社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Online Large-margin Weight Learning for First-order Logic-based Abduction

Naoya INOUE†, Kazeto YAMAMOTO†, Yotaro WATANABE†, Naoaki OKAZAKI†, and Kentaro INUI†

† Tohoku University
Aramaki Aza Aoba 6–3–09, Aoba-ku, Sendai-shi, Miyagi 980–8579 Japan

Abstract Abduction is inference to the best explanation. Abduction has long been studied in a wide range of contexts and is used for modeling artificial intelligence systems, such as diagnostic systems and plan recognition systems. However, less attention has been paid to how to automatically learn *score functions*, which rank explanations in the order of their plausibility. In this paper, we propose a supervised learning approach for first-order logic-based abduction. The contribution of this paper is the following: (i) we show how to formulate the machine learning problem of abduction with the framework of online large-margin training, which has been shown to have both predictive performance and scalability to larger problems; (ii) we extend the state-of-the-art abductive reasoning system [15] to model the score function with a weighted linear model, which is the groundwork for the online large-margin training; (iii) we support partially-specified gold-standard explanations as training examples, where the weights are learned to rank any explanation that includes the gold-standard explanation as the best explanation; (iv) the all-in-one software package for inference and learning is made publicly available.

Key words Abduction, Logic-based reasoning, Online learning, Large-margin training, Structured learning, Latent variables, Passive Aggressive algorithm

1. Introduction

Abduction is inference to the best explanation. Abduction has long been studied in a wide range of contexts. For example, abduction has been viewed as a promising framework for describing the mechanism of human perception [5], [12], [23], [32] etc. The key idea is that the declarative nature of abduction enables us to infer the most plausible, implicitly stated information combining several types of inference, and pieces of explicitly observed information. For instance, Hobbs et al. [12] showed the process of natural language interpretation can reasonably be described as abductive inference; finding the lowest-cost abductive proof provides the solutions to a broad range of natural language pragmatics problems, such as word sense disambiguation, anaphora, and metonymy resolution.

While the lack of world knowledge resources hampered applying abduction to real-life problems in the 1980s and 1990s, a number of techniques that acquire world knowledge resources have been developed in the last decade [1], [4], [13], [30], [31] etc. In addition, the development of an efficient inference technique of abduction warrant the application of abduction with large knowledge bases to real-life problems [15]. Consequently, several researchers have started applying abduction to real-life problems, and exploiting large knowledge bases. For instance, Ovchinnikova et al. [22] propose an abduction-based natural language processing framework using forty thousands of axioms extracted from the popular ontological resources, WordNet [1] and FrameNet [30]. They evaluate their approach on the real-life natural language processing task of Recognizing Textual Entailment (RTE) [9].

However, less attention has been paid to how to automatically learn a function, which rank candidate explanations in order of their plausibility (henceforth, we call it the *score function*). To apply abductive inference to a wide range of tasks, this non-trivial issue needs to be addressed because the criterion of plausibility is highly task-dependent. A notable exception is a series of studies in the context of Statistical Relational Learning [2], [17], [26], [33], where they emulate abduction in the probabilistic deductive inference framework, Markov Logic Networks (MLNs) [27], or Bayesian Logic Programs [16]. These approaches can exploit several choices of machine learning methods originally developed for probabilistic models [14], [18]. However, emulating abduction in these approaches has severe overhead. For example, the em-

ulation in MLNs requires special procedure to convert abduction problems into deduction problems because MLNs are deductive inference framework in nature. This conversion process generates a large number of axioms, and hence hampers the application of MLN-based approaches to larger problems (see Sec. 5. for more detail). Since inference is a subroutine of learning procedure, learning is also intractable on large dataset, as reported in [33].

In this paper, we propose a supervised learning approach for first-order logic-based abduction, extending the tractable first-order abductive inference engine [15]. In order to apply abduction to a wide range of tasks, we support two kinds of gold-standard explanations as training examples: *exactly-specified*, or *partially-specified*. Given *exactly-specified* gold-standard explanations, our framework trains the score function so that it ranks the given explanation itself as the best explanation. Given *partially-specified* gold-standard explanations, on the other hand, the framework trains the score function so that it ranks *any* explanation that *includes* the gold-standard explanation as the best explanation. It is useful to support *partially-specified* gold-standard explanations, because one might want to use abduction for a specific task, where the *subset* of the best explanation is used as the output label of the task. In the case of plan recognition, for example, one might want a system to output *any* explanation that *includes* the correct plan literals, and does not care about any other types of literals in the explanation.

We formulate these learning problems as discriminative structured learning with latent variables. More specifically, we model the score function as a weighted linear feature function, and then apply Passive Aggressive algorithm [7], an online large-margin training algorithm, to tune the weights. The contribution of this paper is as follows:

(i) we show how to formulate the machine learning problem of abduction with the framework of online large-margin training, which has been shown to have both predictive performance and scalability to larger problems;

(ii) we extend the state-of-the-art abductive reasoning system [15] to model the score function with a weighted linear model, which makes it possible for the formulation of learning problem to work;

(iii) we support *partially-specified explanations*, meaning that the framework learns a weight vector that outputs any explanation that *includes* the given explanation;

(iv) the all-in-one software package for inference and learning is made publicly available.

