符号学习简介

命题规则学习（下）

(Press ? for help, n and p for next and previous slide)

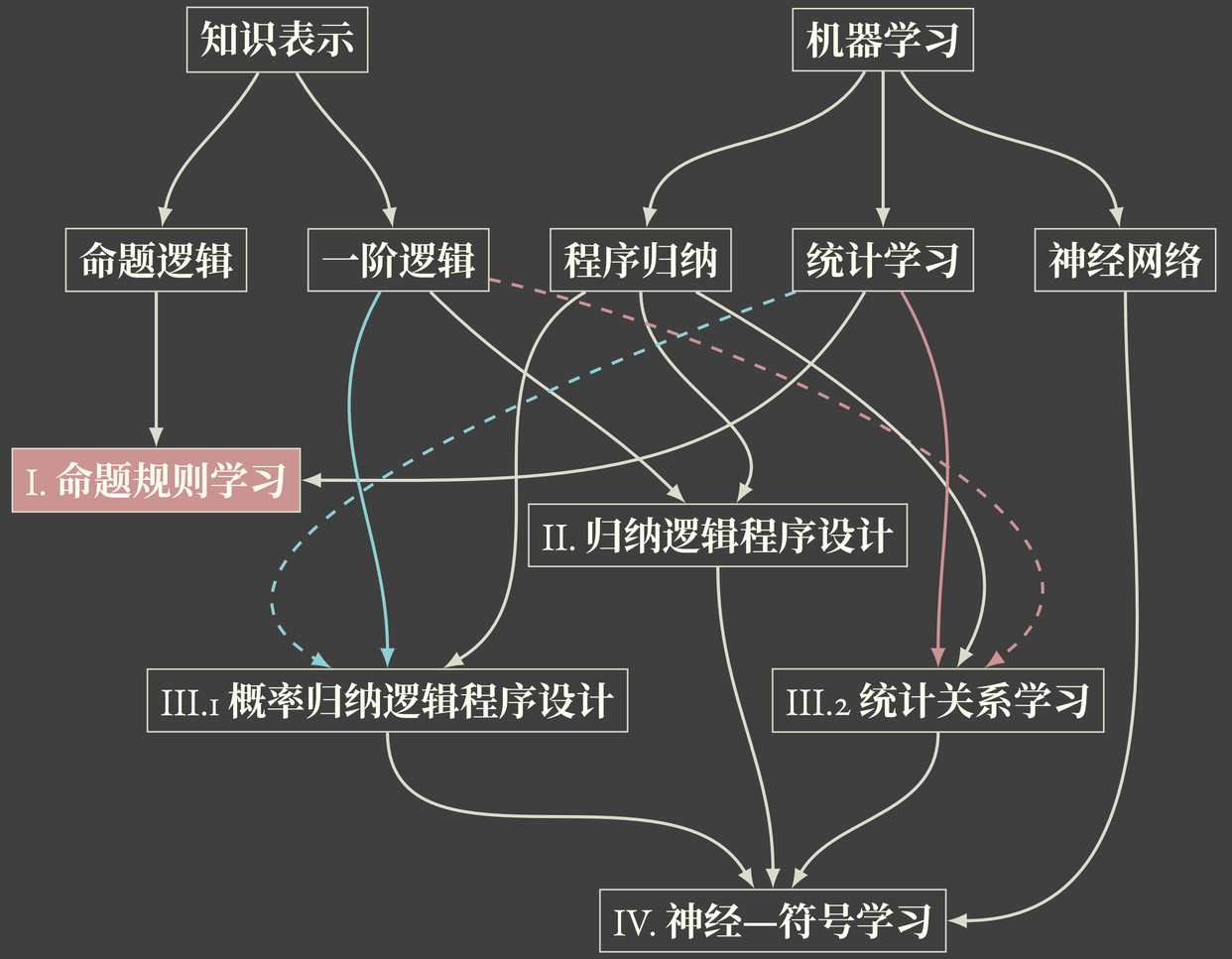
戴望州

南京大学智能科学与技术学院

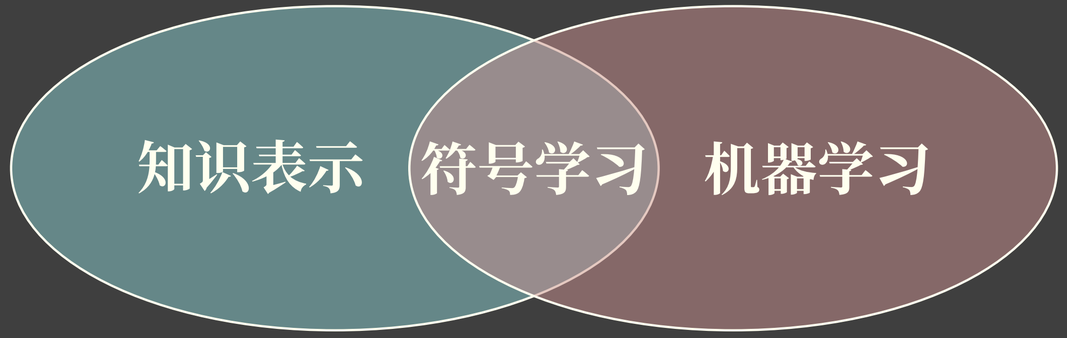
2025年-秋季

https://daiwz.net

路径图


知识表示
机器学习
命题逻辑
一阶逻辑
程序归纳
统计学习
神经网络
I. 命题规则学习
II. 归纳逻辑程序设计
III.1 概率归纳逻辑程序设计
III.2 统计关系学习
IV. 神经—符号学习

命题规则集学习



- **选择“词汇”**：
  - 背景知识B：命题逻辑规则
- **寻找“解释”**：
  - 假设模型H：命题逻辑规则集
  - 寻找过程：启发式搜索

符号学习简介·命题规则（下）

1. **命题规则集**
2. 规则集学习
3. 规则集剪枝
4. 命题规则集的局限

命题规则的描述能力

$$H \leftarrow B_1 \wedge B_2 \wedge \ldots \wedge B_n.$$

- 若存在A个属性，每个属性最多有V个取值：
  - V^A个样例，2^{V^A}种假设模型
- $V \cdot A$种逻辑文字
  - $2^{V \cdot A}$种规则

显然，一条逻辑规则的描述能力远远不够！

析合范式

也叫**析取范式（Disjunctive Normal Form, DNF）**：用“$\vee$”把所有可能的正样例都包含进来

$$positive \leftarrow (\mathbf{f}_1 \wedge \mathbf{f}_2) \vee (\mathbf{f}_3 \wedge \mathbf{f}_4) \vee \ldots.$$

它可以等价地表示为一个**命题逻辑规则集**：

$$positive \leftarrow \mathbf{f}_1 \wedge \mathbf{f}_2$$
$$positive \leftarrow \mathbf{f}_3 \wedge \mathbf{f}_4$$
$$\ldots$$

相当于$(positive \leftarrow \mathbf{f}_1 \wedge \mathbf{f}_2) \wedge (positive \leftarrow \mathbf{f}_3 \wedge \mathbf{f}_4)$

- 集合表示：$\{positive \leftarrow \mathbf{f}_1 \wedge \mathbf{f}_2,\ \ positive \leftarrow \mathbf{f}_3 \wedge \mathbf{f}_4\}$

命题逻辑规则集

命题逻辑集是由命题逻辑规则构成的**集合**。

$$
\begin{aligned}
R_1 &: H^1 \leftarrow B_1^1 \wedge B_2^1 \wedge \ldots \wedge B_{n_1}^1. \\
R_2 &: H^2 \leftarrow B_1^2 \wedge B_2^2 \wedge \ldots \wedge B_{n_2}^2. \\
&\ldots \\
R_m &: H^m \leftarrow B_1^m \wedge B_2^m \wedge \ldots \wedge B_{n_2}^m.
\end{aligned}
$$

它是一种k-CNF，即最大子句长为k的合析范式（Conjunctive Normal Form, CNF, or合取范式）

- **子句**（clause）：逻辑文字构成的析取式
  - $h_1 \vee h_2 \vee \ldots \vee h_m \vee \neg b_1 \vee \neg b_2 \vee \ldots \vee \neg b_n$
  - $h_1 \vee h_2 \vee \ldots \vee h_m \leftarrow b_1 \wedge b_2 \wedge \ldots \wedge b_n$
  - $h_1; h_2; \ldots; h_m \leftarrow b_1, b_2, \ldots, b_n$

闭世界假设

Closed World Assumption (CWA)

$$
\begin{aligned}
&\text{好瓜} \leftarrow \text{色泽} = \text{青绿} \\
&\text{好瓜} \leftarrow \text{敲声} = \text{浊响} \\
&\text{好瓜} \leftarrow \text{脐部} = \text{凹陷} \\
&\text{好瓜} \leftarrow \text{根蒂} = \text{稍蜷}
\end{aligned}
$$

相当于:

$$\text{好瓜} \leftrightarrow \neg(\text{色泽} = \text{青绿}) \vee (\text{敲声} = \text{浊响}) \vee (\text{脐部} = \text{凹陷}) \vee (\text{根蒂} = \text{稍蜷})$$

相当于:

$$\text{坏瓜} \leftrightarrow (\text{色泽} = \text{青绿}) \wedge \neg(\text{敲声} = \text{浊响}) \wedge \neg(\text{脐部} = \text{凹陷}) \wedge \neg(\text{根蒂} = \text{稍蜷})$$

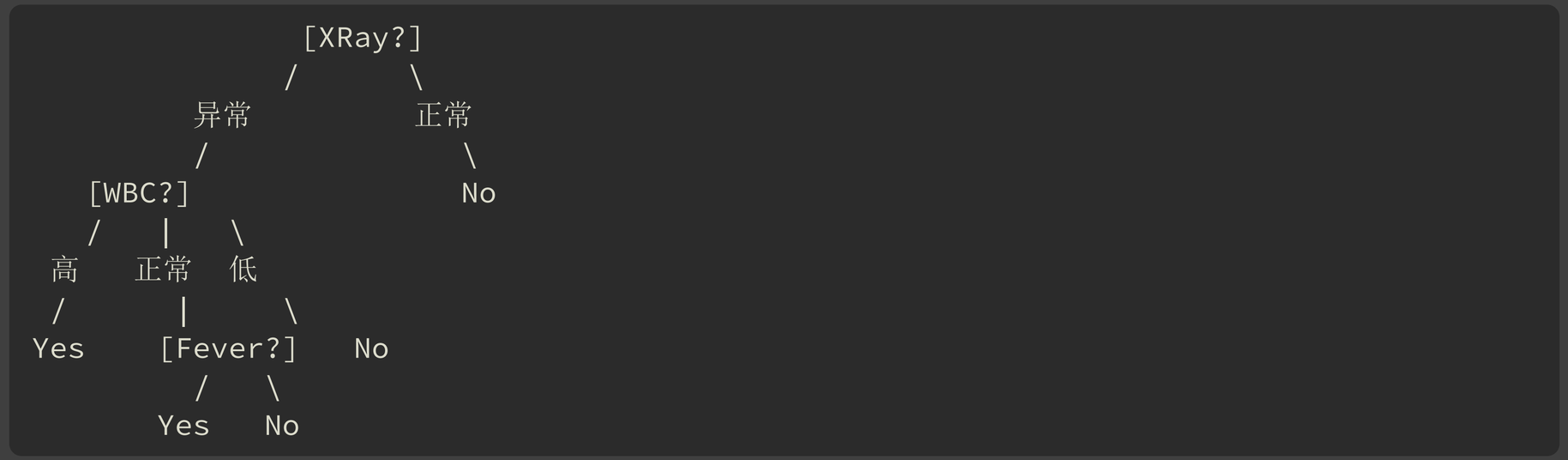
命题规则集 vs 决策树

🧪 场景背景：

- 医院要根据体征和检测结果判断患者是否“可能患有肺炎”。考虑以下特征：
  - Fever（发烧，体温 > 38°C）
  - Cough（咳嗽）
  - ChestPain（胸痛）
  - WBC（白细胞计数，高/正常/低）
  - XRay（胸片结果，正常/异常）
  - Age（年龄）
- 目标：输出 Pneumonia = Yes / No（是否可能患肺炎）

决策树表示

训练后的决策树可能长这样👇



所有判断都沿着单一路径完成，每个病例只能对应树上的一个“叶节点”规则：

- 胸片异常 + 白细胞高 → Yes.
- 胸片异常 + 白细胞正常 + 发烧 → Yes.
- 胸片异常 + 白细胞低 → No.

