哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

分类算法

李春山

2020年10月22日

主要内容

- **决策树分类**
- **贝叶斯分类**
- **基于规则分类**
- **基于实例分类**
- **集成分类**
- **评价方法**
- **小结**

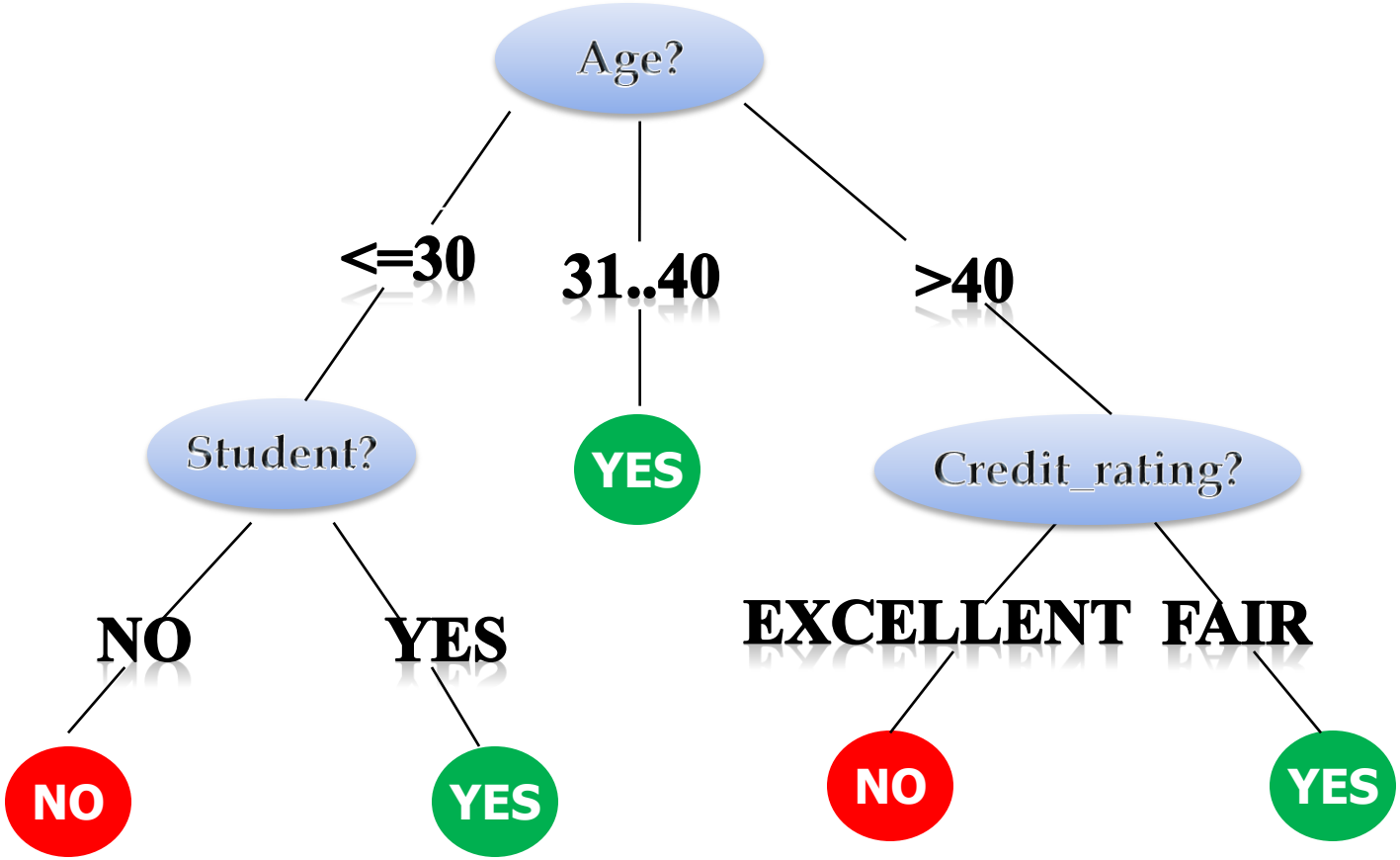
决策树分类器

决策树分类-用判定树归纳分类

Quinlan's ID3算法的例子 (buying computers)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

决策树分类-输出: “buys_computer”的决策树



决策树分类算法

- **判定归纳树算法**
  - 自顶向下的分治方式构造判定树
  - 树以代表训练样本的单个根节点开始
  - 使用分类属性（如果是连续属性，则需先进行离散化）
  - 递归的通过选择相应的***测试属性***，来划分样本，一旦一个属性出现在一个节点上，就不在该节点的任何后代上出现
  - 测试属性是根据某种启发信息或者是统计信息来进行选择（如：信息增益）
- **递归划分步骤停止的条件**
  - 给定节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—使用投票表决
  - 没有剩余的样本

决策树分类-属性选择度量: 信息增益(ID3/C4.5)

- 选择具有最高信息增益（或最大熵压缩）的属性
- 假设p_i为数据集D中任意一个记录属于类C_i的概率，
  $p_i = |C_{i,D}|/|D|$
- 带有类标的数据集D的Expected information (entropy)：

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

- 使用属性A将D划分成V的部分的信息需求:

