间隔最大化**，所以可以将此问题转化为**最优化问题**
- 分类面的函数表示为，$g(x)=wx+b$,
  - 此处的x是样本的向量表示
  - 此表达式不限于二维，若为n维，则w为n维向量
  - 称，$wx+b=0$, 为分类面

H是分类面，而H1和H2是平行于H，且过离H最近的两类样本的直线，H1与H，H2与H之间的距离就是几何间隔。

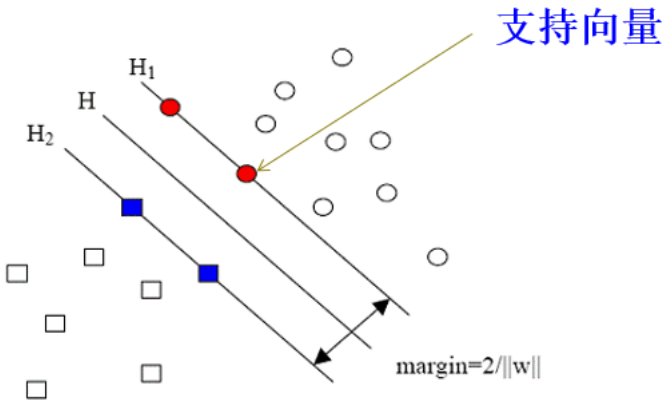
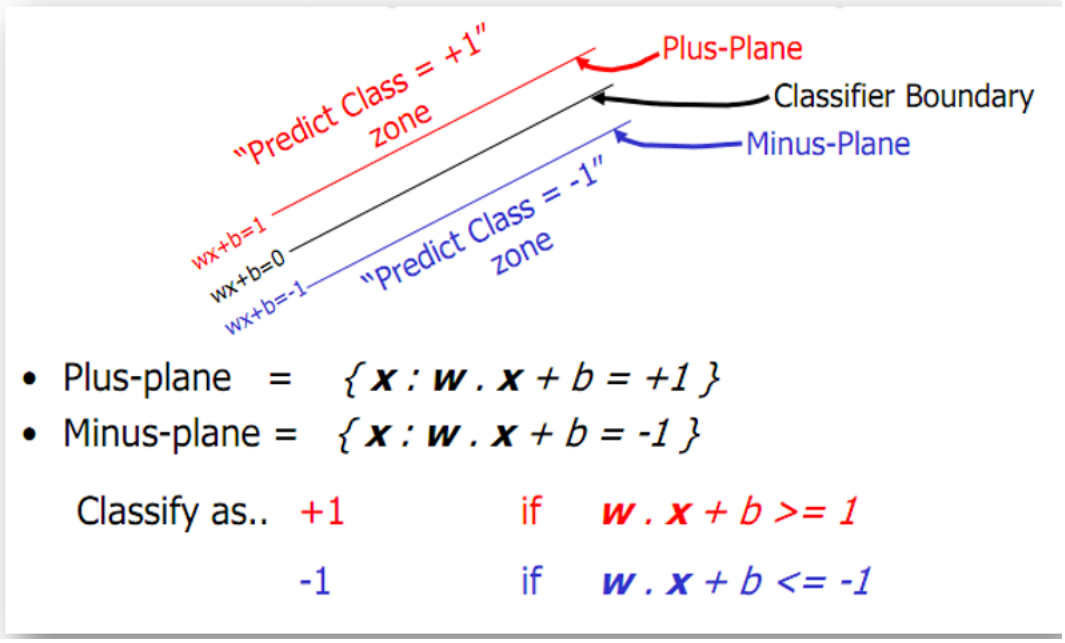


图2 线性可分情况下的最优分类线

- ，H_1，H_2，上的点组成的下向量都称为**支持向量**

转化为最优化问题



- 对于在Plus-Plane内的数据(有,$wx+b\geq 1,$)，归为1类，对于在Minus-Plane内的数据(有,$wx+b\leq -1,$)，归为,-1类,

