2024-2025学年 第1学期(秋)

数据挖掘

第7章 规则和最近邻分类器

2024 年 10 月

南京大学
NANJING UNIVERSITY

目录

基于规则的分类

- 使用一组 “if…then…” 规则进行分类
- 规则:　(Condition) → y
  - 其中
    - Condition 是属性测试的合取
    - y 是类标号
  - 左部: 规则的前件（或前提）　(Rule antecedent)
  - 右部: 规则的后件（或结论）　(Rule consequent)
  - 分类规则的例子:
    - (Blood Type=Warm) ∧ (Lay Eggs=Yes) → Birds
    - (Taxable Income < 50K) ∧ (Refund=Yes) → Cheat =No

基于规则的分类：例

- 脊椎动物数据集

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
人类	恒温	毛发	是	否	否	是	否	哺乳类
蟒蛇	冷血	鳞片	否	否	否	否	是	爬行类
鲑鱼	冷血	鳞片	否	是	否	否	否	鱼类
鲸	恒温	毛发	是	是	否	否	否	哺乳类
青蛙	冷血	无	否	半	否	是	是	两栖类
巨蜥	冷血	鳞片	否	否	否	是	否	爬行类
蝙蝠	恒温	毛发	是	否	是	是	是	哺乳类
鸽子	恒温	羽毛	否	否	是	是	否	鸟类
猫	恒温	软毛	是	否	否	是	否	哺乳类
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类
美洲鳄	冷血	鳞片	否	半	否	是	否	爬行类
企鹅	恒温	羽毛	否	半	否	是	否	鸟类
豪猪	恒温	刚毛	是	否	否	是	是	哺乳类
鳗鲡	冷血	鳞片	否	是	否	否	否	鱼类
蠎蟒	冷血	无	否	半	否	是	是	两栖类

基于规则的分类：例

- 规则 r 覆盖 实例 x（记录），如果该实例的属性满足规则r的条件

r_1：（胎生 = 否）∧（飞行动物 = 是）→ 鸟类

r_2：（胎生 = 否）∧（水生动物 = 是）→ 鱼类

r_3：（胎生 = 是）∧（体温 = 恒温）→ 哺乳类

r_4：（胎生 = 否）∧（飞行动物 = 否）→ 爬行类

r_5：（水生动物 = 半）→ 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
鹰	恒温	羽毛	否	否	是	是	否	?
灰熊	恒温	软毛	是	否	否	是	是	?

- 规则r_1覆盖“鹰” => 鸟类

基于规则的分类：例

- 规则 r 覆盖 实例 x（记录），如果该实例的属性满足规则r的条件

r_1：（胎生 = 否）∧（飞行动物 = 是）→ 鸟类

r_2：（胎生 = 否）∧（水生动物 = 是）→ 鱼类

r_3：（胎生 = 是）∧（体温 = 恒温）→ 哺乳类

r_4：（胎生 = 否）∧（飞行动物 = 否）→ 爬行类

r_5：（水生动物 = 半）→ 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
鹰	恒温	羽毛	否	否	是	是	否	?
灰熊	恒温	软毛	是	否	否	是	是	?

- 规则r_1覆盖“鹰” => 鸟类
- 规则r_3 覆盖“灰熊” => 哺乳类

规则的质量

- 用覆盖率和准确率度量
- 规则的覆盖率（Coverage）：
  - 满足规则前件的记录所占的比例
- 规则的准确率（Accuracy）：
  - 在满足规则前件的记录中，满足规则后件的记录所占的比例
- 规则: (Status=Single) → No
- Coverage = 40%, Accuracy = 50%

Tid	Refund	Marital Stayus	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

如何用规则分类

- **一组规则**

  r_1：（胎生 = 否）∧（飞行动物 = 是）→ 鸟类

  r_2：（胎生 = 否）∧（水生动物 = 是）→ 鱼类

  r_3：（胎生 = 是）∧（体温 = 恒温）→ 哺乳类

  r_4：（胎生 = 否）∧（飞行动物 = 否）→ 爬行类

  r_5：（水生动物 = 半）→ 两栖类

- **待分类记录**

名称	体温	胎生	飞行动物	水生动物	类
狐猴	恒温	是	否	否	?
海龟	冷血	否	否	半水生	?
狗鲨	冷血	是	否	是	?

- 狐猴触发规则 r_3, 它分到哺乳类
- 海龟触发规则r_4和 r_5 —— 冲突
- 狗鲨未触发任何规则

规则分类的特征

- 互斥规则集
  - 每个记录最多被一个规则覆盖
  - 如果规则都是相互独立的，分类器包含互斥规则
- 如果规则集不是互斥的
  - 一个记录可能被多个规则触发
  - 如何处理?
    - 有序规则集

      —— 基于规则的序 vs 基于类的序
    - 无序规则集

      —— 在无序规则方案中，允许一条记录触发多条规则，规则被触发时视为对其相应类的一次投票，然后计算不同类的票数（可以使用加权方式）来决定记录的类所属。

r_1: （胎生 = 否） ∧ （飞行动物 = 是） → 鸟类

r_2: （胎生 = 否） ∧ （水生动物 = 是） → 鱼类

r_3: （胎生 = 是） ∧ （体温 = 恒温） → 哺乳类

r_4: （胎生 = 否） ∧ （飞行动物 = 否） → 爬行类

r_5: （水生动物 = 半） → 两栖类

- **海龟触发规则r4和r5——冲突**

规则分类的特征

- 穷举规则集
  - 每个记录至少被一个规则覆盖
  - 如果规则集涵盖了属性值的所有可能组合，则规则集具有穷举覆盖
- 如果规则集是非穷举的
  - 一个记录可能不被任何规则触发
  - 如何处理?
    - 使用缺省类

有序规则集

- 根据规则优先权将规则排序定秩（rank）
  - 有序规则集又称决策表（decision list）
- 对记录进行分类时
  - 由被触发的，具有最高秩的规则确定记录的类标号
  - 如果没有规则被触发，则指派到缺省类

r_1：（胎生 = 否）∧（飞行动物 = 是）→ 鸟类
r_2：（胎生 = 否）∧（水生动物 = 是）→ 鱼类
r_3：（胎生 = 是）∧（体温 = 恒温）→ 哺乳类
r_4：（胎生 = 否）∧（飞行动物 = 否）→ 爬行类
r_5：（水生动物 = 半）→ 两栖类

名称	体温	胎生	飞行动物	水生动物	类
海龟	冷血	否	否	半水生	?

规则定序方案

- 基于规则的序
  - 根据规则的质量排序：覆盖率(coverage)和准确率(accuracy)
- 基于类的序
  - 属于同一类的规则放在一起
  - 基于类信息（如类的分布、重要性）对每类规则排序

基于规则的排序	基于类的排序
(表皮覆盖=羽毛, 飞行动物=是) ⇒ 鸟类	(表皮覆盖=羽毛, 飞行动物=是) ⇒ 鸟类
(体温=恒温, 胎生=是) ⇒ 哺乳类	(体温=恒温, 胎生=否) ⇒ 鸟类
(体温=恒温, 胎生=否) ⇒ 鸟类	(体温=恒温, 胎生=是) ⇒ 哺乳类
(水生动物=半) ⇒ 两栖类	(水生动物=半) ⇒ 两栖类
(表皮覆盖=鳞片, 水生动物=否) ⇒ 爬行类	(表皮覆盖=无) ⇒ 两栖类
(表皮覆盖=鳞片, 水生动物=是) ⇒ 鱼类	(表皮覆盖=鳞片, 水生动物=否) ⇒ 爬行类
(表皮覆盖=无) ⇒ 两栖类	(表皮覆盖=鳞片, 水生动物=是) ⇒ 鱼类