This paper is organized as follows. We first give a brief review of abduction and the abductive reasoning system [15] (Sec. 2). We then generalize the score function with a weighted linear model, and formally define the learning problem (Sec. 3. 1). We show how to efficiently perform abduction with the weighted linear model (Sec. 3. 2), and then present our learning framework, starting with the simple case where exactly-specified gold-standard explanations are given. (Sec. 3. 3). We then extend the learning framework to learn weights from partially-specified gold-standard explanations (Sec. 3. 4). Finally, we demonstrate that our learning algorithm successfully improve predictive performance in two applications (Sec. 4.).

2. Background

2. 1 Abduction

Abduction is inference to the best explanation. We use function-free first-order logic for the representation language of abduction[⋆1]. Formally, first-order logical abduction is defined as follows:

- **Given:** Background knowledge B, and observations O, where B is a set of first-order logical formulae, and O is a set of literals or substitutions.
- **Find:** An *explanation* (or *hypothesis*) H such that $H \cup B \models O$, $H \cup B \not\models \bot$,[⋆2] where H is a set of literals or substitutions. Each element in H is called an *elemental explanation*.

We define *substitution* to be the form $x = y$ (*positive* substitution) or $x \neq y$ (*negative* substitution), where x and y are either variables or constants. The semantics is that entities indicated by variables or constants are (not) the same (i.e. $\{p(x), p(y), x = y\}$ is identical to $\{p(x)\}$). We say that p is *hypothesized* if $H \cup B \models p$, and that p is *explained* if there exists a set Q of literals such that $Q \cup B \rightarrow p$ and $H \cup B \models Q$. In this paper, we assume that all variables occurring in a logical form of background knowledge are *universally* quantified with the widest possible scope, unless it is explicitly stated as existentially quantified. On the one hand, we assume that variables occuring in an explanation and observation are *existentially* quantified implicitly.

Typically, several explanations H explaining O exist. We call each of them a *candidate explanation*, and represent a set of candidate explanations of O given B as $\mathcal{H}_{O,B}$. The goal of abduction is to find the best explanation among candidate explanations by a specific evaluation measure. In this paper, we formulate abduction as the task of finding the maximum-score explanation $\hat{H}$ among $\mathcal{H}_{O,B}$. Formally, we find $\hat{H} = \arg\max_{H \in \mathcal{H}_{O,B}} score(H)$, where $score$ is a function $\mathcal{H}_{O,B} \rightarrow \mathbb{R}$, which is called the *score function*. In the literature, several kinds of score functions have been proposed, including cost-based and probability-based [5], [12], [24], [26], [33] etc. As shown in Sec. 3., we generalize the score function with a weighted linear model in this paper.

Finally, let us describe the task of abduction with a toy example. Given $B = \{p(x, y) \wedge q(x) \rightarrow r(x), s(x) \rightarrow r(x)\}, O = \{r(z)\}$, we have four candidate explanations: $H_1 = \{r(z)\}$, $H_2 = \{p(z, w), q(z)\}$, $H_3 = \{s(z)\}$, and $H_4 = \{p(z, w), q(z), s(z)\}$. The task of abduction is to select the best explanation among them in terms of score. Suppose $score(H_1) = 5.5$, $score(H_2) = 12.25$, $score(H_3) = 10.8$, and

⋆1: We consider the case of finite domains.

⋆2: Throughout the paper, $\models$, $\bot$ means logical entailment and logical contradiction respectively.

$score(H_4) = 7.13$. The correct prediction is then H_2.

2.2 Integer Linear Programming Formulation of First-order Logic-based Abduction

The problem of best explanation finding is computationally expensive; the number of candidate explanations grows exponentially to the size of observation and knowledge base. To perform efficient inference, we adopt the Integer Linear Programming (ILP)-based formulation of first-order logic-based abduction [15]. This approach is efficient because (i) the abductive reasoning problem on first-order logic is directly performed on first-order level, similarly to the resolution principle in theorem proving [29], and (ii) the state-of-the-art ILP optimization technique is exploited. This section gives a brief description of the solution as a preparation of Sec. 3.2; see [15] for more details.

Given B and O, the framework first enumerates a set P of *potential elemental explanations* (atomic assumptions), which are possible constituents of candidate explanations (i.e. literals or substitutions). It initializes P with O, and iteratively adds new literals through backward-inference on P with axioms in B. For example, the set of potential elemental explanations of the toy problem in Sec. 2.1 is represented by $\{r(z), p(z,w), q(z), s(z)\}$. Then, the framework generates ILP variables and constraints based on this set to represent all possible *candidate explanations* through value assignments to the ILP variables. The two main ILP variables are $h_p \in \{0,1\}$, and $s_{x=y} \in \{0,1\}$, where p is a potential elemental explanation and x, y are variables or constants used in P. h_p is used to represent whether p is hypothesized ($h_p = 1$) or not ($h_p = 0$). $s_{x=y}$ is used to represent whether $x = y$ ($s_{x=y} = 1$) or not ($s_{x=y} = 0$).

The score function of this framework is the sum of scores of each elemental explanation p, each of which has two terms: (i) *cost* of p being hypothesized, and (ii) *reward* of p being explained by other elemental explanations. The framework thus introduces new ILP variables $r_p \in \{0,1\}$ to represent whether p is rewarded (i.e., is explained, $r_p = 1$) or not ($r_p = 0$). The ILP objective function is as follows:

$$\text{max. } score(H) = \sum_{p \in P} [h_p \cdot -cost(p) + r_p \cdot cost(p)], \quad (1)$$

where $cost(p)$ is a cost of hypothesizing p. It assumes that $cost(p)$ is given by a user, and does not address how to automatically learn the value of $cost(p)$. The framework finds the value assignment that optimizes this function, with several constraints to ensure the value assignments to be candidate explanations (e.g. O must be explained).