命题规则集表示

规则集可能是这样的（每条规则是独立的 if-then 语句）：

```
RULE 1: Yes :- XRay = 异常, WBC = 高, !.
RULE 2: Yes :- XRay = 异常, Fever = Yes, !.
RULE 3: Yes :- Cough = Yes, Fever = Yes, Age > 65, !.
RULE 4: Yes :- ChestPain = Yes, XRay = 异常, !.
RULE 5: No :- XRay = 正常, WBC = 正常, AND Fever = No, !.
DEFAULT: No :- true.
```

特殊情况：合并症与例外

例如，有一种特殊情况：“X光正常，但患者高龄、有持续发烧+咳嗽 → 虽然X光没发现异常，也不能完全排除肺炎。”

- 在决策树中：XRay = 正常 直接被划到 “No” 分支；
  - 要处理这种“例外”，就得改动整棵树的结构，重新训练。
- 在规则集里：你只需增加一条规则

```
RULE 6: Yes :- XRay = 正常, Age > 70, Fever = Yes, Cough = Yes
```

命题规则集 vs 决策树

特性	规则集	决策树
表达方式	平行规则集合，可重叠	树结构，单一路径
可解释性	高，规则短小、独立	结构化，但可能冗长
可维护性	易增删改	修改困难，需整体重建
与专家知识结合	方便	较困难
表达复杂逻辑的能力	强（组合规则）	弱，需要深树
适应类别重叠/不平衡	好	一般

命题规则集的缺点

虽然命题规则集有这些优势，但它在一些方面也有挑战：

- 规则学习算法（如 RIPPER、CN2、Bayesian Rule Sets）相对复杂；
- 对于高噪声数据或非线性问题，决策树可能更容易快速拟合；
- 规则集的预测时间可能略慢（因为要检查多条规则）

符号学习简介·命题规则（下）

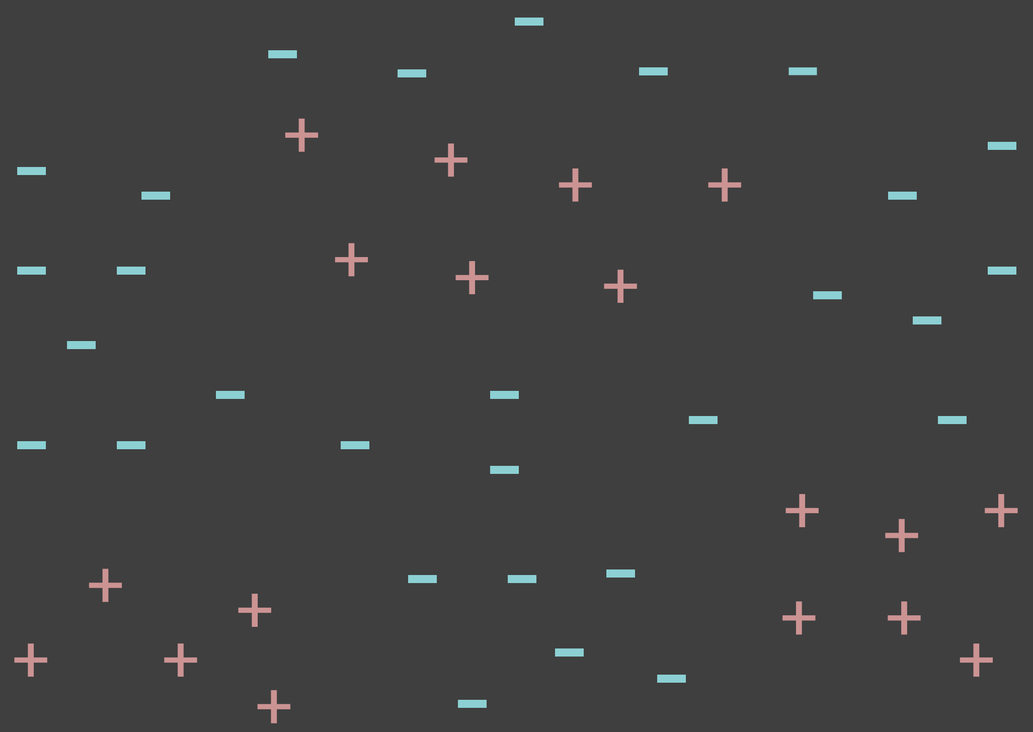
1. 命题规则集
2. **规则集学习**
3. 规则集剪枝
4. 命题规则集的局限

序贯覆盖（Covering）

也叫Sequential Covering

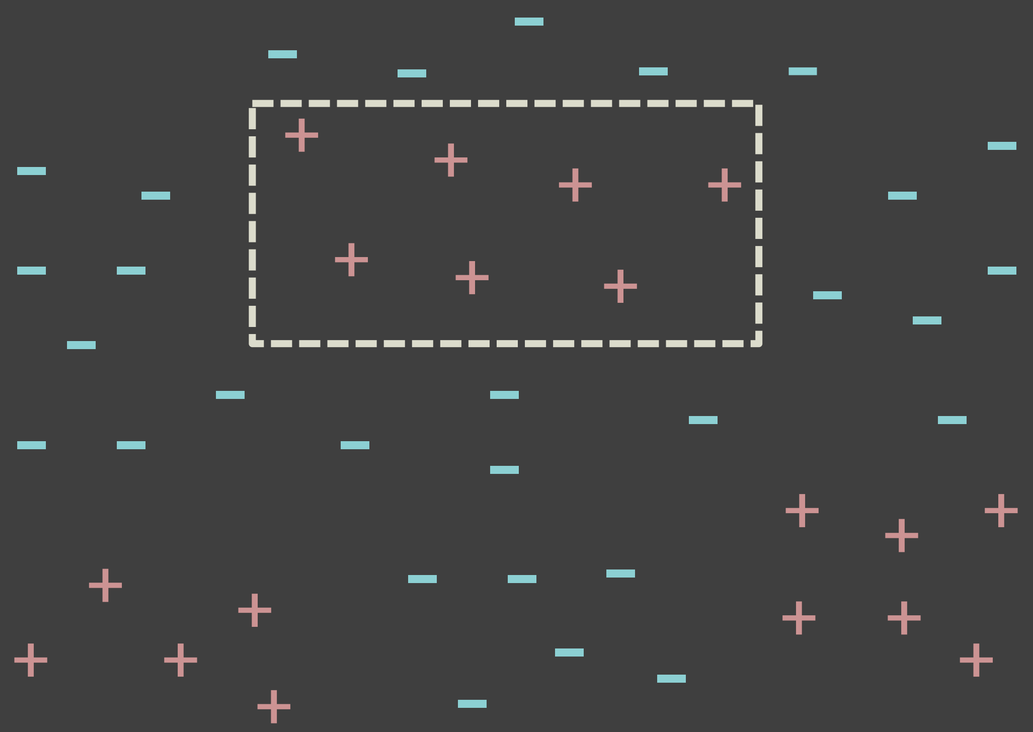
```
# E: 训练样本
function covering(E)
    R = [] # 初始化规则集合
    E = P, N # P/N: 正/负样例集合

    E_rest = E # 当前还未覆盖的样本
    P_rest = P # 当前还未覆盖的负样例

    F = init_features(E) # 初始化特征集合
    while isempty(P_rest) == false
        # 检查是否还需要学更多规则
        if need_more_rule(R, E_rest) == false
            break
        end

        # 学习单条命题规则r, 并加入规则集R
        r = find_best_rule(E_rest, F)
        push!(R, r)

        # 调整样本集合（例如删掉被覆盖的样例）
        E_rest = adjust_examples(R, E_rest)
```