如何建立基于规则的分类器

- 直接方法:
  - 直接由数据提取规则
  - 例如: RIPPER，Holte’s 1R

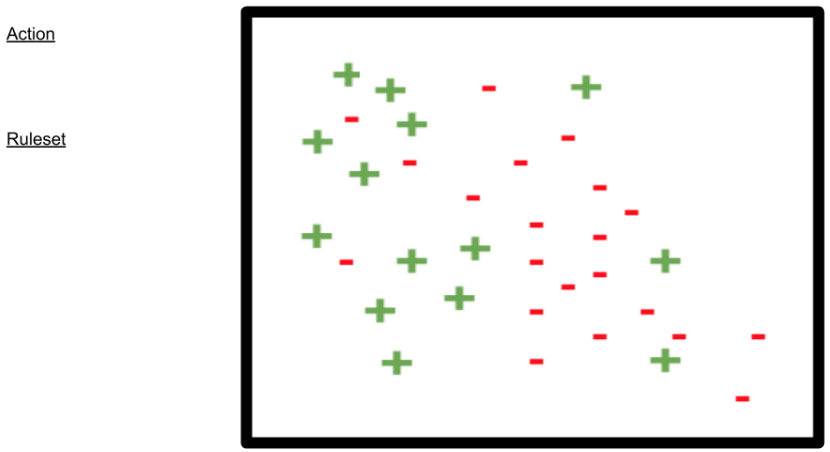
William W. Cohen

Robert Holte

- 间接方法:
  - 由其他分类模型提取规则 (例如，从决策树等).
  - 例如: C4.5rules

直接方法：顺序覆盖

- 基本思想
  - 依次对每个类建立一个或多个规则
  - 对第i类建立规则
    - 第i类记录为正例，其余为负例
    - 建立一个第i类的规则r，尽可能地覆盖正例，而不覆盖负例（即构建一个正例的规则）
    - 删除r覆盖的所有记录，在剩余数据集上学习下一个规则，直到所有第i类记录都被删除



直接方法：顺序覆盖

- 顺序覆盖（sequential covering）算法

1：令E是训练记录，A是属性—值对的集合$\{(A_j, v_j)\}$

2：令Y_o是类的有序集$\{y_1, y_2, \ldots, y_k\}$

3：令$R = \{\}$是初始规则列表

4：for 每个类 $y \in Y_o - \{y_k\}$ do

5： while 终止条件不满足 do

6： $r \leftarrow$ Learn-One-Rule (E, A, y)

7： 从E中删除被r覆盖的训练记录

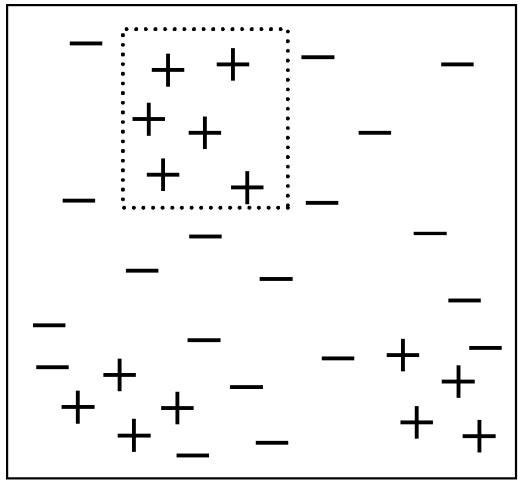
8： 追加r到规则列表尾部：$R \leftarrow R \vee r$

9： end while

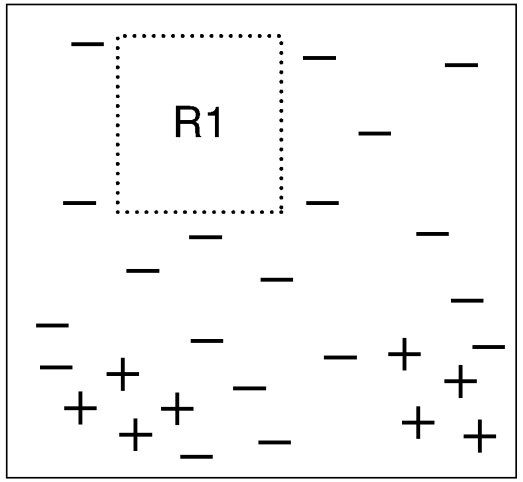
10：end for

11：把默认规则$\{\} \rightarrow y_k$插入到规则列表R尾部

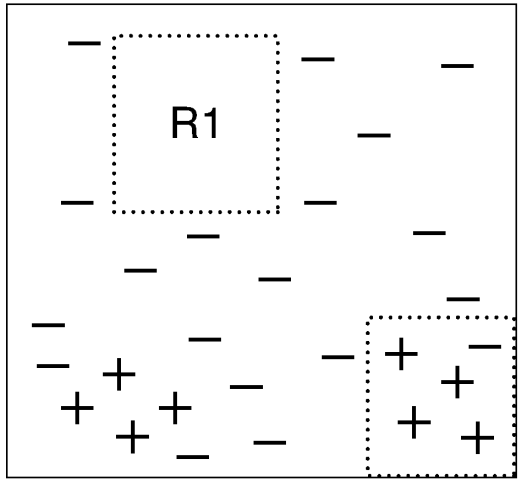
直接方法：顺序覆盖



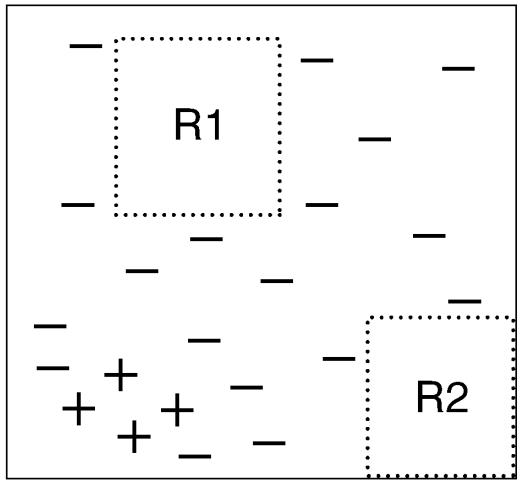
(a) Original data



(b) Step 1



(c) Step 2



(c) Step 3

删除实例

r_1: （胎生 = 否） ∧ （飞行动物 = 是） → 鸟类

r_2: （胎生 = 否） ∧ （水生动物 = 是） → 鱼类

r_3: （胎生 = 是） ∧ （体温 = 恒温） → 哺乳类

r_4: （胎生 = 否） ∧ （飞行动物 = 否） → 爬行类

r_5: （水生动物 = 半） → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
人类	恒温	毛发	是	否	否	是	否	哺乳类
蟒蛇	冷血	鳞片	否	否	否	否	是	爬行类
鲑鱼	冷血	鳞片	否	是	否	否	否	鱼类
鲸	恒温	毛发	是	是	否	否	否	哺乳类
青蛙	冷血	无	否	半	否	是	是	两栖类
巨蜥	冷血	鳞片	否	否	否	是	否	爬行类
蝙蝠	恒温	毛发	是	否	是	是	是	哺乳类
鸽子	恒温	羽毛	否	否	是	是	否	鸟类
猫	恒温	软毛	是	否	否	是	否	哺乳类
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类
美洲鳄	冷血	鳞片	否	半	否	是	否	爬行类
企鹅	恒温	羽毛	否	半	否	是	否	鸟类
豪猪	恒温	刚毛	是	否	否	是	是	哺乳类
鳗鲡	冷血	鳞片	否	是	否	否	否	鱼类
蠎蟒	冷血	无	否	半	否	是	是	两栖类

删除实例

r_1: （胎生 = 否） ∧ （飞行动物 = 是） → 鸟类

r_2: （胎生 = 否） ∧ （水生动物 = 是） → 鱼类

r_3: （胎生 = 是） ∧ （体温 = 恒温） → 哺乳类

r_4: （胎生 = 否） ∧ （飞行动物 = 否） → 爬行类

r_5: （水生动物 = 半） → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
人类	恒温	毛发	是	否	否	是	否	哺乳类
蟒蛇	冷血	鳞片	否	否	否	否	是	爬行类
鲑鱼	冷血	鳞片	否	是	否	否	否	鱼类
鲸	恒温	毛发	是	是	否	否	否	哺乳类
青蛙	冷血	无	否	半	否	是	是	两栖类
巨蜥	冷血	鳞片	否	否	否	是	否	爬行类
蝙蝠	恒温	毛发	是	否	是	是	是	哺乳类
猫	恒温	软毛	是	否	否	是	否	哺乳类
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类
美洲鳄	冷血	鳞片	否	半	否	是	否	爬行类
企鹅	恒温	羽毛	否	半	否	是	否	鸟类
豪猪	恒温	刚毛	是	否	否	是	是	哺乳类
鳗鲡	冷血	鳞片	否	是	否	否	否	鱼类
蠎蟒	冷血	无	否	半	否	是	是	两栖类