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

- 是用属性A分裂数据集后的Information gained

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

决策树分类-属性选择: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0)$$

$$\frac{5}{14}I(2,3) + \frac{5}{14}I(3,2) = 0.694$$

means "age <=30" has 5 out of 14 samples, with 2 yes and 3 no. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

决策树分类-连续值属性的 Information-Gain

- **给定一个连续值属性A,决策树需要计算A的最佳分裂点**
  - 使用增序来排列属性A的值
  - 典型地，将两个相邻值的中点作为可能的分裂点
    - $(a_i+a_{i+1})/2$ 是 a_i 和 a_{i+1} 的中点
  - 属性A的 *minimum expected information* 点将会被选择成为分裂点
- **分裂:**
  - D1 是D的子集，其中的记录满足 A≤ split-point, and D2 中的记录满足 A > split-point

决策树分类-属性选择的增益率 (C4.5)

- 信息增益度量明显的偏向于选择具有多值的属性
- C4.5 (ID3的扩展) 使用增益率来克服这个问题 (标准化信息增益)

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

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex. $SplitInfo_A(D) = -\frac{4}{14} \times \log_2(\frac{4}{14}) - \frac{6}{14} \times \log_2(\frac{6}{14}) - \frac{4}{14} \times \log_2(\frac{4}{14}) = 0.926$
  - gain_ratio(income) = 0.029/0.926 = 0.031
- 具有最大增益率的属性将被选择成为分裂属性

决策树分类- Gini index (CART, IBM IntelligentMiner)

- 若一个数据集D包含N个类别的样本, **gini index**, $gini(D)$ 将被定义为

$$gini(D)=\sum_{j=1}^{n} p_j(1-p_j)=1-\sum_{j=1}^{n} p_j^2$$

其中 p_j 是D中类j的相关概率

- 如果数据集D被属性A分裂成两个子集D_1 和D_2, 那么 **gini index** $gini(D)$ 将被定义为

$$gini_A(D)=\frac{|D_1|}{|D|}gini(D_1)+\frac{|D_2|}{|D|}gini(D_2)$$

- 减少Impurity:

$$\Delta gini(A)=gini(D)-gini_A(D)$$

- 具有最小 $gini_{split}(D)$值 (或者最大 **impurity**) 是选择的分裂点 (需要对属性的每个分裂点进行枚举处理)

决策树分类- Gini index (CART, IBM IntelligentMiner)

- **Ex. 数据集D有9个记录 buys_computer = “yes” 同时5个记录 “no”**

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- **接收属性income 划分 10个样本给子集 D_1: {low, medium}， 4 个样本给 D_2**

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_1)$$

$$= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right)$$

$$= 0.450$$

$$= Gini_{income \in \{high\}}(D)$$

同时 $gini_{\{medium,high\}}$ 的值为 0.30 因此我们选择分裂点为low

- **所有属性被假设为是连续值**
- **有时需要其他的工具来产生可能的分裂点, e.g., clustering,**
- **可以被类别属性修改**

决策树分类-属性选择方法的对比

- 一般来说三种方法都能得到好的分类结果
  - Information gain:
    - 选择多值的属性作为分裂属性
  - Gain ratio:
    - 分裂不平衡的属性，导致其中的一个划分远小于其他划分
  - Gini index:
    - 二元分割
    - 产生大小和纯度相似的属性分割

决策树分类-其他属性选择方法

- **CHAID: 使用 χ^2 test 来确定最佳分割**
- **C-SEP: 在一些数据中性能优于info. gain and gini index**
- **G-statistics: 使用近似χ^2 分布来确定分割点**
- **MDL (Minimal Description Length) principle (i.e.,优先选择最简单的解):**
  - 最优的决策树是使用最少比特位来（1）编码这棵树（2）编码树的异常
- **那个是最好的?**
  - 大都产生好的结果, 没有效果十分优于其他算法的。

决策树分类-过拟合和树剪枝

- **过拟合：决策树出现过分适应训练数据的问题**
  - 由于数据中的噪声和孤立点，许多分枝反应的是训练数据中的异常
  - 对新样本的判定很不精确
- **避免过拟合的2种方式**
  - 预剪枝：通过提前停止树的构造——如果在一个节点划分样本将导致低于预定义临界值的分裂（e.g. 使用信息增益度量）
    - 选择一个合适的临界值往往很困难
  - 后剪枝：由“完全生长”的树剪去分枝——对于树中的每个非树叶节点，计算该节点上的子树被剪枝可能出现的期望错误率
    - 使用一个独立的测试集来评估每颗树的准确率，就能得到具有最小期望错误率的判定树

决策树分类-基础决策树的扩展方法

- **处理连续值属性**
  - 动态定义将连续属性处理为若干离散的分段
- **处理缺失值**
  - 指定缺失值为最常使用的属性值
  - 指定每个可能值得概率
- **构建属性**
  - 根据已有的稀疏属性创造新的属性, 从而减少了碎片， 重复和冗余

决策树分类-决策树算法的优点

- 相对与其他分类算法，决策树训练速度快
- 产生简单、好理解的分类规则
- 可以直接使用数据库中的SQL查询

决策树分类-决策树算法的扩展

- **SLIQ (EDBT'96 — Mehta et al.)**
  - 为每个属性建立索引，只有类标列和当前属性列驻留内存
- **SPRINT (VLDB'96 — J. Shafer et al.)**
  - 创建属性列的数据结构
- **RainForest (Optional)(VLDB'98 — Gehrke, Ramakrishnan & Ganti)**
  - 建立 AVC-list (属性, 值, 类标)
- **BOAT (Optional) (PODS'99 — Gehrke, Ganti, Ramakrishnan & Loh)**
  - 使用bootstrapping 创建小样本集

贝叶斯分类器

贝叶斯分类-贝叶斯分类技术: Why?

