Learning Deduction Rules by Induction

Chiaki Sakama[1], Tony Ribeiro[2] and Katsumi Inoue[3]

[1] Wakayama University
sakama@sys.wakayama-u.ac.jp
[2] The Graduate University for Advanced Studies (Sokendai)
tony_ribeiro@nii.ac.jp
[3] National Institute of Informatics
inoue@nii.ac.jp

Abstract. This paper provides an attempt for learning deductive inference rules by induction. Given a set S of propositional formulas and their logical consequences T, we consider the problem of learning deductive inference rules that produce T from S. To this end, we use an induction framework **LF1T** which learns logic programs from interpretation transitions. Experimental results show that **LF1T** successfully produces deductive inference rules from input transitions.

1 Introduction

Formal systems based on logic reason about knowledge and data using an axiomatic system that is specified and built-in by human engineers. In [3], Sakama and Inoue pose a question whether it is possible to develop artificial (general) intelligence that automatically produces a logic underlying any given data set. They introduce a conceptual framework for learning logics in which a machine can acquire logical inference rules inductively. Induction or ILP has been used as an inference mechanism of machine learning, while little study has been devoted to the challenging topic of learning logics.

In this paper, we implement learning deductive inference rules using the **LF1T** algorithm [1, 2] and examine whether **LF1T** successfully produces deductive inference rules. The rest of this paper is organized as follows. Section 2 reviews a conceptual framework for learning logics. Section 3 realizes learning logics using **LF1T** and provides experimental results.

2 Learning Logics

We review a conceptual framework for learning logics that is introduced in [3]. There are an agent $\mathcal{A}$ and a machine $\mathcal{M}$. The agent $\mathcal{A}$, which could be a human or a computer, is capable of deductive reasoning: it has a set $\mathcal{L}$ of axioms and inference rules in classical logic. Given a (finite) set S of formulas as an input, the agent $\mathcal{A}$ produces a (finite) set of formulas T such that $T \subseteq Th(S)$ where $Th(S)$ is the set of logical consequences of S. On the other hand, the machine $\mathcal{M}$ has no axiomatic system for deduction, while it is equipped with a machine learning algorithm $\mathcal{C}$. Given input-output pairs $(S_1, T_1), \ldots, (S_i, T_i), \ldots$ (where $T_i \subseteq Th(S_i)$) of $\mathcal{A}$ as an input to $\mathcal{M}$,

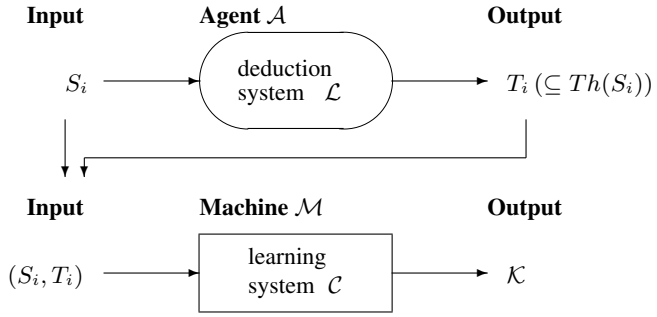

Fig. 1. Learning Logics [3]

the problem is whether one can develop an algorithm $\mathcal{C}$ which successfully produces an axiomatic system $\mathcal{K}$ for deduction. An algorithm $\mathcal{C}$ is *sound* wrt $\mathcal{L}$ if it produces an axiomatic system $\mathcal{K}$ such that $\mathcal{K} \subseteq \mathcal{L}$. An algorithm $\mathcal{C}$ is *complete* wrt $\mathcal{L}$ if it produces an axiomatic system $\mathcal{K}$ such that $\mathcal{L} \subseteq \mathcal{K}$. Designing a sound and complete algorithm $\mathcal{C}$ is called a problem of *learning logics* (Figure 1). In this framework, an agent $\mathcal{A}$ plays the role of a teacher who provides training examples representing premises along with entailed consequences. The output $\mathcal{K}$ is refined by incrementally providing examples. We consider a deduction system $\mathcal{L}$ while it could be a system of arbitrary logic, e.g. nonmonotonic logic, modal logic, fuzzy logic, as far as it has a formal system of inference. Alternatively, we can consider a framework in which a teacher agent $\mathcal{A}$ is absent. In this case, given input-output pairs (S_i, T_i) as data, the problem is whether a machine $\mathcal{M}$ can find an unknown logic (or axiomatic system) that produces a consequence T_i from a premise S_i.

