Inductive Logic Programming Using a MaxSAT Solver

National Institute of Technology, Tokuyama College
Noriaki Chikara

Kyushu University
Miyuki Koshimura, Hiroshi Fujita, Ryuzo Hasegawa
# Table of Contents

- **Introduction**
- **Inductive Logic Programming**
  - Cover set algorithm
  - Restricting ILP
- **MaxSAT Encoding of ILP Problem**
  - Preprocessing
  - MaxSAT encoding
- **Experiment**
  - Data sets for the experiment.
  - Environment
  - Results
- **Conclusion**
- **Future works**
Introduction

• Inductive Logic Programming (ILP)
  – A method of inductive learning
  – ILP systems: Progol, Aleph, etc.
  – A wide variety of applications

• MaxSAT
  – Optimization version of Satisfiability Testing (SAT)
  – Satisfying clauses as many as possible.

  – A lot of progress in SAT/MaxSAT solvers
  – Remarkable success of many applications
ILP using a MaxSAT solver

Our motivation is to increase the performance by using the MaxSAT solver.

• We propose a method which transforms a problem of ILP into that of MaxSAT.
• Mimicing the search in Progol and Aleph.
• A syntactical restriction on the ILP problem.
• Preprocessing in order to prevent the size of the transformed problem growing up.
ILP using a MaxSAT Solver

Search Problem in ILP ➝ MaxSAT encoding ➝ MaxSAT Problem ➝ MaxSAT Solver solving ➝ Model ➝ decoding ➝ Extended Hypothesis
Inductive Logic Programming (ILP)

Given the background knowledge $B$, a set of positive examples $E^+$, and a set of negative examples $E^-$, which satisfy the following relations:

$$\begin{cases} B \not\models E^+ \\ B \cup E^- \not\models \square \end{cases}$$

an ILP system will derive a hypothesized logic program $H$ which satisfies the following relations:

$$\begin{cases} B \cup H \models E^+ \\ H \cup B \cup E^- \not\models \square \end{cases}$$
Cover set algorithm
(The algorithm of Aleph and Progol)

B is the background knowledge, H is hypotheses, and E is a set of positive examples. H is initialized to \( \emptyset \).

(1) If \( E = \emptyset \) then output \( H \).
(2) Let \( e \) be an example in \( E \).
(3) Generate a MSH from \( e \) and \( B \).
(4) Generate the best hypothesis \( H' \) with a **top-down search**.
(5) \( H := H \cup H' \).
(6) \( E' := \{ e' | e' \in E \ and \ B \cup H' \models e' \} \).
(7) Goto (1).

Using MaxSAT solver

(MSH: Most Specific Hypothesis)
Restricting ILP in this study

• Arguments in the predicate do not have structure.

• All predicates are required to having mode declarations

• We do not deal with negated atoms.
Preprocessing
Tree Structure of Hypothesis

Refinement operation of Hypothesis

head(A).

\[ \downarrow \text{refinement} \]

\[ \text{head}(A) : -p_1(A,B), p_2(A,C), p_3(A,D). \]

\[ \downarrow \text{refinement} \]

\[ \text{head}(A) : -p_1(A,B), p_2(A,C), p_3(A,D), p_4(B,E,F). \]

Causality graph of literals in the hypothesis

We have only to take care of its descendant nodes locally.

\[ \rightarrow \text{ Suppressing the size of MaxSAT encoding} \]
Hypothesis:

The Transformation to Tree structure
MaxSAT encoding

• \( r(i; j) \): the \( j \)-th literal of the \( i \)-th literal group in \( MSH \) appears in \( H' \).
• \( a(i; bki) \): the \( i \)-th literal group in \( MSH \) is satisfied. Through \( bki \), we can find out ground unit clauses in the background knowledge which are used for the satisfaction.
• \( e(ia; bkia; ib) \): the necessary condition for \( a(ia; bkia) \) is satisfied in the descendant nodes of the \( ib \)-th literal group in \( MSH \).
• \( p(ex) \): \( H' \) covers the \( ex \)-th positive example.
• \( n(ey) \): \( H' \) covers the \( ey \)-th negative example.