删除实例

r_1: （胎生 = 否） ∧ （飞行动物 = 是） → 鸟类

r_2: （胎生 = 否） ∧ （水生动物 = 是） → 鱼类

r_3: （胎生 = 是） ∧ （体温 = 恒温） → 哺乳类

r_4: （胎生 = 否） ∧ （飞行动物 = 否） → 爬行类

r_5: （水生动物 = 半） → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
人类	恒温	毛发	是	否	否	是	否	哺乳类
蟒蛇	冷血	鳞片	否	否	否	否	是	爬行类
鲑鱼	冷血	鳞片	否	是	否	否	否	鱼类
鲸	恒温	毛发	是	是	否	否	否	哺乳类
青蛙	冷血	无	否	半	否	是	是	两栖类
巨蜥	冷血	鳞片	否	否	否	是	否	爬行类
蝙蝠	恒温	毛发	是	否	是	是	是	哺乳类
猫	恒温	软毛	是	否	否	是	否	哺乳类
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类
美洲鳄	冷血	鳞片	否	半	否	是	否	爬行类
企鹅	恒温	羽毛	否	半	否	是	否	鸟类
豪猪	恒温	刚毛	是	否	否	是	是	哺乳类
鳗鲡	冷血	鳞片	否	是	否	否	否	鱼类
蠎蟒	冷血	无	否	半	否	是	是	两栖类

删除实例

r_1: （胎生 = 否） ∧ （飞行动物 = 是） → 鸟类

r_2: （胎生 = 否） ∧ （水生动物 = 是） → 鱼类

r_3: （胎生 = 是） ∧ （体温 = 恒温） → 哺乳类

r_4: （胎生 = 否） ∧ （飞行动物 = 否） → 爬行类

r_5: （水生动物 = 半） → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
人类	恒温	毛发	是	否	否	是	否	哺乳类
蟒蛇	冷血	鳞片	否	否	否	否	是	爬行类
鲸	恒温	毛发	是	是	否	否	否	哺乳类
青蛙	冷血	无	否	半	否	是	是	两栖类
巨蜥	冷血	鳞片	否	否	否	是	否	爬行类
蝙蝠	恒温	毛发	是	否	是	是	是	哺乳类
猫	恒温	软毛	是	否	否	是	否	哺乳类
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类
美洲鳄	冷血	鳞片	否	半	否	是	否	爬行类
企鹅	恒温	羽毛	否	半	否	是	否	鸟类
豪猪	恒温	刚毛	是	否	否	是	是	哺乳类
蝾螈	冷血	无	否	半	否	是	是	两栖类

删除实例

r_1: （胎生 = 否） ∧ （飞行动物 = 是） → 鸟类

r_2: （胎生 = 否） ∧ （水生动物 = 是） → 鱼类

r_3: （胎生 = 是） ∧ （体温 = 恒温） → 哺乳类

r_4: （胎生 = 否） ∧ （飞行动物 = 否） → 爬行类

r_5: （水生动物 = 半） → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
人类	恒温	毛发	是	否	否	是	否	哺乳类
蟒蛇	冷血	鳞片	否	否	否	否	是	爬行类
鲸	恒温	毛发	是	是	否	否	否	哺乳类
青蛙	冷血	无	否	半	否	是	是	两栖类
巨蜥	冷血	鳞片	否	否	否	是	否	爬行类
蝙蝠	恒温	毛发	是	否	是	是	是	哺乳类
猫	恒温	软毛	是	否	否	是	否	哺乳类
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类
美洲鳄	冷血	鳞片	否	半	否	是	否	爬行类
企鹅	恒温	羽毛	否	半	否	是	否	鸟类
豪猪	恒温	刚毛	是	否	否	是	是	哺乳类
蝾螈	冷血	无	否	半	否	是	是	两栖类

删除实例

r_1: （胎生 = 否） ∧ （飞行动物 = 是） → 鸟类

r_2: （胎生 = 否） ∧ （水生动物 = 是） → 鱼类

r_3: （胎生 = 是） ∧ （体温 = 恒温） → 哺乳类

r_4: （胎生 = 否） ∧ （飞行动物 = 否） → 爬行类

r_5: （水生动物 = 半） → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
蟒蛇	冷血	鳞片	否	否	否	否	是	爬行类
青蛙	冷血	无	否	半	否	是	是	两栖类
巨蜥	冷血	鳞片	否	否	否	是	否	爬行类
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类
美洲鳄	冷血	鳞片	否	半	否	是	否	爬行类
企鹅	恒温	羽毛	否	半	否	是	否	鸟类
蝾螈	冷血	无	否	半	否	是	是	两栖类

删除实例

r_1: (胎生 = 否) ∧ (飞行动物 = 是) → 鸟类

r_2: (胎生 = 否) ∧ (水生动物 = 是) → 鱼类

r_3: (胎生 = 是) ∧ (体温 = 恒温) → 哺乳类

r_4: (胎生 = 否) ∧ (飞行动物 = 否) → 爬行类

r_5: (水生动物 = 半) → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
蟒蛇	冷血	鳞片	否	否	否	否	是	爬行类
青蛙	冷血	无	否	半	否	是	是	两栖类
巨蜥	冷血	鳞片	否	否	否	是	否	爬行类
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类
美洲鳄	冷血	鳞片	否	半	否	是	否	爬行类
企鹅	恒温	羽毛	否	半	否	是	否	鸟类
蝾螈	冷血	无	否	半	否	是	是	两栖类

删除实例

r_1: （胎生 = 否） ∧ （飞行动物 = 是） → 鸟类

r_2: （胎生 = 否） ∧ （水生动物 = 是） → 鱼类

r_3: （胎生 = 是） ∧ （体温 = 恒温） → 哺乳类

r_4: （胎生 = 否） ∧ （飞行动物 = 否） → 爬行类

r_5: （水生动物 = 半） → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
青蛙	冷血	无	否	半	否	是	是	两栖类
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类
企鹅	恒温	羽毛	否	半	否	是	否	鸟类
蝾螈	冷血	无	否	半	否	是	是	两栖类

删除实例

r_1: (胎生 = 否) ∧ (飞行动物 = 是) → 鸟类

r_2: (胎生 = 否) ∧ (水生动物 = 是) → 鱼类

r_3: (胎生 = 是) ∧ (体温 = 恒温) → 哺乳类

r_4: (胎生 = 否) ∧ (飞行动物 = 否) → 爬行类

r_5: (水生动物 = 半) → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
青蛙	冷血	无	否	半	否	是	是	两栖类
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类
企鹅	恒温	羽毛	否	半	否	是	否	鸟类
蝾螈	冷血	无	否	半	否	是	是	两栖类

删除负实例

删除实例

r_1: (胎生 = 否) ∧ (飞行动物 = 是) → 鸟类

r_2: (胎生 = 否) ∧ (水生动物 = 是) → 鱼类

r_3: (胎生 = 是) ∧ (体温 = 恒温) → 哺乳类

r_4: (胎生 = 否) ∧ (飞行动物 = 否) → 爬行类

r_5: (水生动物 = 半) → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
青蛙	冷血	无	否	半	否	是	是	两栖类
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类
蝾螈	冷血	无	否	半	否	是	是	两栖类

删除负实例

删除实例

r_1: (胎生 = 否) ∧ (飞行动物 = 是) → 鸟类

r_2: (胎生 = 否) ∧ (水生动物 = 是) → 鱼类

r_3: (胎生 = 是) ∧ (体温 = 恒温) → 哺乳类

r_4: (胎生 = 否) ∧ (飞行动物 = 否) → 爬行类

r_5: (水生动物 = 半) → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
青蛙	冷血	无	否	半	否	是	是	两栖类
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类
蝾螈	冷血	无	否	半	否	是	是	两栖类

