Logic Tensor Networks: Deep Learning and Logical Reasoning from Data and Knowledge*

Luciano Serafini[1] and Artur d'Avila Garcez[2]

[1] Fondazione Bruno Kessler, Trento, Italy, serafini@fbk.eu
[2] City University London, UK, a.garcez@city.ac.uk

Abstract. We propose *Logic Tensor Networks*: a uniform framework for integrating automatic learning and reasoning. A logic formalism called Real Logic is defined on a first-order language whereby formulas have truth-value in the interval [0,1] and semantics defined concretely on the domain of real numbers. Logical constants are interpreted as feature vectors of real numbers. Real Logic promotes a well-founded integration of deductive reasoning on a knowledge-base and efficient data-driven relational machine learning. We show how Real Logic can be implemented in deep Tensor Neural Networks with the use of Google's TENSORFLOW™ primitives. The paper concludes with experiments applying Logic Tensor Networks on a simple but representative example of knowledge completion.

Keywords: Knowledge Representation, Relational Learning, Tensor Networks, Neural-Symbolic Computation, Data-driven Knowledge Completion.

1 Introduction

The recent availability of large-scale data combining multiple data modalities, such as image, text, audio and sensor data, has opened up various research and commercial opportunities, underpinned by machine learning methods and techniques [5, 12, 17, 18]. In particular, recent work in machine learning has sought to combine logical services, such as knowledge completion, approximate inference, and goal-directed reasoning with data-driven statistical and neural network-based approaches. We argue that there are great possibilities for improving the current state of the art in machine learning and artificial intelligence (AI) thought the principled combination of knowledge representation, reasoning and learning. Guha's recent position paper [15] is a case in point, as it advocates a new model theory for real-valued numbers. In this paper, we take inspiration from such recent work in AI, but also less recent work in the area of neural-symbolic integration [8, 10, 11] and in semantic attachment and symbol grounding [4] to achieve a vector-based representation which can be shown adequate for integrating machine learning and reasoning in a principled way.

* The first author acknowledges the Mobility Program of FBK, for supporting a long term visit at City University London. He also acknowledges NVIDIA Corporation for supporting this research with the donation of a GPU.

This paper proposes a framework called *Logic Tensor Networks (LTN)* which integrates learning based on tensor networks [26] with reasoning using first-order many-valued logic [6], all implemented in TENSORFLOW™ [13]. This enables, for the first time, a range of knowledge-based tasks using rich knowledge representation in first-order logic (FOL) to be combined with efficient data-driven machine learning based on the manipulation of real-valued vectors[1]. Given data available in the form of real-valued vectors, logical soft and hard constraints and relations which apply to certain subsets of the vectors can be specified compactly in first-order logic. Reasoning about such constraints can help improve learning, and learning from new data can revise such constraints thus modifying reasoning. An adequate vector-based representation of the logic, first proposed in this paper, enables the above integration of learning and reasoning, as detailed in what follows.

We are interested in providing a computationally adequate approach to implementing learning and reasoning [28] in an integrated way within an idealized agent. This agent has to manage knowledge about an unbounded, possibly infinite, set of objects $O = \{o_1, o_2, \dots\}$. Some of the objects are associated with a set of quantitative attributes, represented by an n-tuple of real values $\mathcal{G}(o_i) \in \mathbb{R}^n$, which we call *grounding*. For example, a person may have a grounding into a 4-tuple containing some numerical representation of the person's name, her height, weight, and number of friends in some social network. Object tuples can participate in a set of relations $\mathcal{R} = \{R_1, \dots, R_k\}$, with $R_i \subseteq O^{\alpha(R_i)}$, where $\alpha(R_i)$ denotes the arity of relation R_i. We presuppose the existence of a latent (unknown) relation between the above numerical properties, i.e. groundings, and partial relational structure $\mathcal{R}$ on O. Starting from this partial knowledge, an agent is required to: (i) infer new knowledge about the relational structure on the objects of O; (ii) predict the numerical properties or the class of the objects in O.

