Learning Disjunctive Logic Programs from Interpretation Transition

Yi Huang\textsuperscript{1,3}, Yisong Wang\textsuperscript{1,*}, Ying Zhang\textsuperscript{2}, and Mingyi Zhang\textsuperscript{2}

\textsuperscript{1}Guizhou University, Guiyang 550025, P. R. China
\textsuperscript{2}Guizhou Academy of sciences, Guiyang 550001, P. R. China
\textsuperscript{3}Chongqing University of Arts and Sciences, Chongqing 402160, P. R. China

Abstract. We present a new framework for learning disjunctive logic programs from interpretation transitions, called LFDT. It is a nontrivial extension to Inoue, Ribeiro and Sakama’s LF1T learning framework, which learns normal logic programs from interpretation transitions. Two resolutions for disjunctive rules are also presented and used in LFDT to simplify learned disjunctive rules.

1 Introduction

In machine learning and specifically inductive logic programming [1, 2], it is an important task to learn the dynamics of complex systems, such as Boolean networks. A Boolean network consist of a set of Boolean variables each of which has a Boolean function. It is a simple yet a powerful mathematical tool to describe dynamics of complex systems [3, 4].

Since a seminal work on representing Boolean networks by logic programs [5], there is increasing interest in approaching the task from the perspective of logic programming [6–8]. In particular, Inoue, Ribeiro and Sakama proposed a novel framework, named LF1T, for learning normal logic programs from interpretation transitions that are pairs \( \langle I, J \rangle \) of interpretations with \( T_P(I) = J \), where \( T_P \) is the immediate consequence operator for a normal logic program \( P \) [6].

It is well-known that disjunctive logic programs [9, 10] are substantially more expressive than normal logic programs at many aspects. To extend LF1T for learning disjunctive logic programs from interpretation transitions, we present a new immediate consequence operator \( T_d^P \) for a disjunctive logic program \( P \). Informally, \( T_d^P(I) \) consists of all minimal models of the heads of rules in \( P \) whose bodies are satisfied by \( I \).

Comparing with LF1T framework for learning normal logic programs, a nontrivial work in the paper is to handle with nondeterministic interpretation transitions, i.e., it is possible there are two interpretation transitions \( \langle I, J \rangle \) and \( \langle I, J' \rangle \) in an observation with \( J \neq J' \). Combining with two new proposed resolutions for disjunctive rules, we achieve the new framework for learning disjunctive logic programs from interpretation transitions, called LFDT. It is proved being both sound and complete.

* Corresponding author, yswang@gzu.edu.cn.
2 Disjunctive Logic Programs

We assume a underlying first-order language without proper function symbols $\mathcal{L}$ and denote its Herbrand base (the set of all ground atoms) by $\mathcal{A}$. We assume $\mathcal{A}$ is finite for our learning purpose.

A (disjunctive) logic program is a finite set of (disjunctive) rules of the form

$$A_1 \lor \cdots \lor A_k \leftarrow B_1 \land \cdots \land B_m \land \neg C_1 \land \cdots \land \neg C_n,$$

where $k \geq 1$, $m \geq 0$, $n \geq 0$ and $A_i$ (1 \leq i \leq k), $B_j$ (1 \leq j \leq m), and $C_s$ (1 \leq s \leq n) are atoms of $\mathcal{L}$.

For a rule $r$ of the form (1), the head of $r$, written $hd(r)$, is $A_1 \lor \cdots \lor A_k$; the body of $r$, written $bd(r)$, is the conjunction $B_1 \land \cdots \land B_m \land \neg C_1 \land \cdots \land \neg C_n$; the atoms occurring in the body of $r$ positively (resp. negatively) is denoted by $bd^+(r) = \{B_1, \ldots, B_m\}$ (resp. $bd^-(r) = \{C_1, \ldots, C_n\}$). If $k = 1$ then $r$ is normal. A normal logic program is a finite set of normal rules. Given a logic program $P$, we denote $hd(P) = \bigcup_{r \in P} hd(r)$ and $bd(P) = \bigcup_{r \in P} bd(r)$. For convenience, we also write $hd(r)$ as the set $\{A_1, \ldots, A_k\}$, and $bd(r)$ as the set $\{B_1, \ldots, B_m, \neg C_1, \ldots, \neg C_n\}$ when there is no confusion. In this sense, a rule $r$ of the form (1) can be alternatively written as

$$\{A_1, \ldots, A_k\} \leftarrow \{B_1, \ldots, B_m, \neg C_1, \ldots, \neg C_n\}.$$  \hspace{1cm} (2)