- 基于统计的分类器: 采用概率学概率预测, *i.e.*, 预测类成员的概率
- 基础: 贝叶斯概率理论
- 性能: 最简单的贝叶斯分类器, 朴素贝叶斯分类器, 与决策树和神经网络性能相当
- 增量: 每个训练样本都可以增量的增加/降低类别假设的概率 – 先验知识和观测变量的结合 |
- 标准:可以提供与其他算法对比的决策标准

贝叶斯分类-贝叶斯理论基础

- **假设 X 是数据样本 (“*evidence*”): 其类标未知**
- **假设 H 是一个假设，指出X属于类标C**
- **则X的类标由条件概率P(H|X)给出**
- **P(H) (*先验概率*), 初始概率**
- **P(X)是样本被观测到的概率**
- **P(X|H) (后验概率)表示给定一个类标, 随机变量X出现的概率**
  - E.g., **X** 被标记为购买了电脑, P(**X**|H) 表示所有购买了电脑的人群中 X 出现的概率。X是 31..40, medium income

贝叶斯分类-贝叶斯理论基础

- 给定训练数据X, *假设* H的后验概率, P(H|X) *被定义为:*

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})}$$

- 通常, 该工作可以理解为

posteriori = likelihood x prior/evidence

- 当且仅当$P(C_i|X)$在所有的$P(C_k|X)$具有最大值，其中k=1,C，那么我们判断X 属于 C_i
- 运行前提: 许多观测变量的先验概率, 较高的计算成本

贝叶斯分类-朴素贝叶斯分类器

- 假设D包含若干数据元组的训练集以及这些数据的类标信息，其中每个数据元组都是一个N维属性向量 $\mathbf{X} = (x_1, x_2, \ldots, x_n)$
- 假设存在 m 个类 $C_1, C_2, \ldots, C_m$.
- 分类过程就是找到最大后验概率的过程, i.e., 最大的$P(C_i|\mathbf{X})$
- 最大后验概率将来自 Bayes' theorem

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

$$P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$$

- 因为对于每个类P(X)都是常量, 所以上述公式仅需最大化

贝叶斯分类-朴素贝叶斯分类器的原理

- 假设: 属性之间是条件独立的

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times ... \times P(x_n|C_i)$$

- 消减计算成本，仅需要统计类别的属性
- 若属性 $\mathbf{A_k}$ 是类别属性（离散型），$\mathbf{P(x_k|C_i)}$ 是属性$\mathbf{A_k}$ 的第k维值等于$\mathbf{x_k}$ 在类别 $\mathbf{C_i}$ 中出现的概率
- 若$\mathbf{A_k}$ 是连续属性, $\mathbf{P(x_k|C_i)}$ 通常有一个高斯分布（均值$\boldsymbol{\mu}$，方差$\boldsymbol{\sigma}$

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

贝叶斯分类-朴素贝叶斯: 训练数据集

类别:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

数据样本
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

age	income	student	credit_rating	com
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

贝叶斯分类-朴素贝叶斯分类器举例

- **$P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643**

  P(buys_computer = "no") = 5/14= 0.357

- 在每一类中计算 **$P(X|C_i)$**

  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222

  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6

  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444

  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4

  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667

  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2

  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667

  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

贝叶斯分类-朴素贝叶斯分类器举例

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X | buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044

P(X | buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X | buys_computer = "yes") * P(buys_computer = "yes") = 0.028

P(X | buys_computer = "no") * P(buys_computer = "no") = 0.007

因此，X 被判别给类 ("buys_computer = yes")

贝叶斯分类-避免0概率问题

- **朴素贝叶斯分类器要求每一个条件概率都是非0的，否则总体的似然概率也是0**

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

Ex. 假设数据集中有1000 元组, income=low (0), income= medium (990), and income = high (10),

- **使用 Laplacian 平滑 (or Laplacian 估计)**
  - 对于每一项income的统计值都加1

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - 平滑后的“正确”概率 与原始概率相似，但避免了0概率问题。

贝叶斯分类-讨论

- **优势**
  - 易于实现
  - 绝大情况下效果可接受
- **劣势**
  - 假设: 属性之间的独立性， 导致了精度的降低
  - 实际上, 各个变量之间是相关的
    - E.g., 病人: Profile: 年龄, 家族病史, etc.
      症状: 发热, 咳嗽 etc., 疾病: 肺癌, 糖尿病, etc.
    - 这些变量之间的依赖性，朴素贝叶斯是处理不了的
- **处理依赖性的方法?**
  - Bayesian Belief Networks