3. Online Large-margin Weight Learning for First-order Logic-based Abduction

In this section, we propose a machine learning framework for first-order logic-based abduction. We first formalize the abductive reasoning problem as a structured prediction with a weighted linear model, and then define the weight learning problem (Sec. 3.1). We show how to use the weighted linear feature function in the ILP-based formulation (Sec. 3.2), and then show how to learn the weights by instantiating Passive Aggressive algorithm [7]. We start with the simple case where exactly-specified gold-standard explanations are given (Sec. 3.3), and then describe a learning framework for partially-specified gold-standard explanations (Sec. 3.4)

3.1 Problem Formulation

We first generalize the score function with a weighted linear model. Let $\mathbf{\Phi}(H) = \{\phi_1(H), \phi_2(H), ..., \phi_n(H)\}$ be a n-dimensional feature vector of an explanation H, and $\mathbf{w} = \{w_1, w_2, ..., w_n\}$ be a n-dimensional weight vector. We then define the score function as follows:

$$score(H; \mathbf{w}) = \mathbf{w} \cdot \mathbf{\Phi}(H) = \sum_{i=1}^{n} w_i \cdot \phi_i(H) \quad (2)$$

We refer to $\mathbf{w}$ as the *parameter* of score function. We assume each element $\phi_i(H)$ to be the following:

$$\phi_i(H) = \begin{cases} V_i & \text{if } H \cup B \models C_i; \\ 0 & \text{otherwise,} \end{cases} \quad (3)$$

where V_i is a real-valued constant, and C_i is a first-order logical formula where each element is a literal or substitution included in H. We call V_i the *feature value*, $\phi_i(H)$ the *feature function*, and C_i the *feature condition*. The feature vector is designed by a user. For example, one might create a feature function ϕ_i such that $(V_i, C_i) = (1, x = y \wedge cat(x) \wedge dog(y))$. The task of abductive reasoning is then formalized as follows:

$$H = \arg\max_{H \in \mathcal{H}_{O,B}} score(H; \mathbf{w}) = \arg\max_{H \in \mathcal{H}_{O,B}} \mathbf{w} \cdot \mathbf{\Phi}(H) \quad (4)$$

Notice that this formulation is equivalent to a structured prediction problem (or multi-class classification problem), where the input is O, B, and the set of possible output structures (or classes) is $\mathcal{H}_{O,B}$. We find the best H in the modified ILP-based framework, which is described in the next section.

Let us formalize the supervised learning problem of first-order logic-based abduction. Let $\mathbb{D} = \{(O_i, H_i)\}_{i=1}^n$ be a set of training examples, where O_i is an observation (i.e. input) and H_i is either exactly-specified, or partially-specified gold-standard explanation for O_i. Based on the definition in Sec. 2.1, we assume that O_i and H_i are given by a set of literals or substitutions. The goal of supervised learning is to learn $score(H; \mathbf{w})$, which has minimal prediction errors on $\mathbb{D}$. To achieve this goal, we estimate a weight vector $\mathbf{w}$ that minimizes the value $\sum_{i=1}^n \Delta(\hat{H}_i, H_i)$, where $\hat{H}_i$ is the best explanation for O_i inferred by the system, and $\Delta(\hat{H}_i, H_i)$ is a non-negative function that measures the difference between $\hat{H}_i$ and H_i. Henceforth, we call $\Delta(\hat{H}_i, H_i)$ the *loss function*. Because the definition of loss is task-dependent, the loss function is designed by the user. The simple example of loss function for exactly-specified gold-standard explanations is the following (a.k.a 0-1 loss function):

$$\Delta(\hat{H}_i, H_i) = \begin{cases} 1 & \text{if } \hat{H}_i \neq H_i; \\ 0 & \text{otherwise (i.e. } \hat{H}_i = H_i) \end{cases} \quad (5)$$

In this paper, we assume that there is enough knowledge to infer the gold-standard explanation for each problem (*the knowledge completeness assumption*). If this assumption were not satisfied, which means that the gold-standard explanation is not included in the candidate explanations, then we could not infer the gold-standard explanation even if we change the weight vector.

irrespectively of parameters.

3.2 ILP-based Abduction with Weighted Linear Model

In order to exploit the weighted linear feature function as the score function, we replace the ILP-based objective function (1) with equation (6). We introduce new ILP variables $f_i \in \{0,1\}$ such that $f_i = 1$ if and only if the feature condition C_i is entailed by $H \cup B$; $f_i = 0$ otherwise. The extended ILP objective function is as follows:

$$\text{max. } score(H;\mathbf{w}) = \sum_{i=1}^{n} w_i \cdot (V_i \cdot f_i) \tag{6}$$

Following the definition of f_i above, we associate the feature condition C_i with the assignment of f_i by introducing new ILP constraint so that $f_i = 1 \Leftrightarrow H \cup B \models C_i$. In general, however, this association cannot be represented as a single ILP constraint. Therefore, we first decompose C_i into a *Conjunctive Normal Form* $\text{CNF}(C_i)$, a set of disjunctive clause, and then introduce ILP constraints for each disjunctive clause.