A substitution $\theta$ is a function from variables to terms, which is written in the following form

$$\{x_1/t_1, \ldots, x_n/t_n\}$$ \hspace{1cm} (3)

where $x_i$s (1 \leq i \leq n) are pair-wise distinct variables and $t_i$s (1 \leq i \leq n) are terms (of the language $\mathcal{L}$). The application $e\theta$ of $\theta$ to an expression $e$ is obtained from $e$ by simultaneously replacing all occurrences of each variable $x_i$ in $e$ with the same term $t_i$, and $e\theta$ is called an instance of $e$ [11]. The substitution $\theta$ is ground if $t_i$ contains no variables for every $x_i/t_i$ in $\theta$. If the instance $e\theta$ of $e$ contains no variable then it is a ground instance of $e$. The ground of a logic program $P$, written $ground(P)$, is the set $\bigcup_{r \in P} ground(r)$, where

$$ground(r) = \{r\theta | r\theta \text{ is a ground instance of } r\}.$$ \hspace{1cm} (4)

For simplicity, we assume logic programs are always ground in the following of the paper, unless explicitly stated otherwise.

Let $r_1, r_2$ be two rules. We say that $r_1$ subsumes $r_2$, written $r_1 \preceq r_2$, if there exists a substitution $\theta$ such that $hd(r_1)\theta \subseteq hd(r_2)$ and $bd(r_1)\theta \subseteq bd(r_2)$. In this sense, we say that $r_1$ is more (or equally) general than $r_2$, and $r_2$ is less (or equally) general than $r_1$. By $r_1 \prec r_2$ we mean $r_1$ subsumes $r_2$ but $r_2$ does not subsume $r_1$. For a logic program $P$, we denote $SR(P)$ the logic program obtained from $P$ by removing all rules that are properly subsumed by some other rules in $P$, i.e.,

$$SR(P) = \{r \in P | \exists r' \in P \text{ s.t. } r' \prec r\}.$$ \hspace{1cm} (5)

A (Herbrand) interpretation $I$ is a set of ground atoms. An interpretation $I$ satisfies a ground rule $r$ if $I$ satisfies $bd(r)$ implies $I$ satisfies $hd(r)$. It satisfies a rule $r$ if $I$ satisfies $bd(r)$ and $\exists I. \ x, I \models r$. For example, $\{x_1, x_2\}$ satisfies the rule

$$x_1 \lor x_2 \leftarrow x_1 \land x_2 \land \neg x_1 \land \neg x_2$$

with $\theta = \{x_1/t_1, x_2/t_2\}$.
satisfies every ground rule in \(\text{ground}(r)\). It satisfies a logic program \(P\) if it satisfies every rule in the logic program \(P\). In this case we call \(I\) a model of a (ground) rule (resp., logic program). In the following we use “\(\models\)” to denote the satisfaction relation, and “\(\equiv\)” to denote the classical equivalence relation.

A rule \(r\) is applicable w.r.t. an interpretation \(I\) if \(I \models \text{bd}(r)\). Let \(P\) be a logic program. We denote \(\text{app}(P, I)\) the set of rules in \(P\) that are applicable w.r.t. \(I\).

