# Logical inference as cost minimization in vector spaces

**Taisuke Sato[1], Ryosuke Kojima[2]**
**[1] AI research center AIST, Japan**
**[2] Graduate School of Medicine, Kyoto University, Japan**
**satou.taisuke@aist.go.jp, kojima.ryosuke.8e@kyoto-u.ac.jp**

## Abstract

We propose a differentiable framework for logic program inference as a step toward realizing flexible and scalable logical inference. The basic idea is to replace symbolic search appearing in logical inference by the minimization of a cost function $\mathbf{J}$ in a continuous space. $\mathbf{J}$ is made up of matrix (tensor), Frobenius norm and non-linear functions just like neural networks and specifically designed for each task (relation abduction, answer set computation, etc) in such a way that $\mathbf{J}(\mathbf{X}) \geq 0$ and $\mathbf{J}(\mathbf{X}) = 0$ holds if-and-only-if $\mathbf{X}$ is a 0-1 tensor representing a solution for the task. We compute the minimizer X of $\mathbf{J}$ giving $\mathbf{J}(\mathbf{X}) = 0$ by gradient descent or Newton's method. Using artificial data and real data, we empirically show the potential of our approach by a variety of tasks including abduction, random SAT, rule refinement and probabilistic modeling based on answer set (supported model) sampling.

## 1 Introduction

We propose a differentiable framework for logic program inference as a step toward realizing flexible and scalable logical inference. The basic idea is to replace symbolic search by the minimization of a cost function $\mathbf{J}$ in a continuous space. We choose $\mathbf{J}$ specifically to each task and perform from abduction to probabilistic inference.

Consider the case of abduction where the task is to abduce a binary relation $r_2(Y,Z)$ satisfying $r_3(X,Z) \Leftrightarrow \exists Y\, r_1(X,Y) \wedge r_2(Y,Z)$ for given $r_3(X,Z)$ and $r_1(X,Y)$[1]. $\mathbf{J}$ is chosen as

$$\mathbf{J}^{abd}(\mathbf{X}) = \frac{1}{2}\{\|\mathbf{R}_3 - \min_1(\mathbf{R}_1\mathbf{X})\|_F^2 + \ell \cdot \|\mathbf{X} \odot (\mathbb{1} - \mathbf{X})\|_F^2\}. \quad (1)$$

Here $\mathbf{R}_1$ and $\mathbf{R}_3$ are adjacency matrices (i.e. 0-1 matrices) representing $r_1(X,Y)$ and $r_3(X,Z)$ respectively. $\min_1(x)$ is a non-linear function defined by $\min_1(x) = \min(x,1)$(the lesser of $x$ and 1)[2], $\odot$ denotes element-wise product, $\|\cdot\|_F$ Frobenius norm and $\mathbb{1}$ a matrix of all ones. By construction, $\mathbf{J}^{\rm abd}(\mathbf{X}) = 0$ if-and-only-if $\mathbf{X}$ is a 0-1 matrix[3] satisfying $\mathbf{R}_3 = \min_1(\mathbf{R}_1\mathbf{X})$. The latter implies $\mathbf{X}$ is an adjacency matrix representing a binary relation $r_2(Y,Z)$ s.t. $r_3(X,Z) \Leftrightarrow \exists Y\, r_1(X,Y) \wedge r_2(Y,Z)$.

Note that our abductive setting of solving $\mathbf{R}_3 = \min_1(\mathbf{R}_1\mathbf{X})$ is broad and includes SAT problem as a special case where $\mathbf{R}_3 = \mathbf{1}$ is a column vector with every element being one, $\mathbf{R}_1 = \mathbf{Q}_{\rm CNF}$ is a 0-1 matrix encoding clauses and $\mathbf{X}$ is a 0-1 column vector representing a truth-assignment (see Subsection 3.3).

Also the problem of finding supported models (answer sets)[Gelfond and Lifshcitz, 1988; Marek and V.S.Subrahmanian, 1992] is similarly formulated as a cost minimization problem of some $\mathbf{J}$ derived from normal logic programs.

In general, in our approach, $\mathbf{J}$ is constructed in such a way that $\mathbf{J}(\mathbf{X}) \geq 0$ and $\mathbf{J}(\mathbf{X}) = 0$ holds if-and-only-if $\mathbf{X}$ is a 0-1 tensor (multi-linear function, a generalization of vector and matrix) representing a solution such as relations and answer sets. Logical inference is therefore achieved either by minimizing $\mathbf{J}$ to zero or by root finding of $\mathbf{J}$. We carry out the former by gradient descent (GD) and the latter by Newton's method. On the numerical side however, it is difficult and time-consuming to reduce $\mathbf{J}$ to completely zero. We therefore incorporate a "jumping to a solution" strategy which means while updating $\mathbf{X}$ by GD or by Newton's method, we threshold $\mathbf{X}$ into a 0-1 tensor using an appropriate thresholding value to see if the resulting $\mathbf{X}$ constitutes a solution. Although combining continuous relaxation of symbolic model search with thresholding is somewhat ad hoc, it works and opens a new way of logical inference using tensors.

One notable thing with $\mathbf{J}$ is that it enables random sampling of solutions (target relations and/or models). Consider for example the sampling of answer sets, or more specifically the sampling of supported models of a normal logic program DB[Gelfond and Lifshcitz, 1988; Marek and V.S.Subrahmanian, 1992]. Usually DB has multiple supported models and which model is reached by minimizing $\mathbf{J}$ to zero depends on the initialization of $\mathbf{X}$ in GD or Newton's method. By uniformly sampling an initial point from $[0,1]^n$, we can expect to perform (semi) uniform sampling of sup-

[1]We follow Prolog convention and logical variables begin with upper case letters.

[2]For a matrix $\mathbf{A}$, $\min_1(\mathbf{A})$ indicates element-wise application of $\min_1(x)$ to $\mathbf{A}$.

[3]$\|\mathbf{X} \odot (\mathbb{1} - \mathbf{X})\|_F^2 = \sum_{ij} \mathbf{X}_{ij}^2 (1 - \mathbf{X}_{ij})^2 = 0$ implies $\mathbf{X}_{ij}(1 - \mathbf{X}_{ij}) = 0$ for all $i, j$ where $\mathbf{X}_{ij}$ denotes the $(i,j)$ element of $\mathbf{X}$.

ported models as exemplified later by our experiments. Also by introducing a dynamic cost function that stochastically changes as sampling proceeds, "distribution-aware" sampling is realized and applied to probabilistic inference for supported models.

