Using Binary Decision Diagrams to Enumerate Inductive Logic Programming Solutions

Hikaru Shindo*, Masaaki Nishino**, Akihiro Yamamoto*

September 4, 2018

* Graduate School of Informatics, Kyoto University
** NTT Communication Science Laboratories
Abstract

• We propose an efficient algorithm for enumerating solutions of Inductive Logic Programming problem with Binary Decision Diagrams.
  • Basic formalization of ILP allows many potential solutions, and we might miss important solutions. ➞ Enumeration is fundamental technique to avoid such missing.
  • Key idea: We use Binary Decision Diagram for enumeration.
    • Binary Decision Diagram (BDD) is a directed acyclic graph representing compactly a Boolean function.

• We show how to build recursively a Binary Decision Diagram that represents the set of solutions.
# Table of contents

1. Introduction

2. Binary Decision Diagram and Enumeration of Solutions

3. Applications

4. Experiments

5. Conclusion and Future work
Introduction
Motivation

• ILP system generate solutions for given positive examples and negative examples. On the view point of logic, a lot of candidates of solutions might be generated.

• Every ILP system choose some appropriate solutions based on some criteria or its search method.

Example

\[ \mathcal{E}^+ = \{p(a)\}, \quad \Rightarrow \quad \Sigma = \{p(a)\}, \]
\[ \mathcal{E}^- = \{p(b)\}, \quad \Sigma = \{p(x) \leftarrow q(x), q(a)\}, \]
\[ \mathcal{B} = \{\} \]

We call the solution of ILP problem as hypothesis.
Fundamental idea: Enumeration of hypotheses

Enumeration of hypotheses is keeping all hypotheses.

Merits of the enumeration:

1. **Preventing hypothesis omission**
   The importance of a hypothesis depends on the case, so algorithms that give only one hypothesis may not return the best hypothesis.

2. **Hypothesis selection**
   Users can select a hypothesis or compare some hypotheses using an evaluation function.

3. **Online-learning**
   We can efficiently perform online learning, i.e., updating the current set of hypothesis when new examples are added.
• We assume that a finite set of clauses that can be an element of hypotheses is given explicitly.
  • Even in that finite space, enumerating all hypotheses naively is an implausible task because there are a serious amount of candidate hypotheses.

• To treat such large scale sets of hypotheses, we use Binary Decision Diagrams (BDDs) that give compressed representation of hypotheses for enumeration.

• In this work, we developed an efficient recursive algorithm for constructing a BDD.
Contribution

- An efficient algorithm for enumerating hypotheses using BDDs.
- The class of ILP problems that we can apply our algorithm.
- An efficient algorithm to get the best hypothesis with an evaluation function.
- We empirically show that our method can be applied to real data.
Binary Decision Diagram and Enumeration of Solutions
A Binary Decision Diagram (BDD) is a directed acyclic graph that represents a Boolean function.

BDD that represents $F(x_0, x_1, x_2) = (x_0 \land x_1) \lor x_2$

Binary operations between BDDs can be executed efficiently. For example, given two BDDs representing logical functions $F$ and $G$, then the BDD representing $H = F \land G$ can be computed in time linear to $F$ and $G$ sizes.
Inductive Logic Programming

In Inductive Logic Programming (ILP), all data, background knowledge, and hypotheses are represented by first-order logic.

ILP Problem

<table>
<thead>
<tr>
<th>Input</th>
<th>Finite sets $\mathcal{E}^+$, $\mathcal{E}^-$, and $\mathcal{B}$ of ground atoms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output</td>
<td>A set of definite clauses $\Sigma$ such that</td>
</tr>
<tr>
<td></td>
<td>1. for all $A \in \mathcal{E}^+$ $\Sigma \cup \mathcal{B} \models A$</td>
</tr>
<tr>
<td></td>
<td>2. for all $A \in \mathcal{E}^-$ $\Sigma \cup \mathcal{B} \not\models A$</td>
</tr>
</tbody>
</table>

Example

$\mathcal{E}^+ = \{p(a)\}, \mathcal{E}^- = \{p(b)\}, \mathcal{B} = \{\}$

$\Sigma = \{p(a)\}, \{p(x) \leftarrow q(x), q(a)\}, \ldots$
Using BDDs for enumerating ILP solutions

- To enumerate ILP hypotheses with BDDs, we introduce Boolean variables, because BDD is a representation of a Boolean function.

- Boolean variables make the hypothesis enumeration problem equivalent to the problem of identifying a Boolean function.

- Hypothesis space $\mathcal{H}$ is a finite set of clauses that can be an element of the hypothesis. We assume that $\mathcal{H}$ is given explicitly.

For each clause $C \in \mathcal{H}$, we introduce a propositional variable $v_{C \in \Sigma}$ that becomes true if and only if clause $C \in \Sigma$.