对于n维空间中的某个点x_i,对应的类的值为y_i,若能有 $y_i(w^Tx_i+b)\geq 1$, 因为在minus-plane中$y_i=-1$,虽然$wx+b\leq -1$，乘上y_i仍为$y_i(w^Tx_i+b)\geq 1$ 说明该分类面

👉 如何最大化几何间隔 即plus-plane(minus-plane)到分类面的距离,$max\frac{1}{||w||}$,

- 将,$max\frac{1}{||w||}$,作为目标函数，,$y_i(w^Tx_i+b)\geq 1$,作为约束条件，问题变为有约束的优化问题
- 将目标函数变为,$min\frac{1}{2}||w||^2$,

👉 如何求解该**凸二次规划问题**——SMO算法

SVM非线性分类器

👉 **核函数**，将数据映射到高维空间使其变为线性可分，但是运算还是在低维进行向量内积运算

见Blog

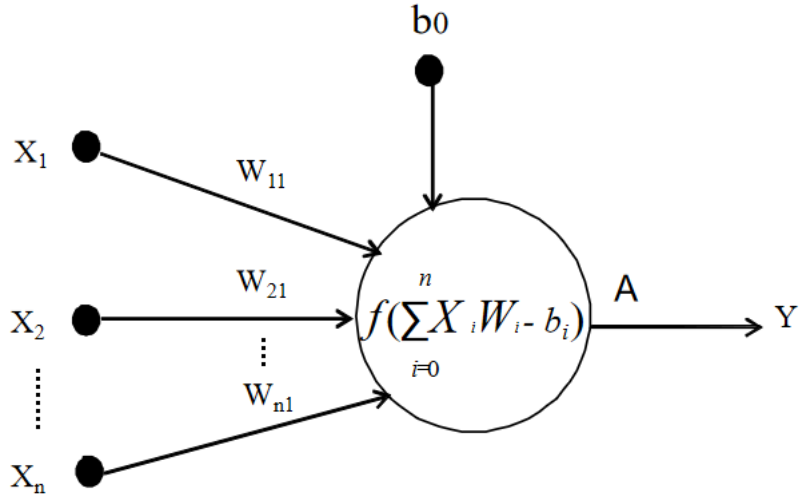
ANN

👉 人工神经网络

- 将大量功能简单的神经元通过一定的拓扑结构组织起来，构成群体并行分布式处理的计算结构

人工神经元模型

👉 是一个多输入单输出的非线性阈值器件



ANN的学习技术

- 权值修正学派
  - 神经网络学习过程即为不断调整网络的连接权重，以获得期望的输出

- 典型的权值修正方法
  - 相关学习
  - 误差修正学习
    - θ学习规则

（1）选择一组初始权值W_{ji} (0);

（2）计算某一输入模式对应的实际输出与期望输出的误差；

（3）更新权值：

$$W_{ji}(t+1) = W_{ji}(t) + \eta[\, d_j - y_j(t)\,]\, x_i(t)$$

式中，η为学习因子；d_j、y_j分别表示第j个神经元的期望输出与实际输出；x_i为第j个神经元的输入。

（4）返回步骤（2），直到对所有训练模式网络输出均能满足要求。

    - 感知机学习
    - BP网络学习

感知机学习

👉 典型的监督学习

👉 权值更新 采用θ学习规则

BP网络

👉 通过修改神经元连接的权重，使网络输出层的**误差平方和最小**

- 对误差函数中的权值求偏导，即其梯度方向
- 求梯度之后，乘上学习率，来更新权值

集成学习

👉 构建并结合多个学习器来完成任务

- 通常可以获得比单一学习器更显著的泛化性能
- 基学习器既要有**准确性**又要有**多样性**(即差异性)

Bagging

Bagging步骤

👉 对于包含m个样本的数据集

- 随机抽取一个样本放入采样集，**再把该样本放回初始数据集**
- 经过m次随机采样，得到含m个样本的采样集
- 重复T次，得到T个采样集
- 基于每一个采样集训练出一个基学习器，再将基学习器结合

输入：训练集 $D = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$，$y \in \{-1, +1\}$；基学习算法 $\mathfrak{L}$；训练轮数 T。