Performing logical inference or more generally symbolic reasoning in a continuous space is a rather old idea but the recent explosion of deep learning technologies reignites interest in relating symbolic approaches to neural networks [Widdows and Cohen, 2015; Rocktäschel and Riedel, 2017; Cohen *et al.*, 2017; Kazemi and Poole, 2018; Manhaeve *et al.*, 2018]. Our approach has a lot in common with these differentiable approaches in that we exploit the flexibility and scalability of linear algebraic operations combined with nonlinear activity functions supported by modern computer technologies such as many cores and GPGPUs. We however differ in that we entirely reformulate logical inference as a simple cost minimization problem and are focused on logical/probabilistic inference, not on raw data processing such as image data and text data.

In what follows, after a preliminary section, we show how to perform exact abduction in a continuous space in Section 3. Then we apply this technique to random SAT problem and real data, i.e. rule discovery from knowledge graphs. Then we move on to the problem of sampling supported models (answer sets) [Gelfond and Lifshcitz, 1988; Baral *et al.*, 2009; Eiter *et al.*, 2009] with constraints in Section 4 and apply it to probabilistic modeling by distribution-aware sampling in Section 5. The reader is supposed to be familiar with the basics of logic programming. Due to space limitations, description tends to be sketchy and example-based for intuitiveness.

## 2 Preliminaries

### 2.1 Tensorized semantics

To perform logical inference in terms of vectors and matrices, we adopt a tensorized semantics introduced in [Sato, 2017]. For a syntactic object $A$, we use $[\![A]\!]$ to denote its interpretation (entities, relations, truth values). We assume a finite Herbrand domain $\{e_1,\ldots,e_n\}$ and each entity $e_i$ $(1 \leq i \leq n)$ is represented by a one-hot column vector $\mathbf{e}_i = [0..,1,0..]^T$ which is a zero vector except the $i$-th element being 1. An $m$-ary predicate $p/m$ is identified as an $m$-ary multi-linear mapping in the $n$-dimensional vector space and a proposition (ground atom) $p(e_{i_1},\ldots,e_{i_m})$ takes a truth value 1 when true else 0, i.e, $[\![p(e_{i_1},\ldots,e_{i_m})]\!] \in \{1,0\}$. Note a unary predicate is computed by a dot product like $[\![p(e_i)]\!] = (\mathbf{p} \bullet \mathbf{e}_i)$ using a 0-1 vector $\mathbf{p}$ representing $p$. Likewise a binary predicate is computed as $[\![r(e_i,e_j)]\!] = \mathbf{e}_i^T \mathbf{R} \mathbf{e}_j = \mathbf{R}_{ij}$ using a 0-1 matrix $\mathbf{R}$ encoding $r$. Compound boolean formulas are inductively computed by $[\![\neg A]\!] = 1 - [\![A]\!]$, $[\![A \wedge B]\!] = [\![A]\!] \cdot [\![B]\!]$ and $[\![A \vee B]\!] = \min_1([\![A]\!] + [\![B]\!])$. Existential quantifiers are treated by grounding them to disjunctions. So we have $[\![\exists X p(X)]\!] = \min_1([\![p(e_1)]\!] + \cdots + [\![p(e_n)]\!])$. Fortunately however, oftentimes this grounding process is eliminable in particular in the case of existentially quantified conjunctions like

$$\begin{aligned} [\![\exists Y r_1(X,Y) \wedge r_2(Y,Z)]\!] &= \min_1(\sum_j \mathbf{x}^T \mathbf{R}_1 \mathbf{e}_j \mathbf{e}_j^T \mathbf{R}_2 \mathbf{z}) \\ &= \mathbf{x}^T \min_1(\mathbf{R}_1 \mathbf{R}_2) \mathbf{z} \qquad (2) \end{aligned}$$

where $\mathbf{x}$ and $\mathbf{z}$ are arbitrary entity vectors substituted for $X$ and $Z$ respectively.

Let $\mathbf{X}$ be a matrix. We use $|\mathbf{X}|$ to denote the number of nonzero elements of $\mathbf{X}$. So when $\mathbf{R}$ is a 0-1 matrix representing a binary relation, $|\mathbf{R}|$ coincides with the size of the relation.

### 2.2 Supported model and tensor equation

Let DB be a ground normal logic program and consider clauses $\{h \Leftarrow B_1,\ldots,h \Leftarrow B_k\}$ about a ground atom $h$ in DB. $B_j$s are conjunctions of ground literals and can be empty (empty body is considered true). Denote the boolean formula $h \Leftrightarrow B_1 \vee \cdots \vee B_k$ by $\mathrm{iff}(h)$ and define $\mathrm{iff}(\mathrm{DB}) = \{\mathrm{iff}(h) \mid h \text{ occurs in DB}\}$. $\mathrm{iff}(\mathrm{DB})$ is said to be a *completion form* of DB.

A *model* of $\mathrm{iff}(\mathrm{DB})$, i.e., a truth assignment making every $\mathrm{iff}(h) \in \mathrm{iff}(\mathrm{DB})$ true is called a *supported model* of DB[Gelfond and Lifshcitz, 1988; Baral *et al.*, 2009; Eiter *et al.*, 2009]. Supported models constitute a super class of stable models (answer sets) and when DB is tight, i.e. no looping dependency through positive atoms in DB, supported models and stable models coincide. In this paper, we assume programs are tight. When DB is non-ground, we always treat it as the set of all ground instantiations of the clauses in DB.

Now look at non-ground Datalog programs[Kakas *et al.*, 1992; Eiter *et al.*, 1997; Gottlob *et al.*, 2010]. We show how tensor equations are derived from their supported models. Although more complex classes are possible to deal with, we concentrate on a simple class $\mathscr{C}$ of DB containing only binary predicates such that each clause of the form $r_0(X_0,X_m) \Leftarrow r_1(X_0,X_1) \wedge \cdots \wedge r_m(X_{m-1},X_m)$. We furthermore assume that when DB is recursive, clauses have only one recursive goal in their clause body.