序贯覆盖

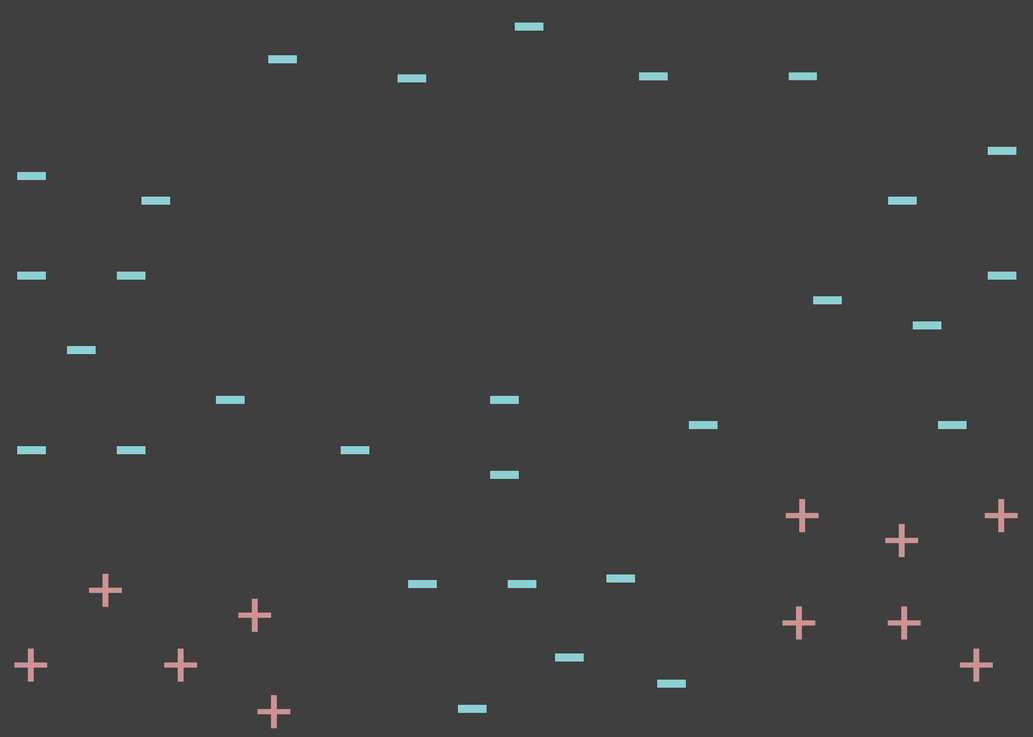


序贯覆盖

- R_1



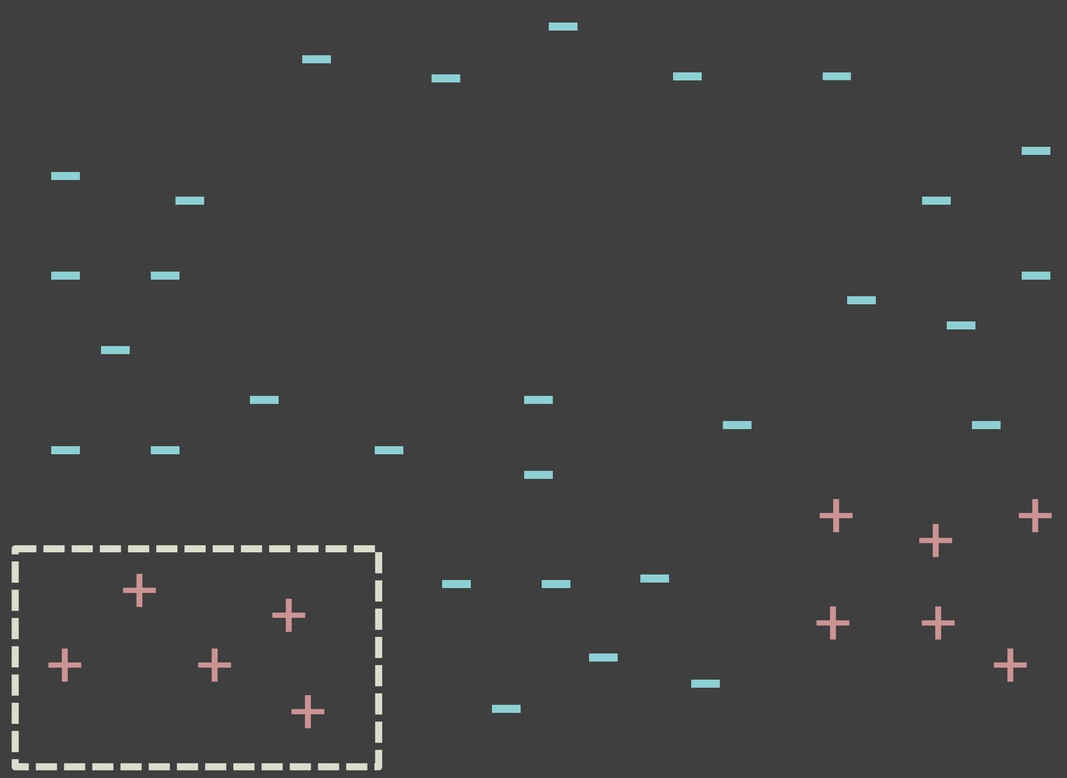
序贯覆盖



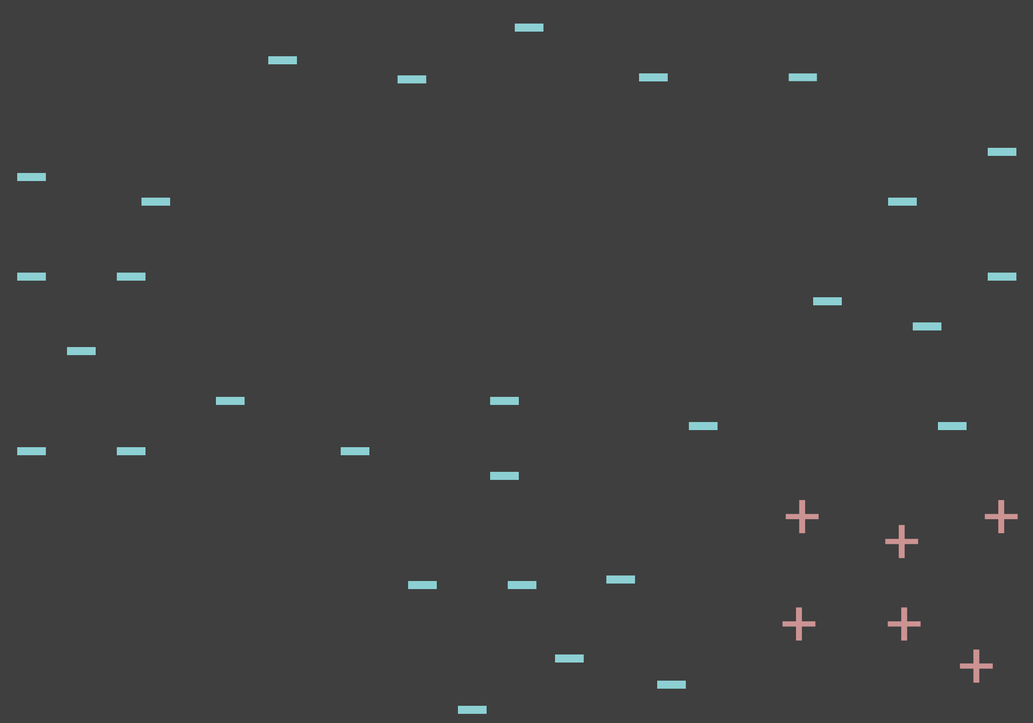
- R_1
- **继续生成**

序贯覆盖

- R_1
- R_2

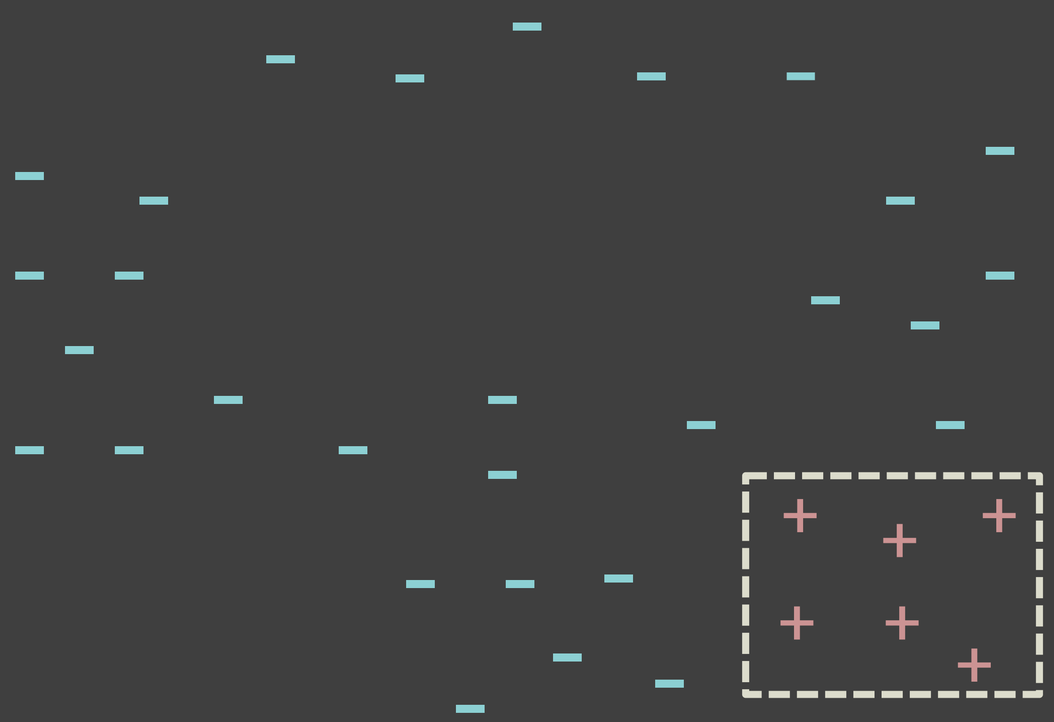


序贯覆盖



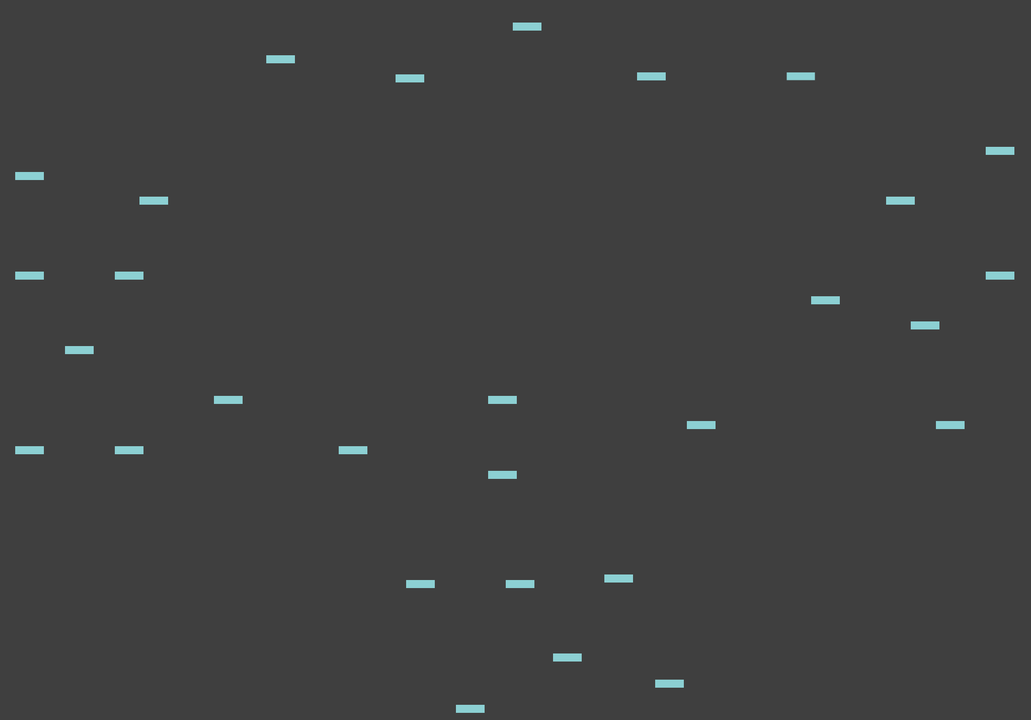
- R_1
- R_2
- **继续生成**

序贯覆盖



- R_1
- R_2
- R_3

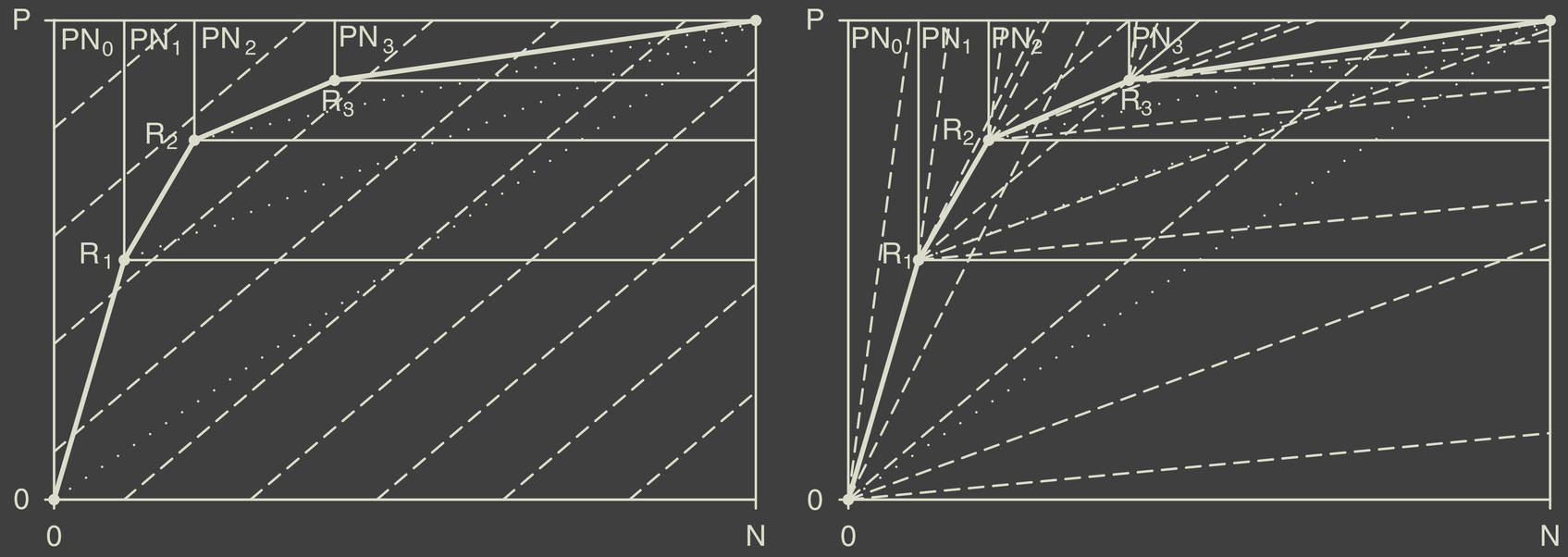
序贯覆盖



- R_1
- R_2
- R_3
- **终止生成**

打分函数对序贯覆盖的影响

精度（accuracy，$(TP+TN)/TTL$）vs准确率（precision, $TP/(TP+FP)$）



Accuracy作为规则评价函数

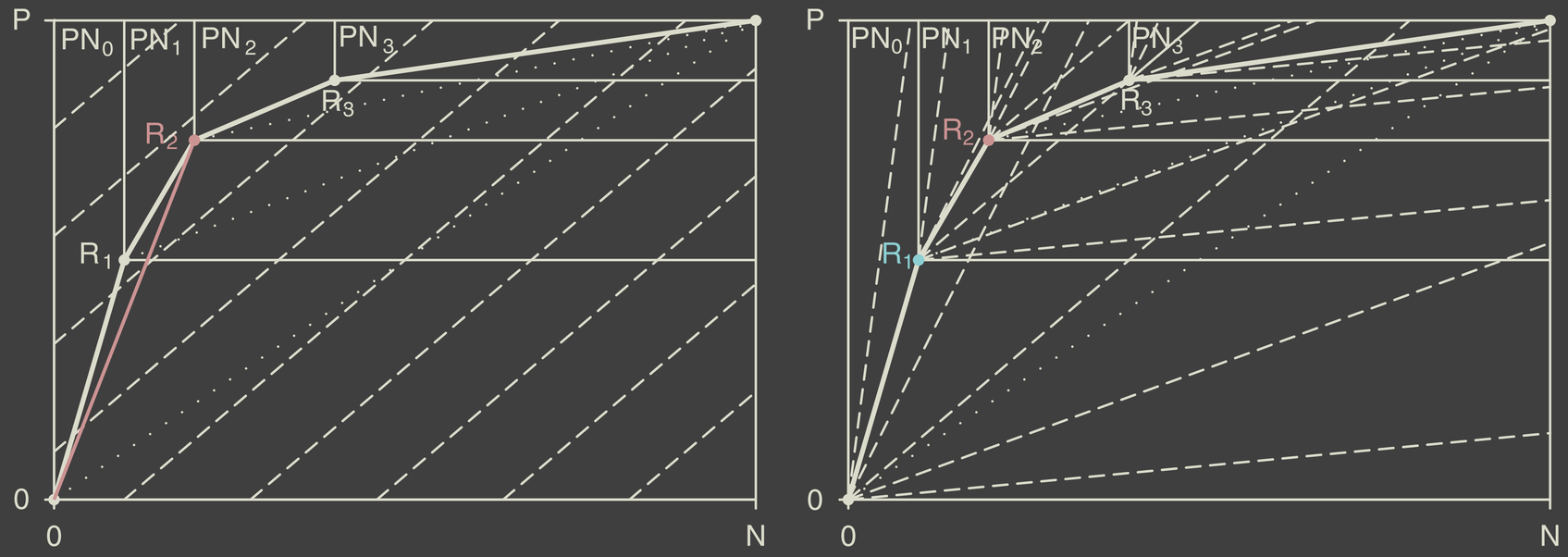
$$\frac{TP+TN}{TP+TN+FP+FN} = 1 - \frac{FP+FN}{Total}$$