The abstract framework has challenging issues of AI including the questions:

1. *Can we develop a sound and complete algorithm $\mathcal{C}$ for learning a classical or non-classical logic $\mathcal{L}$?*
2. *Is there any difference between learning axioms and learning inference rules?*
3. *Does a machine $\mathcal{M}$ discover a new axiomatic system $\mathcal{K}$ such that $\mathcal{K} \vdash F$ iff $\mathcal{L} \vdash F$ for any formula F?*

The first question concerns the possibility of designing machine learning algorithms that can learn existing logics (or axiomatic systems) from given formulas. The second question concerns differences between learning Gentzen-style logics and Hilbert-style logics. The third question is more ambitious: it asks the possibility of AI's discovering new logics that are unknown to human mathematicians.

In [3], Sakama and Inoue provide simple case studies concerning the first question. They represent a formal system $\mathcal{L}$ of propositional logic using *metalogic programming* and show that deductive inference rules can be reproduced as meta-rules of logic programs. In the next section, we implement the framework using the **LF1T** induction algorithm and show experimental results.

3 Learning Deductive Rules by LF1T

Learning from 1-step transitions (**LF1T**) [1] is a framework for learning normal logic programs from transitions of interpretations. Here we apply **LF1T** for learning *definite logic programs*, a subclass of normal logic programs that do not contain negation as failure. Let $\mathcal{B}$ be the set of all ground atoms (Herbrand base) and P a *definite logic program* (or simply, a *program*) that consists of *rules* of the form:

$$a \leftarrow b_1, \ldots, b_n \qquad (n \geq 0) \tag{1}$$

where $a, b_1, \ldots, b_n$ are ground atoms from $\mathcal{B}$. For each rule r of the form (1), the atom a is the *head* (written $h(r)$) and the conjunction $b_1, \ldots, b_n$ is the *body* of the rule (written $b(r)$). The body is identified with the set of atoms $\{b_1, \ldots, b_n\}$. A rule with the empty body is a *fact*. **LF1T** produces a program from a pair of interpretations as follows:

Input: $E \subseteq 2^{\mathcal{B}} \times 2^{\mathcal{B}}$
Output: a program P such that $J = T_P(I)$ holds for any $(I, J) \in E$

where $T_P(I) = \{\, h(r) \mid r \in P \text{ and } b(r) \subseteq I \,\}$ [4]. A rule r is *consistent* with (I, J) if $b(r) \subseteq I$ implies $h(r) \in J$, otherwise, r is *inconsistent* with (I, J). A program P is *consistent* with (I, J) if every rule in P is consistent with (I, J). In **LF1T**, a positive example is input as a one-step state transition from I to J, which is given as a pair of Herbrand interpretations. **LF1T** outputs a single program which realizes all state transitions given in the input.

To build a program, we use a *top-down* version of **LF1T** [2], which generates hypotheses by specialization from the most general rules until a program is consistent with all input state transitions. More precisely, **LF1T** starts with the initial program $P = \{\, a \leftarrow \mid a \in \mathcal{B}\}$. Then **LF1T** iteratively analyzes each transition (I, J). For each atom a that does *not* appear in J, **LF1T** produces an *anti-rule*:

$$a \leftarrow \bigwedge_{b_i \in I} b_i \,.$$

Any rule of P that subsumes such an anti-rule is not consistent with the transition and must be revised. To this end, a rule is minimally specialized by introducing an atom $c_j \in \mathcal{B} \setminus I$ to the body of the rule to make P consistent with the new transition by avoiding the subsumption of all anti-rules produced by (I, J). After such minimal specialization, P becomes consistent with the new transition while remaining consistent with all previously analyzed transitions.

We use **LF1T** as a learning system $\mathcal{C}$ in Figure 1. We assume a (propositional logic) deduction system $\mathcal{L}$ represented by a *metalogic program* P that provides transitions (I, J) satisfying $J = T_P(I)$. Given (I, J) as input, our goal is to examine whether **LF1T** can reproduce correct inference rules of deduction represented by meta-rules in P. To represent formulas by interpretations, we use a meta-predicate $hold$ and consider a set of atoms of the form

$$hold(F)$$

where F is a formula in propositional logic. Using the meta-expression, for example, two sets of formulas $S = \{\, p,\, p \rightarrow q \,\}$ and $T = \{\, p,\, p \rightarrow q,\, q \,\}$ where $T \subseteq Th(S)$

are respectively represented as the sets of atoms $I = \{\, hold(p),\ hold(p \rightarrow q)\}$ and $J = \{\, hold(p),\ hold(q),\ hold(p \rightarrow q)\,\}$. We next provide the results of experiments.