The following is a non-ground Datalog program (in the completion form) in $\mathscr{C}$ which takes the transitive closure $r_2(X,Y)$ of a base relation $r_1(X,Y)$.

$$r_2(X,Z) \quad \Leftrightarrow \quad r_1(X,Z) \vee \exists Y (r_1(X,Y) \wedge r_2(Y,Z)) \qquad (3)$$

In every supported model, both sides of (3) denote the same truth value for any instantiations of $X,Z$. Hence we have

$$[\![r_2(X,Z)]\!] \quad = \quad [\![r_1(X,Z) \vee \exists Y (r_1(X,Y) \wedge r_2(Y,Z))]\!].$$

By applying the interpretation mapping (2) to $\exists Y (r_1(X,Y) \wedge r_2(Y,Z))$, we obtain a matrix equation

$$\mathbf{R}_2 \quad = \quad \min_1(\mathbf{R}_1 + \mathbf{R}_1 \mathbf{R}_2) \qquad (4)$$

where $\mathbf{R}_1$ and $\mathbf{R}_2$ are 0-1 matrices respectively encoding $r_1(X,Z)$ and $r_2(X,Z)$ in a supported model. Since the derivation from (3) to (4) can go the other way around, computing supported models of (3) is equivalent to solving (4) using 0-1 matrices. By generalization, we see that supported models of a program in $\mathscr{C}$ are characterized as a solution of matrix equation derived from it (proof omitted).

## 3 Abducing relations by cost minimization

In this section, we solve the abduction problem described in the beginning of Section 1: given $r_3(X,Z)$ and $r_1(X,Y)$, find a binary relation $r_2(Y,Z)$ satisfying $r_3(X,Z) \Leftrightarrow \exists Y r_1(X,Y) \wedge$

$r_2(Y,Z)$ by minimizing $\mathbf{J}^{abd}(\mathbf{X}) = \frac{1}{2}\{\|\mathbf{R}_3 - \min_1(\mathbf{R}_1\mathbf{X})\|_F^2 + \ell \cdot \|\mathbf{X} \odot (\mathbb{1} - \mathbf{X})\|_F^2\}$ w.r.t. $\mathbf{X}$ to zero. Here $\mathbf{R}_3$ and $\mathbf{R}_1$ are respectively 0-1 matrices representing $r_3(X,Z)$ and $r_1(X,Y)$.

## 3.1 Jacobian

To tackle this problem, we first derive a Jacobian $\partial\mathbf{J}^{abd}/\partial\mathbf{X}$[4]. For a matrix $\mathbf{A}$, write $\mathbf{A}_{\leq 1}$ to denote a 0-1 matrix defined by $(\mathbf{A}_{\leq 1})_{ij} = \begin{cases} 1 & \text{if } \mathbf{A}_{ij} \leq 1 \\ 0 & \text{otherwise} \end{cases}$. Then we can compute the Jacobian as follows (derivation omitted).

$$\begin{aligned} \mathbf{J}_{\mathbf{a}}^{abd} &= \partial\mathbf{J}^{abd}/\partial\mathbf{X} \\ &= \mathbf{R}_1^T((\mathbf{R}_1\mathbf{X})_{\leq 1} \odot (\mathbf{R}_1\mathbf{X} - \mathbf{R}_3)) \\ &\quad + \ell \cdot (\mathbf{X} \odot (\mathbb{1} - \mathbf{X}) \odot (\mathbb{1} - 2\mathbf{X})) \end{aligned} \tag{5}$$

Using this Jacobian, we update $\mathbf{X}$ by

$$\text{Gradient descent: } \mathbf{X}_{new} \leftarrow \mathbf{X} - \alpha\mathbf{J}_{\mathbf{a}}^{abd} \quad \text{or} \tag{6}$$

$$\text{Newton's method: } \mathbf{X}_{new} \leftarrow \mathbf{X} - (\mathbf{J}^{abd}/\|\mathbf{J}_{\mathbf{a}}^{abd}\|_F^2)\mathbf{J}_{\mathbf{a}}^{abd}. \tag{7}$$

Here $\alpha$ in (6) is a learning rate. (7) is derived from the first order Taylor polynomial of $\mathbf{J}^{abd}$ and obtained by solving $\mathbf{J}^{abd} + (\mathbf{J}_{\mathbf{a}}^{abd} \bullet \mathbf{X}_{new} - \mathbf{X}) = 0$[5].

Below is an abduction algorithm to search for a 0-1 matrix $\mathbf{R}_2$ representing $r_2(X,Y)$. It starts from an approximate solution (line 3) and then iteratively reduces $\mathbf{J}^{abd}$ toward zero while checking whether thresholding $\mathbf{X}$ gives an exact solution (line 5,6) in each iteration. We stop updating $\mathbf{X}$ when an exact solution is obtained (line 7) or $i$ reaches *max_itr*.

**Algorithm 1:** Relation abduction by matrix

```
1 Input: 0-1 matrices $\mathbf{R}_3(l \times n)$, $\mathbf{R}_1(l \times m)$
2 Output: 0-1 matrix $\mathbf{R}_2(m \times n)$ s.t. $\mathbf{R}_3 = \min_1(\mathbf{R}_1\mathbf{R}_2)$
3 $\mathbf{X} \leftarrow \min_{\mathbf{X}} \|\mathbf{R}_3 - \mathbf{R}_1\mathbf{X}\|_F^2 + \lambda\|\mathbf{X}\|_F^2$
4 for $i \leftarrow 1$ to max_itr do
5     $\mathbf{R}_2 \leftarrow \mathbf{X}_{>\theta}$ for some $\theta$
6     error $\leftarrow |\mathbf{R}_3 - \min_1(\mathbf{R}_1\mathbf{R}_2)|$
7     if error = 0 then
          break
8     Update $\mathbf{X}$ by (6) or by (7)
9 return $\mathbf{R}_2$
```

The initial value of $\mathbf{X}$ (line 3) that minimizes the r.h.s. is given by $\mathbf{X} = (\lambda\mathbf{I} + \mathbf{R}_1^T\mathbf{R}_1)^{-1}(\mathbf{R}_1^T\mathbf{R}_3)$ where $\mathbf{I}$ is an identity matrix(derivation omitted). $\mathbf{X}_{>\theta}$ (line 5) denotes a 0-1 matrix computed by thresholding like $(\mathbf{X}_{>\theta})_{ij} = \begin{cases} 1 & \text{if } \mathbf{X}_{ij} > \theta \\ 0 & \text{otherwise} \end{cases}$. The best $\theta$ that minimizes *error* is chosen by grid search between the minimum and maximum element of $\mathbf{X}$ divided into 50 steps.

[4]$\min_1(x)$ is differentiable except at one point $x = 1$ and hence $\partial\mathbf{J}^{\mathrm{abd}}/\partial\mathbf{X}$ is almost everywhere differentiable.

[5]For matrices $\mathbf{A}, \mathbf{B}$, $(\mathbf{A} \bullet \mathbf{B}) = \sum_{ij}\mathbf{A}_{ij}\mathbf{B}_{ij}$.