Classes and relations are not normally independent. For example, it may be the case that if an object x is of class C, $C(x)$, and it is related to another object y through relation $R(x, y)$ then this other object y should be in the same class $C(y)$. In logic: $\forall x \exists y((C(x) \wedge R(x, y)) \rightarrow C(y))$. Whether or not $C(y)$ holds will depend on the application: through reasoning, one may derive $C(y)$ where otherwise there might not have been evidence of $C(y)$ from training examples only; through learning, one may need to revise such a conclusion once examples to the contrary become available. The vectorial representation proposed in this paper permits both reasoning and learning as exemplified above and detailed in the next section.

The above forms of reasoning and learning are integrated in a unifying framework, implemented within tensor networks, and exemplified in relational domains combining data and relational knowledge about the objects. It is expected that, through an adequate integration of numerical properties and relational knowledge, differently from the immediate related literature [9, 2, 1], the framework introduced in this paper will be capable of combining in an effective way first-order logical inference on open domains with efficient relational multi-class learning using tensor networks.

The main contribution of this paper is two-fold. It introduces a novel framework for the integration of learning and reasoning which can take advantage of the repre-

[1] In practice, FOL reasoning including function symbols is approximated through the usual iterative deepening of clause depth.

sentational power of (multi-valued) first-order logic, and it instantiates the framework using tensor networks into an efficient implementation which shows that the proposed vector-based representation of the logic offers an adequate mapping between symbols and their real-world manifestations, which is appropriate for both rich inference and learning from examples.

The paper is organized as follows. In Section 2, we define Real Logic. In Section 3, we propose the Learning-as-Inference framework. In Section 4, we instantiate the framework by showing how Real Logic can be implemented in deep Tensor Neural Networks leading to Logic Tensor Networks (LTN). Section 5 contains an example of how LTN handles knowledge completion using (possibly inconsistent) data and knowledge from the well-known *smokers and friends* experiment. Section 6 concludes the paper and discusses directions for future work.

2 Real Logic

We start from a first order language $\mathcal{L}$, whose signature contains a set $\mathcal{C}$ of constant symbols, a set $\mathcal{F}$ of functional symbols, and a set $\mathcal{P}$ of predicate symbols. The sentences of $\mathcal{L}$ are used to express relational knowledge, e.g. the atomic formula $R(o_1, o_2)$ states that objects o_1 and o_2 are related to each other through binary relation R; $\forall xy.(R(x, y) \rightarrow R(y, x))$ states that R is a symmetric relation, where x and y are variables; $\exists y.R(o_1, y)$ states that there is an (unknown) object which is related to object o_1 through R. For simplicity, without loss of generality, we assume that all logical sentences of $\mathcal{L}$ are in prenex conjunctive, skolemised normal form [16], e.g. a sentence $\forall x(A(x) \rightarrow \exists y R(x, y))$ is transformed into an equivalent clause $\neg A(x) \vee R(x, f(x))$, where f is a new function symbol.

As for the semantics of $\mathcal{L}$, we deviate from the standard abstract semantics of FOL, and we propose a *concrete* semantics with sentences interpreted as tuples of real numbers. To emphasise the fact that $\mathcal{L}$ is interpreted in a "real" world, we use the term *(semantic) grounding*, denoted by $\mathcal{G}$, instead of the more standard *interpretation*[2].

- $\mathcal{G}$ associates an n-tuple of real numbers $\mathcal{G}(t)$ to any closed term t of $\mathcal{L}$; intuitively $\mathcal{G}(t)$ is the set of numeric features of the object denoted by t.
- $\mathcal{G}$ associates a real number in the interval $[0, 1]$ to each clause ϕ of $\mathcal{L}$. Intuitively, $\mathcal{G}(\phi)$ represents one's confidence in the truth of ϕ; the higher the value, the higher the confidence.

A grounding is specified only for the elements of the signature of $\mathcal{L}$. The grounding of terms and clauses is defined inductively, as follows.