Sequential covering 使用 Accuracy 作为打分标准时，它倾向于

- 选择覆盖范围更大（包括一些负例）的规则，因为总体正确率提高了
  - 这种“宽网兜更多样本”的规则可能不够“纯”，但能快速提高总体 accuracy
- 缺点：当正负样本分布不平衡（负样本多）时，只要规则覆盖了大量负样本不犯错，或者即使有些 FP，accuracy 仍可能很高
  - 容易导致学习到一些“泛泛而谈”的规则

打分函数对序贯覆盖的影响

精度（accuracy，$(TP+TN)/TTL$）vs准确率（precision, $TP/(TP+FP)$）



Precision作为规则评价函数

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

等高线不是平行线，而是从原点发散的射线（iso-precision lines）：

- Precision 越高，斜率越大（更多 TP，少 FP）。

在 sequential covering 使用 precision 作为打分标准时，算法倾向于选择：

- 规则“纯度高”（几乎不覆盖负样本）的规则；
  - 即使 TP 数量少，只要 FP 几乎为零，也可能得高分。
  - 每条规则都尽量覆盖“干净”的正样本子集。
- 缺点：这种规则虽然“很准”，但可能覆盖的样本很少，导致 recall（召回率）下降，整体规则集的覆盖率低。

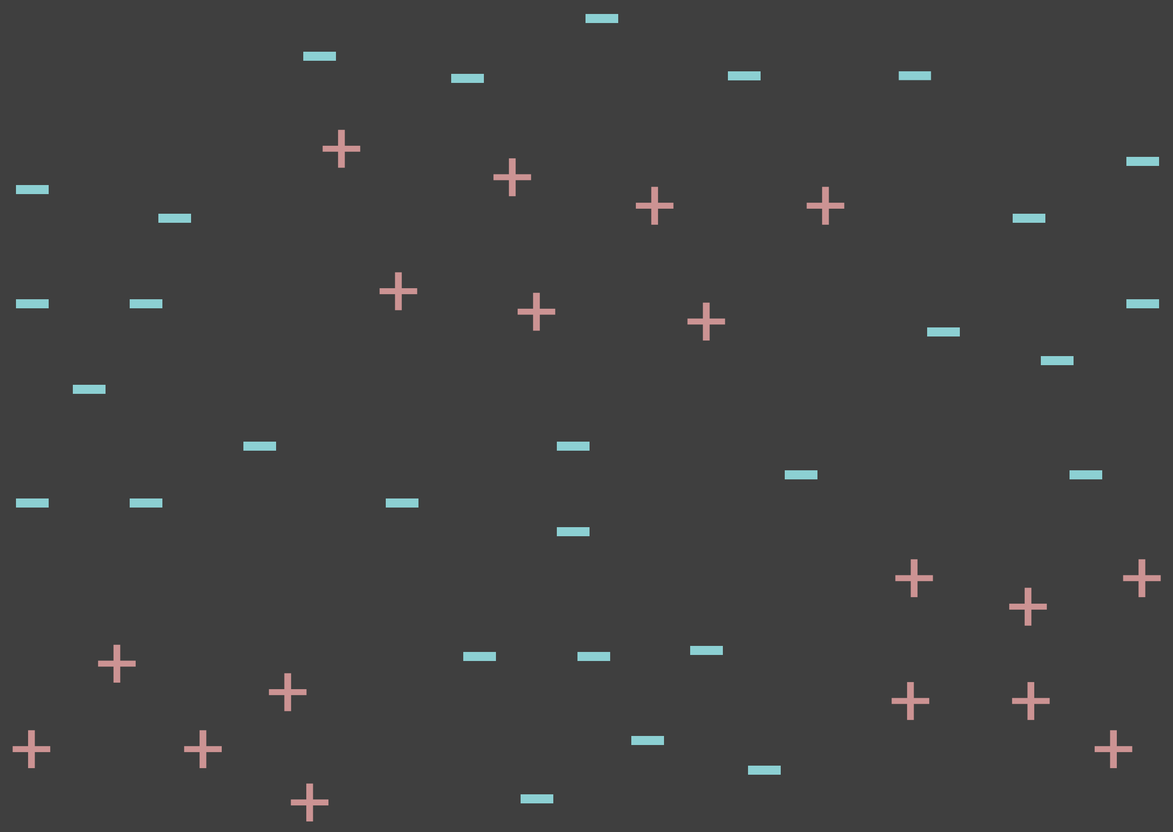
序贯覆盖到底是什么？

1. 分治（divide-and-conquer）？
2. 集成（ensemble）
   - Boosting
   - 加权序贯覆盖（weighted covering）
     - γ^C，C为被覆盖的次数
     - $\frac{1}{C+1}$

自底向上的方法

```
function RISE(E)
    # P/N为正/负样例
    P, N = E

    # 生成bottom rules, 即只覆盖一个正样例的规则
    R = bottom_rules(E)

    while true
        for r in R
            # 在P中找到:
            # 1. 距离规则r最近（Hamming）的
            # 2. 未覆盖的正样例
            e = select_nearest_uncovered_neighbor(P, r)

            # 泛化规则，令r_1能覆盖r和e
            r_1 = generalize(r, e)

            # 如果新规则不导致准确率下降，则保留
            R_1 = R - r + r_1
            if ! accuracy(R_1) < accuracy(R)
                R = R_1
```

自底向上的方法

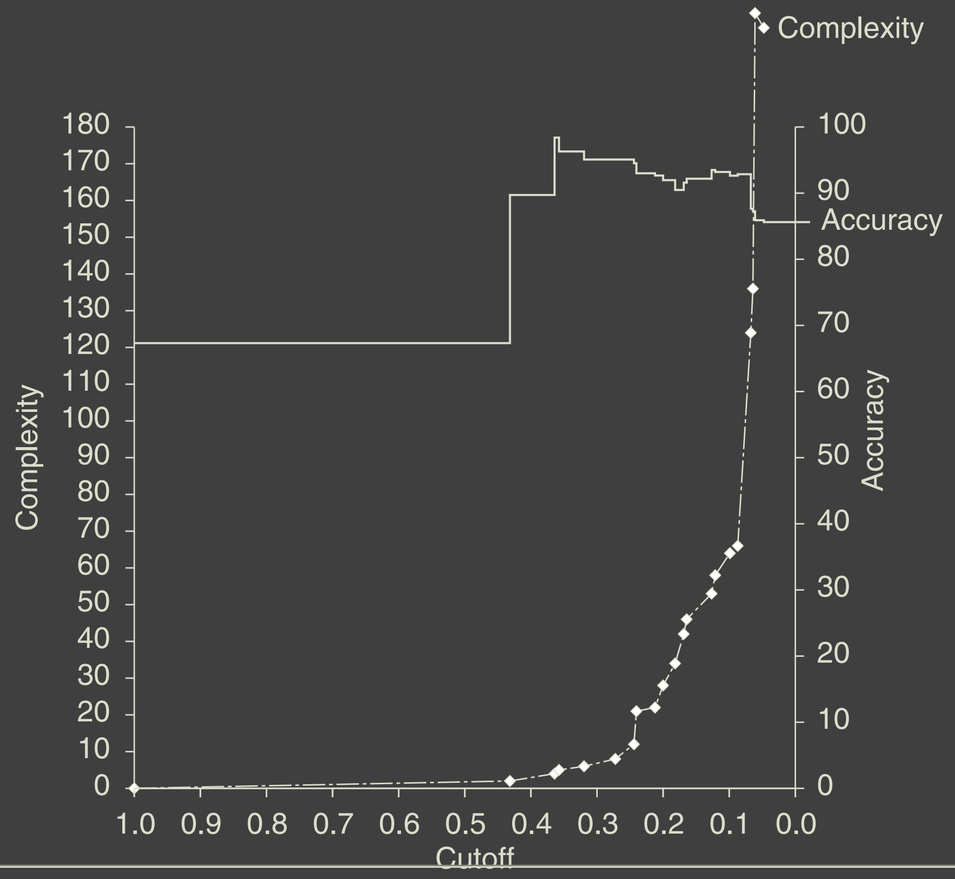


符号学习简介·命题规则（下）

1. 命题规则集
2. 规则集学习
3. **规则集剪枝**
4. 命题规则集的局限

过拟合

与所有统计学习模型一样，在有噪声的数据中学习可能造成命题规则集**过拟合**（**overfitting**）



剪枝方法

与决策树一样，可分为：

- 预剪枝（pre-pruning）
- 后剪枝（post-pruning）

预剪枝

利用启发式函数约束，在规则（集）生长时及时停止

- 最小覆盖样例个数
  - 正样例个数
  - 总样例个数
- 当前最佳规则的精度（纯度）
- 规则复杂度阈值
  - 描述覆盖样本需要的信息量：$\log_2(P+N)+\log_2\binom{P+N}{\hat{P}}$
    - P, N为训练集正负样例个数，$\hat{P}$为规则覆盖的正样例个数
  - 描述规则本身需要的信息量：$\log_2\binom{F}{L_r}$
    - F为特征个数，L_r为规则里的文字个数

后剪枝操作

对于已学得的规则集进行以下行为，并评估打分函数

- 删除任意文字
- 删除最差文字
- 删除规则最后的文字
- 删除规则的最后文字序列
- 删除一条规则
- ……

后剪枝策略

REP [Brunk and Pazzani, 1991]和IREP [Fürnkranz and Widmer, 1994]