Let D_i^j be the j-th disjunctive clause in $\text{CNF}(C_i)$. For all $j \in \{1,2,...,|\text{CNF}(C_i)|\}$, we first introduce new ILP variable $f_i^j \in \{0,1\}$ such that $f_i^j = 1 \Leftrightarrow H \cup B \models D_i^j$. To allow to set $f_i^j = 1$ iff $H \cup B \models D_i^j$, we impose the following ILP constraint:

$$0 \leqq |l(D_i^j)| f_i^j - [\sum_{L \in l(D_i^j)} I(L)] \leqq |l(D_i^j)| - 1, \tag{7}$$

where $l(D_i^j)$ is a set of literals or substitutions in D_i^j, and $I(L)$ is a function that returns h_L if L is a literal; s_L if L is a positive substitution; $1 - s_L$ if L is a negative substitution. On the most-right of the term, we add -1 because at least one $L \in l(D_i^j)$ must be hypothesized (remember that D_i^j is a disjunctive clause) when $f_i^j = 1$.

Finally, to ensure that $f_i = 1$ iff $f_i^j = 1$ for all $j \in \{1,2,...,|\text{CNF}(C_i)|\}$, we introduce the following ILP constraint:

$$-|\text{CNF}(C_i)| + 1 \leqq |\text{CNF}(C_i)| f_i - \sum_{j=1}^{|\text{CNF}(C_i)|} f_i^j \leqq 0 \tag{8}$$

Note that we are able to use a constant instead of f_i in equation (6) when the value of feature is decidable from observations (i.e. $O \models C_i$). In this case, the constraints (7), (8) need not be introduced.

Let us describe the ILP constraints (7), (8) with an example. Suppose that we have the feature condition $C_k = \neg p(x) \wedge (p(y) \vee q(y) \vee x \neq y)$ for k-th feature. The CNF of this formula is $\{\neg p(x), p(y) \vee q(y) \vee x \neq y\}$. We thus introduce two ILP variables for each clause: $f_k^1, f_k^2 \in \{0,1\}$, and then introduce the ILP constraints $f_k^1 - h_{\neg p(x)} = 0$ (i.e. $f_k^1 = 1 \Leftrightarrow H \cup B \models \neg p(x)$), and $0 \leqq 3f_k^2 - [h_{p(y)} + h_{q(y)} + (1 - s_{x,y})] \leqq 2$ (i.e. $f_k^2 = 1 \Leftrightarrow H \cup B \models [p(y) \vee q(y) \vee x \neq y]$). Finally, we introduce $-1 \leqq 2f_k - (f_k^1 + f_k^2) \leqq 0$ to ensure that $f_k = 1 \Leftrightarrow f_k^1 = 1 \wedge f_k^2 = 1$.

Algorithm 1 learnExact(training examples $\mathbb{D}$, background knowledge B, int N, double C)

```
1: w ← 0
2: for n = 1 to N do
3:   for all (O_i, H_i) ∈ D do
4:     Ĥ ← argmax_{H ∈ H_{O_i,B}} score(H; w)
5:     if Ĥ ≠ H_i then
6:       τ ← min(C, (score(H_i; w) − score(Ĥ; w) + Δ(Ĥ, H_i)) / ||Φ(Ĥ) − Φ(H_i)||^2)
7:       w ← w + τ(Φ(H_i) − Φ(Ĥ))
8:     end if
9:   end for
10: end for
11: return w
```

3.3 Learning from Exactly-specified Explanations

In order to train the weight vector $\mathbf{w}$, we employ Passive-Aggressive (PA) algorithm [7], which is a supervised large-margin online learning algorithm applicable to a wide range of linear classifiers ranging from binary classifiers to structured predictors. The motivation is that (i) an online learning makes our framework scalable, and (ii) it has been empirically shown that large-margin approaches demonstrate a superior generalization ability on unseen datasets. In this section, we consider the simplest setting where *exactly-specified* explanations are given as training examples. The framework learns the score function so that it ranks the given explanation itself as the best explanation.

Algorithm 1 depicts our learning algorithm. Every time we receive a training instance (O_i, H_i) from a set $\mathbb{D}$ of training instances, we first find the highest-score explanation $\hat{H}$ given the current weight vector (line 4). If the current prediction has a prediction error, we train the weight vector (line 5–8). A new weight vector $\mathbf{w}$ should satisfy the following conditions: (i) $score(H_i;\mathbf{w})$ is greater than $score(\hat{H};\mathbf{w})$ by at least a margin $\Delta(\hat{H}, H_i)$, and (ii) the difference between the current weight vector $\mathbf{w}'$ and the new weight vector $\mathbf{w}$ is minimal. In line 6, we calculate how much $\mathbf{w}$ should be corrected, where C is a parameter of PA algorithm, meaning the aggressiveness of weight updates. Intuitively, the more different $\hat{H}$ and H_i are, the larger an ensured margin is.

3.4 Learning from Partially-specified Explanations

Let us consider the case where we use abduction for a specific task, and the *subset* of the best explanation is used as the output label of the task. In plan recognition, for example, one might use only plan literals (i.e. literals that represent a plan) in the best explanation to decide the system output, and might not care about any other types of literals in the

Algorithm 2 learnPartial(training examples $\mathbb{D}$, background knowledge B, int N, double C)