删除实例

r_1: (胎生 = 否) ∧ (飞行动物 = 是) → 鸟类

r_2: (胎生 = 否) ∧ (水生动物 = 是) → 鱼类

r_3: (胎生 = 是) ∧ (体温 = 恒温) → 哺乳类

r_4: (胎生 = 否) ∧ (飞行动物 = 否) → 爬行类

r_5: (水生动物 = 半) → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号
虹鳉	冷血	鳞片	是	是	否	否	否	鱼类

删除实例

r_1: (胎生 = 否) ∧ (飞行动物 = 是) → 鸟类

r_2: (胎生 = 否) ∧ (水生动物 = 是) → 鱼类

r_3: (胎生 = 是) ∧ (体温 = 恒温) → 哺乳类

r_4: (胎生 = 否) ∧ (飞行动物 = 否) → 爬行类

r_5: (水生动物 = 半) → 两栖类

名称	体温	表皮覆盖	胎生	水生动物	飞行动物	有腿	冬眠	类标号

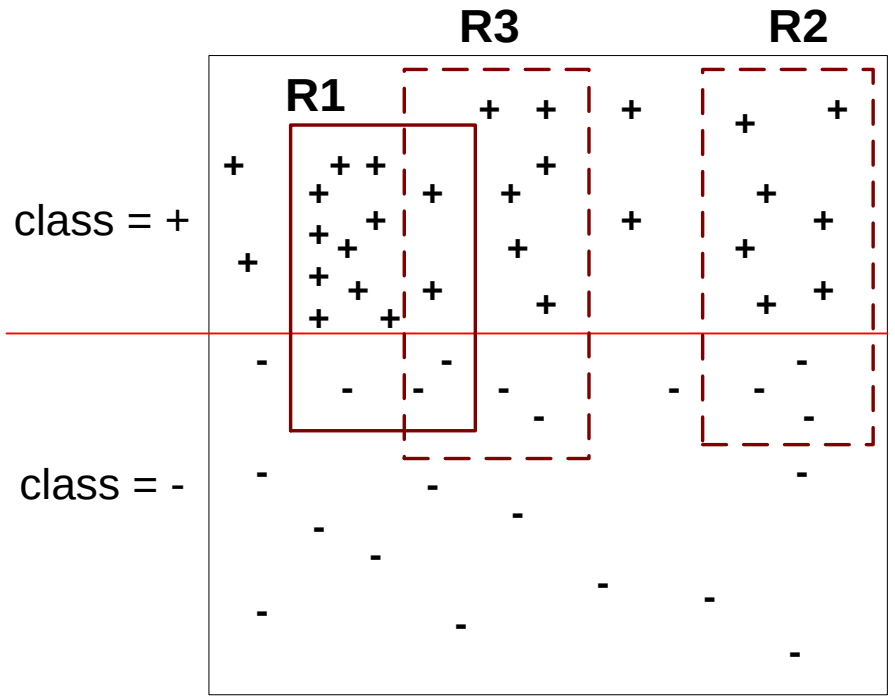
该规则集是穷举规则还是非穷举规则?

虹鳉	冷血	鳞片	是	是	否	否	否	鱼类

删除实例

- 为什么要删除实例?
  - 否则, 下一个规则将与前面的规则相同

    (规则可能重复)
- 为什么删除正实例?
  - 防止高估后面规则的准确率
  - 确保下一个规则不同
- 为什么删除负实例?
  - 防止过拟合错误训练集
  - 防止低估后面规则的准确率
  - 比较图中的规则 R2 和 R3



删除实例

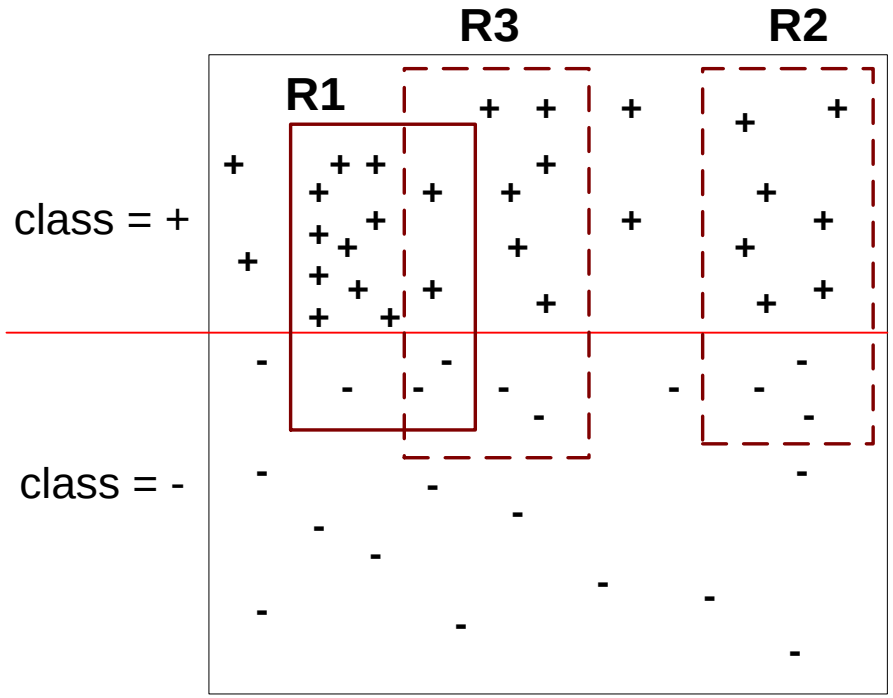
- R1：12/15=80%
- R2：7/10=70%
- R3：8/12=66.7%

1）产生R1（第一步）

2）产生R2？R3？

- R1 U R2：19/25=76%
- R1 U R3：

3）产生R？（第二步）



删除实例

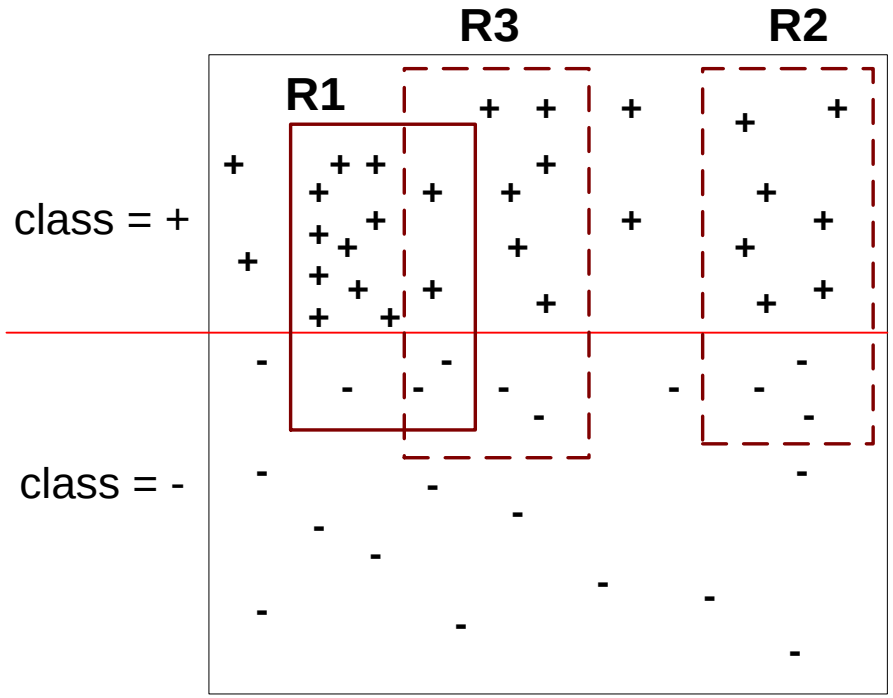
- R1：12/15=80%
- R2：7/10=70%
- R3：8/12=66.7%