过程：

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

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

end for

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

随机森林RF

👉 与传统的DT训练相比，进行了一个改造——**随机属性选择**

- 在选取某个结点的分裂属性时，不是在所有属性选取，而是在候选属性集合中抽取一个子集，在其中选取分裂属性

Boosting

👉 将弱学习器提升为强学习器的方法

- 先训练出一个基学习器，调整样本——**提高分类错误的样本的权重**
- 根据调整后的样本，训练下一个基学习器
- 将训练好的所有基学习器**加权结合**

AdaBoost 算法

输入：训练集 $D=\{(\boldsymbol{x}_1,y_1),\dots,(\boldsymbol{x}_m,y_m)\}$，$y\in\{-1,+1\}$；基学习算法 $\mathfrak{L}$；训练轮数 T。

过程：

$\mathcal{D}_1(\boldsymbol{x})=\frac{1}{m}$

for $t=1,2,\dots,T$ **do**

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

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

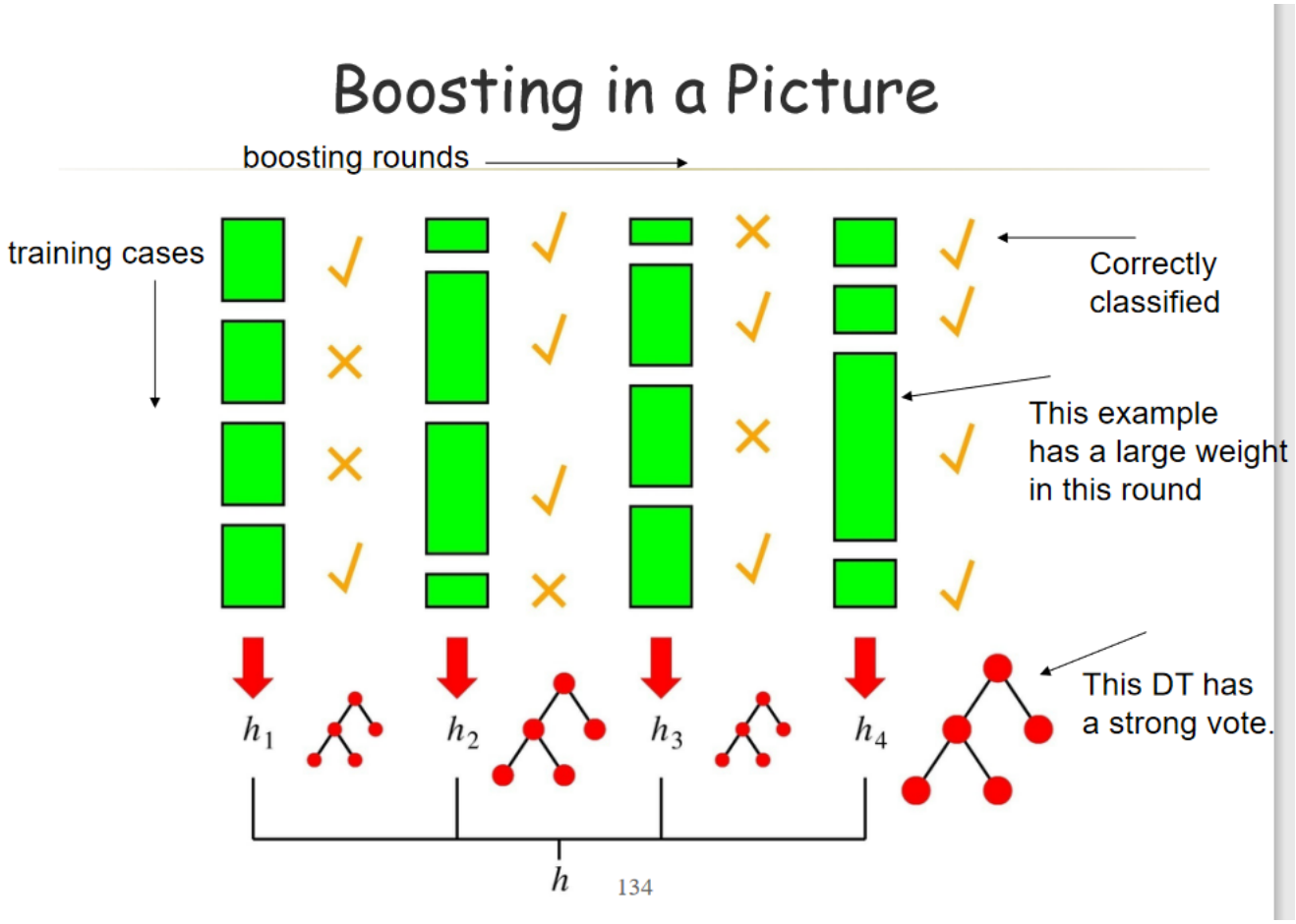
if $\epsilon_t>0.5$ **then break** （检测是否优于随机猜测）