基于规则的分类

基于规则分类器

- **通过一系列的“if…then…” 规则来分类**
- **Rule:** **(*条件*) → *y***
  - 条件
    - *条件是若干属性的连接*
    - *y* 是类标信息
  - *LHS*: 规则前提或者条件
  - *RHS*: 规则结论
  - 分类规则示例:
    - (Blood Type=Warm) ∧ (Lay Eggs=Yes) → Birds
    - (Taxable Income < 50K) ∧ (Refund=Yes) → Evade=No

基于规则分类器(示例)

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

基于规则分类器(示例)

R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds

R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes

R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals

R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles

R5: (Live in Water = sometimes) → Amphibians

5.5 基于规则分类器-应用

- 如果规则*r* 覆盖 样本**x**，那么**x**的属性必须满足规则的前提条件

R1: (Give Birth = no) ∧ (Can Fly = yes) → Birds

R2: (Give Birth = no) ∧ (Live in Water = yes) → Fishes

R3: (Give Birth = yes) ∧ (Blood Type = warm) → Mammals

R4: (Give Birth = no) ∧ (Can Fly = no) → Reptiles

R5: (Live in Water = sometimes) → Amphibians

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
hawk	warm	no	yes	no	?
grizzly bear	warm	yes	no	no	?

The rule R1 covers a hawk => Bird

The rule R3 covers the grizzly bear => Mammal

基于规则分类器-规则覆盖和精度

- **规则覆盖率:**
  - 满足规则前提条件的记录在整个数据集中的比率
- **规则准确率:**
  - 在规则的前提条件下，能够得到规则结果的记录的比率

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	**Single**	125K	**No**
2	No	Married	100K	**No**
3	No	**Single**	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	**Single**	85K	**Yes**
9	No	Married	75K	**No**
10	No	**Single**	90K	**Yes**

(Status=Single) → No

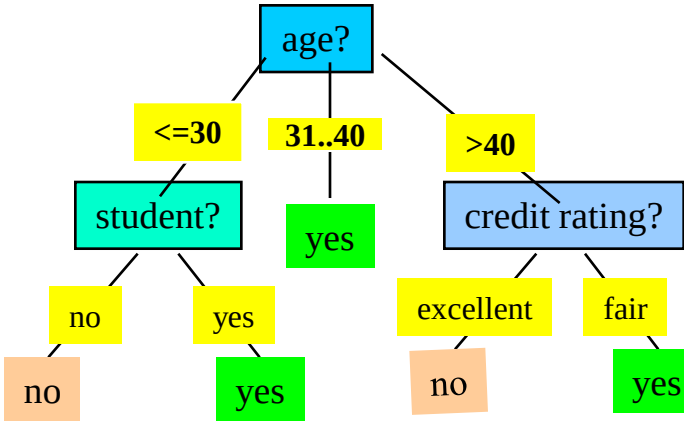
Coverage = 40%, Accuracy = 50%

基于规则分类器-建立分类规则

- 直接方法
  - 从数据集中直接抽取规则
  - e.g.: RIPPER, CN2, Holte's 1R
- 间接方法:
  - 从其他分类模型中抽取规则 (e.g. decision trees, neural networks, etc).
  - e.g: C4.5rules

基于规则分类器-从决策树中抽取规则

- 规则表达比树结构更易让人理解
- 每条从根节点到叶子节点的路径就是一条规则
- 连接路径上的每个节点作为规则的条件，叶节点是规则的结果
- 规则是互斥的，并且是具体的



- **Example: 在*buys_computer* 决策树上抽取规则**

IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*

IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*

IF *age* = mid-age THEN *buys_computer* = *yes*

IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *yes*

IF *age* = young AND *credit_rating* = *fair* THEN *buys_computer* = *no*

基于规则分类器-从训练数据中抽取规则

- **连续覆盖算法: 从训练数据中直接抽取规则**
- **典型地算法: FOIL, AQ, CN2, RIPPER**
- **连续地学习规则, 给定某一个类，规则将被从这个类中的数据抽取，同时要避免规则覆盖其他类的数据。**
- **Steps:**
  - 一次只学习一条规则
  - 当一条规则被学到, 被这条规则所覆盖的元组将被删除
  - 一直重复上述学习过程，直到满足终止条件 e.g., 没有更多的训练样本或者规则的质量低于预先设定的阈值
- **Tips：决策树推到同时要学习多个规则**