## 3.2 Experiment with random relations

To see the performance of our cost minimization approach, we conduct an experiment with relatively large relations, i.e., matrices[6]. Given $n$, we generate two $(n \times n)$ random 0-1 matrices $\mathbf{R}_1$ and $\mathbf{Y}$ s.t. an element is one with probability $p = 0.001$ and compute $\mathbf{R}_3 = \min_1(\mathbf{R}_1\mathbf{Y})$ as test data. Then we run the **Algorithm1** using Newton's method (7) with $\mathbf{R}_3$ and $\mathbf{R}_1$ as input to abduce $\mathbf{R}_2$ and measure an *error* $= |\mathbf{R}_3 - \min_1(\mathbf{R}_1\mathbf{R}_2)|$. *max_itr* is set to 1,000. For $n$ varying from 1,000 to 10,000, we repeat this process five times and abduce five solutions for $\mathbf{R}_2$ and compute averages of $|\mathbf{R}_3|$, *error* and execution time. The result is summarized in Table 1. Figures are averages over five trials. There "ini_error" is the error caused by an initial value of $\mathbf{X}$ (line 3) before iteration and "final_error" is the error by the returned $\mathbf{R}_2$.

According to Table 1, we have achieved zero error for every $n$ and have successfully obtained an exact $\mathbf{R}_2$ that makes $\mathbf{R}_3 = \min_1(\mathbf{R}_1\mathbf{R}_2)$ true. Execution time looks rather linear w.r.t. $n$, which empirically supports the scalability of our approach. Note that the recursive case (4) of abducing $\mathbf{R}_1$ from $\mathbf{R}_2 = \min_1(\mathbf{R}_1 + \mathbf{R}_1\mathbf{R}_2) = \min_1(\mathbf{R}_1(\mathbf{I} + \mathbf{R}_2))$ is treated similarly by setting $\mathbf{R}_3 \leftarrow \mathbf{R}_2^T$, $\mathbf{R}_1 \leftarrow (\mathbf{I} + \mathbf{R}_2)^T$ and $\mathbf{R}_2 \leftarrow \mathbf{R}_1^T$.

## 3.3 3-SAT

Here we briefly show how to apply our approach to SAT problem. Consider a boolean formula in CNF: $(a \vee b \vee \bar{c}) \wedge (a \vee \bar{b})$. We encode this CNF as a 0-1 matrix $\mathbf{Q}_{CNF}$ by separately encoding positive literals and negative literal as follows where rows represent clauses.

$$\mathbf{Q}_{CNF} = \begin{array}{c} \begin{array}{cccccc} a & b & c & \bar{a} & \bar{b} & \bar{c} \end{array} \\ \left[\begin{array}{cccccc} 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \end{array}$$

As is obvious from this example, we can encode a SAT problem with $n$ variables and $m$ clauses by an $(m \times 2n)$ 0-1 matrix $\mathbf{Q}_{CNF}$. Let $\mathbf{u}$ be a 0-1 vector representing an assignment to $n$ variables (1:true,0:false). Then a model (solution) of $\mathbf{Q}_{CNF}$ is represented by $(2n \times 1)$ 0-1 vector of the form $[\mathbf{u}; \mathbf{1}_n - \mathbf{u}]$ s.t. $\mathbf{1}_m = \min_1(\mathbf{Q}_{CNF}[\mathbf{u}; \mathbf{1}_n - \mathbf{u}])$. Here $\mathbf{1}_n$ is a vector of ones with length $n$. $[\mathbf{u}; \mathbf{v}]$ stands for a vertical concatenation of column vectors $\mathbf{u}$ and $\mathbf{v}$. Thus, solving a SAT problem is considered as a form of abduction that abduces $\mathbf{u}$ satisfying $\mathbf{1}_m = \min_1(\mathbf{Q}_{CNF}[\mathbf{u}; \mathbf{1}_n - \mathbf{u}])$ for the given $\mathbf{Q}_{CNF}$.

Write $\mathbf{Q}_{CNF} = [\mathbf{Q}_1\,\mathbf{Q}_2]$ where $\mathbf{Q}_i (i = 1,2)$ is an $(m \times n)$ matrix and introduce a cost function $\mathbf{J}^{sat}$ and its Jacobian $\mathbf{J}_{\mathbf{a}}^{sat}$ as follows (derivation omitted).

$$\begin{aligned} \mathbf{J}^{sat} &= (\mathbf{1}_m \bullet \mathbf{1}_m - \min_1(\mathbf{Q}_{CNF}[\mathbf{u}; \mathbf{1}_n - \mathbf{u}])) \\ &\quad + (\ell/2) \cdot \|\mathbf{u} \odot (\mathbf{1}_n - \mathbf{u})\|_F^2 \end{aligned} \tag{8}$$

$$\begin{aligned} \mathbf{J}_{\mathbf{a}}^{sat} &= (\mathbf{Q}_2 - \mathbf{Q}_1)^T(\mathbf{Q}_1\mathbf{u} + \mathbf{Q}_2(\mathbf{1}_n - \mathbf{u}))_{\leq 1} \\ &\quad + \ell \cdot (\mathbf{u} \odot (\mathbf{1}_n - \mathbf{u}) \odot (\mathbf{1}_n - 2\mathbf{u})) \end{aligned} \tag{9}$$

It is easily proved that $\mathbf{J}^{sat} = 0$ if-and-only-if $\mathbf{u}$ is a 0-1 vector representing a satisfying assignment for the original SAT problem.