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

$\mathcal{D}_{t+1}(\boldsymbol{x})=\frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t}\times\begin{cases}\exp(-\alpha_t), & h_t(\boldsymbol{x})=\boldsymbol{f}(\boldsymbol{x})\\ \exp(\alpha_t), & h_t(\boldsymbol{x})\neq\boldsymbol{f}(\boldsymbol{x})\end{cases}=\frac{\mathcal{D}_t(\boldsymbol{x})\exp\big(-\alpha_t\boldsymbol{f}(\boldsymbol{x})h_t(\boldsymbol{x})\big)}{Z_t}$ （Z_t是规范化因子，确保$\mathcal{D}_{t+1}$是一个分布）

end for

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



Stacking

👉 主要思想——将初级学习器的输出结果作为次级学习器的输入

👉 注意：原始特征也要输入到次级学习器，因为初级学习器的输出结果只是一个**补充信息**

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

过程：

for $t=1,2,\dots,T$ **do**

$h_t=\mathfrak{L}_t(D)$

end for

$D'=\emptyset$

for $i=1,2,\dots,m$ **do**

for $t=1,2,\dots,T$ **do**

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

end for

$D'=D'\bigcup\left((z_{i1},\dots,z_{iT}),y_i\right)$

end for

$h'=\mathfrak{L}(D')$

输出：$H(\boldsymbol{x})=h'\left(h_1(\boldsymbol{x}),\dots,h_T(\boldsymbol{x})\right)$

三种集成学习的比较

- Bagging
  - 通过**随机抽样**增加数据和属性的多样性
- Boosting
  - 错误驱动
- Stacking
  - 从底层特征向高层特征的映射

分类评价

混淆矩阵 Confusion Matrix

A\Pr	C	¬C	
C	TP	FN	P
¬C	FP	TN	N
	P’	N’	All

A：实际的类

Pr：预测的类

TP：真正例，被正确分类的正样本

TN：真负例，被正确分类的负样本

FP：假正例，被错误分类的负样本

FN：假负例，被错误分类的正样本

P：正样本数；N：负样本数

P’：被分类器标记为正的样本数

N’：被分类器标记为负的样本数

评价指标

准确度 Accuracy

$$Accuracy=\frac{TP+TN}{All}$$

误差率 Error Rate

$$ER=\frac{FP+FN}{All}$$

精度 Precision

👉 被分类器标记为正类的样本实际为正类的比例

$$precision = \frac{TP}{TP+FP}$$

召回率 Recall

👉 正样本被标记为正的比例

$$recall = \frac{TP}{TP+FN}$$

F值

👉 F_1,: 精度和召回率的**调和平均值**

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$

👉 F_β,: 精度和召回率的加权量(,$\beta = 2, or, 0.5,$)

$$F_\beta = \frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$$

类分布不平衡——灵敏度 特效性

👉 **One class may be rare**

👉 灵敏度(Sensitivity)——正样本被标记为正的比例(即为召回率)

$$Sensitivity = \frac{TP}{TP+FN}$$

👉 特效性(Specificity)——负样本被标记为负的比例

$$Specificity = \frac{TN}{FP+TN}$$