**Definition 1.** Let \(P\) be a disjunctive logic program. The immediate consequence operator \(T^d_P : 2^\mathcal{A} \rightarrow 2^\mathcal{A}\) is defined as follows, for \(I \subseteq \mathcal{A}\),

\[
T^d_P(I) = \{S | S \text{ is a minimal (under set inclusion) model of } \text{hd}(\text{app}(P, I))\}. \tag{6}
\]

Please note that the operator \(T^d_P\) is a generalization to the operator \(T_P\) for normal logic programs [12], and it is similar to the operator \(T^d_{P'}\) for logic programs with abstract constraint atoms [13].

### 3 Two resolutions

To extend the learning algorithm for normal logic programs in [6] to disjunctive logic programs, we extend its ground resolution for disjunctive rules and present a combined resolution. Recall that a literal \(l\) is either an atom or its classical negation. The complement of \(l\), written \(\overline{l}\), is defined as \(\overline{A} = \neg A\) and \(\neg \overline{A} = A\) where \(A\) is an atom. For a set \(S\) of atoms, we denote \(\neg S = \{\neg A | A \in S\}\).

**Definition 2 (disjunctively ground resolution).** Let \(r\) and \(r'\) be two ground rules. The rule \(r\) is disjunctively ground resolvable w.r.t. \(r'\) on a literal \(l\) whenever

\((a)\) \(l \in \text{bd}(r)\) and \(\overline{l} \in \text{bd}(r')\),  
\((b)\) \(\text{bd}(r') \setminus \{\overline{l}\} \subseteq \text{bd}(r) \setminus \{l\}\), and  
\((c)\) \(\text{hd}(r') \subseteq \text{hd}(r)\).

The disjunctive ground resolvent of \(r\) w.r.t. \(r'\) on \(l\) is the rule \(\text{hd}(r) \leftarrow \text{bd}(r) \setminus \{l\}\). We denote it by \(\text{gr}(r, r')\). In particular, if the above condition \((b)\) is strengthen to

\[
\text{bd}(r') \setminus \{l\} = \text{bd}(r) \setminus \{\overline{l}\} \tag{7}
\]

then we say that \(r\) is disjunctively naive resolvable w.r.t. \(r'\) on \(l\).

The following proposition shows that the disjunctive ground resolution preserves the equivalence of the \(T^d_P\) operator in terms of \(T^d_P(I) = T^d_{P'}(I)\) for every \(I \subseteq \mathcal{A}\) where \(P'\) is obtained from \(P\) by adding some disjunctive ground resolvent.

**Proposition 1.** Let \(P\) be a ground logic program containing two disjunctive rules \(r\) and \(r'\) such that \(r\) is disjunctively ground resolvable w.r.t. \(r'\) on a literal \(l\), and \(Q = P \cup \{\text{gr}(r, r')\}\). Then \(\text{hd}(\text{app}(P, I)) \equiv \text{hd}(\text{app}(Q, I))\) for every \(I \subseteq \mathcal{A}\).
Definition 3 (combined resolution). Let \( r_1, \ldots, r_k \) and \( r \) be the following rules:

\[
\begin{align*}
  r_1 : hd(r_1) & \leftarrow bd^+ \cup \neg(bd^- \cup \{b_1\}), \\
  \vdots \\
  r_k : hd(r_k) & \leftarrow bd^+ \cup \neg(bd^- \cup \{b_k\}), \\
  r : hd(r) & \leftarrow bd^+ \cup \{b_i | 1 \leq i \leq k\} \cup \neg bd^-
\end{align*}
\]

such that

\[ - bd^- \cap \{b_i | 1 \leq i \leq k\} = \emptyset, \text{ and} \]
\[ - hd(r_i) \subseteq hd(r) \text{ for every } i (1 \leq i \leq k). \]

Then the combined resolvent of \( r, r_1, \ldots, r_k \), written \( \text{cr}(r, r_1, \ldots, r_k) \), is the rule

\[
r^* : hd(r) \leftarrow bd^+ \cup \neg(bd^- \cup bd^{'})
\]

In this case we say that the rules \( r, r_1, \ldots, r_k \) are combined resolvable.

The next proposition shows that, similar to the disjunctive ground resolution, the combined resolution preserves the equivalence of the \( T^p_d \) operator as well.