[6]All experiments in this paper are carried out using GNU Octave 4.2.2 and Python 3.6.3 on a PC with Intel(R) Core(TM) i7-3770@3.40GHz CPU, 28GB memory.

| $n$ | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 | 8000 | 9000 | 10000 |
|---|---|---|---|---|---|---|---|---|---|---|
| $\|\mathbf{R}_3\|$ | 1005.4 | 7985.2 | 27036.0 | 63582.0 | 125245.0 | 215527.0 | 343080.0 | 50777.0 | 730999.0 | 995653.0 |
| ini_error | 32.8 | 226.2 | 285.0 | 297.0 | 182.0 | 116.4 | 56.6 | 39.7 | 17.3 | 10.0 |
| final_error | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| time(s) | 2.6 | 73.8 | 661.8 | 1608.2 | 1920.0 | 1870.2 | 2300.6 | 2874.7 | 3306.7 | 4255.3 |

Table 1: Abducing relation $\mathbf{R}_2 : \mathbf{R}_3 = \min_1(\mathbf{R}_1\mathbf{R}_2)$

We preliminarily implement our approach as a SAT solver named AbdSat using GNU Octave 4.2.2 and conduct a random 3-SAT experiment comparing AbdSat and MapleLCMDistChronoBT(MapleBT for short here)[Heule *et al.*, 2018], the SAT Competition 2018 Main Track winner as follows. Given $n$, the number of boolean variables, we generate a satisfiable SAT instance by first randomly generating a "hidden" assignment $\mathbf{v}$ over $n$ variables, and then randomly generating $m$ clauses consisting of 3 literals satisfied by $\mathbf{v}$. We collect them as an $(m \times 2n)$ 0-1 matrix $\mathbf{Q}_{CNF}$ as a generated instance. We generate 100 such instances and we run the **Algorithm1** using Newton's method (7) with the Jacobian (9) where $\mathbf{R}_3 = \mathbf{1}_m$, $\mathbf{R}_1 = \mathbf{Q}_{CNF}$, $\mathbf{R}_2 = [\mathbf{u}; \mathbf{1}_n - \mathbf{u}]$. Also we initialize $\mathbf{u}$ (line 3) appropriately using a random 0-1 vector. Restart is allowed 100 times when failing to find a satisfying assignment. We repeat this process 10 times. Table 2 summarizes the average run time with std to solve all of 100 instances. On average, when $n = 100$, AbdSat runs 7 times slower than MapleBT but when $n = 500$, it runs 40 times faster than MapleBT (though implemented in Octave). Table 2 suggests the potential of our approach to solution finding for random 3-SAT problems.

| $(n,m)$ | AbdSat | MapleBT |
|---|---|---|
| (100,426) | 39.4(13.6) | 5.91(0.17) |
| (500,2130) | 103.3(8.8) | 4231.8(1465.3) |

Table 2: Average run time(s) for solving 100 random 3-SAT instances

## 3.4 Rule refinement for knowledge graph

Here we apply our relation abduction technique to refine rules extracted from a knowledge graph FB15k. FB15k is a well-known knowledge graph containing triples of the form (subject, relation, object) in RDF format for 1,345 binary relations and 14,951 entities [Bordes *et al.*, 2013]. Given two 0-1 matrices $\mathbf{R}_3$ and $\mathbf{R}_1$ representing two known relations $r_1(X,Y)$ and $r_3(X,Z)$ respectively in FB15k, it is possible to find a 0-1 matrix $\mathbf{R}_2$ representing a new relation $r_2(X,Y)$ which makes $r_3(X,Z) \Leftrightarrow \exists Y r_1(X,Y) \wedge r_2(Y,Z)$ *approximately* true in two steps. First compute $\mathbf{X} = (\lambda\mathbf{I} + \mathbf{R}_1^T\mathbf{R}_1)^{-1}(\mathbf{R}_1^T\mathbf{R}_3)$ which minimizes $\|\mathbf{R}_3 - \mathbf{R}_1\mathbf{X}\|_F^2 + \lambda\|\mathbf{X}\|_F^2$ and then put $\mathbf{R}_2 = \mathbf{X}_{>\theta}$ for some $\theta$ that makes $error = |\mathbf{R}_3 - \min_1(\mathbf{R}_1\mathbf{R}_2)|$ minimum. This way of new relation discovery is proposed in [Sato *et al.*, 2018] and shown to give new rules such as

$$\text{language}(X,Z) \Leftarrow \text{genre}(X,Y) \wedge \mathit{genre\_lang}(Y,Z) \qquad (10)$$

which connects two existing relations, language/2 and genre/2 in the film domain of FB15k, by a new abduced relation genre_lang/2. The limitation is that this approach only provides approximation and there is no guarantee of exact abduction. Here we apply our abduction technique to reduce approximation error.

Let $\mathbf{R}_l$, $\mathbf{R}_g$ and $\mathbf{R}_{gl}$ be matrices representing language$(X,Z)$, genre$(X,Y)$ and genre_lang$(Y,Z)$ respectively. Put $\mathbf{A} = \mathbf{R}_l$ and $\mathbf{B} = \min_1(\mathbf{R}_g\mathbf{R}_{gl})$. To measure the quality of extracted rules containing abduced relations, we pretend that $\mathbf{A}$, the l.h.s. in the rule, is predicted by $\mathbf{B}$, the r.h.s. in the rule and measure the quality in terms of F-measure $F(\mathbf{A},\mathbf{B}) = 2|\mathbf{A} \cap \mathbf{B}|/(|\mathbf{A}| + |\mathbf{B}|)$[7] and $error = |\mathbf{A} - \mathbf{B}|$.

Since **Algorithm 1** contains the abduction procedure described in [Sato *et al.*, 2018] as its initial part as lines from (line 3) to (line 6), it is expected to return better abduced relations than those found by [Sato *et al.*, 2018]. We ran **Algorithm 1** with $\mathbf{R}_l$ and $\mathbf{R}_g$ as input using Newton's method and have obtained an abduced relation $\mathbf{R}_{gel}$. We then compared the quality of the rule (10) between the one described in [Sato *et al.*, 2018] and the other containing $\mathbf{R}_{gl}$ returned by **Algorithm 1**. We observed that while the former yields F-measure = 0.654 and $error$ = 1614, the latter rule found by **Algorithm 1** gives F-measure = 0.666 and $error$ = 1498. That is, F-measure increased by 1.2% and $error$ decreased by 7.1%. We observed similar improvement with other rules such as nationality$(X,Z) \Leftarrow$ live_in$(X,Y) \wedge$ *nationality_live_in*$(Y,Z)$ discovered by relation abduction.

# 4 Probabilistic inference

## 4.1 Sampling supported models

In this section, we apply our cost minimization approach to a relatively unexplored area of probabilistic modeling by probabilistic normal logic programs. Our programs look like DB shown in Figure 1[8]. We assume DB consists of unit clauses

[7]We here consider $\mathbf{A}$ as a set $\{(i,j) \mid \mathbf{A}_{ij} = 1\}$ and use $|\mathbf{A}|$ as its cardinality.