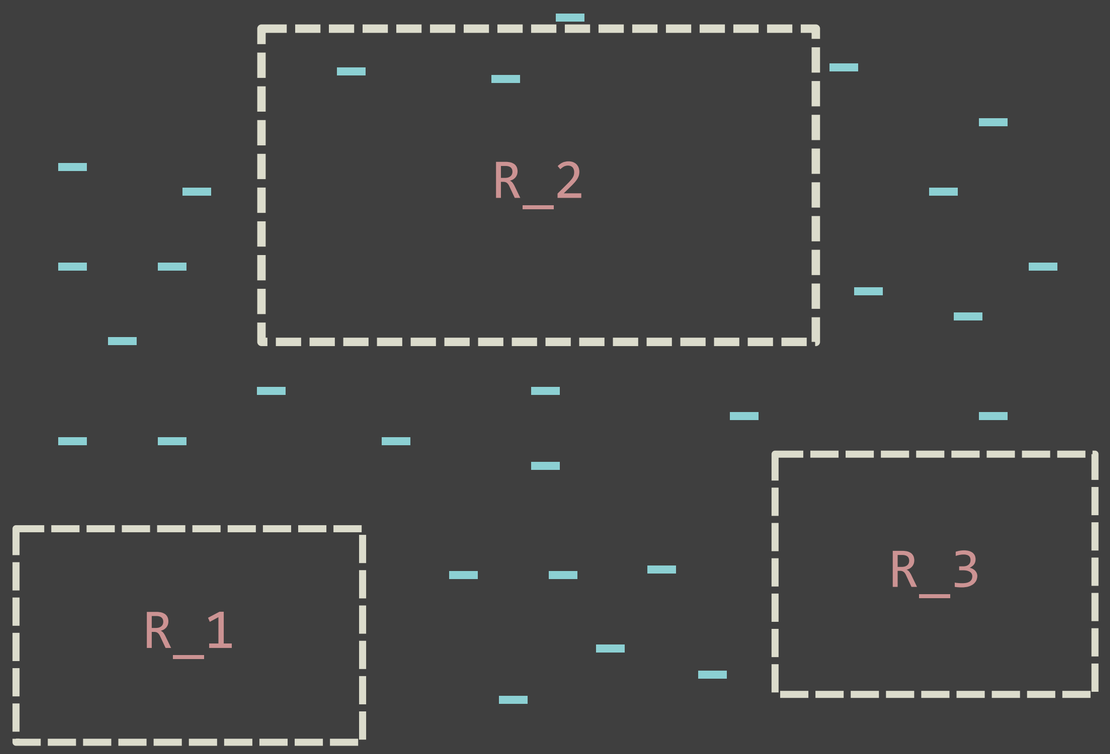
- 减错剪枝（Reduced Error Pruning, REP）
  - 利用以上操作，对规则集进行后剪枝，最小化验证集上的误差
  - 最差情况下复杂度为$O(|E|^4)$
- 递增减错剪枝（Incremental REP）
  - 每生成一条规则，即刻对其进行剪枝

后剪枝中加入预剪枝

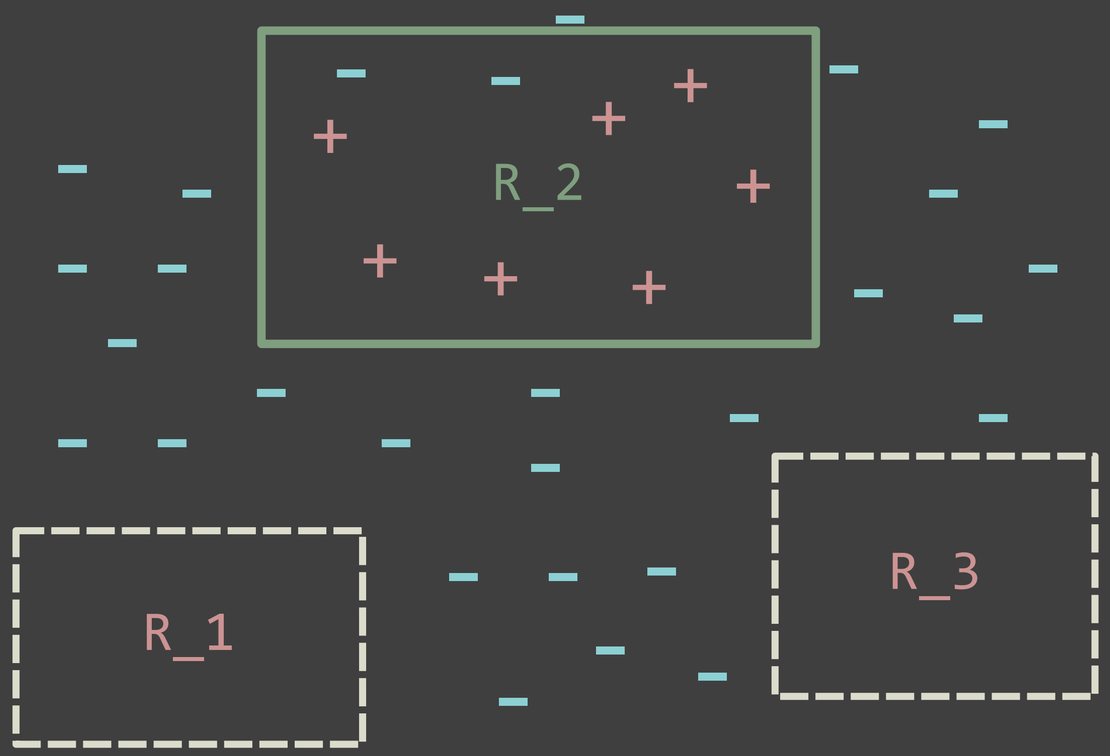
- RIPPER [Cohen, 1995]

RIPPER

I. 生成规则集

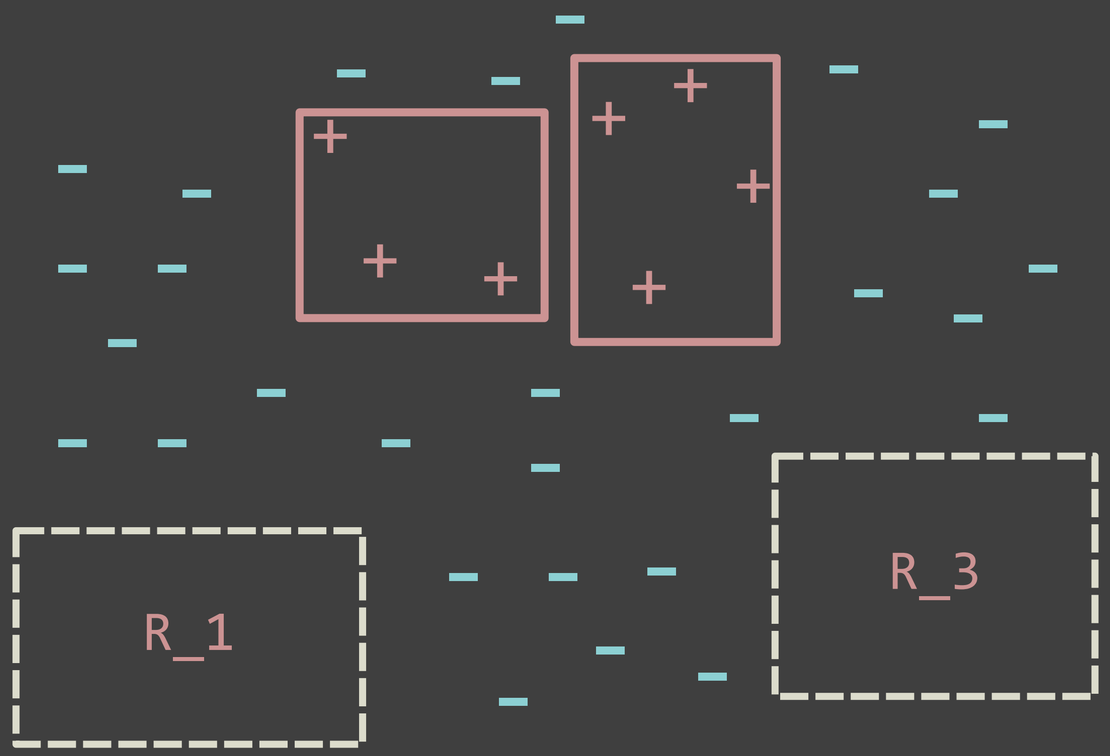


RIPPER



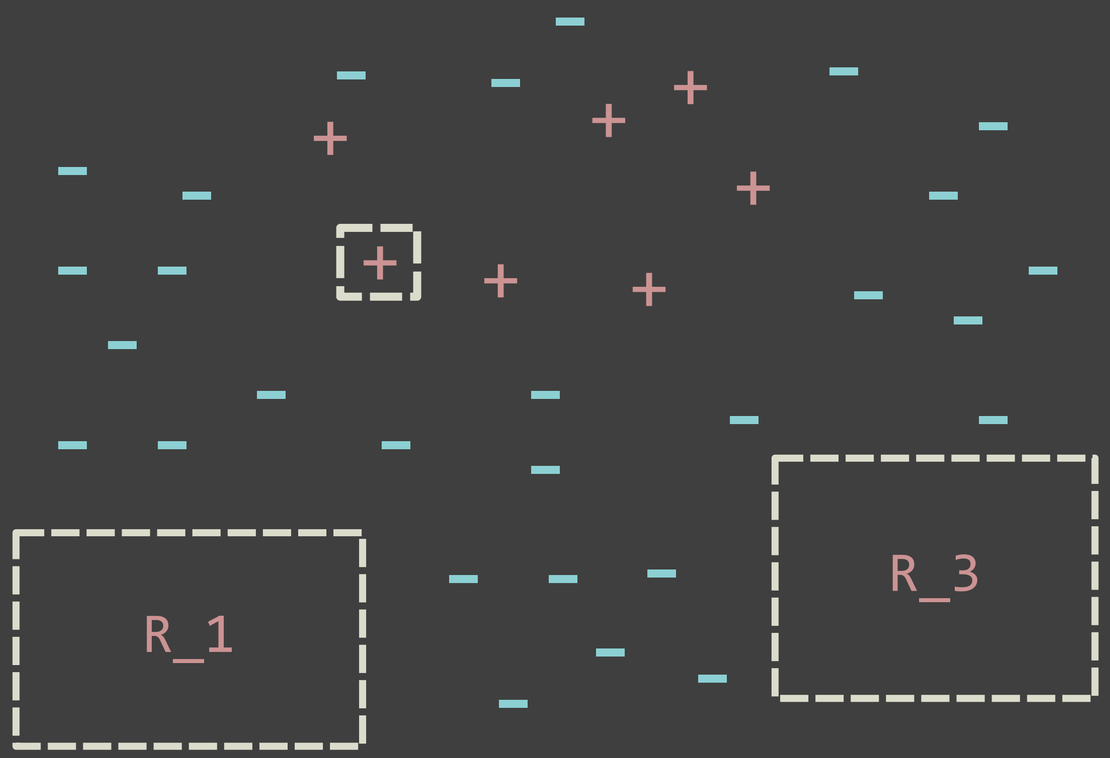
1. 生成规则集
2. 选取一条规则，找到其覆盖的样本

RIPPER



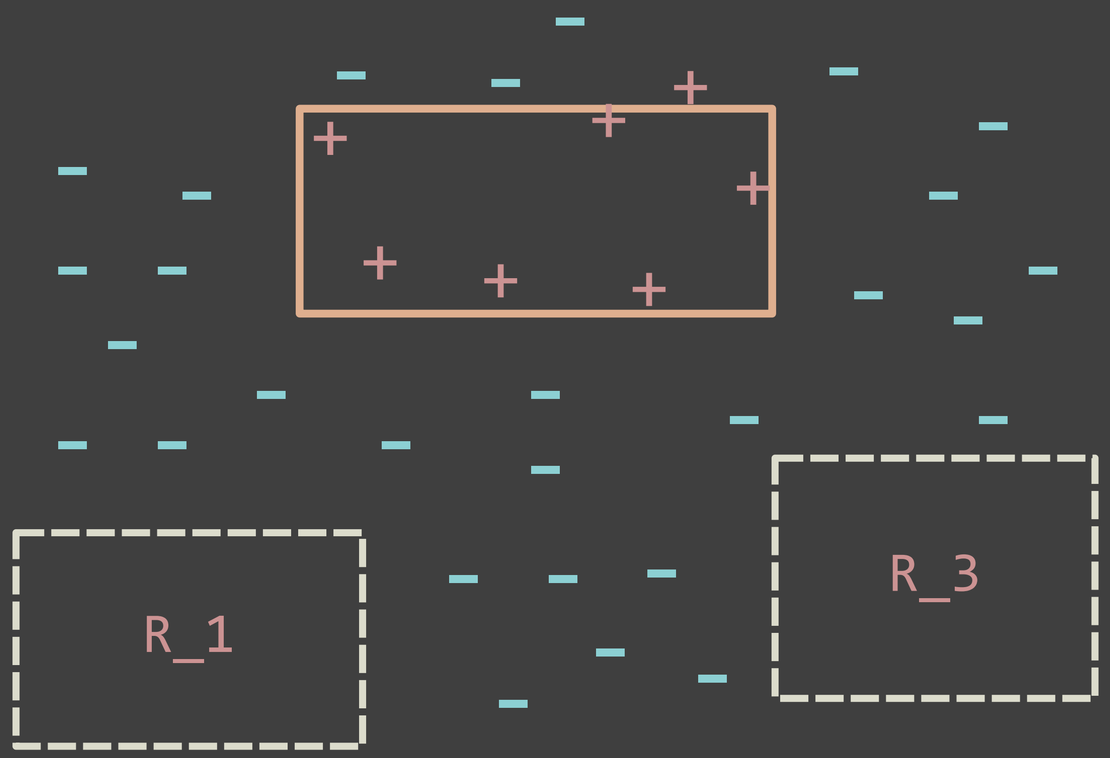
1. 生成规则集
2. 选取一条规则，找到其覆盖的样本
   - » 重新生成规则