1: Initialize $\mathbf{w}$
2: **for** $n = 1$ to N **do**
3: **for all** $(O_i, H_i) \in \mathbb{D}$ **do**
4: $\hat{H} \leftarrow \underset{H \in \mathcal{H}_{O_i,B}}{\arg\max}\ score(H; \mathbf{w})$
5: **if** $H_i \not\subseteq \hat{H}$ **then**
6: $\overline{H} \leftarrow \underset{H \in \mathcal{H}_{O_i,B}}{\arg\max}\ score(H; \mathbf{w})$ subject to $H_i \subseteq H$
7: $\tau \leftarrow \min(C, \frac{score(\overline{H};\mathbf{w}) - score(\hat{H};\mathbf{w}) + \Delta(\overline{H}, H_i)}{||\mathbf{\Phi}(\hat{H}) - \mathbf{\Phi}(\overline{H})||^2})$
8: $\mathbf{w} \leftarrow \mathbf{w} + \tau(\mathbf{\Phi}(\overline{H}) - \mathbf{\Phi}(\hat{H}))$
9: **end if**
10: **end for**
11: **end for**
12: **return** $\mathbf{w}$

explanation. In this situation, the learning framework is required to have the capability of learning the score function from *partially-specified* gold-standard explanations: training a weight vector that can rank *any* explanation that *includes* the gold-standard explanation as the best explanation, because one wants the system to output *any* explanation that *includes* the correct plan literals. Of course, one could exhaustively give all explanations that include the correct plan literals as exactly-specified explanations, but it is intractable in many cases due to the exponential growth of the number of candidate explanations.

Therefore, in this section, we extend the learning algorithm in the previous section to allow the setting where H_i is partially-specified gold-standard explanation. We formulate the learning problem as a discriminative structured learning with latent variables [6], [10], [37] etc., where the output label is a set of literals that are specified in H_i, and the rest are regarded as latent variables.

Algorithm 2 depicts the extended learning algorithm. The key extensions from Algorithm 1 are two folds: (i) we update the weight vector if the partially-specified gold-standard explanation H_i is *not included in* the current prediction $\hat{H}$ (line 5–9), and (ii) we perform *latent variable completion*, the inference to complete the unspecified part of the partially-specified gold-standard explanation H_i (line 6). We refer to the completed explanation as the *pseudo exactly-specified explanation*. Note that one can use k-best completed explanations as pseudo exactly-specified explanations, instead of using one completed explanation. In future, we will compare the performance of the k-best explanations approach with the 1-best explanation approach.

In order to infer $\overline{H}$ in latent variable completion, we follow Yamamoto et al. [36]'s learning framework for abduction, where $\overline{H}$ is the highest-score explanation among candidate explanations that *are the super set of* H_i. To find such $\overline{H}$, we perform abduction with (O_i, B), satisfying the following two constraints: (i) for all *literal* $L \in H_i$, there exists a literal U and a set of substitutions θ in $\overline{H}$ such that $U\theta = L$ (i.e. $\overline{H} \models H_i$), and (ii) for all *substitution* $x = y \in H_i$, $x = y$ must be hypothesized in $\overline{H}$ i.e. $(\overline{H} \models x = y)$. These constraints ensure that the best explanation for O_i entails H_i. To impose constraint (i), we create the feature function $\mathbf{\Phi}_L(H)$ for all $L \in H_i$, which returns $-\infty$ if none of the literals unifiable with L are hypothesized:

$$\mathbf{\Phi}_L(H) = \begin{cases} -\infty & \text{if } H \not\models \bigvee_{L' \in H}(\text{UNIF}(L, L') \wedge L'); \\ 0 & \text{otherwise,} \end{cases} \quad (9)$$

where $\text{UNIF}(L, L')$ is true only if $L \equiv p(x_1, x_2, ..., x_n)$ and $L' \equiv q(y_1, y_2, ..., y_n)$ are unifiable (i.e. $p \equiv q$ and $x_1 = y_1 \wedge x_2 = y_2 \wedge ... \wedge x_n = y_n$); false otherwise. For constraint (ii), we add the following ILP constraints: $s_{x=y} = 1$ for all $x = y \in H_i$, and $s_{x=y} = 0$ for all $x \neq y \in H_i$. Note that the score of $\overline{H}$ will be $-\infty$ when knowledge complete assumption is not satisfied. We skip the weight update if the score is $-\infty$.

4. Evaluation

In this section, we evaluate our online large-margin learning algorithm in two applications to answer the following questions: (i) does the weight vector trained by partially-specified explanations indeed give predictive performance better than the untuned weight vector does? (ii) can machine learning-based abductive reasoning be combined with the powerful existing feature-based classifiers (e.g. Support Vector Machines [35]) for boosting predictive performance? For all experiments, we run our own implementation for the extended version of ILP-based reasoner shown in Sec. 3. 2. The implementation is made publicly available on the web.[⋆3] We used a 12-core Opteron 6174 (2.2GHz) 128 GB RAM machine. We used Gurobi optimizer 5.0[⋆4] as an ILP solver, and 8 cores for solving ILP problems in parallel processing. The parameter C of PA algorithm is set to 1.0 in the experiments. To make the framework more scalable, we implemented the training algorithm in a distributed structure learning framework, following [19].

4. 1 Story Understanding

The task of story understanding is to abductively infer the top-level plans of characters from observed actions. For example, given "*Bill went to the liquor-store. He pointed a gun at the owner*," we need to infer *Bill*'s plan, e.g. *Bill* is robbing at *the liquor store*. By evaluating our algorithm on this task, we want to empirically check whether our algortihm has the capability to learn the signals of "good" explanation from *partially-specified* gold-standard explanations or not.

We used [20]'s story understanding dataset, which is widely used for evaluation of abductive plan recognition systems [17], [26], [33]. The dataset consists of development set and test set, each of which includes 25 pairs of observed actions and its gold-standard plan.[⋆5] In the

⋆3: http://github.com/naoya-i/henry-n700/
⋆4: http://www.gurobi.com/
⋆5: To the best of our knowledge, this dataset is a only public dataset

Table 1: Feature set used for abductive story understanding.

