


当前进度汇报：21-01-18

1. 一个新的搜索方法

这段时间我找到了一种在Inductive Learning的搜索空间进行类似“度下降”的方式，算法如下：

Input:

1. 知识库 B
2. 覆盖率阈值： c_{min}

Output: 具有最大压缩率的规则 r_{max}

$heap \leftarrow$ 初始化一个最大堆

For each predicate p in B **do**:

$r \leftarrow$ a new rule $\langle p(X_0, \dots, X_{\phi(p)}), \emptyset \rangle$ // $\phi(p)$ 表示谓词 p 的参数数量，这一行的意思是将 B 中每一个谓词都作为一条新的规则的head加入备选

$heap.add(r, score(r))$

End For

While $heap$ is not empty **do**:

$r \leftarrow heap.poll()$ // 取出当前得分最高的rule

If r 是一条完整的规则 **AND** r 起到了压缩的效果 **then**:

Return r

End If

For each r_e extends r **do**:

If r_e 形式上不包含trivial的部分 **AND** $\rho(r) > c_{min}$ **AND** $score(r_e) > score(r)$ **then**:

$heap.add(r_e, score(r_e))$

End If

End For

End While

Return Null

1.1. 规则的拓展

对于一条规则 r ，我们定义其拓展是下面两种形式之一：

1. 拓展一个已知的变量：

即：1) 在 r 的body中当前还没有确定变量的参数设置为已知变量; 2) 或添加一个新的谓词，并在新谓词中的未确定参数位置尝试已知变量。

例：

假设 r 为： $h(X, Y) \leftarrow p(? , X)$

其中“?”表示还未确定变量的参数，假设 B 中还存在一个谓词 $q/1$ ，则 r 的已知变量的拓展有：

- $h(X, Y) \leftarrow p(X, X)$
- $h(X, Y) \leftarrow p(Y, X)$
- $h(X, Y) \leftarrow p(? , X), h(X, ?)$
- $h(X, Y) \leftarrow p(? , X), h(Y, ?)$
- $h(X, Y) \leftarrow p(? , X), h(? , X)$
- $h(X, Y) \leftarrow p(? , X), h(? , Y)$
- $h(X, Y) \leftarrow p(? , X), q(X)$
- $h(X, Y) \leftarrow p(? , X), q(Y)$

2. 拓展一个新的变量：

拓展一个新的变量的时候，如果仅在一个未知参数处拓展，则和原本的规则在查询时等价，因为未确定的参数实际上代表一个独立的自由变量。

还是举上述的例子 r ，在考虑其效果的时候做的查询等价于以下规则：

$h(X, Y) \leftarrow p(Z, X)$

其中 Z 就是一个独立的自由变量。此时在增加新变量的时候新的规则就和原规则等价，不存在梯度，因此一定要在两处未知参数处进行拓展。这两处未知参数可以是 r 中的，也可以是在添加了一个新的谓词之后的。在上例中有以下新变量拓展：

- $h(X, Y) \leftarrow p(Z, X), h(Z, ?)$
- $h(X, Y) \leftarrow p(Z, X), h(?, Z)$
- $h(X, Y) \leftarrow p(?, X), h(Z, Z)$
- $h(X, Y) \leftarrow p(Z, X), q(Z)$

1.2. Score Metric

评价 r 的好坏的标准有两个：

$$1. \text{ 压缩率: } \tau(r) = \frac{|E_r^+|}{|E_r^+| + |E_r^-|}$$

其中 E_r^+ 表示 r 可以推出的正例， E_r^- 表示 r 推出的负例（也就是 r 的副作用）

$$2. \text{ 覆盖率: } \rho(r) = \frac{|E_r^+|}{|E_p^+|}$$

其中 E_p^+ 表示谓词 p 的所有正例， p 就是 r 的head的谓词

这两个标准都是越大越好，但是往往不可兼得。目前在上述算法中，用来作为“梯度”的指标是压缩率，即：

$$\text{score}(r) = \tau(r)$$

1.3. 算法中的几点判别条件

1. 完整的规则：指 r 符合generative, closed的形式要求
2. 起到了压缩的效果： $\tau(r) > 0.5$
3. 形式上不包含trivial的部分：即body中不包含和head相同谓词且参数列表相同的term，否则这样的 r 是一条trivial的规则，对归纳本身没有任何用处。

例如： $sibling(X, Y) \leftarrow aunt(Z, X), sibling(X, Y), aunt(Z, Y)$

这条规则的head和body中就都包含了 $sibling(X, Y)$ 。另外，这种形式也将表达对称性或者参数旋转等价性的规则一并排除在外，因为目前来看，这些形式的规则并没有什么意义。

2. 这个搜索过程的几个性质

2.1. Completeness

即：当前的搜索方式可以遍历整个搜索空间（或者说搜索空间中有意义的部分）。

Proof: 对于任意规则 r ，可以找到这样的构造序列：在序列分为两部分，第一部分逐次将head中的变量在body中的出现的位置构造出来，剩下的位置暂时未知；在序列的第二部分，逐次添加仅在body中出现的变量，当这样的变量第一次出现的时候，同时添加两个，之后可以每次添加一个。按照这样的构造方式，即可构造出 r ，且这个序列中的每一步操作都在前文关于拓展的定义范围之内。