RIPPER



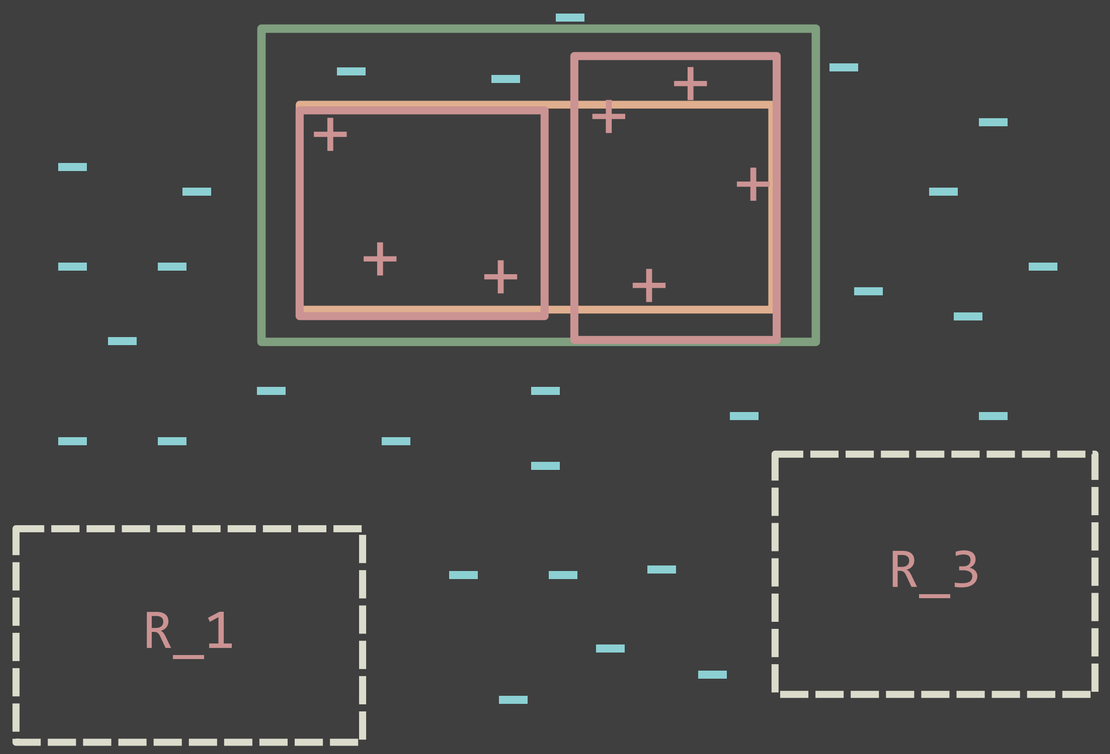
1. 生成规则集
2. 选取一条规则，找到其覆盖的样本
   - » 重新生成规则
   - » 特化规则再泛化

RIPPER



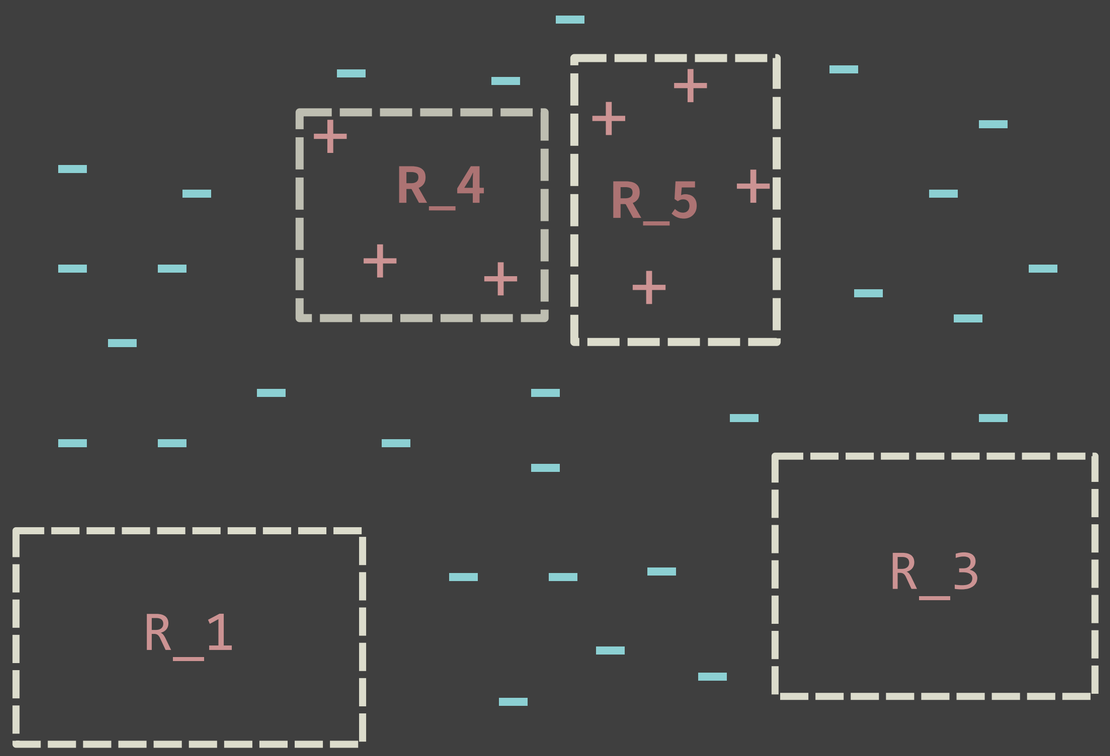
1. 生成规则集
2. 选取一条规则，找到其覆盖的样本
   - » 重新生成规则
   - » 特化规则再泛化

RIPPER



1. 生成规则集
2. 选取一条规则，找到其覆盖的样本
   - » 重新生成规则
   - » 特化规则再泛化
3. 把原规则和新规则分别置入规则集进行评价，留下最好的

RIPPER



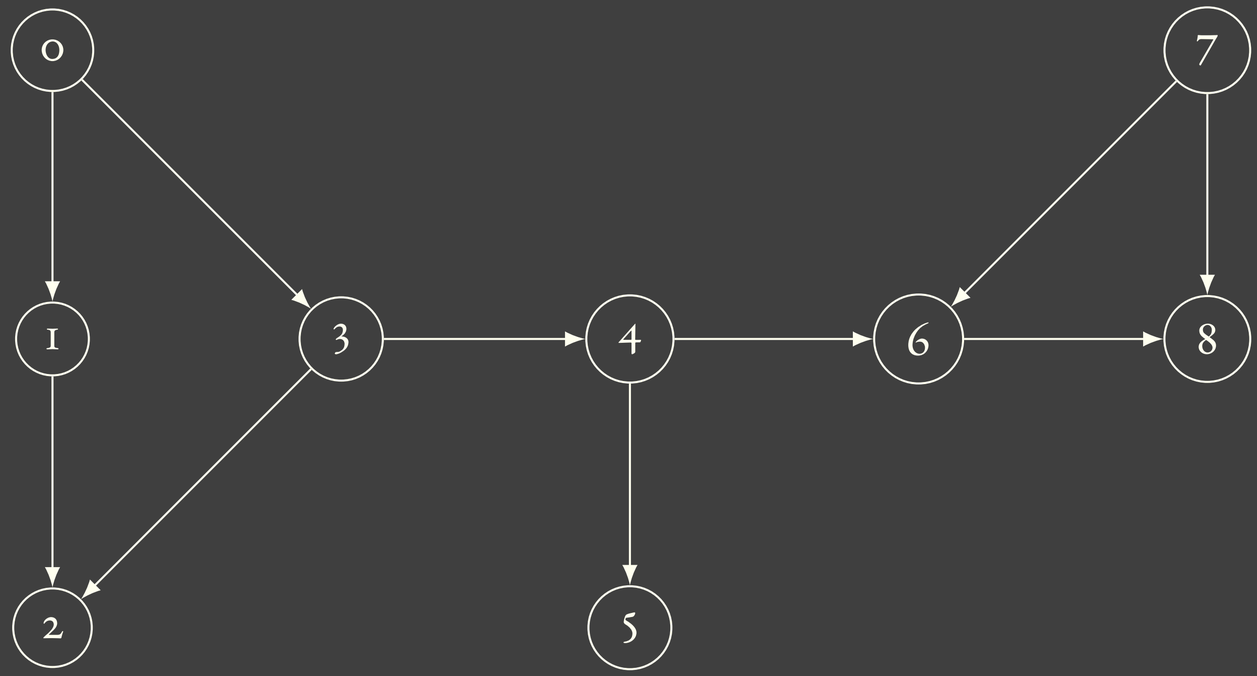
1. 生成规则集
2. 选取一条规则，找到其覆盖的样本
   - » 重新生成规则
   - » 特化规则再泛化
3. 把原规则和新规则分别置入规则集进行评价，留下最好的
4. 反复优化，直到性能不再提升

符号学习简介·命题规则（下）

1. 命题规则集
2. 规则集学习
3. 规则集剪枝
4. **命题规则集的局限**

一个例子

"A small network illustrating the limitations of prepositional descriptions." [Quinlan, 1990]



如何定义一组能够用来描述任意网络结构的命题特征？

一个例子

假设网络：

- 至多有10个节点
- 每个节点的出度至多为3

可以这样设计：

- A_1, B_1, C_1：节点1连接的点
- A_2, B_2, C_2：节点2连接的点
- ……

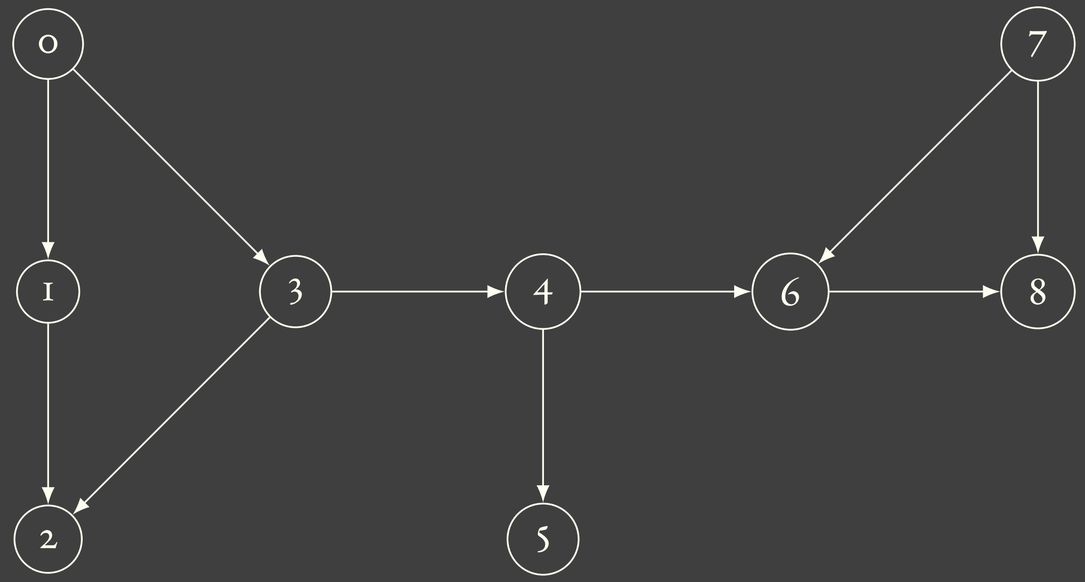
一个例子

需要学习的目标概念：**“两个节点相接”**

$$
\begin{aligned}
& [(A_1 = 2 \vee B_1 = 2 \vee C_1 = 2) \wedge (A_2 = 1 \vee B_2 = 1 \vee C_2 = 1)] \\
\vee\ & [(A_1 = 3 \vee B_1 = 3 \vee C_1 = 3) \wedge (A_3 = 1 \vee B_3 = 1 \vee C_3 = 1)] \\
\vee\ & [(A_1 = 4 \vee B_1 = 4 \vee C_1 = 4) \wedge (A_4 = 1 \vee B_4 = 1 \vee C_4 = 1)] \\
& \cdots \\
\vee\ & [(A_2 = 3 \vee B_2 = 3 \vee C_2 = 3) \wedge (A_3 = 2 \vee B_3 = 2 \vee C_3 = 2)] \\
\vee\ & [(A_2 = 4 \vee B_2 = 4 \vee C_2 = 4) \wedge (A_4 = 2 \vee B_4 = 2 \vee C_4 = 2)] \\
& \cdots
\end{aligned}
$$