Let $\mathcal{B} = \{\, hold(p),\ hold(q),\ hold(r),\ hold(p \rightarrow r)\,\}$. We address the process of constructing a rule with the atom $hold(r)$ in the head.

Step 0: **LF1T** starts with the most general rule:

$$hold(r) \leftarrow . \tag{2}$$

Step 1: Suppose that the transition $(\emptyset, \emptyset)$ is given. The rule (2) is inconsistent with this transition, so that (2) is (minimally) specialized into four rules by introducing an atom from $\mathcal{B}$:

$$hold(r) \leftarrow hold(p), \tag{3}$$
$$hold(r) \leftarrow hold(q), \tag{4}$$
$$hold(r) \leftarrow hold(r), \tag{5}$$
$$hold(r) \leftarrow hold(p \rightarrow r). \tag{6}$$

Those rules are consistent with the transition $(\emptyset, \emptyset)$.

Step 2: Suppose that the transition $(\{hold(p)\}, \{hold(p)\})$ is given. The rule (3) is inconsistent with this transition, so that (3) is specialized into three rules:

$$hold(r) \leftarrow hold(p), hold(q),$$
$$hold(r) \leftarrow hold(p), hold(r),$$
$$hold(r) \leftarrow hold(p), hold(p \rightarrow r).$$

These 3 rules are respectively subsumed by the rules (4), (5) and (6) in Step 1, hence removed. As a result, the rules (4), (5) and (6) remain.

Step 3: Suppose that the transition $(\{hold(q)\}, \{hold(q)\})$ is given. The rule (4) is inconsistent with this transition, and is removed after specialization. As a result of subsumption, three rules (5), (6) and the newly constructed rule (7) remain.

$$hold(r) \leftarrow hold(p), hold(q). \tag{7}$$

Step 4: Suppose that the transition $(\{hold(p \rightarrow r)\}, \{hold(p \rightarrow r)\})$ is given. The rule (6) is inconsistent with this transition, and is removed after specialization. As a result of subsumption, two rules are newly constructed:

$$hold(r) \leftarrow hold(p \rightarrow r), hold(p), \tag{8}$$
$$hold(r) \leftarrow hold(p \rightarrow r), hold(q), \tag{9}$$

Now four rules (5), (7), (8) and (9) remain.

Step 5: Suppose that the transition $(\{hold(p), hold(q)\}, \{hold(p), hold(q)\})$ is given. The rule (7) is inconsistent with this transition, and is removed after specialization. As a result of subsumption, three rules (5), (8) and (9) remain.

Step 6: Suppose that the transition $(\{hold(p \to r), hold(q)\}, \{hold(p \to r), hold(q)\})$ is given. The rule (9) is inconsistent with this transition, and is removed after specialization. As a result of subsumption, two rules (5) and (8) remain.

Step 7: Suppose that the transition $(\{hold(p \to r), hold(p)\}, \{hold(p \to r), hold(p), hold(r)\})$ is given. Two rules (5) and (8) are consistent with this transition and remain as they are.

The remaining two rules (5) and (8) are consistent with any other transitions (I, J) such that J represents logical consequences of I under a metalogic program P. Then **LF1T** produces those rules as output. The rule (5) is tautology and (8) represents **Modus Ponens**. The input-output of **LF1T** is summarized in Table 1.[4]

Table 1. LF1T Input-Output

input	output
$(\emptyset, \emptyset)$	$hold(r) \leftarrow hold(p).$ $hold(r) \leftarrow hold(q).$ $hold(r) \leftarrow hold(r).$ $hold(r) \leftarrow hold(p \to r).$
$(\{hold(p)\}, \{hold(p)\})$	~~$hold(r) \leftarrow hold(p).$~~ $hold(r) \leftarrow hold(q).$ $hold(r) \leftarrow hold(r).$ $hold(r) \leftarrow hold(p \to r).$
$(\{hold(q)\}, \{hold(q)\})$	~~$hold(r) \leftarrow hold(q).$~~ $hold(r) \leftarrow hold(r).$ $hold(r) \leftarrow hold(p \to r).$ $hold(r) \leftarrow hold(p), hold(q).$
$(\{hold(p \to r)\}, \{hold(p \to r)\})$	$hold(r) \leftarrow hold(r).$ ~~$hold(r) \leftarrow hold(p \to r).$~~ $hold(r) \leftarrow hold(p), hold(q).$ $hold(r) \leftarrow hold(p \to r), hold(p).$ $hold(r) \leftarrow hold(p \to r), hold(q).$
$(\{hold(p), hold(q)\}, \{hold(p), hold(q)\})$	$hold(r) \leftarrow hold(r).$ ~~$hold(r) \leftarrow hold(p), hold(q).$~~ $hold(r) \leftarrow hold(p \to r), hold(p).$ $hold(r) \leftarrow hold(p \to r), hold(q).$
$(\{hold(p \to r), hold(q)\}, \{hold(p \to r), hold(q)\})$	$hold(r) \leftarrow hold(r).$ $hold(r) \leftarrow hold(p \to r), hold(p).$ ~~$hold(r) \leftarrow hold(p \to r), hold(q).$~~
$(\{hold(p \to r), hold(p)\}, \{hold(p \to r), hold(p), hold(r)\})$	$hold(r) \leftarrow hold(r).$ $hold(r) \leftarrow hold(p \to r), hold(p).$