基于规则分类器-基于规则的分类算法的优势

- 类似决策树，分类数据很直观
- 易于解释
- 易于产生
- 能快速分类新样本
- 分类性能与决策树相似

基于实例的分类

基于实例的分类器

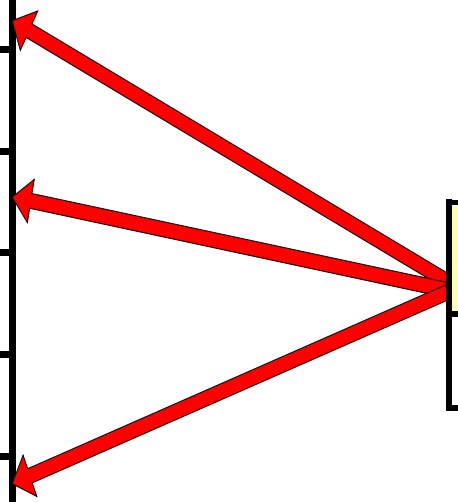
Set of Stored Cases

Atr1	……	AtrN	Class
			A
			B
			B
			C
			A
			C
			B

- 只存储训练实例
- 使用训练数据来分类未知的实例

Unseen Case

Atr1	……	AtrN

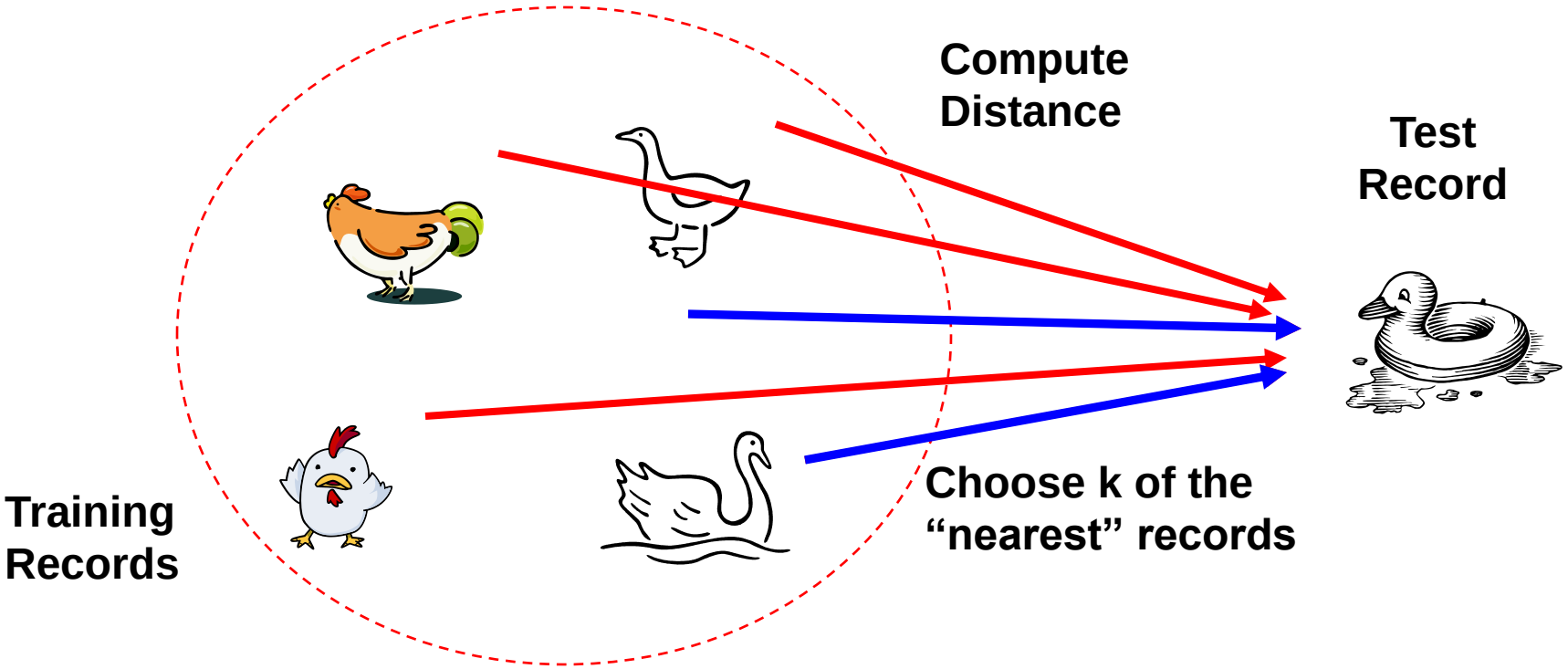


基于实例的分类器

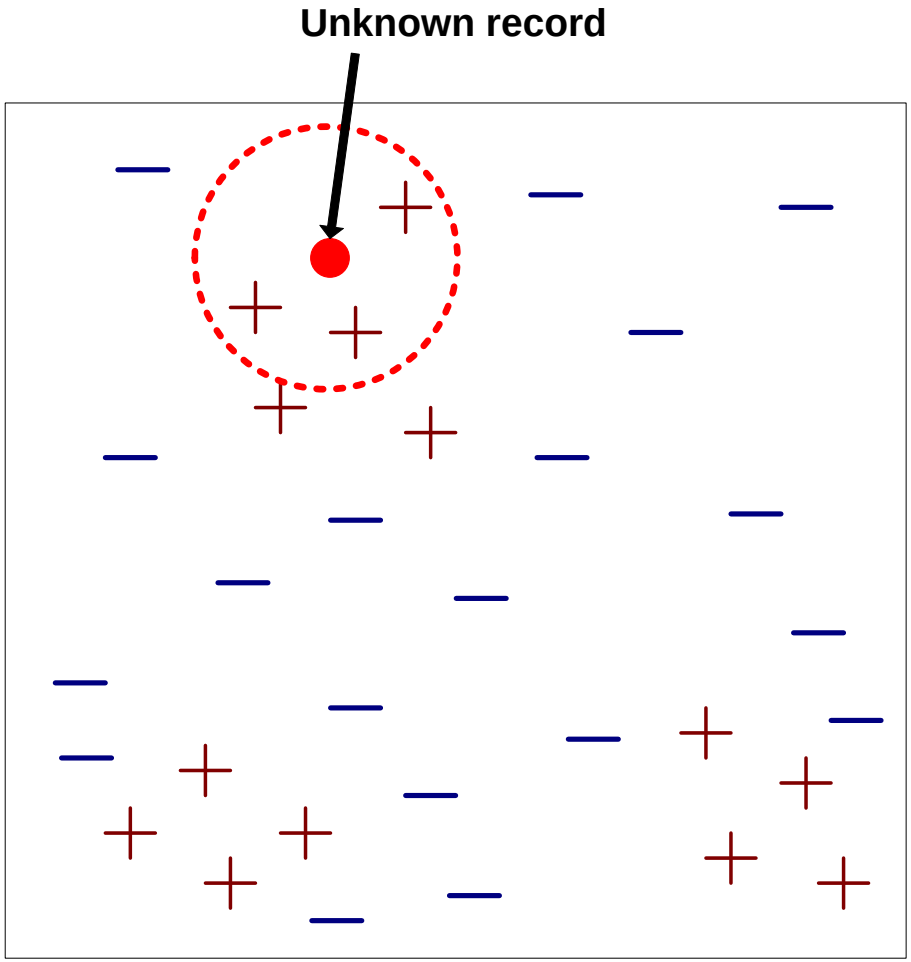
- **Examples:**