"Horrific." [Quinlan, 1990]

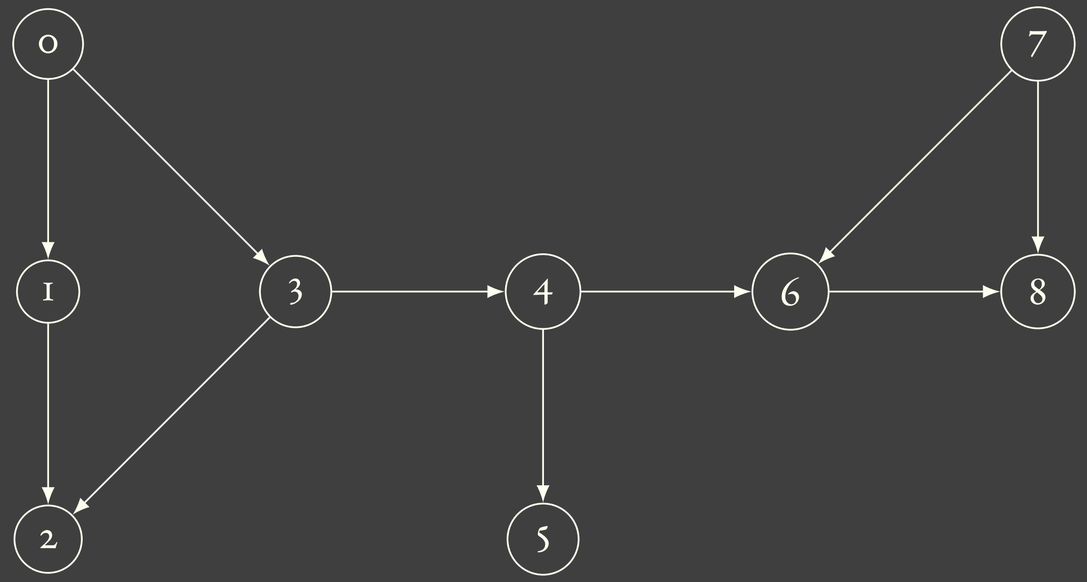
一阶逻辑表示



只需要用一个谓词：linked-to(X, Y)

$$\begin{aligned}\text{linked-to} = \{ &\langle 0,1\rangle, \langle 0,3\rangle, \langle 1,2\rangle, \langle 3,2\rangle, \langle 3,4\rangle \\ &\langle 4,5\rangle, \langle 4,6\rangle, \langle 6,8\rangle, \langle 7,6\rangle, \langle 7,8\rangle\}\end{aligned}$$

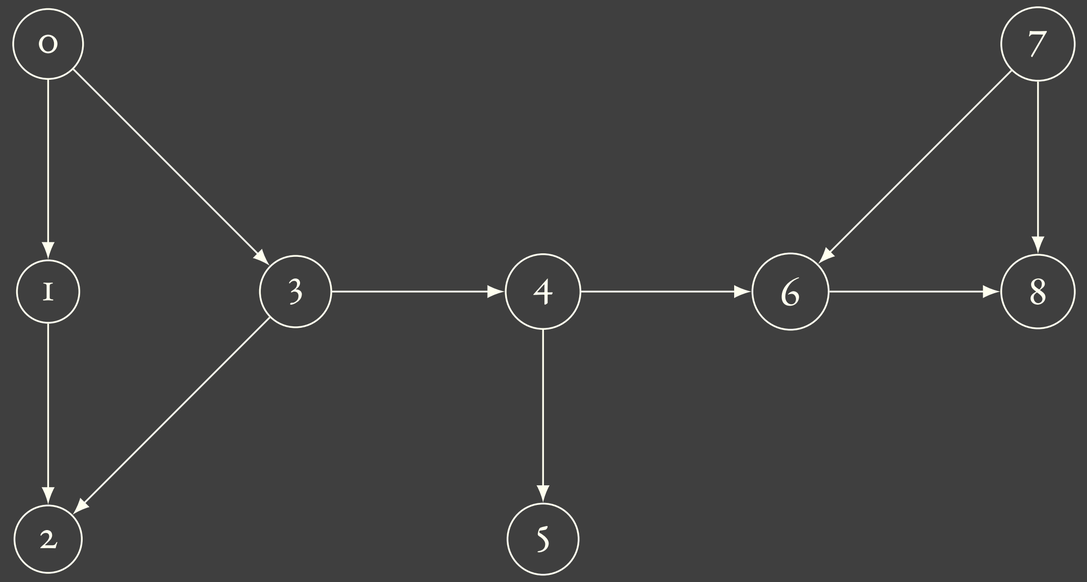
一阶逻辑表示



“两个节点相接”：

$$\text{linked-to-each-other}(X, Y) \leftarrow \text{linked-to}(X, Y) \wedge \text{linked-to}(Y, X)$$

一阶逻辑表示



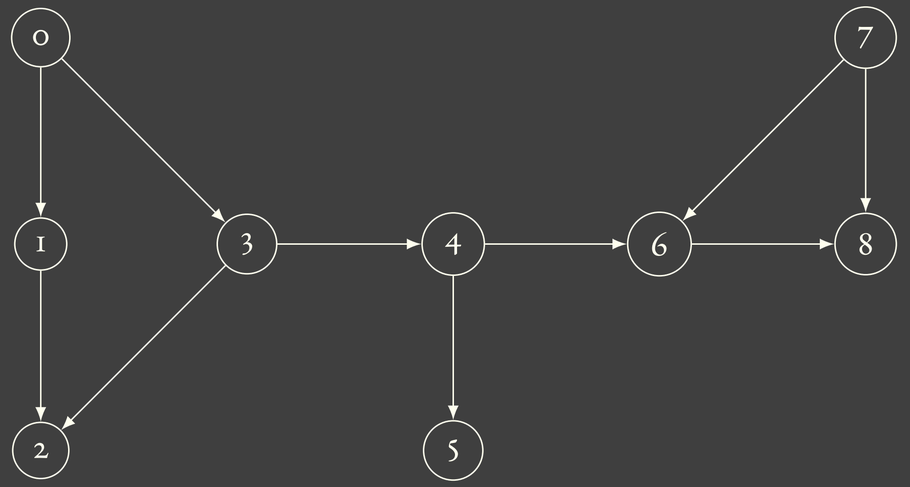
“两个节点连通”：

$$\text{can-reach}(X, Y) \leftarrow \text{linked-to}(X, Y)$$
$$\text{can-reach}(X, Y) \leftarrow \text{can-reach}(X, Z) \wedge \text{linked-to}(Z, Y)$$

FOIL的规则集学习

1. 构建可能的一阶逻辑文字
2. 自顶向下生成规则
3. 序贯覆盖生成规则集

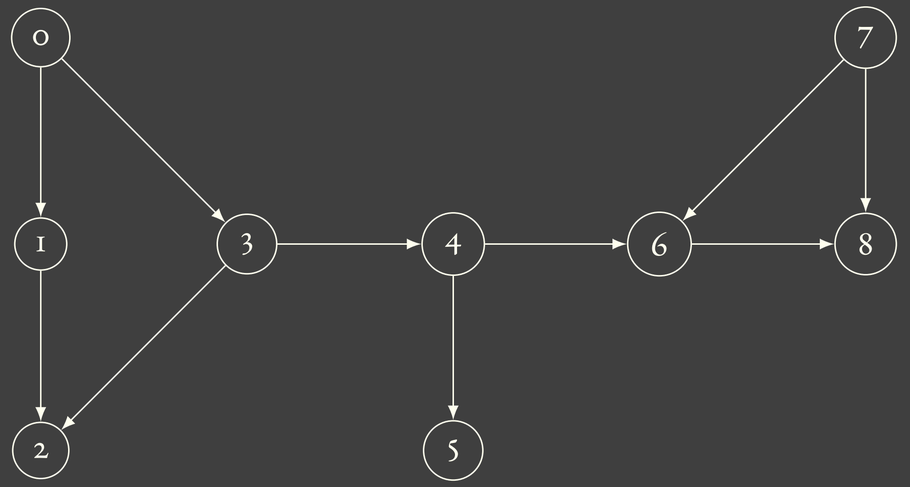
FOIL的规则集学习



“两个节点连通”：

- 正例：$\langle 0,1\rangle, \langle 0,2\rangle, \langle 0,3\rangle, \langle 0,4\rangle, \langle 0,5\rangle, \langle 0,6\rangle, \langle 0,8\rangle, \ldots$
- 负例：$\langle 0,0\rangle, \langle 0,7\rangle, \langle 1,0\rangle, \langle 1,1\rangle, \langle 1,3\rangle, \langle 1,4\rangle, \langle 1,5\rangle, \ldots$

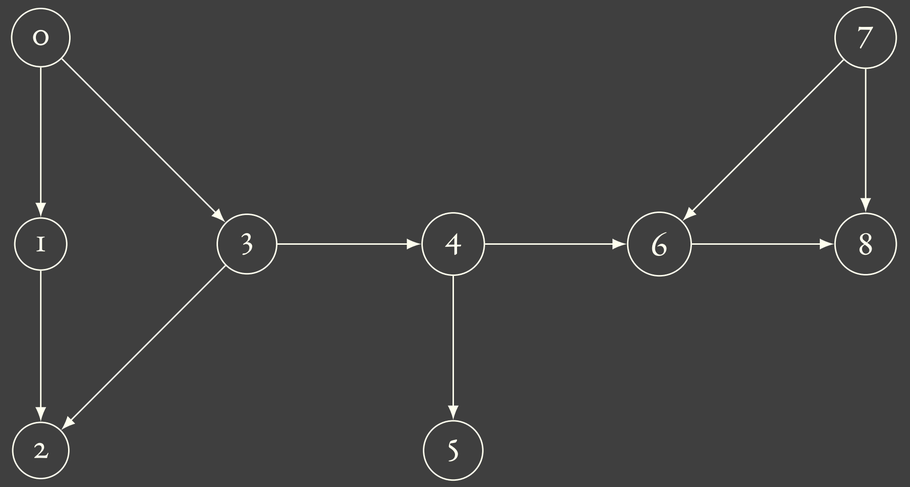
FOIL的规则集学习



初始规则1：$\text{can-reach}(X, Y) \leftarrow$

- 正例：$\langle 0,1\rangle, \langle 0,2\rangle, \langle 0,3\rangle, \langle 0,4\rangle, \langle 0,5\rangle, \langle 0,6\rangle, \langle 0,8\rangle, \ldots$
- 负例：$\langle 0,0\rangle, \langle 0,7\rangle, \langle 1,0\rangle, \langle 1,1\rangle, \langle 1,3\rangle, \langle 1,4\rangle, \langle 1,5\rangle, \ldots$

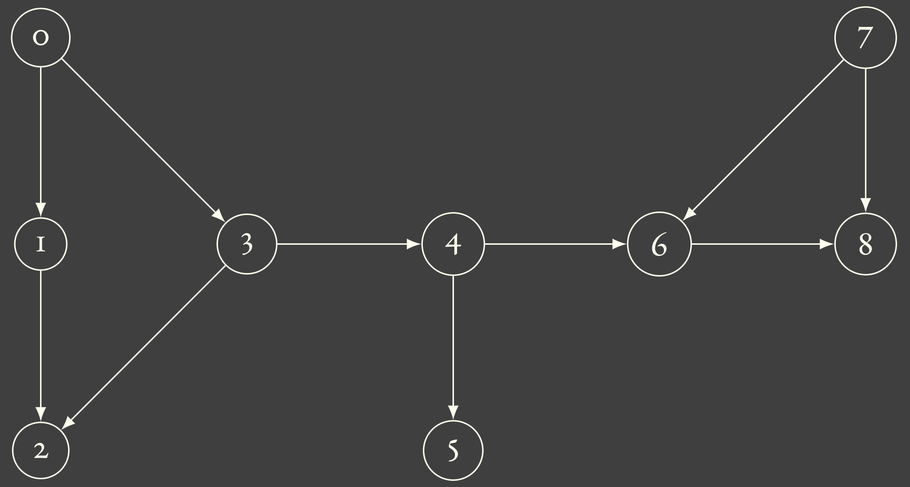
FOIL的规则集学习



加入文字：$\text{can-reach}(X, Y) \leftarrow \text{linked-to}(X, Y)$

- 正例：
  - 覆盖：$\langle 0,1\rangle, \langle 0,3\rangle, \langle 1,2\rangle, \langle 3,2\rangle, \langle 3,4\rangle, \langle 4,5\rangle, \ldots$
  - **未覆盖**：$\langle 0,2\rangle, \langle 0,4\rangle, \langle 0,5\rangle, \langle 0,6\rangle, \langle 0,8\rangle, \langle 3,5\rangle, \langle 3,6\rangle, \ldots$
- 不覆盖负例