For readability, we represent $[C \in \Sigma]$ instead of $v_{C \in \Sigma}$.

$$C \in \Sigma \iff [C \in \Sigma] = T.$$  

(1)
Building a BDD that represents hypotheses

We define $F_A$ as a BDD that represents the Boolean function that becomes true if and only if $\Sigma \cup \mathcal{B} \models A$.

Then, a BDD that represents the set of hypotheses is

$$\bigwedge_{A \in \mathcal{E}^+} F_A \land \bigwedge_{A \in \mathcal{E}^-} \neg F_A.$$

Example

Given:

$$\mathcal{E}^+ = \{p(a)\}, \mathcal{E}^- = \{p(b)\}, \mathcal{B} = \{\}.$$ 

The BDD to be built:

$$F_{p(a)} \land \neg F_{p(b)} =$$
$I_C$: the BDD that represents the Boolean variable $[C \in \Sigma]$

$BK_A$: the BDD that represents a constant that becomes true if and only if $A \in \mathcal{B}$.

Then $F_A$ for $A \in \mathcal{E}^+ \cup \mathcal{E}^-$ is recursively defined as

$$F_A = BK_A \lor \bigvee_{\begin{array}{c} C \in \mathcal{H} \\ \exists \theta \\ C \theta = A \leftarrow B_1 \land \ldots \land B_n \end{array}} \left( I_C \land \land F_{B_i} \right). \quad (2)$$

The right side of equation (2) represents the fact that $\Sigma \cup \mathcal{B} \models A$ if

1. $A \in \mathcal{B}$, or
2. $A$ is deduced by a substitution.
Solving ILP problem on the BDD

Example

Introduced variables:

① \([p(a) \in \Sigma]\), ② \([p(b) \in \Sigma]\), ③ \([q(a) \in \Sigma]\), ④ \([q(b) \in \Sigma]\), ⑤ \([p(x) \leftarrow q(x) \in \Sigma]\)

\[
F_{p(a)} = I_{p(a)} \lor (I_{p(x)} \leftarrow q(x) \land F_{q(a)})
\]

\[
F_{p(b)} = I_{p(b)} \lor (I_{p(x)} \leftarrow q(x) \land F_{q(b)})
\]
Solving ILP problem on the BDD

Problem

\[ \mathcal{E}^+ = \{ p(a) \}, \mathcal{E}^- = \{ p(b) \}, \mathcal{B} = \{ \} , \]
\[ \mathcal{H} = \left\{ p(a), p(b), q(a), q(b), p(x) \leftarrow q(x) \right\} . \]

Introduced variables:

\[ \begin{align*}
\textcircled{0} [p(a) \in \Sigma] & \quad \textcircled{1} [p(b) \in \Sigma] \\
\textcircled{2} [q(a) \in \Sigma] & \quad \textcircled{3} [q(b) \in \Sigma] \\
\textcircled{4} [p(x) \leftarrow q(x) \in \Sigma] & \\
\end{align*} \]

Enumerated hypotheses:

\[ \Sigma = \{ p(a) \} \]
\[ \Sigma = \{ q(a), p(x) \leftarrow q(x) \} \]
\[ \vdots \]
Applications
Search for the best hypothesis

Introduced variables:

0. \([p(a) \in \Sigma]\)
1. \([p(b) \in \Sigma]\)
2. \([q(a) \in \Sigma]\)
3. \([q(b) \in \Sigma]\)
4. \([p(x) \leftarrow q(x) \in \Sigma]\)

Example

The hypothesis with minimum number of atoms:

\[\Sigma_{best} = \{p(a)\}\]

This corresponds to the minimum-weight path colored red.
Experiments
Classification of natural numbers

When \( n \) is even,

\[
\mathcal{E}^+ = \{e(0), e(s^2(0)), \ldots, e(s^n(0))\}, \\
\mathcal{E}^- = \{e(s(0)), e(s^3(0)), \ldots, e(s^{n+1}(0))\}.
\]

When \( n \) is odd,

\[
\mathcal{E}^+ = \{e(0), e(s^2(0)), \ldots, e(s^{n+1}(0))\}, \\
\mathcal{E}^- = \{e(s(0)), e(s^3(0)), \ldots, e(s^n(0))\}.
\]

**Example**

In the case of \( n = 1 \), \( \mathcal{E}^+, \mathcal{E}^-, \mathcal{B}, \) and \( \mathcal{H} \) are, respectively,

\[
\mathcal{E}^+ = \{e(0), e(s^2(0))\}, \quad \mathcal{E}^- = \{e(s(0))\}, \quad \mathcal{B} = \emptyset, \quad \text{and}
\]

\[
\mathcal{H} = \begin{cases} 
  e(0), & e(x), \\
  e(s(0)), & e(s(x)), \\
  e(s^2(0)), & e(s^2(x)), \\
  e(s(x)) \leftarrow e(x), & e(s^2(x)) \leftarrow e(x), \\
  e(s^2(x)) \leftarrow e(s(x)), & e(s^2(x)) \leftarrow e(s(x)) \land e(x)
\end{cases}.
\]
## Results