Feature	Description
PREDICATES_HYPOTHESIZED	a set of predicate names of literals that are hypothesized.
PREDICATES_EXPLAINED	a set of predicate names of literals that are explained by at least one set of literals.
PREDICATES_UNIFIED	a set of predicate names of literals that have at least one equivalent literal in a explanation.
AXIOMS_SATISFIED	a set of names of axioms that are satisfied by a explanation.

Table 2: Performance of plan recognition in two settings.

	LOGICAL ABDUCTION				TRAINED			
	Loss	P	R	F	Loss	P	R	F
Closed Test	0.24	0.20	0.40	0.27	**0.12**	**0.35**	**0.69**	**0.46**
Open Test	0.26	0.18	0.44	0.25	**0.18**	**0.28**	**0.57**	**0.37**

dataset, the actions and gold-standard plans are given by a set of first-order literals (e.g. $\{inst(get2, getting)$, $agent\_get(get2, bob2), name(bob2, bob)\}$). The dataset contains on average 12.6 literals in the actions, and 12.0 literals in the gold-standard plans. The dataset also provides the background knowledge base, which contains 107 first-order logical Horn clauses (e.g. $inst(R, robbing) \wedge get\_weapon\_step(R, G) \rightarrow inst(G, getting)$). We use the development set for training, and the test set for measuring predictive performance. We gave the gold standard plan literals as partially-specified gold-standard explanations for training.

To perform plan recognition, we apply abduction with the background knowledge base, giving the observed actions as observations. We summarize the feature vector used for this setting in Table 1. To capture the feature of explanations, we introduce a feature that represents what kinds of literals are included (PREDICATES_HYPOTHESIZED), and explained (PREDICATES_EXPLAINED) in an explanation. We also incorporate the information of axioms satisfied by an explanation (AXIOMS_SATISFIED). PREDICATES_UNIFIED feature captures the following intuition: the information that is supported by many observations (i.e. the situation where the same kind of literal is hypothesized from multiple observations) is more reliable. All the features are encoded by 0-1 features, and each one represents whether each element is included in an explanation.

For the loss function, we want to measure the difference between predicted explanation $\overline{H}$ and the gold-standard explanation H, in terms of plan literals. We used the following function:

$$\Delta(\overline{H}, H) = |H| - |H \cap \overline{H}| + n(\overline{H}), \tag{10}$$

where $n(\overline{H})$ is the number of plan literals in $\overline{H}$ that are not included in H. We considered 10 types of literals as plan literals, following [33]. It is clear that this function is a non-negative, and its value is zero iff (i) $H \subseteq \overline{H}$, and (ii) $\overline{H}$ includes plan literals only specified in H.

For evaluating the prediction performance of our system, we focused on how well the system infers plan literals, including their role fillers, following [33]. More specifically, we use precision (ratio of inferred literals that are correct), recall (ratio of correct literals that are inferred by the system), and F-measure (harmonic mean of precision and recall), because the gold data often has multiple plan literals.

Results and discussion: To see the effect of weight learning, we show the value of loss function averaged for all the problems, and predictive performances for closed test and open test in Table 2. We consider two settings here. In LOGICAL ABDUCTION setting, we try to simulate classical logical abduction that favors the fewer number of elemental explanations: we thus set -1.0 to PREDICATES_HYPOTHESIZED, and 1.0 to PREDICATES_EXPLAINED and PREDICATES_UNIFIED, and do not tune the weights. In TRAINED setting, we used our learning procedure for tuning a weight vector.[⋆6] In both tests, Table 2 indicates that the training algorithm reduced the loss value than classical logical abduction did, so that it improved the predictive performance. The results of open test also reveal that our learning algorithm shows the generalization ability to unseen data.

4.2 NP Coreference Resolution

Noun-phrase (NP) coreference resolution is the task of identifying the group of NPs that refer to the same entity in the world. For example, in the sentence "*Tim shouted at Ed because he was angry.*", we need to identify the group {*he*, *Tim*}. On the other hand, in the sentence "*Tim shouted at Ed because he crashed the car.*", we need to identify the group {*he*, *Ed*}. As the reader can see, coreference resolution requires commonsense reasoning using world knowledge, such as causal relations of events, and synonymous relations of words, etc.

The question here is: what benefits could we receive from the development of machine learning framework for abduction? Our hypothesis is that combining the learning of logical inference with the existing powerful feature-based classifier (e.g. Support Vector Machines [35]) would improve the performance of knowledge-intensive tasks such as coreference resolution. Therefore, we compare the predictive performance of feature-based classifier with a machine learning-based abductive reasoning procedure combined with the existing feature-based classifiers, using coreference resolution as a test bed. To simulate the feature-based classifiers, we created a feature function for each pair of literals that represent NPs, following the feature set proposed by Soon et al. [34], which is widely used as the simple baseline model of

that provides a complete test environment for abduction, although it is small. We plan to create the bigger dataset for future evaluation.

⋆6: A weight vector is initialized with the zero vector.

Table 3: Performance of NP coreference resolution, provided by feature-based classifier and abductive reasoner combined with feature-based classifier.

Setting	System	Pairwise Loss
Closed Test	Soon	0.40
	Soon+Abduction	**0.29**
Open Test	Soon	0.55
	Soon+Abduction	**0.48**

coreference resolution. Henceforth, we call it Soon system.