Proposition 2. Let \( P \) be a logic program containing rules \( r, r_1, \ldots, r_k \) such that \( r, r_1, \ldots, r_k \) are combined resolvable, and \( Q = P \cup \{\text{cr}(r, r_1, \ldots, r_k)\} \). It holds that \( hd(app(P, I)) \equiv hd(app(Q, I)) \) for any \( I \subseteq A \).

4 Learning from 1-step Transitions

In the section we present our inductive learning task for disjunctive logic programs and its learning algorithm. Properties of the algorithm are investigated as well.

4.1 The Learning Task

A background theory is a logic program. An example (or observation) is a state transition (or interpretation transition), i.e., a tuple \((I, J)\) with \( I \subseteq A \) and \( J \subseteq A \), which means that the state \( J \) is a candidate successor of the state \( I \) in a Boolean network, or \( J \in T^p_d(I) \) for a disjunctive logic program \( P \). Let \( E \) be a set of examples. We denote \( E^i = \{I | (I, J) \in E\}, E^o = \{J | (I, J) \in S\} \) and \( E(I) = \{J | (I, J) \in E\} \) for \( I \subseteq A \). The set \( E \) is total whenever \( E^i = 2^A \).

Definition 4 (the learning task). An inductive learning task from nondeterministic state transitions is, given a background theory \( B \) and a set \( E \) of examples (state transitions), to find a hypothesis (a logic program) \( H \) such that, for every example \((I, J) \in E, J \in T^p_d_{B \cup H}(I) \) holds.
The above inductive learning task is written as $ILT(B, E)$. Such a hypothesis $H$ to the inductive learning task is called a solution to $ILT(B, E)$. For our learning purpose, the given examples have to be restricted. For instance, let $E = \{\langle 0, 0 \rangle, \langle 0, \{p\} \rangle\}$ and $B = \emptyset$. There will be no logic program $H$ satisfying $\{0, \{p\}\} \subseteq T^d_{B \cup H}(\emptyset)$, since the sets in the collection $T^d_{B \cup H}(\emptyset)$ are incomparable under set inclusion, while $\emptyset$ and $\{p\}$ are comparable under set inclusion.

A set $E$ of state transitions is coherent if $J$ and $J'$ are incomparable under set inclusion for every $\langle I, J \rangle$ and $\langle I, J' \rangle$ in $E$, i.e., $J$ and $J'$ are all minimal under set inclusion. A set $E$ of state transitions is consistent w.r.t. a logic program $P$, if for each $\langle I, J \rangle \in E$, $I \models bd(r)$ implies $J \models hd(r)$ holds for every rule $r$ in $P$.

The following property identifies the sufficient and necessary condition for the existence of a solution to an inductive learning task.

**Proposition 3.** Given an inductive learning task $ILT(B, E)$ where $B$ is a background theory and $E$ is a set of observations, there exists a solution $H$ to $ILT(B, E)$ if and only if $E$ is coherent and $E$ is consistent w.r.t. $B$.

### 4.2 An Inductive Learning Algorithm

In the following we present a bottom-up method to compute a logic program for our inductive learning tasks. This method generates hypothesis by generalization from the most specific rules until all examples are covered.

Firstly, let $q \in \mathcal{A}$ and $I \subseteq \mathcal{A}$. We denote $r^q$ the following rule:

$$ q \leftarrow I \cup \neg \mathcal{T} \tag{9} $$

which is the most specific normal rule such that $q$ belongs to a candidate successor of the state $I$. Now the \textit{LFDT} algorithm is showed in Algorithm 1. Intuitively, this algorithm is to construct the following rules for these examples with the same first state in the state transitions $\langle I, J_1 \rangle, \ldots, \langle I, J_m \rangle$ of $E$:

$$ H \leftarrow I \cup \neg \mathcal{T}, \quad H \text{ is a minimal hitting set of } J_1, \ldots, J_m. \tag{10} $$

The algorithm \textbf{AddRule}, shown in Algorithm 2, adds these rules into the result. It also simplifies the result by removing being subsumed rules through disjunctive ground resolution and combined resolution. Since disjunctive ground resolution is a generalization of ground resolution, this algorithm is also a generalization of the algorithm \textit{LFIT} in [6], which learns normal logic programs from (deterministic) state transitions, i.e., $I_1 \neq I_2$ for any two distinct state transitions $\langle I_1, J_1 \rangle$ and $\langle I_2, J_2 \rangle$ in $E$.