1）产生R1（第一步）

2）产生R2？R3？

- R1 U R2：19/25=76%
- R1 U R3：18/23=78.3%

3）产生R3（第二步，6/8=75%）



Learn-One-Rule

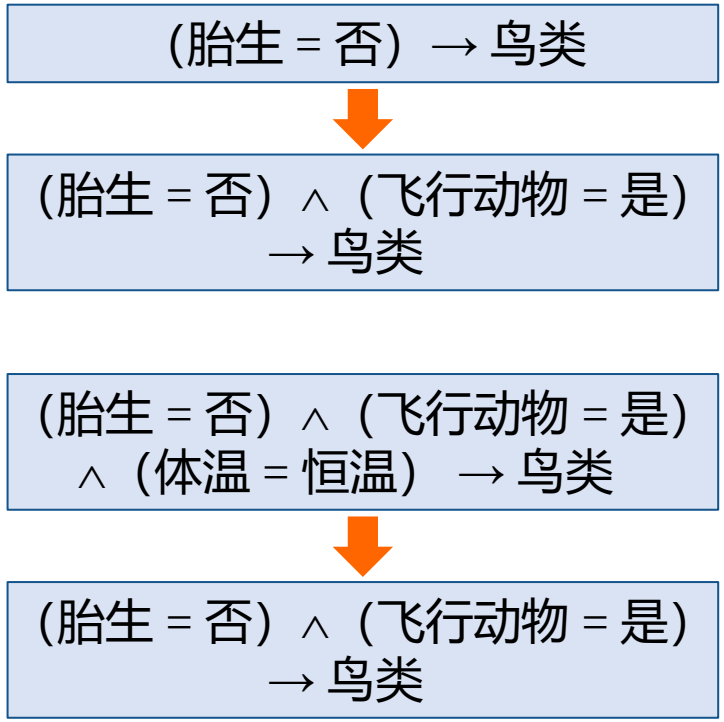
- 规则增长
- 规则评估
- 停止准则
- 规则剪枝

顺序覆盖（sequential covering）算法

1. 令E是训练记录，A是属性 — 值对的集合$\{(A_j, v_j)\}$
2. 令Y_o是类的有序集$\{y_1, y_2 \ldots, y_k\}$
3. 令$R=\{\}$是初始规则列表
4. for 每个类$y \in Y_o - \{y_k\}$ do
5. while 终止条件不满足 do 6
6. $r \leftarrow$ Learn-One-Rule (E,A,y)
7. 从E中删除被r覆盖的训练记录
8. 追加r到规则列表尾部：$R \leftarrow R \vee r$
9. end while
10. end for
11. 把默认规则$\{\} \rightarrow y_k$插入到规则列表R尾部

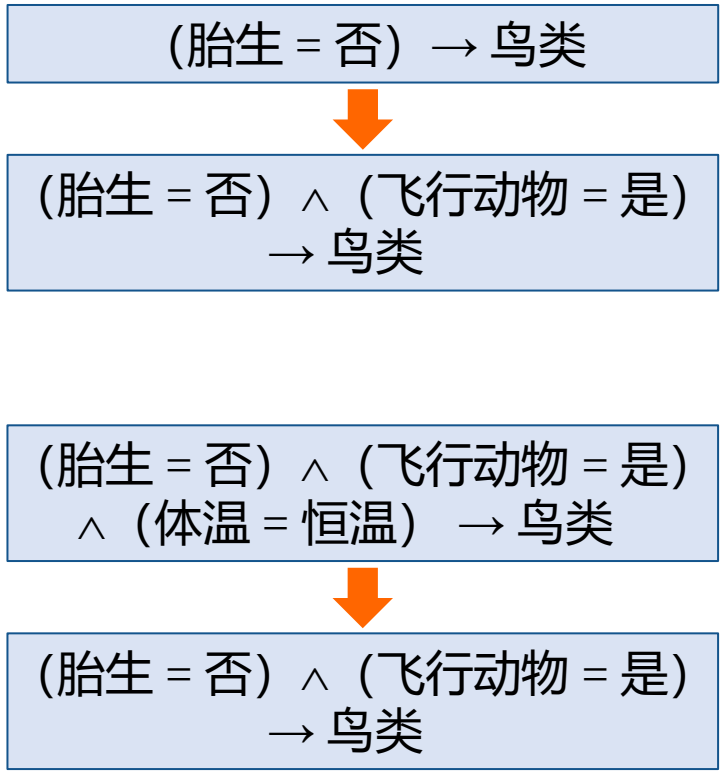
规则增长

- 两种策略
  - 一般到特殊（通常采用的策略）
    - 从初始规则r: {}→y开始
    - 反复加入合取项，得到更特殊的规则，直到不能再加入
  - 特殊到一般（适用于小样本情况）
    - 随机地选择一个正例作为初始规则
    - 反复删除合取项，得到更一般的规则，直到不能再删除



规则增长

- 两种策略
  - 一般到特殊（通常采用的策略）
    - 从初始规则r: {}→y开始
    - 反复加入合取项，得到更特殊的规则，直到不能再加入
  - 特殊到一般（适用于小样本情况）
    - 随机地选择一个正例作为初始规则
    - 反复删除合取项，得到更一般的规则，直到不能再删除



- 问题
  - 加入/删除合取项有多种选择，如何选择?
  - 何时停止加入/删除合取项?

  → 需要评估标准

规则增长（一般到特殊）

编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否

- 规则r后件为（好瓜=是），前件从空开始，先依次添加一个（属性，值），计算覆盖的部分记录编号及它的分类准确率。

规则增长（一般到特殊）

- 可以看到，（纹理=清晰）的正确率最高，因此首先在规则r的前件中添加（纹理=清晰），接着在（纹理=清晰）覆盖的记录中，继续规则r前件中（属性，值）添加。

属性-值	覆盖的记录编号	准确率
色泽=青绿	1,4,6,10,13,17	1/2
色泽=乌黑	2,3,7,8,9,15	2/3
根蒂=蜷缩	1,2,3,4,5,12,16,17	5/8
敲声=浊响	1,3,5,6,7,8,12,13,15,16	3/5
纹理=清晰	1,2,3,4,5,6,8,10,15	7/9
脐部=凹陷	1,2,3,4,5,13,14,17	5/8

规则增长（一般到特殊）

- 可以看到，在（纹理=清晰）覆盖的记录中，（根蒂=蜷缩）与（脐部=凹陷）覆盖记录的准确率都达到了100%，可以任选一个（属性，值），这里可以选择（根蒂=蜷缩），此时也达到了（属性，值）添加的终止条件，故在类（好瓜=是）中，函数生成了第一条规则。

属性-值	覆盖的记录编号	准确率
色泽=青绿	1,4,6,10	3/4
色泽=乌黑	2,3,8,15	3/4
根蒂=蜷缩	1,2,3,4,5	5/5
敲声=浊响	1,3,5,6,8,15	5/6
脐部=凹陷	1,2,3,4,5	5/5

- {（纹理=清晰）^（根蒂=蜷缩）}→（好瓜=是）

规则增长（一般到特殊）

- 从一般到特殊的规则生成策略中，每次只考虑一个最优的（属性，值）
- 这显得过于贪心，容易陷入局部最优麻烦
- 为了缓解该问题，可以采用一种“集束搜索(Beam search)”的方式
  - 具体做法为：每次选择添加的（属性，值)时，可以保留前k个最优的（属性，值），而不是只选择最优的那个，然后对这k个最优的（属性，值）继续进行下一轮的（属性，值）添加。
- {（纹理=清晰） ^ （根蒂=蜷缩）} → （好瓜=是）

Learn-One-Rule

- 规则增长
- 规则评估
- 停止准则
- 规则剪枝

顺序覆盖（sequential covering）算法

1. 令E是训练记录，A是属性 — 值对的集合$\{(A_j, v_j)\}$
2. 令Y_o是类的有序集$\{y_1, y_2 \ldots, y_k\}$
3. 令$R=\{\}$是初始规则列表
4. for 每个类$y \in Y_o - \{y_k\}$ do
5. while 终止条件不满足 do 6
6. $r \leftarrow$ Learn-One-Rule (E,A,y)
7. 从E中删除被r覆盖的训练记录
8. 追加r到规则列表尾部：$R \leftarrow R \vee r$
9. end while
10. end for
11. 把默认规则$\{\} \rightarrow y_k$插入到规则列表R尾部

规则评估

- 常用的度量
  - 准确率
  - 似然比
  - Laplace
  - FOIL信息增益

例如 考虑一个训练集，它包含60个正例和100个反例，现有两个候选规则：

r1:覆盖50个正例和5个反例

r2:覆盖2个正例和0个反例

规则评估

- 准确率
  - Accuracy= $\frac{n_c}{n}$
    - n : 被规则覆盖的实例数
    - n_c : 被规则正确分类的实例数
  - 问题：准确率高的规则可能覆盖率太低