To solve coreference problems with abduction using world knowledge, we adopt the idea of Interpretation as Abduction [12]. The idea is that the interpretation of sentences is an abductive explanation to the logical forms (LFs) of sentences, where substitutions correspond to the identification of coreference relations. We thus perform abduction with world knowledge, giving the LFs of text as an observation. We then extract substitutions from the best explanation for identifying the coreference relations. For combining the abductive reasoning with Soon system, we use the feature set summarized in Table 1 and the feature set of Soon system simultaneously in the score function. The resulting system is called Soon+Abduction.

We use the CoNLL-2011 shared task dataset [25].⋆7 We used 100 documents of training dataset for training, and 100 documents from development dataset for testing. We convert the dataset into the logical forms, and encode the gold-standard coreference annotations as substitutions. We then give the substitutions as partially-specified gold-standard explanations. We used Boxer semantic parser [3] for the logical form conversion. As a world knowledge, we used WordNet [1] and FrameNet [30]. We convert the world knowledge to the form of axioms, such as $synsetX(s) \rightarrow dog(s)$, following Ovchinnikova [21].

For the loss function, we used a pairwise loss function $\Delta_P(\overline{H}, H) = W_O/T_O$, where T_O is the number of pairs of variables in the observation and W_O is the number of substitutions for observed variables (i.e. variables representing NPs) in H that disagrees with $\overline{H}$. The pairwise loss function is also used for supervised clustering-based coreference resolution [11]. Again, it is clear that this function is a non-negative, and its value is zero iff there are no disagreement.

Results and discussion: Table 3 shows the values of pairwise loss function in closed test and open test setting. For Soon+Abduction, we initialized the weight vector with the same value as Logical Abduction setting in the story understanding setting, and then trained the weights. In both settings, the loss of Soon+Abduction system is less than Soon system. This indicates that combining the learning of logical inference using the world knowledge with feature-based classifier has a positive impact to the predictive performance of feature-based classifier. In our future work, we will conduct an additional experiment to check the best way to exploit the world knowldge: comparing the results with the performance of feature-based classifier using the world knowledge as a feature.

⋆7: http://conll.cemantix.org/2011/.

5. Related Work

Probabilistic logical abduction has been studied in the context of Statistical Relational Learning [2], [17], [26], [33] etc. They assume to use the standard learning algorithms of probabilistic models (e.g. EM) for learning the score function. However, the inference of probabilistic models for first-order logical inference is computationally expensive, because the inference is performed on a propositional level. Due to the intractability of inference, some work report that they could not learn weights on large dataset [2], [33]. Raghavan and Mooney [26] propose Bayesian Abductive Logic Programs, which constructs a Bayesian Network by using the backward-chaining procedure similar to the ILP-based approach, but they use a task-specific heuristic rule to unify literals to reduce the computational complexity of inference during the construction of the network. Given much larger and dataset in general domain, their framework would not be a scalable solution. Other researchers [2], [17], [33] employ Markov Logic Networks (MLNs) [27] to emulate abductive inference. MLNs provide well-studied software packages of inference and learning; however, MLN-based approaches require special procedures to convert abduction problems into deduction problems because of the deductive nature of MLNs. The pioneering work of MLN-based abduction [17] converts background axioms into MLN logical formulae by (i) reversing implication and (ii) constructing axioms representing mutual exclusiveness of explanation (e.g. the set of background knowledge axioms $\{p_1 \rightarrow q, p_2 \rightarrow q, p_3 \rightarrow q\}$ is converted into the following MLN formulae: $q \rightarrow p_1 \vee p_2 \vee p_3$, $q \rightarrow \neg p_1 \vee \neg p_2$, $q \rightarrow \neg p_1 \vee \neg p_3$ etc.). As the readers can imagine, MLN-based approach suffers from the inefficiency of inference due to the increase of converted axioms. In addition, the current solution of MAP inference for MLNs, which is needed for the best explanation finding, works on a propositional level. Therefore, learning would not scale to larger problems due to the severe overhead [15]. Singla and Mooney [33] report that their MLN-based abduction models cannot be trained on larger dataset.

As mentioned in Sec. 3. 4, Yamamoto et al. formulate the learning problem of first-order logic abduction as the framework similar to us. The key difference is that they use score function that is non-linear in terms of weights, and thus use a different optimization strategy for optimizing the weights. Comparing the performance of our work with them is interesting and important future direction. Our work is also related to a structured learning approaches that exploit latent variables, which demonstrate a superior performance in many tasks ranging from natural language processing to graphical processing. For example, Latent Support Vector Machines, a variant of structured learning model with latent variables, is widely used [6], [10], [37] etc. for many classification tasks, and shown to outperform the existing systems.

6. Conclusion

We have proposed a supervised approach for learning the score function of weighted abduction. We formulated the learning procedure in the framework of structured learning with latent variables. Our approach enables us to learn the score function from partially-specified gold-standards, which is a useful feature in real-life tasks. In our evaluation, we found that our learning procedure can reduce the loss, and improve predictive performance of story understanding tasks in both open test and closed test. We also explored the potential use of machine learning-based abductive reasoning, i.e. the integration of learning of logical inference and feature-based classifiers. The experiments showed that the integration of these two approaches is promising.

Our future direction includes improving the training time, where the bottleneck of training is inference procedure. We thus need to develop more efficient inference method that returns high-quality solution in a short time. We are planning to apply Cutting Plane Inference for Markov Logic Networks [28], which gradually instantiates logical formulae in knowledge base on the fly. Another direction is extending our learning framework. As mentioned in Sec. 3. 4, we plan to start with incorporating the k-best update into our framework such as [8].