  - Rote-learner
    - 存储整个训练集，分类过程就是寻找训练集中与测试数据相同的数据样本的过程
  - Nearest neighbor
    - 使用k “closest” 点 (nearest neighbors) 来预测测试样本的类标

基于实例的分类器

- **Basic idea:**
  - If 测试对象走路像鸭子，叫声像鸭子，那么它可能就是鸭子

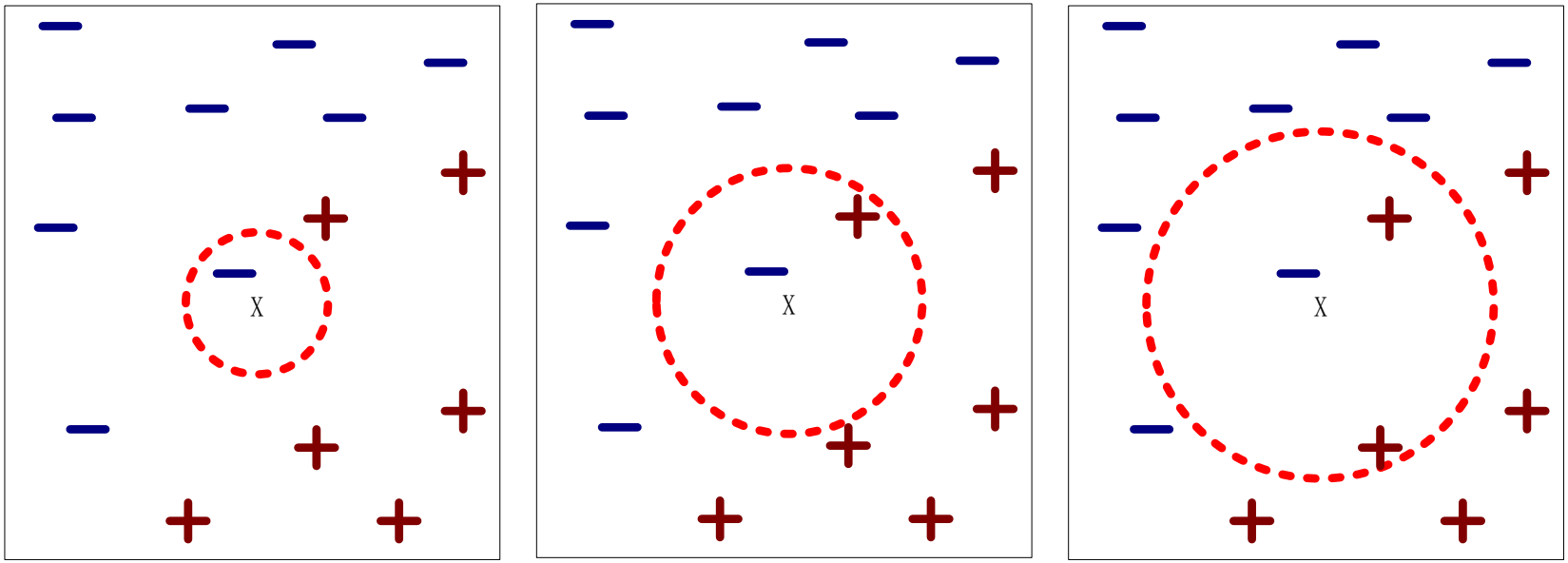


基于实例的分类器



- 分类器的充分条件
  - 实例的集合
  - 计算实例之间距离的度量
  - 最近邻的数目K
- 对未知样本的分类方法:
  - 计算与其他样本之间的距离
  - 找到K最近邻
  - 使用K最近邻的类标来预测未知样本的类标 (e.g., 投票法)