$Acc(r_1)$: 90.9%

$Acc(r_2)$: 100%

例如 考虑一个训练集，它包含60个正例和100个反例，现有两个候选规则：

r1:覆盖50个正例和5个反例

r2:覆盖2个正例和0个反例

规则评估

- 准确率
  - Accuracy= $\frac{n_c}{n}$
    - n : 被规则覆盖的实例数
    - n_c : 被规则正确分类的实例数
  - 问题：准确率高的规则可能覆盖率太低

$Acc(r_1)$: 90.9%

$Acc(r_2)$: 100%

例如 考虑一个训练集，它包含60个正例和100个反例，现有两个候选规则：

r1:覆盖50个正例和5个反例

r2:覆盖2个正例和0个反例

规则评估

- 似然比 （越高越好）
  - k是类别数
  - f_i是被规则覆盖的类i的样本的观测频度
  - e_i是规则作随机猜测的期望频度

$$R = 2\sum_{i=1}^{k} f_i \log(f_i/e_i)$$

简单理解就是当前规则分类效果比随机效果越高，说明规则越好

$$LRS(r_1) = 2 \times \left[50 \times \log_2 \frac{50}{55\times 60/160} + 5 \times \log_2 \frac{5}{55\times 100/160}\right] = 99.99$$

例如 考虑一个训练集，它包含60个正例和100个反例，现有两个候选规则：

r1:覆盖50个正例和5个反例

r2:覆盖2个正例和0个反例

规则评估

- 似然比 （越高越好）
  - k是类别数
  - f_i是被规则覆盖的类i的样本的观测频度
  - e_i是规则作随机猜测的期望频度

$$R = 2\sum_{i=1}^{k} f_i \log(f_i/e_i)$$

简单理解就是当前规则分类效果比随机效果越高，说明规则越好

$$LRS(r_1) = 2 \times \left[50 \times \log_2 \frac{50}{55\times 60/160} + 5 \times \log_2 \frac{5}{55\times 100/160}\right] = 99.99$$

$$LRS(r_2) = 2 \times \left[2 \times \log_2 \frac{2}{2\times 60/160} + 0 \times \log_2 \frac{0}{2\times 100/160}\right] = 5.66$$

例如 考虑一个训练集，它包含60个正例和100个反例，现有两个候选规则：

r1:覆盖50个正例和5个反例

r2:覆盖2个正例和0个反例

规则评估

- Laplace估计
  - k是类别数
  - n_+是被规则覆盖的的正例数
  - n是被规则覆盖的样例数

例如考虑一个训练集，它包含60个正例和100个反例，现有两个候选规则：

r1:覆盖50个正例和5个反例

r2:覆盖2个正例和0个反例

$$Laplace = \frac{n_+ + 1}{n + k} \qquad Laplace(r_1) = \frac{50+1}{55+2} = 0.8947$$

- 准确率
  - $\text{Accuracy} = \frac{n_+}{n}$ ➡

简单理解：Laplace估计即为准确率的平滑

规则评估

- Laplace估计
  - k是类别数
  - n_+是被规则覆盖的的正例数
  - n是被规则覆盖的样例数

例如考虑一个训练集，它包含60个正例和100个反例，现有两个候选规则：

r1:覆盖50个正例和5个反例

r2:覆盖2个正例和0个反例

$$Laplace = \frac{n_+ + 1}{n + k}$$

$$Laplace(r_1) = \frac{50+1}{55+2} = 0.8947$$

$$Laplace(r_2) = \frac{2+1}{2+2} = 0.75$$

- 准确率
  - Accuracy $= \frac{n_+}{n}$

简单理解：Laplace估计即为准确率的平滑

规则评估

- FOIL信息增益 ➡ 类似决策树的信息增益
  - 假设规则r:A→覆盖n_{0+}个正例和n_{0-}个反例，增加新的合取项B后，扩展的规则r:B →覆盖n_{1+}个正例和n_{1-}个反例，此时扩展规则后FOIL信息增益定义为：

使用规则1后数据熵值　　使用规则2后数据熵值

$$FOIL(r) = n_{1+} \times \left[\log_2 \frac{n_{1+}}{n_{1+}+n_{1-}} - \log_2 \frac{n_{0+}}{n_{0+}+n_{0-}} \right]$$

  - 该度量倾向于选择那些高支持度计数和高准确率的规则

例如 考虑一个训练集，它包含60个正例和100个反例，现有两个候选规则：

r1:覆盖50个正例和5个反例

r2:覆盖2个正例和0个反例

$$FOIL(r_1) = 50 \times \left[\log_2 \frac{50}{50+5} - \log_2 \frac{60}{60+100} \right] = 63.87$$

规则评估

- FOIL信息增益 ➡ **类似决策树的信息增益**
  - 假设规则r:A→覆盖$\boldsymbol{n_{0+}}$个正例和$\boldsymbol{n_{0-}}$个反例，增加新的合取项B后，扩展的规则r:B →覆盖$\boldsymbol{n_{1+}}$个正例和$\boldsymbol{n_{1-}}$个反例，此时扩展规则后FOIL信息增益定义为：

    使用规则1后数据熵值　　使用规则2后数据熵值

$$FOIL(r) = n_{1+} \times \left[\log_2 \frac{n_{1+}}{n_{1+}+n_{1-}} - \log_2 \frac{n_{0+}}{n_{0+}+n_{0-}} \right]$$

  - 该度量倾向于选择那些**高支持度计数和高准确率**的规则

例如 考虑一个训练集，它包含60个正例和100个反例，现有两个候选规则：

r1:覆盖50个正例和5个反例

r2:覆盖2个正例和0个反例

$$FOIL(r_1) = 50 \times \left[\log_2 \frac{50}{50+5} - \log_2 \frac{60}{60+100} \right] = 63.87$$

$$FOIL(r_2) = 2 \times \left[\log_2 \frac{2}{2} - \log_2 \frac{60}{60+100} \right] = 2.83$$

Learn-One-Rule

- 规则增长
- 规则评估
- 停止准则
- 规则剪枝

顺序覆盖（sequential covering）算法

1. 令E是训练记录，A是属性 — 值对的集合$\{(A_j, v_j)\}$
2. 令Y_o是类的有序集$\{y_1, y_2 \ldots, y_k\}$
3. 令$R=\{\}$是初始规则列表
4. for 每个类$y \in Y_o - \{y_k\}$ do
5. while 终止条件不满足 do 6
6. $r \leftarrow$ Learn-One-Rule (E,A,y)
7. 从E中删除被r覆盖的训练记录
8. 追加r到规则列表尾部：$R \leftarrow R \vee r$
9. end while
10. end for
11. 把默认规则$\{\} \rightarrow y_k$插入到规则列表R尾部

停止条件与规则剪枝

- 停止条件
  - 计算增益
  - 如果增益不显著, 则丢弃新规则
- 规则剪枝
  - 类似于决策树后剪枝
  - 降低错误剪枝：
    - 删除规则中的合取项
    - 比较剪枝前后的错误率
    - 如果降低了错误率，则剪掉该合取项

如何建立基于规则的分类器

- 直接方法:
  - 直接由数据提取规则
  - 例如: RIPPER, CN2, Holte's 1R
- 间接方法:
  - 由其他分类模型提取规则 (例如，从决策树等).
  - 例如: C4.5rules

直接方法：RIPPER

- 对于2类问题, 选定一个类为正类，另一个为负类
  - 从正类学习规则
  - 负类时缺省类
- 多类问题
  - 按类的大小（属于特定类的实例所占的比例）对诸类排序
  - 从最小的类开始学习规则，其余类都看做负类
  - 对次小类学习规则，如此下去

直接方法：RIPPER

- 规则增长:
  - 由空规则开始
  - 只要能够提高FOIL信息增益就增加一个合取项
  - 当规则开始覆盖负实例时就停止
  - 剪枝
    - 剪枝度量：　　$v = (p-n)/(p+n)$

      p: 验证集中被规则覆盖的正实例数

      n: 验证集中被规则覆盖的负实例数
    - 剪枝方法:

      如果剪掉合取项可以提高v就剪

直接方法：RIPPER

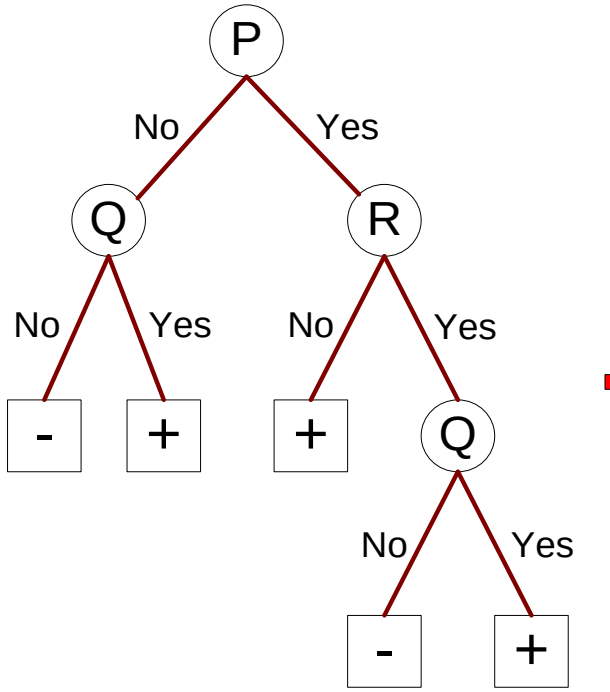
- 建立规则集:
  - 使用顺序覆盖算法
    - 找出覆盖当前正实例的最佳规则
    - 删除被规则覆盖的所有正实例和负实例
  - 当一个规则加入规则集时，计算新的最小描述长度MDL
    - 当新的MDL比已经得到的MDL多d位时，就停止增加新规则
    - 当在确认集上的错误率超过50%时，停止增加新规则

直接方法：RIPPER

- 优化规则集：
  - 对规则集R中的每个规则 r
    - 考虑2个替换的规则：
      - 替换规则 (r*): 重新增长新规则
      - 编辑的规则(r'): 把一个新的合取项增加到规则 r
    - 比较替换前后的规则集
    - 选择最小描述长度（MDL）的规则集
  - 对剩下的正实例，重复规则产生和优化

规则提取的间接方法

- 决策树从根结点到叶结点的每一条路径都可以表示为一个分类规则
  - 路径中的测试条件构成规则前件的合取项，叶结点的类标号赋给规则后件



Rule Set

r1: (P=No,Q=No) ==> -
r2: (P=No,Q=Yes) ==> +
r3: (P=Yes,R=No) ==> +
r4: (P=Yes,R=Yes,Q=No) ==> -
r5: (P=Yes,R=Yes,Q=Yes) ==> +

规则分类的特点

优 点

- 表达能力与决策树一样高
- 容易解释
- 容易产生
- 能够快速对新实例分类
- 性能可与决策树相媲美

缺 点

- 规则库难以维护
- 规则匹配计算量大
- 模型缺乏泛化能力

南京大学
NANJING UNIVERSITY

目录

急切学习 vs 惰性学习

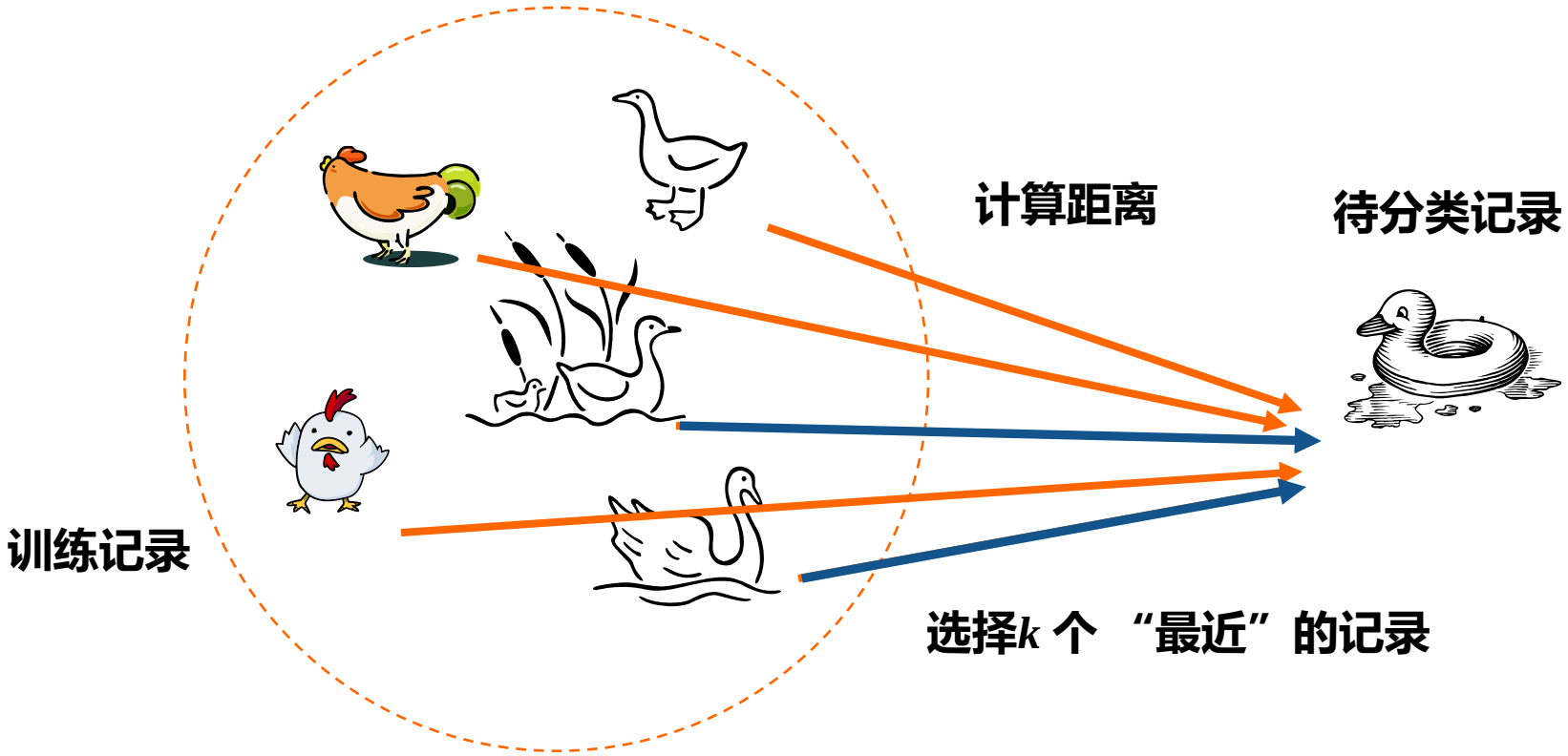
- 急切学习（Eager Learner）
  - 两步过程: (1) 归纳 (2) 演绎
- 惰性学习（Lazy Learner）
  - 把训练数据建模过程推迟到需要对样本分类时
  - 例子
  - Rote-learner（死记硬背）
    - 记住所有的训练数据，仅当记录的属性值与一个训练记录完全匹配才对它分类
  - 最近邻（Nearest neighbor）
    - 使用“最近”的 k 个点 (最近邻) 进行分类