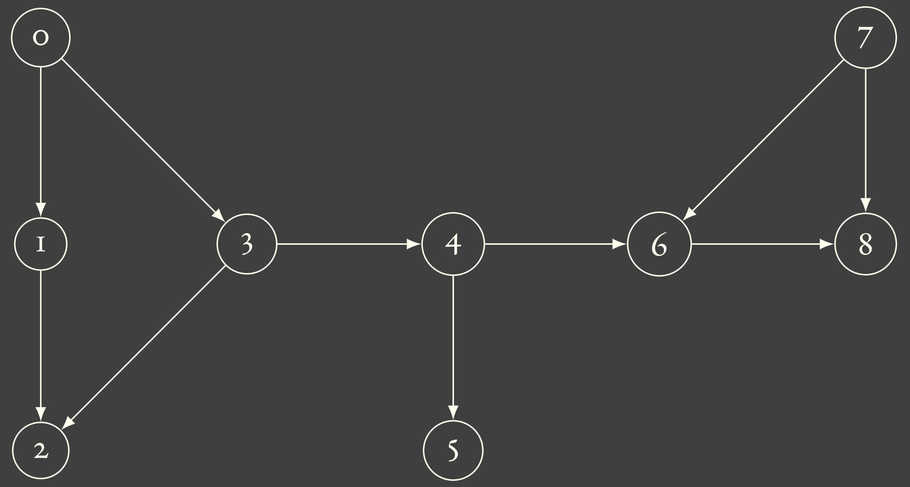
FOIL的规则集学习



初始规则2：$\text{can-reach}(X, Y) \leftarrow$

- 正例：$\langle 0,2\rangle, \langle 0,4\rangle, \langle 0,5\rangle, \langle 0,6\rangle, \langle 0,8\rangle, \langle 3,5\rangle, \langle 3,6\rangle, \ldots$
- 负例：$\langle 0,0\rangle, \langle 0,7\rangle, \langle 1,0\rangle, \langle 1,1\rangle, \langle 1,3\rangle, \langle 1,4\rangle, \langle 1,5\rangle, \ldots$

FOIL的规则集学习

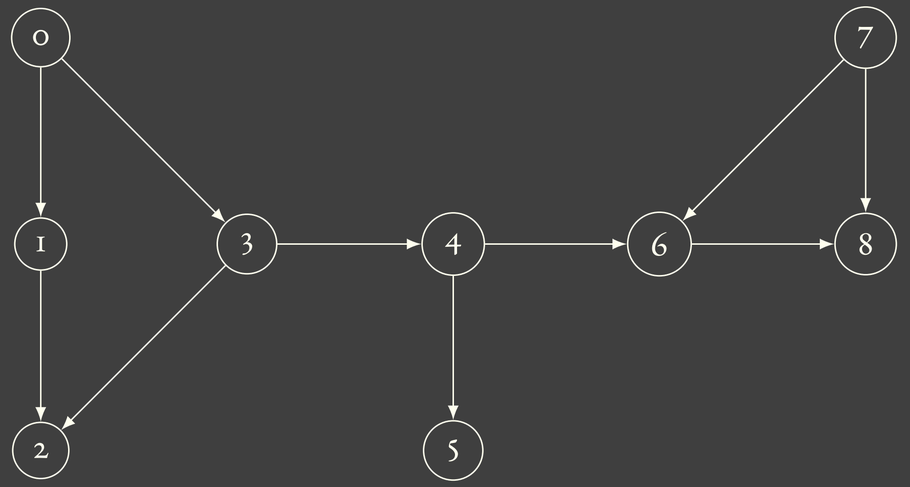


加入文字：$\text{can-reach}(X, Y) \leftarrow \text{linked-to}(X, Z)$

包含变量$\langle X, Y, Z\rangle$，扩展样例：

- 覆盖正例：$\langle 0,2,1\rangle, \langle 0,2,3\rangle, \langle 0,4,1\rangle, \langle 0,4,3\rangle, \langle 0,5,1\rangle, \langle 0,5,3\rangle, \langle 0,6,1\rangle, \ldots$
- **覆盖负例**：$\langle 0,0,1\rangle, \langle 0,0,3\rangle, \langle 0,7,1\rangle, \langle 0,7,3\rangle, \langle 1,0,2\rangle, \langle 1,1,2\rangle, \langle 1,3,2\rangle, \ldots$

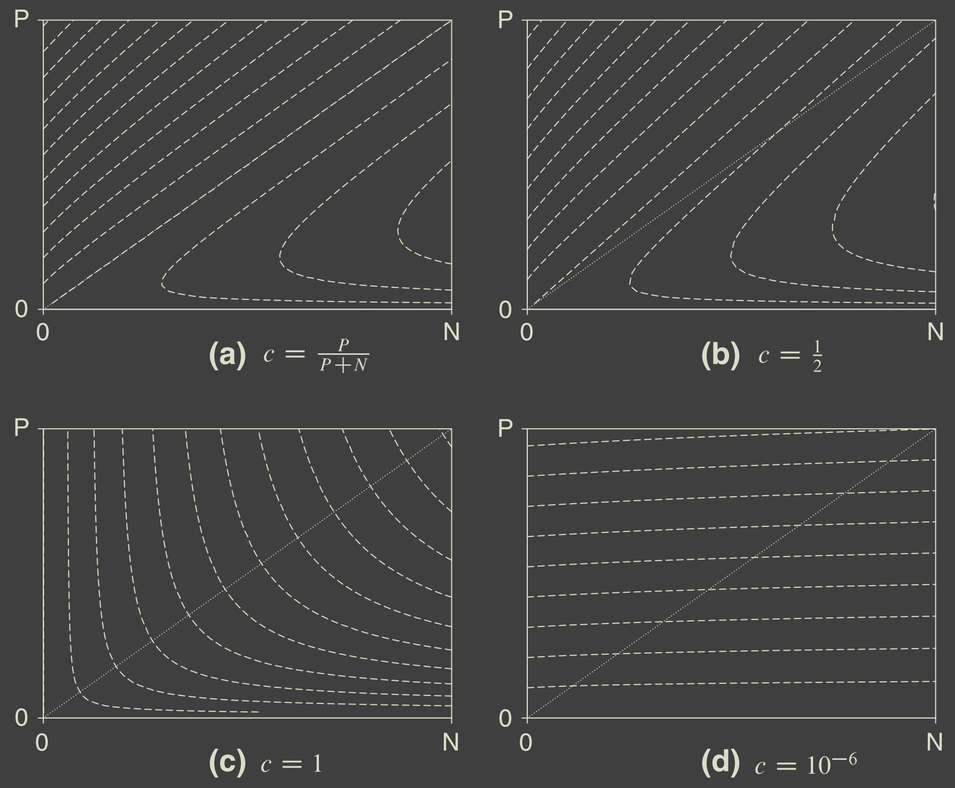
FOIL的规则集学习



加入文字：$\text{can-reach}(X, Y) \leftarrow \text{linked-to}(X, Z) \wedge \text{can-reach}(Z, Y)$

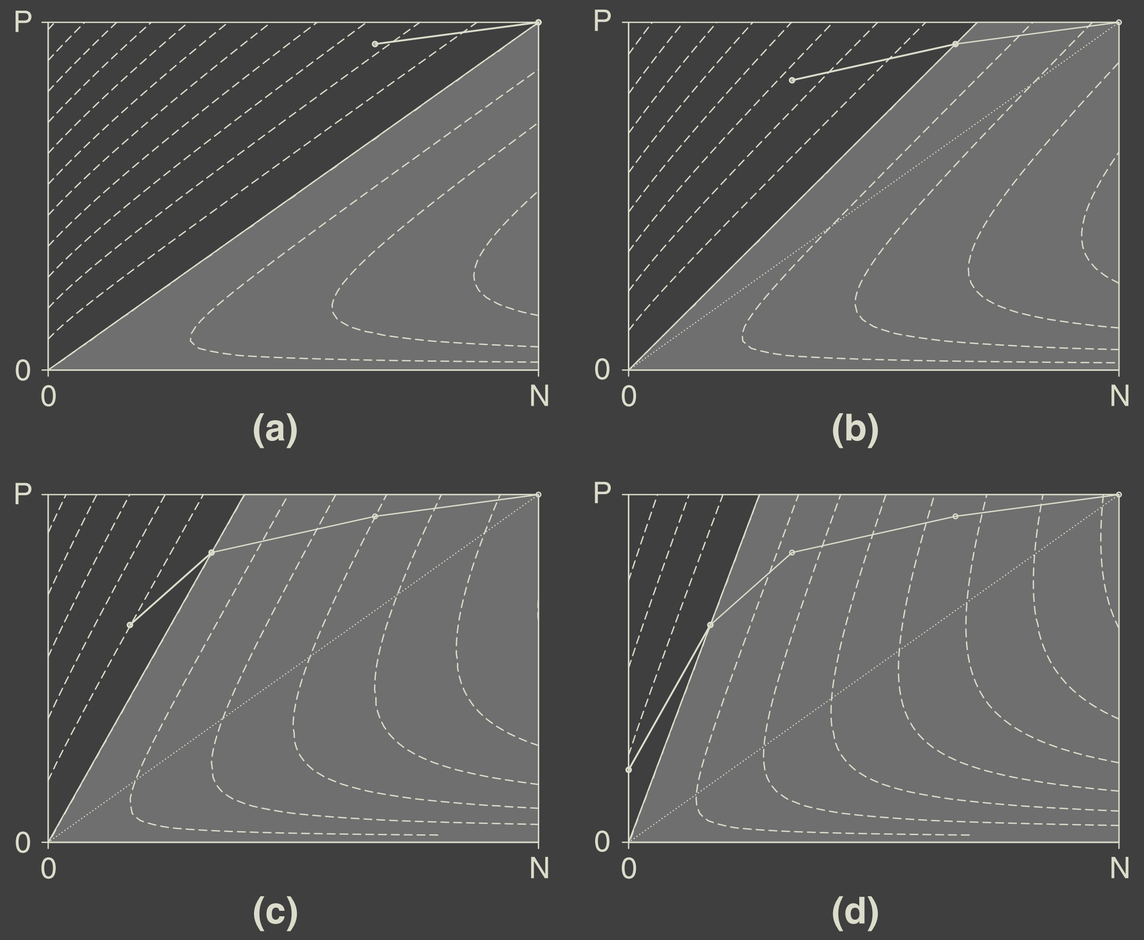
- 仅覆盖所有剩余正例：$\langle 0, 2, 1\rangle, \langle 0, 2, 3\rangle, \langle 0, 4, 3\rangle, \langle 0, 5, 3\rangle, \langle 0, 6, 3\rangle, \langle 0, 8, 3\rangle, \ldots$
- 不覆盖负例
- 终止学习，进入后处理（剪枝）

FOIL的打分函数



$$\text{正样本覆盖率} \cdot (\text{信息量} - \text{精化前信息量}) = \hat{P} \cdot \left(\log_2 \frac{\hat{P}}{\hat{P} + \hat{N}} - \log_2 c \right)$$

使用FOIL增益的规则精化



小结

命题规则（下）小结

1. **命题规则集**是由命题逻辑规则构成的集合
   - » 是一种特殊的集成模型
2. 命题规则集的**搜索**
   - » 可与Boosting类比
3. 规则集的剪枝
4. 命题规则的表示能力十分有限
   - » 思考：FOIL的优缺点？