<table>
<thead>
<tr>
<th>$n$</th>
<th>variables</th>
<th>nodes</th>
<th>hypotheses</th>
<th>BDD construction time</th>
<th>best hypothesis search time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>8</td>
<td>28</td>
<td>7.56msec</td>
<td>0.62msec</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>14</td>
<td>192</td>
<td>9.63msec</td>
<td>0.68msec</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>27</td>
<td>$1.25 \times 10^7$</td>
<td>1.90 $\times$ 10msec</td>
<td>1.02msec</td>
</tr>
<tr>
<td>4</td>
<td>69</td>
<td>42</td>
<td>$1.31 \times 10^{13}$</td>
<td>3.08 $\times$ 10msec</td>
<td>1.16msec</td>
</tr>
<tr>
<td>5</td>
<td>134</td>
<td>69</td>
<td>$4.82 \times 10^{32}$</td>
<td>7.00 $\times$ 10msec</td>
<td>1.48msec</td>
</tr>
<tr>
<td>6</td>
<td>263</td>
<td>101</td>
<td>$9.77 \times 10^{63}$</td>
<td>$3.50 \times 10^2$msec</td>
<td>2.21msec</td>
</tr>
<tr>
<td>7</td>
<td>520</td>
<td>156</td>
<td>$2.26 \times 10^{141}$</td>
<td>$1.68 \times 10^3$msec</td>
<td>1.68msec</td>
</tr>
<tr>
<td>8</td>
<td>1033</td>
<td>219</td>
<td>$1.80 \times 10^{308}+$</td>
<td>$1.20 \times 10^4$msec</td>
<td>2.66msec</td>
</tr>
</tbody>
</table>

**Table 1:** The results of the natural number problem
Classification of real data

(1) Soybean(small)\(^1\) and (2) Shuttle Landing Control\(^2\) from UCI Machine Learning Repository\(^3\).

Target concept: \(D1, no\_auto\) respectively.

<table>
<thead>
<tr>
<th>Problem</th>
<th>variables</th>
<th>nodes</th>
<th>hypotheses</th>
<th>BDD construction time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soybean</td>
<td>2243</td>
<td>788498</td>
<td>(1.80 \times 10^{308}+)</td>
<td>13495msec</td>
</tr>
<tr>
<td>Shuttle</td>
<td>117</td>
<td>2345</td>
<td>(6.76 \times 10^{10})</td>
<td>30msec</td>
</tr>
</tbody>
</table>

**Table 2:** The results of real data problem

One of the best hypotheses found in problem of Soybean(small) is,

\[
\Sigma_{best} = \{\text{class}(x, D1) \leftarrow \text{stem\_canker}(x, above\_soil)\}.
\]

---

\(^1\)https://archive.ics.uci.edu/ml/datasets/soybean+(small)
\(^2\)https://archive.ics.uci.edu/ml/datasets/Shuttle+Landing+Control
\(^3\)http://archive.ics.uci.edu/ml/index.php
Conclusion and Future work
Conclusion

• We proposed a BDD-based method to enumerate hypotheses of an ILP.
• We showed that users can get the best hypothesis following an evaluation function from the constructed BDD.

Future Work

• Enumerating hypotheses that have some errors
• Combination with other ILP approaches
• Enumeration with other data structures
Hypothesis space is a finite set of clauses that can be an element of the hypothesis.

We assume that the hypothesis space is given explicitly, and it satisfies the following two requirements.

Requirement 1
The hypothesis space does not contain any mutually recursive clauses.

Requirement 2
The hypothesis space is variable-bounded.
Mutually recursive clauses

Let $\mathcal{H}$ be a hypothesis space that is a finite set of definite clauses. If a series of definite clauses $\{C_i \in \mathcal{H}\}_{i=0,...,n}$ and substitutions $\theta_1, \ldots, \theta_n$ exist, and they are expressed as

\[
C_1 \theta_1 = A \leftarrow \ldots \land X_1 \land \ldots,
\]
\[
C_2 \theta_2 = X_1 \leftarrow \ldots \land X_2 \land \ldots,
\]
\[\vdots\]
\[
C_n \theta_n = X_{n-1} \leftarrow \ldots \land A \land \ldots,
\]

then $C_1, C_2, \ldots, C_n$ are mutually recursive clauses.

Having no mutually recursive clauses ensures that we can trace all literals present in the hypothesis space in a finite number of steps.
Definite clause $A \leftarrow B_1 \land \ldots \land B_n$ is variable-bounded if $v(A) \supseteq v(B_i)$ ($i = 1, \ldots, n$), where $v(C)$ is the set of all variables in $C$. The hypothesis space $\mathcal{H}$ is variable-bounded if all $C \in \mathcal{H}$ are variable-bounded.

Being variable-bounded ensures that for a clause $C\theta = A \leftarrow B_1 \land \ldots \land B_n \in \mathcal{H}$, if $A$ has no variables, then $B_i$ ($i = 1, \ldots, n$) also has no variables.