南京大学
NANJING UNIVERSITY

目录

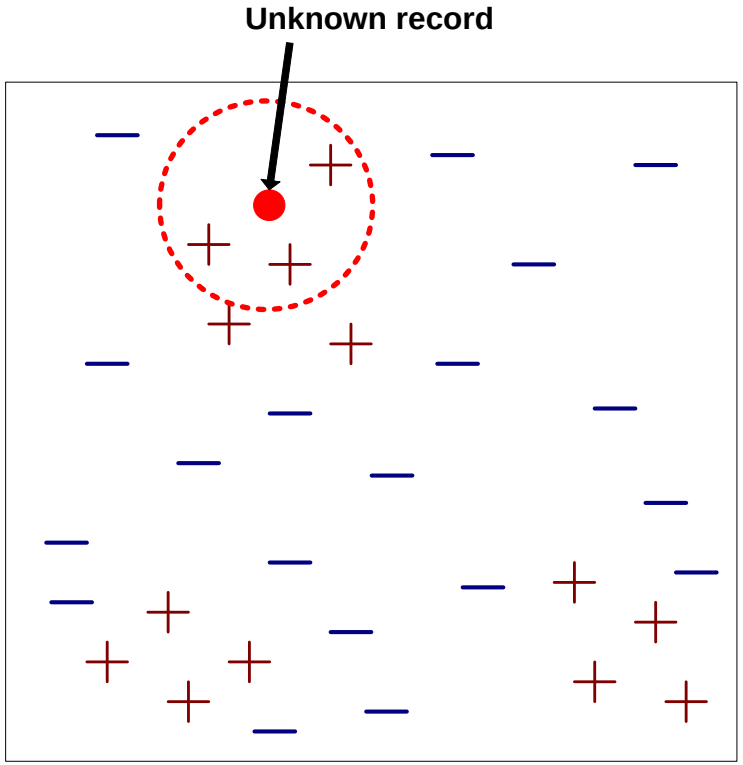
最近邻分类器

- 基本思想:
  - If it walks like a duck, quacks like a duck, then it's probably a duck

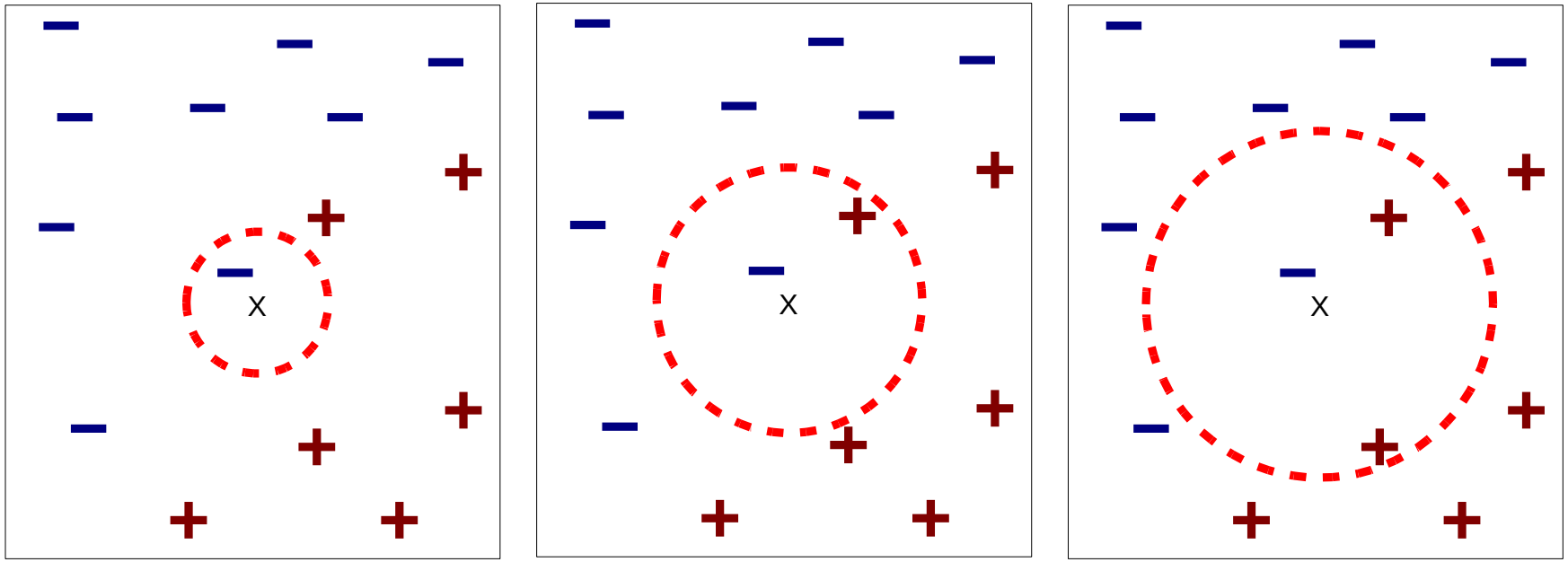


最近邻分类器

- 要求
  - 存放训练记录
  - 计算记录间距离的度量
  - k值, 最近邻数
- 对未知记录分类:
  - 计算域各训练记录的距离
  - 找出 k 个最近邻
  - 使用最近邻的类标号决定未知记录的类标号 (例如, 多数表决)



最近邻定义



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

记录x的k-最近邻是与x之间距离最小的k个训练数据点

k-最近邻分类算法

- k-最近邻分类算法

  1：令k是最近邻数目，D是训练样例的集合

  2：for 每个测试样例 z = (x', y') do

  3：　计算z和每个样例(x, y) ∈ D之间的距离d(x', x)

  4：　选择离z最近的k个训练样例的集合Dz　　D

  5：　$y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$

  6：end for

- 距离加权表决

$$y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} w_i \times I(v = y_i)$$

k-最近邻分类算法

- k-最近邻分类算法

  1：令k是最近邻数目，D是训练样例的集合

  2：for 每个测试样例 z = (x', y') do

  3：　计算z和每个样例(x, y) ∈ D之间的距离d(x', x)　　计算开销大

  4：　选择离z最近的k个训练样例的集合Dz　　D

  5：$$y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$$

  6：end for

- 距离加权表决

$$y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} w_i \times I(v = y_i)$$

k-最近邻分类算法

- k-最近邻分类算法

  1：令k是最近邻数目，D是训练样例的集合

  2：for 每个测试样例 z = (x', y') do

  3：　计算z和每个样例(x, y) ∈ D之间的距离d(x', x)　　计算开销大

  4：　选择离z最近的k个训练样例的集合Dz　　D

  5：　$y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$

  6：end for

  两种特殊的数据结构提前对训练集进行优化存储

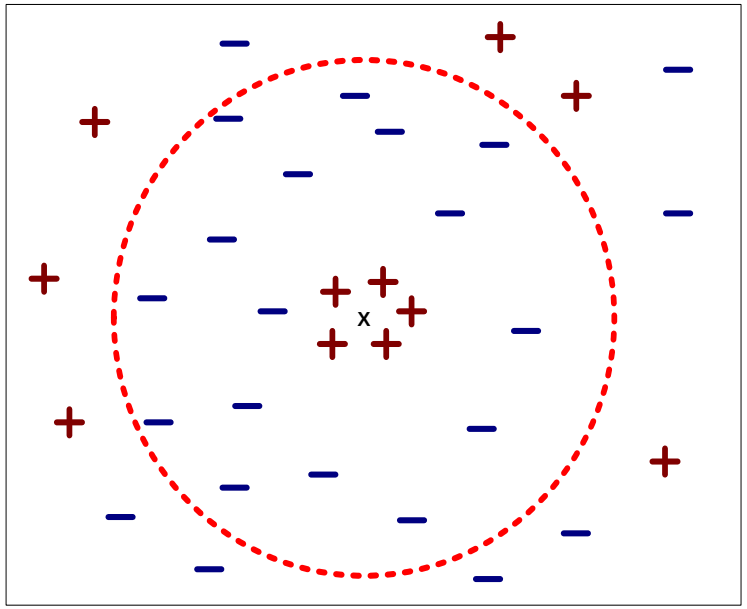
  - Kd-Tree
  - Kd-Ball

- 距离加权表决

$$y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} w_i \times I(v = y_i)$$

k-最近邻注意的问题

- k值的选择：
  - 如果 k 太小, 则对噪声点敏感
  - 如果 k 太大, 邻域可能包含很多其他类的点
- 定标问题（规范化）
  - 属性可能需要规范化，防止距离度量被具有很大值域的属性所左右



k-NN的特点

- k-NN的特点
  - 是一种基于实例的学习
    - 需要一个邻近性度量来确定实例间的相似性或距离
  - 不需要建立模型，但分类一个测试样例开销很大
    - 需要计算域所有训练实例之间的距离
  - 基于局部信息进行预测，对噪声非常敏感
  - 最近邻分类器可以生成任意形状的决策边界
    - 决策树和基于规则的分类器通常是直线决策边界
  - 需要适当的邻近性度量和数据预处理
    - 防止邻近性度量被某个属性左右