Let $P$ be a logic program, and $E$ be a coherent set of state transitions which is consistent w.r.t. a background theory $B$. The logic program $P$ is complete for $E$ w.r.t. $B$ if $\{J \mid \langle I, J \rangle \in E\} \subseteq T^d_{B \cup P}(I)$ for any $I \in E^q$, it is sound for $E$ if $T^d_{B \cup P}(I) \subseteq \{J \mid \langle I, J \rangle \in E\}$ for any $I \in E^q$. A learning algorithm is complete (resp. sound) for $E$ w.r.t. $B$ if its output is complete (resp. sound) for $E$ w.r.t. $B$. In the following we show the correctness of the \textit{LFDT} algorithm according to its soundness and completeness.

**Theorem 1.** The algorithm \textit{LFDT} is sound and complete (with disjunctive ground resolution, combined resolution, and/or subsumption reduction). Namely, if $E$ is coherent and $B$ is consistent w.r.t. $E$ then the output $P$ by the algorithm \textit{LFDT} is sound and complete for $E$ w.r.t. $B$. 


Algorithm 1: LFDT\((E, B)\)

**Input:** A set \(E\) of state transitions and a background theory \(B\) such that \(E\) is coherent and it is consistent w.r.t. \(B\)

**Output:** A logic program \(P\)

1. \(P \leftarrow B;\)
2. foreach \((I, J) \in E\) do
   3. \(Q \leftarrow \{r_i^{J}|q \in J\};\)
   4. \(E \leftarrow E \setminus \{(I, J)\};\)
   5. foreach \((I', J') \in E\) with \(I' = I\) do
      6. \(Q \leftarrow Q \cup \{hd(r) \cup \{p\} \leftarrow bd(r)\};\)
   7. end
   8. foreach \(r \in Q\) do
      9. AddRule\((r, P)\);
   10. end
11. \(P \leftarrow P \setminus B;\)
12. return \(P;\)

Algorithm 2: AddRule\((r, P)\)

**Input:** A rule \(r\) and a logic program \(P\)

1. if \(\exists r' \in P\) s.t. \(r' \prec r\) then return;
2. foreach \(r' \in P\) do
   3. if \(r \prec r'\) then \(P \leftarrow P \setminus \{r\};\)
   4. \(P \leftarrow P \cup \{r\};\)
   5. while \(r, r_1, \ldots, r_k \in P\) are combined resolvable do AddRule\((cr(r, r_1, \ldots, r_k))\);
6. foreach \(r' \in P\) do
   7. if \(r\) is disjunctively ground resolvable w.r.t. \(r'\) then
      8. AddRule\((gr(r, r'), P)\);
   9. else if \(r'\) is disjunctively ground resolvable w.r.t. \(r\) then
      10. AddRule\((gr(r', r), P)\);
   11. end
12. end
13. return \(P;\)

5 Concluding Remarks and Future Work

In this paper we proposed a new framework LFDT for learning disjunctive programs from interpretation transitions. It is a nontrivial and substantial extension to the LF1T framework. One remaining challenge work is to apply the learning approach to practical domains, such as bio-informatics for which LF1T is successfully applied.

Acknowledgement. We thank reviewers for their helpful comments. This work is partially supported by NSFC under grant 63170161, Stadholder Fund of Guizhou Province under grant (2012)62, Outstanding Young Talent Training Fund of Guizhou Province under grant (2015)01 and Science and Technology Fund of Guizhou Province under grant [2014]7640, Scientific and Technological Research Program of Chongqing Municipal Education Commission under Grant No. KJ1601129.
References