基于实例的分类器-最近邻的定义



(a) 1-nearest neighbor

(b) 2-nearest neighbor

(c) 3-nearest neighbor

记录X的k最近邻是与距离最近的k个数据点

基于实例的分类器-最近邻分类

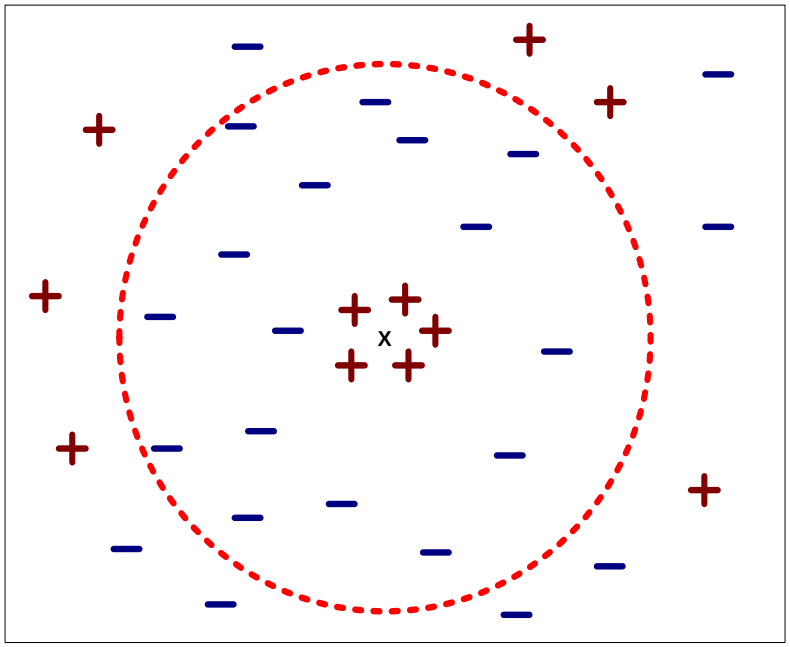
- **计算两点之间的距离:**
  - 欧式距离

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- **根据最近邻计算样本的类标**
  - 投票法，少数服从多数
  - 加权投票法，根据距离计算权重
    - 加权因子, $w = 1/d^2$

基于实例的分类器-最近邻分类

- **选择k的值:**
  - k太小，对噪声敏感
  - K太大, 会包含其他类的节点



基于实例的分类器-最近邻分类

- 范围的问题
  - 属性要被转换，以防距离度量没某一属性影响太大
  - Example:
    - 身高 1.5m to 1.8m
    - 体重 60kg to 90kg
    - 收入 $10K to $1M

基于实例的分类器-最近邻分类

- **欧式距离的问题:**
  - 高维数据
    - 维灾难
  - 可能产生违反常识的结果

1 1 1 1 1 1 1 1 1 1 1 0		1 0 0 0 0 0 0 0 0 0 0 0	
0 1 1 1 1 1 1 1 1 1 1 1	vs	0 0 0 0 0 0 0 0 0 0 0 1	
d = 1.4142		d = 1.4142	

- Solution: 标准化到单位长度

基于实例的分类器-最近邻分类

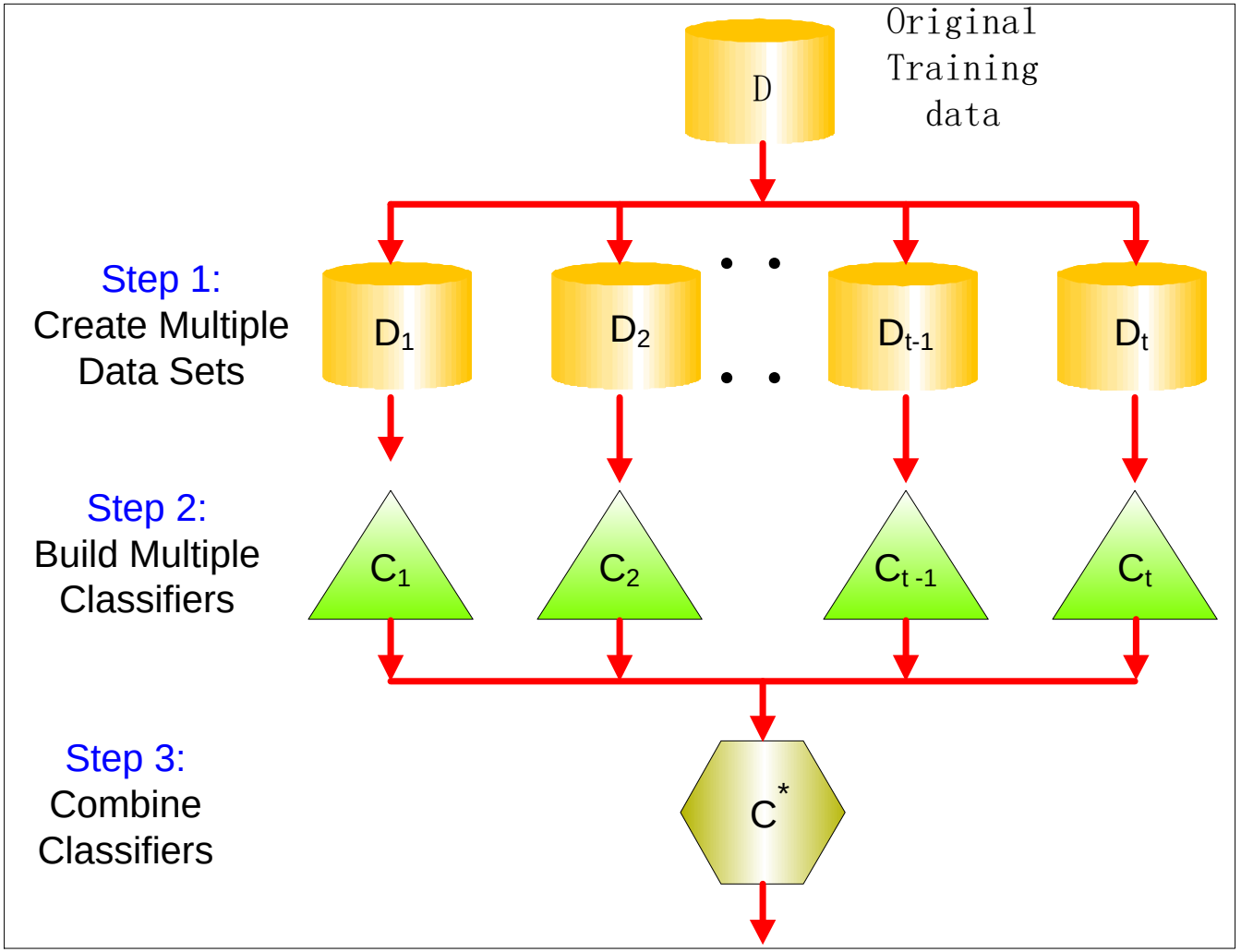
- **k-NN 分类器是懒惰学习器**
  - 不明确建立模型
  - 分类新样本消耗时间