[8]This is a variant of 'Friends & Smokers' program from ProbLog's tutorial (`https://dtai.cs.kuleuven.be/problog/tutorial/basic/05_smokers.html`).

labeled with probabilities and non-unit clauses without probability labels. To make matters simple, we also assume predicates are unary or binary and clauses have at most one recursive goal. Hereafter we focus on the 'Friends & Non-smokers' program due to space limitations but generalization is not difficult.

$$
\begin{aligned}
&0.3 :: \mathrm{stress(X)}\\
&0.2 :: \mathrm{influences(X,Y)}\\
&\mathrm{smokes}(X) \Leftarrow \mathrm{stress}(X)\\
&\mathrm{smokes}(X) \Leftarrow \mathrm{friend}(X,Y) \wedge \mathrm{influences}(Y,X) \wedge \neg\mathrm{smokes}(Y)
\end{aligned}
$$

Figure 1: Friends & Non-smokers program

Mathematically, DB is considered as a probabilistic program specifying a probability distribution over its supported models (distribution semantics [Sato, 1995]). Here we define a distribution by a sampling process; first we sample $\mathrm{DB}^g$[9], a ground instantiation of DB, by sampling ground unit clauses with probabilities specified by their labels together with all ground instantiations of non-unit clauses in DB. Then we sample uniformly one of the supported models of $\mathrm{DB}^g$. By sampling repeatedly, we collect a set $S$ of supported models $\mathbf{m}$ and compute various probabilities $P(\mathrm{A})$ of atoms A as a ratio (empirical probability) $|\{\mathbf{m} \mid \mathbf{m} \models \mathrm{A}, \mathbf{m} \in S\}|/|S|$.

The point is that the latter sampling process can be carried out in a vector space. Recall that the equivalence (completion form):

$$
\begin{aligned}
&\mathrm{smokes}(X)\\
&\quad\Leftrightarrow\quad \mathrm{stress(X)} \vee\\
&\qquad\quad \exists Y \mathrm{friend}(X,Y) \wedge \mathrm{influences}(Y,X) \wedge \neg\mathrm{smokes}(Y)
\end{aligned} \tag{11}
$$

holds in any supported model of DB and vice versa[Gelfond and Lifshcitz, 1988; Baral *et al.*, 2009; Eiter *et al.*, 2009]. Suppose there are $n$ people. We rewrite the equivalence (11) to a tensor equation (12) as explained in Section 2 by introducing tensors representing relations in DB, i.e., $(n \times 1)$ vectors $\mathbf{S}_m$ and $\mathbf{S}_t$ for representing unary predicates $\mathrm{smokes}(X)$ and $\mathrm{stress}(X)$ respectively and $(n \times n)$ matrices $\mathbf{F}_r$ and $\mathbf{I}_n$ representing binary predicates $\mathrm{friend}(X,Y)$ and $\mathrm{influences}(X,Y)$ respectively.

$$
\mathbf{S}_m \quad = \quad \min_1(\mathbf{S}_t + (\mathbf{F}_r \odot \mathbf{I}_n^T)(\mathbf{1}_n - \mathbf{S}_m)) \tag{12}
$$

Here $\mathbf{F}_r \odot \mathbf{I}_n^T$ stands for $\mathrm{friend}(X,Y) \wedge \mathrm{influences}(Y,X)$ and $\mathbf{1}_n - \mathbf{S}_m$ for $\neg\mathrm{smokes}(Y)$. Since supported models of (11) and solutions of (12) have a one-to-one correspondence when stress/1, friend/2 and influences/2 are given, we can perform various types of sampling via (12) with the help of a cost function specifically designed for each purpose.

## 4.2 Posterior computation

We conduct a small experiment of posterior computation as a proof of concept following [Nickles, 2018]. We assume there are $n = 4$ people and a friend relation is given by { friend(1,2), friend(2,1), friend(2,4), friend(3,2), friend(4,2) }. Suppose we have observed that smokes(2) = true but influences(4,2) = false. Under this condition, we would like to compute the posterior probability $P(\mathrm{smokes}(1) \mid \mathrm{smokes}(2))$, $P(\mathrm{smokes}(3) \mid \mathrm{smokes}(2))$ and $P(\mathrm{smokes}(4) \mid \mathrm{smokes}(2))$ by *constrained sampling*. So we introduce a cost function $\mathbf{J}^{samp}$:

$$
\begin{aligned}
&\mathbf{J}^{samp}(\mathbf{S}_m)\\
&\quad = \quad \frac{1}{2}\{\|\mathbf{S}_m - \min_1(\mathbf{S}_t + (\mathbf{F}_r \odot \mathbf{I}_n^T)(\mathbf{1}_n - \mathbf{S}_m))\|_F^2\\
&\qquad\qquad + \ell_1 \cdot \|\mathbf{S}_m \odot (\mathbf{1}_n - \mathbf{S}_m)\|_F^2 + \ell_2 \cdot (\mathbf{S}_m(2) - 1)^2\}
\end{aligned} \tag{13}
$$

and perform the sampling of supported models satisfying the constraint $\{\mathrm{smokes}(2) = \mathrm{true}, \mathrm{influences}(4,2) = \mathrm{false}\}$ as follows. First sample a 0-1 $(n \times 1)$ vector $\mathbf{S}_t$ where each element is one with probability 0.3. Similarly sample a 0-1 $(n \times n)$ matrix $\mathbf{I}_n$ using probability 0.2 and put $\mathbf{I}_n(4,2) = 0$. After having sampled $\mathbf{S}_t$ and $\mathbf{I}_n$, sample $\mathbf{S}_m$ s.t. $\mathbf{J}^{samp}(\mathbf{S}_m) = 0$ by minimizing $\mathbf{J}^{samp}$ to zero with random initialization as described in the previous section. Put $\mathbf{C} = \mathbf{S}_t + (\mathbf{F}_r \odot \mathbf{I}_n^T)(\mathbf{1}_n - \mathbf{S}_m)$. The Jacobian $\mathbf{J}_\mathbf{a}^{samp} = \partial\mathbf{J}^{samp}/\partial\mathbf{S}_m$ used to minimize $\mathbf{J}^{samp}$ is given by (derivation omitted)

$$
\begin{aligned}
\mathbf{J}_\mathbf{a}^{samp} \quad &= \quad (\mathbf{E} + \mathrm{diag}(\mathbf{C}_{\leq 1})(\mathbf{F}_r \odot \mathbf{I}_n^T))^T(\mathbf{S}_m - \min_1(\mathbf{C}))\\
&\qquad + \ell_1 \cdot \mathbf{S}_m \odot (\mathbf{1}_n - \mathbf{S}_m) \odot (\mathbf{1}_n - 2\mathbf{S}_m)\\
&\qquad + \ell_2 \cdot (\mathbf{S}_m(2) - 1)\mathbf{I}_2.
\end{aligned} \tag{14}
$$

Here $\mathbf{E}$ is an $n \times n$ identity matrix, $\mathbf{I}_2$ is a zero vector except $\mathbf{I}_2(2) = 1$. $\mathrm{diag}(\mathbf{v})$ is the diagonalization of a vector $\mathbf{v}$ defined by $\mathrm{diag}(\mathbf{v})_{ij} = \begin{cases} \mathbf{v}(i) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$.