Soft Clauses:
- \(-r(i,j). \) \{weight:1\}
- \( p(ex). \) \{weight: (number of body literal in MSH)+1\}
head(A)
p(ex₁) = T : {A/ex₁}

e(head, ex₁, 1) = T : {A/ex₁}

p₁(A, B), p₂(A, C)
a(1, bki₉) = T : {p₁(A, B)/p₁(a₁, b₁), p₂(A, C)/p₂(a₁, c₁)}

e(1, bki₉, 3) \land e(1, bki₉, 4) \land r(1, 1) \land r(1, 2) \rightarrow a(1, bki₉)

e(1, bki₉, 3) = T : {B/b₁, C/c₁}
a(3, bki₉) \rightarrow e(1, bki₉, 3)

p₄(B, E, F), p₅(C, E)
a(3, bki₉) = T : {p₄(B, E, F)/p₄(b₁, e₁, f₁), p₅(C, E)/p₅(c₁, e₁)}

r(3, 1) \land r(3, 2) \rightarrow a(3, bki₉)

Most Specific Hypothesis (MSH):

Positive Example:
head(ex₁).
...

Background Knowledge:
p₁(a₁, b₁). p₂(a₁, c₁). p₃(a₁, d₁). p₄(b₁, e₁, f₁). p₅(c₁, e₁).
...

The case of
r(1, 1) = T, r(1, 2) = T, r(2, 1) = T, r(3, 1) = T, r(3, 2) = T, r(4, 1) = F

a(2, bki₉) = T : {p₃(A, D)/p₃(a₁, d₁)}

a(1, bki₉) = T : {p₁(A, B)/p₁(a₁, b₁), p₂(A, C)/p₂(a₁, c₁)}

\neg r(4, 1) \rightarrow e(1, bki₉, 4)

\neg r(4, 1) \rightarrow \neg a(4, bki)

p₆(C, G)
a(4, bki₉) = F

r(3, 1) \land r(3, 2) \rightarrow a(3, bki₉)
Data set of UCI for the Experiment

• **Connect-4**
  A two-player connection game. This database contains all legal 8-ply positions of the game.

• **Audiology(Standardized)**
  Nominal audiology dataset from Baylor.

• **Molecular Biology(Splice-junction Gene Sequences)**
  Arimate splice-junction gene sequences (DNA) with associated imperfect domain theory.
A Environment of the Experiment

MaxSAT Solver: QMaxSAT14.04
  based on Glucose3.0, on Cygwin1.7.28(64bit)

pre-processing and post-processing: Java
  java 1.8.40(64bit)
  Paser uses JavaCC

PC for Experiments:
  OS: Windows7 (64bit)
  CPU: Core i7-2620M(2.6GHz)
  Memory: 8GHz
  SSD
The execution time of the experiment

<table>
<thead>
<tr>
<th>Problem</th>
<th>Target of positive examples</th>
<th>Num. of extracted rules</th>
<th>Max. length of extracted rules</th>
<th>Our method (using MaxSAT)</th>
<th>Aleph+ YAProlong</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect-4</td>
<td>win</td>
<td>7</td>
<td>5 literals</td>
<td>1.501s</td>
<td>9.360s</td>
</tr>
<tr>
<td></td>
<td>draw</td>
<td>1</td>
<td>7 literals</td>
<td>0.877s</td>
<td>56.566s</td>
</tr>
<tr>
<td></td>
<td>loss</td>
<td>1</td>
<td>4 literals</td>
<td>0.970s</td>
<td>0.640s</td>
</tr>
<tr>
<td>Audiology (Standardized)</td>
<td>cochlear age</td>
<td>2</td>
<td>6 literals</td>
<td>2.624s</td>
<td>1m39.169s</td>
</tr>
<tr>
<td></td>
<td>cochlear age and noise</td>
<td>1</td>
<td>4 literals</td>
<td>2.530s</td>
<td>0.218s</td>
</tr>
<tr>
<td></td>
<td>cochlear unknown</td>
<td>4</td>
<td>6 literals</td>
<td>3.934s</td>
<td>10m47.807s</td>
</tr>
<tr>
<td></td>
<td>mixed cochlear age otitis media</td>
<td>1</td>
<td>3 literals</td>
<td>1.672s</td>
<td>0.187s</td>
</tr>
<tr>
<td></td>
<td>possible menieres</td>
<td>1</td>
<td>2 literals</td>
<td>1.438s</td>
<td>0.172s</td>
</tr>
<tr>
<td>Molecular Biology(Splice-junction Gene Sequences)</td>
<td>EI</td>
<td>4</td>
<td>5 literals</td>
<td>4.418s</td>
<td>0.328s</td>
</tr>
<tr>
<td></td>
<td>Neither</td>
<td>14</td>
<td>4 literals</td>
<td>1m4.166s</td>
<td>0.905s</td>
</tr>
</tbody>
</table>
Conclusion

• We proposed a new method that transforms a problem of ILP into that of MaxSAT.
• We converted an ILP problem to a tree structure, thereby we suppressed the size of MaxSAT encoding.
• Experimental results show that Our method works fairly well.

Future works:
• We apply the method to other ILP problems and to evaluate the performance.
• We speed up I/O of MaxSAT instances.