集成分类

集成学习

- 使用训练数据来构建一组分类器
- 通过聚集多个分类器的分类结果来预测未知样本的类标

集成学习-基本思路


D
Original
Training
data
Step 1:
Create Multiple
Data Sets
D1
D2
Dt-1
Dt
Step 2:
Build Multiple
Classifiers
C1
C2
Ct -1
Ct
Step 3:
Combine
Classifiers
C*

集成学习-工作原理

- **假设有25个基本分类器**
  - 每个分类器的错误率是, $\varepsilon = 0.35$
  - 假设分类器之间是相互独立的
  - 集成学习误分的概率是：

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

集成学习-集成学习的方式

- 如何对分类器进行集成?
  - Bagging
  - Boosting

集成学习-Bagging

- 有放回的采样

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- 根据每次采样建立弱分类模型

- 每个样本在相同的概率下被选择

集成学习-Boosting

- 自动地修改训练集中数据没选择的概率，在迭代过程中动态的增加之前被错分样本被选中的概率（为什么？）。
  - 首先，所有N个样本被给予相同的权重
  - 与bagging不同之处在于, 样本权重在每轮 boosting之后被修改

集成学习-Boosting (Schapire 1989)

- *在D中无放回的随机选择$n_1 < n$个样本产生样本子集 D_1*
  - 训练弱分类器 C_1
- *在D中无放回的随机选择$n_2 < n$个样本产生样本子集 D_2，其中有一半的样本是分类器C_1，误分的样本*
  - 训练弱分类器 C_2
- *使用C_1 and C_2 分类结果不同的样本*
  - 训练弱分类器 C_3
- *通过投票来集成最终的分类器*

集成学习-Adaboost (Freund and Schapire, 1997)

- **给定一组具有d个类别的样本 $(X_1, y_1), \ldots, (X_d, y_d)$**
- **初始化, 将所有样本的权设置为 (1/d)**
- **通过k轮迭代产生k个分类器. 在每一轮i,**
  - 在D中有放回的采样，形成一个与D大小相同的数据集D_i
  - 每个样本基于它的权重被采样
  - 在 D_i中训练分类器M_i
  - 在D_i 中计算分类器的误分率
  - 若样本被误分, 其权重会增加, 反之. 权重下降

集成学习-Adaboost (Freund and Schapire, 1997)

- 误分率: $\mathbf{err(X_j)}$ 表示对样本的错误分类。分类器 $\mathbf{M_i}$ 误分率 是所有误分样本权重之和

$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

- 弱分类器$\mathbf{M_i}$ 的投票权重为

$$\log \frac{1-error(M_i)}{error(M_i)}$$

集成学习- Adaboost (2)

- 使用权重更新来采样新样本集
- 采样多个弱分类器
- 最后通过弱分类器的加权投票来得到分类结果

集成学习- Adaboost (3)

- 误分的样本采样权重上升
- 正确分类的样本采样权重下降

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 很难被前2个分类器正确分类
- 它的采样权重一直在增加，在第3轮被选中多次

集成学习-集成分类的应用

- 高维数据
- 并行化算法

小结

- 分类是数据分析的重要技术之一，通过建立模型，学习类标知识来预测新出现的样本的类别知识
- 分类技术中的方法主要包括：决策树，朴素贝叶斯、基于规则分类、基于实例分类等等
- 集成分类：聚集多个弱分类器的分类结果得到超过强分类器的分类性能
- 评价方法：精度、混淆矩阵和ROC曲线

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

结束

2020年10月22日