Fast Estimation of First-Order Clause Coverage through Randomization and Maximum Likelihood

Ondřej Kuželka and Filip Železný

Czech Technical University in Prague
Inductive Logic Programming

**ILP learns first-order logic rules, such as**

\[
\text{mutagenic}(C) \leftarrow \text{atom}(C, A_1) \land \text{atom}(C, A_2) \land \text{bond}(A_1, A_2)
\]

- Usually Horn clauses

**From examples, such as**

\[
\text{mutagenic}(c_{12}) \leftarrow \text{atom}(c_{12}, a_{121}) \land \text{carb}(a_{121}) \\
\land \text{atom}(c_{12}, a_{122}) \land \text{hydro}(a_{122}) \\
\land \text{bond}(a_{121}, a_{122}) \land \ldots
\]

- Usually (ground) Horn clauses
- ILP algos search for clauses *covering* many positive and few negative examples.
Coverage

- Clause *Coverage* (on set $E$) = number of examples from set $E$ covered by the clause
- Clause *covers* another clause if it logically implies it
- Implication between clauses $C \models D$ undecidable.
- If no recursion, $\models$ equivalent to $\theta$-subsumption

**$\theta$-subsumption**

Clause $C$ $\theta$-subsumes clause $D$ (written $C \leq_{\theta} D$) iff there exist a substitution $\theta$ such that $\text{lits}(C\theta) \subseteq \text{lits}(D)$.

- A generalization of the subgraph homomorphism relation
- NP-complete to verify
Θ-Subsumption Algorithm

- A basic CSP-inspired subsumption tester
- backtracking, forward checking, variable-selection heuristic (randomized tie-breaking), pruning

Subsumption check

```plaintext
Input: Clause C, example e;
if C ⊆ e then
    return YES
else
    Choose variable V from C using a heuristic function
    for ∀S ∈ PossibleSubs(V, C, e) do
        C' ← Substitute V with S
        if ∀W ∈ Adjacency(V) : PossibleSubs(W, C', e) ≠ ∅ then
            SearchedNodes ← SearchedNodes + 1
            if SubsumptionCheck(C', e) = YES then
                return YES
        end if
    end for
end if
return NO
```

Ondřej Kuželka and Filip Železný (Czech Technical University in Prague)
Basic vs. restarted version of the algorithm

- Restarts reduce runtimes in both regions
- Runtimes much larger in the NO region.
- Can we avoid (part of) the NO-region complexity?
Alleviating the Coverage Problem

- Standard ILP approach decides $C \preceq_\theta e$ or $C \not\preceq_\theta e$ for $\forall e \in E$
- Telling us more than we need to know.
- All we need to rank $C$’s is the number $A$ of subsumed examples
- Goal: obtain an estimate of $A$

**Restarted Coverage Tester (“ReCovEr”)**

```plaintext
for $i = 1, 2, \ldots k$ do
    for every $e \in E$ do
        run a single try (# of steps bounded by $\gamma$) of the subsumption check $C \preceq_\theta e$
        $M_i =$examples in $E$ proved to be subsumed, $m_i = |M_i|$
        $E := E \setminus M_i$
    end for
end for
RETURN $m_1, m_2, \ldots m_k$
```

- What does the sequence $m_1, m_2, \ldots m_k$ tell us about $A$?
Coverage estimation: Intuition

# of examples proved in try #1

0
Coverage estimation: Intuition

- Number of examples proved in try #1
- Actual number of subsumed examples

# of examples proved in try #1
Actual # of subsumed examples
Coverage estimation: Intuition

- Actual # of subsumed examples:
  - # of examples proved in try #1
  - # of examples proved in tries #1,2
- # of examples proved in try #1
Coverage estimation: Intuition

- Actual # of subsumed examples
- # of examples proved in tries #1,2
- # of examples proved in try #1

Ondřej Kuželka and Filip Železný (Czech TechFast Estimation of First-Order Clause Coverage)
Coverage estimation: Intuition

Actual # of subsumed examples

# of examples proved in tries #1,2

# of examples proved in try #1

Limit
Coverage estimation: Intuition

- Actual # of subsumed examples
- # of examples proved in tries #1,2
- # of examples proved in try #1
- Limit
- Extrapolate
- Estimate the limit

Ondřej Kuželka and Filip Železný (Czech TecFast Estimation of First-Order Clause Coverage)
Assumption

- Given clause $C$, example set $E$ and cutoff value $R$
- Probability $p$ that the subsumption test algorithm returns YES before exploring $R$ search space nodes is the same for all $e \in E$ such that $C \preceq_{\theta} e$
Coverage estimation: Formalization

- Probability of value $m_1$

\[
P(m_1) = \binom{A}{m_1} p^{m_1} (1 - p)^{A - m_1}
\]

where $A = |\{e \in E | C \leq_{\theta} e\}|$.