We address other results of experiments.

[4] The experimental archive is found at http://www.wakayama-u.ac.jp/~sakama/ILP2015-short/

– Let $\mathcal{B} = \{\, hold(p),\ hold(\neg p),\ hold(q),\ hold(\neg q),\ hold(p \rightarrow\ q),\ hold(q \rightarrow r),\ hold(p \rightarrow r)\,\}$. Then **LF1T** produces

$$hold(\neg p) \leftarrow hold(p \rightarrow q), hold(\neg q) \quad (\textbf{Modus Tollens})$$
$$hold(p \rightarrow r) \leftarrow hold(p \rightarrow q), hold(q \rightarrow r) \quad (\textbf{Hypothetical Syllogism})$$

– Let $\mathcal{B} = \{\, hold(p),\ hold(\neg p),\ hold(q),\ hold(\neg q),\ hold(p \vee q),\ hold(\neg p \vee \neg q),\ hold(r \vee s),\ hold(\neg r \vee \neg s),\ hold(p \rightarrow\ r), hold(q \rightarrow s)\,\}$. Then **LF1T** produces

$$hold(p) \leftarrow hold(p \vee q), hold(\neg q) \quad (\textbf{Disjunctive Syllogism})$$
$$hold(r \vee s) \leftarrow hold(p \vee q), hold(p \rightarrow r), hold(q \rightarrow s) \quad (\textbf{Constructive Dilemma})$$
$$hold(\neg p \vee \neg q) \leftarrow hold(\neg r \vee \neg s), hold(p \rightarrow r), hold(q \rightarrow s) \quad (\textbf{Destructive Dilemma})$$

– Let $\mathcal{B} = \{\, hold(p),\ hold(q),\ hold(r),\ hold(p \wedge q),\ hold(q \wedge r),\ hold(p \vee q),\ hold(q \vee r)\,\}$. Then **LF1T** produces

$$hold(p) \leftarrow hold(p \wedge q) \quad (\textbf{Conjunction Elimination})$$
$$hold(p \wedge q) \leftarrow hold(p), hold(q) \quad (\textbf{Conjunction Introduction})$$
$$hold(p \vee q) \leftarrow hold(p) \quad (\textbf{Disjunction Introduction})$$

Thus, **LF1T** produces inference rules of natural deduction. Moreover, if a transition $(I, J) = (\{hold(q), hold(p \rightarrow\ q)\}, \{hold(p)\})$ is provided, **LF1T** produces the **Fallacy of Affirming the Consequent**:

$$hold(p) \leftarrow hold(q) \wedge hold(p \rightarrow q)$$

that is used for *abductive inference*. In this way, **LF1T** can produce deductive or non-deductive inferences rules from transitions that specify premises and their consequences. Generally speaking, providing all possible transitions, **LF1T** will output production rules that are minimal with respect to subsumption. A limitation is that the number of possible transactions increases exponentially in proportion to the size of the Herbrand base. Further optimization is needed for discovering logics from huge data.

References

1. K. Inoue, T. Ribeiro and C. Sakama. Learning from interpretation transition. *Machine Learning*, 94(1):51–79 (2014).
2. T. Ribeiro and K. Inoue. Learning prime implicant conditions from interpretation transition. In: *Proc. 24th International Conference on Inductive Logic Programming, Lecture Notes in Artificial Intelligence*, 9046, Springer (2015).
3. C. Sakama and K. Inoue. Can machines learn logics? In: *Proc. 8th International Conference on Artificial General Intelligence, Lecture Notes in Artificial Intelligence*, 9205, pp. 341–351, Springer (2015).
4. M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *J. ACM* 23(4):733–742 (1976).