Unfortunately sometimes sampling fails or returns a supported model not satisfying $\mathbf{S}_m(2) = 1$. So we try sampling $10^5$ times with $max\_itr = 10$, collect "correct models", i.e. those that satisfy $\mathbf{S}_m = 1$ and compute the target posteriors using sampled correct models. We run this process five times. The result is summarized as follows. On average, sampling is done in 41.2 seconds and we get 99,904 supported models including 38,842 correct ones. Inferred posteriors by the correct sampled models are shown in Table 3 (figures are averages over 5 runs) together with manually computed exact posteriors (derivation omitted). Seeing it, we may say that sampling by cost minimization fairly works well as long as the current example is concerned.

| $P(\cdot \mid \mathrm{smokes}(2))$ | smokes(1) | smokes(3) | smokes(4) |
|---|---|---|---|
| inferred posterior | 0.232 | 0.299 | 0.299 |
| exact posterior | 0.231 | 0.300 | 0.300 |

Table 3: Inferred and exact posteriors

## 4.3 Unconstrained sampling

Here we examine the scalability of our approach using the 'Friends & Non-smokers' program in Figure 1. We simply

[9]We assume $\mathrm{DB}^g$ has supported models.

sample supported models of the program for various $n$ without any constraint. We set $max\_itr = 10$, perform sampling $10^4$ times and plot the execution time for each $n$ up to 250. We also apply a quadratic fit to the plotted data. The result is shown in Figure 2. The fitting curve seems to reflect the fact that the time complexity of computing $\mathbf{J}^{samp}$ and its Jacobian $\mathbf{J}_{\mathbf{a}}^{samp}$ is $O(n^2)$.

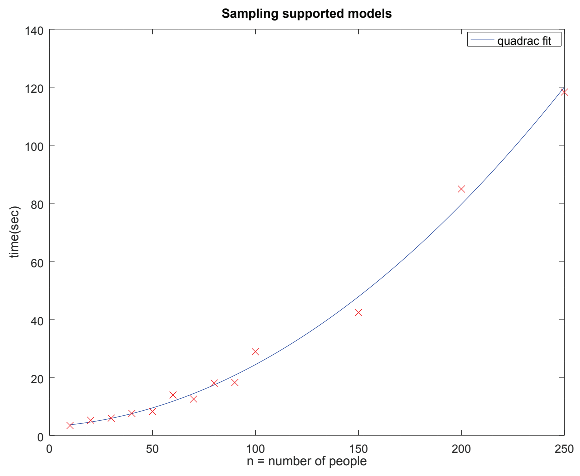

Figure 2: Unconstrained sampling

## 5 Distribution-aware sampling

Now we deal with a more intricate problem of *distribution-aware sampling*[Chakraborty *et al.*, 2014; Nickles, 2018]. By distribution-aware sampling, we mean the one to obtain a set of samples whose empirical distribution best matches the observed distribution of target random variables.

Suppose we have the 'Friends & Non-smokers' program in Figure 1 and observed $P(\text{smokes}(1)) = 0.2$ and $P(\text{smokes}(2)) = 0.8$ in a domain $\{1,\ldots,n\}$. Our task is to sample a set of supported models by distribution-aware sampling whose empirical distribution gives (approximately) these target probabilities. We implement distribution-aware sampling by cost minimization by using a cost function $\mathbf{J}^{dist}(\mathbf{S}_m)$:

$$\mathbf{J}^{dist}(\mathbf{S}_m) = \frac{1}{2}\{\|\mathbf{S}_m - \min_1(\mathbf{C})\|_F^2 + \ell_1 \cdot \|\mathbf{S}_m \odot (\mathbf{1} - \mathbf{S}_m)\|_F^2 + \ell_2 \cdot \|\mathbf{t}_{\text{v\_ep}} - \mathbf{t}_{\text{v\_p}}\|_F^2\}. \quad (15)$$

Here $\mathbf{C} = \mathbf{S}_t + (\mathbf{F}_r \odot \mathbf{I}_n^T)(\mathbf{1}_n - \mathbf{S}_m)$. Let $\mathbf{t}_\text{v}$ be a list of target variables, $\mathbf{t}_{\text{v\_p}}$ their target probabilities and $\mathbf{t}_{\text{v\_ep}}$ an approximation to the empirical distribution by a set $\Delta$ of sampled $\mathbf{S}_m$s. Then $\ell_2 \cdot \|\mathbf{t}_{\text{v\_ep}} - \mathbf{t}_{\text{v\_p}}\|_F^2$ is a penalty term to force $\mathbf{t}_{\text{v\_ep}} \approx \mathbf{t}_{\text{v\_p}}$.

Let $\mathbf{t}_{\text{v\_fq}}$ be a list of frequencies of target variables in $\Delta$. Suppose we are in the process of minimizing $\mathbf{J}^{dist}$ and currently $|\Delta| = N$. By an $N+1$-th sample $\mathbf{S}_m$, $\mathbf{t}_{\text{v\_ep}}$ is updated to $(\max(\min(\mathbf{S}_{m\text{\_v}},1),0) + \mathbf{t}_{\text{v\_fq}})/(N+1)$ where $\mathbf{S}_{m\text{\_v}}$ is a subvector of $\mathbf{S}_m$ for $\mathbf{t}_\text{v}$. Since $\mathbf{t}_{\text{v\_ep}}$ depends on the sampled $\mathbf{S}_m$s, the cost function $\mathbf{J}^{dist}$ *stochastically* changes during its minimization unlike $\mathbf{J}^{samp}$.

We conduct an experiment of distribution-aware sampling for target variables $\mathbf{t}_\text{v} = [\mathbf{S}_m(1)\ \mathbf{S}_m(2)]^T$ and their target probabilities $\mathbf{t}_{\text{v\_p}} = [0.2\ 0.8]^T$ by minimizing $\mathbf{J}^{dist}$ for sampled $\mathbf{S}_t$ and $\mathbf{I}_n$ using **Algorithm 1** with necessary modifications (not included in the paper). Experimental parameters are $n = 100, \ell_1 = 1, \ell_2 = 1000, max\_itr = 300$. We try to sample $\mathbf{S}_m$ maximum 1,000 times until 50 models are sampled (sometimes sampling fails). We then estimate the empirical probability of $\mathbf{t}_\text{v}$ using the sampled $\mathbf{S}_m$s.

