

- 规则学习的基本概念
  - 规则、覆盖、冲突、命题规则、一阶规则等
- 序贯覆盖
  - 解决什么问题，如何处理组合爆炸
  - 自顶向下和自底向上
- 剪枝优化
  - 解决什么问题
  - 预剪枝 (CN2), 后剪枝 (REP)
- 一阶规则学习
  - 要解决什么问题
  - FOIL学习算法
- 归纳逻辑程序设计

剪枝优化，对应算法，优缺点

为防止规则学习中出现组合爆炸，通常用哪两种策略产生规则？它们分别对应于哪类规则学习任务？

解答：

自顶向下、自底向上。

前者适用于命题规则学习，后者适用于一阶规则学习。

## 12. 描述基于"自顶向下"或者"自底向上"方法的一条规则的产生过程，如何避免每次只选择一条"最优"文字产生的局部最优解？

答：自顶向下方法：自顶向下方法是一种逐渐"特化"的过程。对于训练集每一个属性的所有取值产生一个文字，根据一定的评价方法评价每一个候选文字，如准确率（或基尼系数、信息熵增益等），根据准确率的大小和覆盖样本的多少以及属性的次序选择一个候选文字，之后从其他未选中为最终文字的属性中添加新的候选文字，并使用相同的过程，直到准确率为100%，最终这些选中的文字组合产生一条规则。

避免局部最优解可以使用集束搜索（beam search）的方法，每次保留最优的多个候选规则。

## 一、概念

规则：指语义明确、能描述数据分布所隐含的客观规律或领域概念、可写成"若……、则……"形式的逻辑规则

狭义的，省略了逻辑

规则学习：从训练数据中学习出一组能用于（泛化的）对未见示例进行判别的规则

优点：

- 可解释性更好
- 能更自然的在学习过程中引入领域知识
- 能基于逻辑规则进行推理，抽象描述能力强

覆盖：符合该规则的样本称为被规则覆盖（不等价）

冲突：同一个示例被判别结果不同的多条规则覆盖

冲突消解办法：

- 投票法
- 排序法：规则有优先级顺序
- 元规则法：根据领域知识事先设定的元规则（用于解决冲突，其实排序法也算一种元规则）

默认规则（缺省规则）：用来处理规则集合未覆盖到的样本

也是一种元规则

命题规则：由原子命题和逻辑连接词构成的简单陈述句

一阶规则：包含量词和逻辑变量

命题规则是一阶规则的特例

文字：原子公式及其否定

## 二、序贯覆盖

规则学习目标：产生最大覆盖的规则集

序贯覆盖=逐条归纳=分治策略：在训练集上每学到一条规则，就把该规则覆盖的样例去除

穷尽搜索会组合爆炸

一般的规则：空规则

特殊规则：直接以样例属性取值形成的规则

### 自顶向下（生成-测试）

从比较一般规则开始，逐渐添加新文字缩小覆盖范围，规则逐渐特化

- 容易产生泛化性能好的规则
- 对噪声的鲁棒性更强
- 适合命题规则学习

集束搜索：每轮保留最优的b个逻辑文字，在下一轮均用于构建候选集，再把候选集中最优的b个留待再下一轮

### 自底向上（数据驱动）

从比较特殊规则开始，逐渐删除文字扩大覆盖范围，规则逐渐泛化

- 适合小样本
- 适合一阶规则学习（假设空间非常复杂的任务）

适用于多分类问题：当学习关于第c类的规则时，将所有属于类别c的样本作为正例，其他类别的样本作为反例

## 三、剪枝优化

贪心搜索需要剪枝缓解过拟合风险

通常是基于某种性能度量指标来评估增/删逻辑文字前后的规则性能，或增/删规则前后的规则集性能，从而判断是否要进行剪枝

### 预剪枝：CN2算法

用规则集预测必须显著优于直接基于训练集后验概率分布进行预测

似然率统计量LRS，越大说明越好，通常设置0.99才停止规则集生长

### 后剪枝：REP减错剪枝

在学得的规则集上穷举所有剪枝可能，选在验证集性能最好的，用验证集反复剪枝直到准确率无法提高

很有效，但复杂度太高

### IREP:

每生成一条新规则即对其进行REP剪枝

### RIPPER:

后处理优化过程

从全局考虑，缓解序贯覆盖贪心算法局部性

## 四、一阶规则学习

### 解决什么问题

命题规则难以处理对象间关系，而关系信息在很多任务中很重要

### FOLL

优点：把命题规则学习过程通过变量替换等操作直接转化为一阶规则学习，因此比一般归纳逻辑程序设计技术更高效

缺点：大致看作命题规则学习与归纳逻辑程序设计之间的过渡，其自顶向下的规则生成过程不能支持函数和逻辑表达式嵌套，因此规则表达能力仍有不足