Acknowledgement

This work was partially supported by Grant-in-Aid for JSPS Fellows (22-9719), Grant-in-Aid for Scientific Research (23700157, 23240018), and JST.

References

[1] WordNet: an electronic lexical database. 1998.

[2] J. Blythe, J. R. Hobbs, P. Domingos, R. J. Kate, and R. J. Mooney. Implementing Weighted Abduction in Markov Logic. In *IWCS*, pages 55–64, 2011.

[3] J. Bos. Wide-Coverage Semantic Analysis with Boxer. In J. Bos and R. Delmonte, editors, *STEP*, Research in Computational Semantics, pages 277–286. College Publications, 2008.

[4] N. Chambers and D. Jurafsky. Unsupervised Learning of Narrative Schemas and their Participants. In *ACL*, pages 602–610, 2009.

[5] E. Charniak and R. P. Goldman. A Probabilistic Model of Plan Recognition. In *AAAI*, pages 160–165, 1991.

[6] C. Cherry and C. Quirk. Discriminative, syntactic language modeling through latent svms. In *In AMTA*, 2008.

[7] K. Crammer, O. Dekel, J. Keshet, and Y. Singer S. Shalev-Shwartz. Online Passive-Aggressive Algorithms. pages 551–585, 2006.

[8] K. Crammer, R. Mcdonald, and F. Pereira. Scalable large-margin online learning for structured classification, 2005.

[9] I. Dagan, B. Dolan, B. Magnini, and D. Roth. Recognizing textual entailment: Rational, evaluation and approaches - Erratum. *NLG*, 16(1):105, 2010.

[10] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans.*, 32(9):1627–1645, 2010.

[11] T. Finley and T. Joachims. Supervised clustering with support vector machines. In *ICML*, pages 217–224, 2005.

[12] J. R. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. *Artificial Intelligence*, 63:69–142, 1993.

[13] D. Hovy, C. Zhang, E. Hovy, and A. Penas. Unsupervised discovery of domain-specific knowledge from text. In *ACL*, pages 1466–1475, 2011.

[14] T. N. Huynh and R. J. Mooney. Max-Margin Weight Learning for Markov Logic Networks. In *SRL*, 2009.

[15] N. Inoue and K. Inui. Large-scale Cost-based Abduction in Full-fledged First-order Predicate Logic with Cutting Plane Inference, 2012.

[16] k. Kersting and L. De Raedt. Bayesian logic programs. Technical report, 2001.

[17] R. J. Kate and R. J. Mooney. Probabilistic abduction using markov logic networks. In *PAIR*, 2009.

[18] D. Lowd and P. Domingos. Efficient Weight Learning for Markov Logic Networks. In *PKDD*, pages 200–211, 2007.

[19] R. McDonald, K. Hall, and G. Mann. Distributed training strategies for the structured perceptron. In *NAACL2010*, pages 456–464, 2010.

[20] H. T. Ng and R. J. Mooney. Abductive plan recognition and diagnosis: A comprehensive empirical evaluation. In *KR*, pages 499–508, 1992.

[21] E. Ovchinnikova. *Integration of World Knowledge for Natural Language Understanding*. Atlantis Press, Springer, 2012.

[22] E. Ovchinnikova, N. Montazeri, T. Alexandrov, J. R Hobbs, M. McCord, and R. Mulkar-Mehta. Abductive Reasoning with a Large Knowledge Base for Discourse Processing. In *IWCS*, pages 225–234, 2011.

[23] S. E. Peraldi, A. Kaya, S. Melzer, R. Möller, and M. Wessel. Multimedia Interpretation as Abduction. In *Int'l Workshop on Description Logics*, 2007.

[24] D. Poole. Logic programming, abduction and probability: a top-down anytime algorithm for estimating prior and posterior probabilities. *New Gen. Comput.*, 11(3-4):377–400, 1993.

[25] S. Pradhan, L. Ramshaw, M. Marcus, M. Palmer, R. Weischedel, and N. Xue. CoNLL-2011 shared task.

[26] S. Raghavan and R. J. Mooney. Bayesian Abductive Logic Programs. In *Star-AI 10*, pages 82–87, 2010.

[27] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, pages 107–136, 2006.

[28] S. Riedel. Improving the Accuracy and Efficiency of MAP Inference for Markov Logic. In *UAI*, pages 468–475, 2008.

[29] J. A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12:23–41, 1965.

[30] J. Ruppenhofer, M. Ellsworth, M.R. Petruck, C.R. Johnson, and J. Scheffczyk. FrameNet II: Extended Theory and Practice. Technical report, 2010.

[31] S. Schoenmackers, J. Davis, O. Etzioni, and D. Weld. Learning First-order Horn Clauses from Web Text. In *EMNLP*, pages 1088–1098, 2010.

[32] M. Shanahan. Perception as Abduction: Turning Sensor Data Into Meaningful Representation. *Cognitive Science*, 29(1):103–134, 2005.

[33] P. Singla and P. Domingos. Abductive Markov Logic for Plan Recognition. In *AAAI-11*, pages 1069–1075, 2011.

[34] W. M. Soon, H. T. Ng, and D. C. Y. Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics.*, 27(4):521–544, 2001.

[35] N. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[36] K. Yamamoto, N. Inoue, Y. Watanabe, N. Okazaki, and K. Inui. Backpropagation Learning for Weighted Abduction (in Japanese). In *IPSJ SIG Technical Reports*, volume 2012-NL-206.

[37] C. J. Yu and T. Joachims. Learning structural SVMs with latent variables. In *ICML*, 2009.