例如规则 $h(X, Y) \leftarrow p(X, Z), q(Z, Z, W), r(W, Y)$ 按照上述流程可以找到序列：

- 第一部分：
 - $h(X, Y) \leftarrow$
 - $h(X, Y) \leftarrow p(X, ?)$
 - $h(X, Y) \leftarrow p(X, ?), r(? , Y)$
- 第二部分：
 - $h(X, Y) \leftarrow p(X, Z), r(? , Y), q(Z, ? , ?)$
 - $h(X, Y) \leftarrow p(X, Z), r(? , Y), q(Z, Z, ?)$
 - $h(X, Y) \leftarrow p(X, Z), r(W, Y), q(Z, Z, W)$

□

从另一个角度讲，它不会引入一些无关的谓词，比如和body中其他谓词以及head的变量集完全无关的片段，这种片段是没有任何意义的，它们的引入只有两种结果：

- 片段在 B 中为true，则对剩余部分完全没有影响
- 片段在 B 中为false，则整条规则不会推出任何结果

比如规则： $h(X, Y) \leftarrow p(X), q(Y), n(Z, W), m(W, Z)$

其中的谓词 $n(Z, W), m(W, Z)$ 就是完全无关的片段，它们的存在毫无必要，虽然整条规则是符合约束的，但是却毫无意义。

2.2. Necessity

即：Ground Truth中的规则在这种搜索方式中一定存在一条路径，使得路径上的每一步都有一个向上的梯度。

Remark: 这个证明我还没想好具体怎么证，但是从感觉上来说应该是有很大概率成立的。只要知识库 B 足够完善，Completeness中的那种构造序列其实就基本符合这个要求。因为在这种构造方式下，每次添加新的变量约束（不论是用已知变量还是未知变量），都是对原有约束条件的细化，在这种逐步求精的方式下，每次排除的负例的比例应该可以高于被排除的正例的比例，使得压缩率在每一步都增大。但是具体怎么证可能需要用概率说明了。

当这个性质成立的时候，整个搜索空间在当前的搜索方式下就使得Ground Truth中的每一条规则都至少是一个Local Optimal。

2.3. Length Sensitive

即：在这种搜索方式下，理论上不需要预设一个关于规则长度的阈值限制。

Proof: 因为 B 是有限的，因此每一个 E_r^+ 和 E_r^- 都是有限的，每次都要求 $\tau(r_e)$ 严格大于 $\tau(r)$ ，则搜索路径长度不可能大于 $|E_r^+| + |E_r^-|$ 。

□

需要说明的是，如果不设置搜索长度限制，AMIE是有可能不停机的（一直在添加dangling term）。

3. 这个搜索过程的缺点

3.1. Local Optimal

这个问题根本原因是当前的梯度只能作为目标规则的必要条件而不是充分条件。虽然Ground Truth中的所有规则都是Local Optimal，但是反之不成立，因此在当前实验的运行过程中出现了一些质量并不太好的局部最优解。

例如，在手动构造的家庭关系知识库中，有*sibling*, *brother*, *sister*三个谓词。手动解释*sibling*的最优解为：

- $sibling(X, Y) \leftarrow brother(X, Y)$
- $sibling(X, Y) \leftarrow sister(X, Y)$

但是在实际运行中，则得出了局部最优解：

- $sibling(X, Y) \leftarrow aunt(Z, X), brother(X, Y), aunt(Z, Y)$

或

- $sibling(X, Y) \leftarrow aunt(Z, X), aunt(Z, Y)$

这种情况就是在取得了很高的压缩率的情况下，覆盖率却很低。

3.2. 尝试次数多

虽然当前的这个过程已经将一个纯剪枝的过程提升为了可以按照梯度的方式进行有向搜索的过程，但是不得不说对于一个较长的 r 的扩展是很多的，即使通过梯度排除了一些，但是剩下的也不少，而且计算梯度的过程中的数据库查询操作开销很高。

尝试次数多的另外一个可能原因是搜索存在重复，因为一条规则只要足够长，就存在不止一条符合搜索条件的构造序列，实际可以构造它的序列不会比这些更少。

目前考虑解决这个问题的思路有两条：

1. 寻找对得分的估计方法
2. 寻找一种更好的搜索过程，保持当前的性质，同时还能减少分支，减少重复

3.3. 搜索路径虽然不是无限的，但是有时候也很长

这个问题对于一些存在“传递”性质的谓词尤其明显，比如 $brother$ 关系，在实验中就曾出现过这样的规则：

$$brother(X, Y) \leftarrow brother(X, Z), brother(Z, W), brother(W, Y)$$

这个问题还没想好怎么解决。

3.4. 参数不能设置为常量

目前对于规则中可能出现的常量参数还没有考虑，后续可以考虑加入对这种参数的考察。但是常量对于知识库来说过多（比如我用来测试的数据库，1100个facts的时候常量就有142个，对于任意一条 r 来说，这都是比可以尝试的变量数量多好几个数量级的存在），可能需要通过一些预处理的方式进行剪枝才不会引入过多的分支。比如，对于一些参数的对应常量列表分布相对集中于少数几个常量的情况下才考虑把常量纳入考虑的范围。例如 $gender(X, Y)$ 的第二个参数基本只有 $male$ 和 $female$ 两个（其他的基本都可以看做数据的噪音），此时可以考虑这里的 Y 被替换为二者其一。而 $sibling(Z, W)$ 的两个参数的分布都很分散，不需要考虑替换为常量的可能性。

3.5. 对递归规则的处理还需要继续探索

现在的试验中还没有考虑通过递归的规则进行压缩的状况，这种情况还有待继续探索。