We repeated sampling five times and estimated target probabilities as empirical probabilities computed from the sampled $\mathbf{S}_m$s. The result is shown in Table 4 (figures are averages). As seen from it, target probabilities are reasonably estimated, which demonstrates the viability of our proposal, i.e., distribution-aware sampling by cost minimization for probabilistic normal logic programs.

| target var. | $\mathbf{S}_m(1)$ | $\mathbf{S}_m(2)$ |
|---|---|---|
| target prob. | 0.200 | 0.800 |
| estimated prob. | 0.222 | 0.785 |
| #sampled model | 50.0 (all different) | |

Table 4: Distribution-aware sampling

## 6 Related Work

Concerning logic and tensor, Grefenstette reformulates quantifier free first-order logic in tensor spaces[Grefenstette, 2013]. Tensorized first-order logic with full quantification is proposed by Sato in [Sato, 2017]. Partial evaluation of logic programs with matrix encoding is developed by Sakama et al. [Sakama *et al.*, 2018]. Abduction is one of the major categories of symbolic logical inference and has been studied in logic programming in particular [Kakas *et al.*, 1992; Eiter *et al.*, 1997; Gottlob *et al.*, 2010]. A formulation of abduction in vector spaces is proposed by Sato et al. and applied to rule discovery [Sato *et al.*, 2018]. Probabilistic logic programming (PLP) provides high-level probabilistic modeling by first-order logic [Raedt and Kimmig, 2015]. In the context of PLP, our work is most closely related to [Nickles, 2018] where Nickles introduced a differential cost function combined with discrete solution search for SAT/Answer set programming. Our approach seems the first of PLP by supported models via matrix (tensor) equation in vector spaces.

## 7 Conclusion

We have proposed to perform logical inference in vector spaces by minimizing a cost function, combined with thresholding, designed for each task. We demonstrated the effectiveness of our approach by several applications from relation abduction to random SAT, to rule discovery, to constrained sampling and distribution-aware sampling by probabilistic normal logic programs.

## Acknowledgments

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

## References

[Baral *et al.*, 2009] C. Baral, M. Gelfond, and N. Rushton. Probabilistic reasoning with answer sets. *Theory and Practice of Logic Programming (TPLP)*, 9(1):57–144, 2009.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. 2013.

[Chakraborty *et al.*, 2014] Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, and Moshe Y. Vardi. Distribution-aware sampling and weighted model counting for sat. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1722–1730. AAAI Press, 2014.

[Cohen *et al.*, 2017] William W. Cohen, Fan Yang, and Kathryn Mazaitis. Tensorlog: Deep learning meets probabilistic dbs. *CoRR*, abs/1707.05390, 2017.

[Eiter *et al.*, 1997] Thomas Eiter, Georg Gottlob, and Nicola Leone. Abduction from Logic Programs: Semantics and Complexity. *Theoretical Computer Science*, 189(1-2):129–177, 1997.

[Eiter *et al.*, 2009] Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. Reasoning web. semantic technologies for information systems. chapter Answer Set Programming: A Primer, pages 40–110. Springer-Verlag, 2009.

[Gelfond and Lifshcitz, 1988] M. Gelfond and V. Lifshcitz. The stable model semantics for logic programming. pages 1070–1080, 1988.

[Gottlob *et al.*, 2010] Georg Gottlob, Reinhard Pichler, and Fang Wei. Tractable Database Design and Datalog Abduction Through Bounded Treewidth. *Information Systems*, 35(3):278–298, 2010.

[Grefenstette, 2013] Edward Grefenstette. Towards a Formal Distributional Semantics: Simulating Logical Calculi with Tensors. *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, pages 1–10, 2013.

[Heule *et al.*, 2018] M J H Heule, M J Järvisalo, and M (eds) Suda. In *Proceedings of SAT Competition 2018 : Solver and Benchmark Descriptions, Department of Computer Science Series of Publications B, vol. B-2018-1*. Department of Computer Science, University of Helsinki, 2018.

[Kakas *et al.*, 1992] Antonis C. Kakas, Robert Kowalski, and Francesca Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1992.

[Kazemi and Poole, 2018] Seyed Mehran Kazemi and David Poole. Relnn: A deep neural model for relational learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, pages 6367–6375, 2018.

[Manhaeve *et al.*, 2018] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Deepproblog: Neural probabilistic logic programming. *CoRR*, abs/1805.10872, 2018.

[Marek and V.S.Subrahmanian, 1992] W. Marek and V.S.Subrahmanian. The relationship between stable, supported, default and autoepistemic semantics for general logic programs. *Theoretical Computer Science*, 103(2):365–386, 1992.

[Nickles, 2018] Matthias Nickles. Differentiable SAT/ASP. In *Proceedings of the 5th International Workshop on Probabilistic Logic Programming, PLP 2018*, pages 62–74, 2018.

[Raedt and Kimmig, 2015] Luc De Raedt and Angelika Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100(1):5–47, 2015.

[Rocktäschel and Riedel, 2017] Tim Rocktäschel and Sebastian Riedel. End-to-end differentiable proving. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3788–3800. Curran Associates, Inc., 2017.

[Sakama *et al.*, 2018] Chiaki Sakama, Hien Nguyen, Taisuke Sato, and Katsumi Inoue. Partial evaluation of logic programs in vector spaces. In *Proceedings of the 11th Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP 2018)*, pages –, 2018.

[Sato *et al.*, 2018] Taisuke Sato, Katsumi Inoue, and Chiaki Sakama. Abducing relations in continuous spaces. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI-ECAI-18)*, pages 1956–1962, 2018.

[Sato, 1995] T. Sato. A statistical learning method for logic programs with distribution semantics. In *Proceedings of the 12th International Conference on Logic Programming (ICLP'95)*, pages 715–729, 1995.

[Sato, 2017] Taisuke Sato. Embedding Tarskian semantics in vector spaces. In *AAAI-17 Workshop on Symbolic Inference and Optimization (SymInfOpt-17)*, 2017.

[Widdows and Cohen, 2015] Dominic Widdows and Trevor Cohen. Reasoning with Vectors: A Continuous Model for Fast Robust Inference. *Logic journal of the IGPL / Interest Group in Pure and Applied Logics*, 23(2):141–173, 2015.