- Probability of $m_2$ given $m_1$

\[
P(m_2|m_1) = \binom{A - m_1}{m_2} p^{m_2} (1 - p)^{A - m_1 - m_2}
\]

- Generally for $m_i$

\[
P(m_i|m_{i-1}, \ldots, m_1) = \binom{A - \sum_{j=1}^{i-1} m_j}{m_i} p^{m_i} (1 - p)^{A - \sum_{j=1}^{i} m_j}
\]
Coverage estimation: Formalization

- Probability of obtaining sequence $m_1, m_2, \ldots m_k$

$$P(m_1, \ldots, m_k) = \prod_{i=1}^{k} P(m_i|m_{i-1}, \ldots, m_1)$$

- Substituting from previous slide and taking logarithm

$$\ln P(m_1, \ldots, m_k) = \sum_{i=1}^{k} \ln \left( A - \sum_{j=1}^{i-1} m_j \right)$$

$$+ m_i \ln p + \ln \left( A - \sum_{j=1}^{i-1} m_j \right)$$

- Looking for $p \in [0; 1]$ and $A \in \{1, 2, \ldots |E|\}$ maximizing this equation.
Setting \( \frac{\partial}{\partial p} \ln P(m_1, \ldots, m_k) = 0 \) yields

\[
 p_A = \frac{\sum_{i=1}^{k} m_i}{\sum_{i=1}^{k} m_i + \sum_{i=1}^{k} \left( A - \sum_{j=1}^{i} m_j \right)}
\]

For every \( A \in \{1, 2, \ldots |E|\} \)

- check the value \( P(m_1, \ldots, m_k) \) for all of \( p \in \{0, p_A, 1\} \)

The pair \( \hat{A}, \hat{p} \) yielding the maximum value is the maximum likelihood estimate of the true \( A \) and \( p \).

How to choose \( k \) in \( m_1, m_2, \ldots m_k \)? E.g. stop when \( |\hat{A}_{k-1} - \hat{A}_{k}| \leq \Delta \).
Experiments: Estimation precision

- Randomly generated clauses, *approx* scale-free and Erdos-Renyi graphs, 100 e’s × 1000 C’s

![Graphs showing real count vs. estimated count with different cutoffs and runtimes.]

- Comparison: Django (complete CSP-based subsumption algorithm [Maloberti, Sebag]): 483s
**Experiments: Learning Performance**

- RoCoVer and Django employed in a simple bottom-up learner

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Recover [s]</th>
<th>Django [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muta-v1</td>
<td>42</td>
<td>29</td>
</tr>
<tr>
<td>Muta-v2</td>
<td>513</td>
<td>1627</td>
</tr>
<tr>
<td>Muta-v3</td>
<td>1695</td>
<td>&gt;5h</td>
</tr>
<tr>
<td>CAD</td>
<td>121</td>
<td>&gt;2h</td>
</tr>
</tbody>
</table>

**Table:** Average runtimes
## Experiments: Learning Performance

### Table: Quality of learned hypotheses for RECOVER

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Avg. Precision</th>
<th>Avg. Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muta-v1</td>
<td>0.84</td>
<td>0.61</td>
</tr>
<tr>
<td>Muta-v2</td>
<td>0.81</td>
<td>0.65</td>
</tr>
<tr>
<td>Muta-v3</td>
<td>0.83</td>
<td>0.84</td>
</tr>
<tr>
<td>CAD</td>
<td>0.92</td>
<td>0.7</td>
</tr>
</tbody>
</table>

### Table: Quality of learned hypotheses for Django

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Avg. Precision</th>
<th>Avg. Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Muta-v1</td>
<td>0.86</td>
<td>0.6</td>
</tr>
<tr>
<td>Muta-v2</td>
<td>0.82</td>
<td>0.65</td>
</tr>
<tr>
<td>Muta-v3</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>CAD</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
</tbody>
</table>