Definition 1. *A* grounding $\mathcal{G}$ *for a first order language* $\mathcal{L}$ *is a function from the signature of* $\mathcal{L}$ *to the real numbers that satisfies the following conditions:*

1. $\mathcal{G}(c) \in \mathbb{R}^n$ *for every constant symbol* $c \in \mathcal{C}$*;*
2. $\mathcal{G}(f) \in \mathbb{R}^{n \cdot \alpha(f)} \longrightarrow \mathbb{R}^n$ *for every* $f \in \mathcal{F}$*;*

[2] In logic, the term "grounding" indicates the operation of replacing the variables of a term/formula with constants. To avoid confusion, we use the term "instantiation" for this.

3. $\mathcal{G}(P) \in \mathbb{R}^{n \cdot \alpha(R)} \longrightarrow [0,1]$ *for every* $P \in \mathcal{P}$*;*

A grounding $\mathcal{G}$ is inductively extended to all the closed terms and clauses, as follows:

$$\begin{aligned}
\mathcal{G}(f(t_1, \ldots, t_m)) &= \mathcal{G}(f)(\mathcal{G}(t_1), \ldots, \mathcal{G}(t_m)) \\
\mathcal{G}(P(t_1, \ldots, t_m)) &= \mathcal{G}(P)(\mathcal{G}(t_1), \ldots, \mathcal{G}(t_m)) \\
\mathcal{G}(\neg P(t_1, \ldots, t_m)) &= 1 - \mathcal{G}(P(t_1, \ldots, t_m)) \\
\mathcal{G}(\phi_1 \vee \cdots \vee \phi_k) &= \mu(\mathcal{G}(\phi_1), \ldots, \mathcal{G}(\phi_k))
\end{aligned}$$

where μ is an s-norm operator, also known as a t-co-norm operator (i.e. the dual of some t-norm operator). [3]

Example 1. Suppose that $O = \{o_1, o_2, o_3\}$ is a set of documents defined on a finite dictionary $D = \{w_1, ..., w_n\}$ of n words. Let $\mathcal{L}$ be the language that contains the binary function symbol $concat(x, y)$ denoting the document resulting from the concatenation of documents x with y. Let $\mathcal{L}$ contain also the binary predicate Sim which is supposed to be *true* if document x is deemed to be similar to document y. An example of grounding is the one that associates to each document its bag-of-words vector [7]. As a consequence, a natural grounding of the *concat* function would be the sum of the vectors, and of the *Sim* predicate, the cosine similarity between the vectors. More formally:

- $\mathcal{G}(o_i) = \langle n^{o_i}_{w_1}, \ldots, n^{o_i}_{w_n} \rangle$, where n^d_w is the number of occurrences of word w in document d;
- if $\mathbf{v}, \mathbf{u} \in \mathbb{R}^n$, $\mathcal{G}(concat)(\mathbf{u}, \mathbf{v}) = \mathbf{u} + \mathbf{v}$;
- if $\mathbf{v}, \mathbf{u} \in \mathbb{R}^n$, $\mathcal{G}(Sim)(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| ||\mathbf{v}||}$.

For instance, if the three documents are o_1 = *"John studies logic and plays football"*, o_2 = *"Mary plays football and logic games"*, o_3 = *"John and Mary play football and study logic together"*, and W=*{John, Mary, and, football, game, logic, play, study, together}* then the following are examples of the grounding of terms, atomic formulas and clauses.

$$\begin{aligned}
\mathcal{G}(o_1) &= \langle 1, 0, 1, 1, 0, 1, 1, 1, 0 \rangle \\
\mathcal{G}(o_2) &= \langle 0, 1, 1, 1, 1, 1, 1, 0, 0 \rangle \\
\mathcal{G}(o_3) &= \langle 1, 1, 2, 1, 0, 1, 1, 1, 1 \rangle \\
\mathcal{G}(concat(o_1, o_2)) &= \mathcal{G}(o_1) + \mathcal{G}(o_2) = \langle 1, 1, 2, 2, 1, 2, 2, 1, 0 \rangle \\
\mathcal{G}(Sim(concat(o_1, o_2), o_3) &= \frac{\mathcal{G}(concat(o_1, o_2)) \cdot \mathcal{G}(o_3)}{||\mathcal{G}(concat(o_1, o_2))|| \cdot ||\mathcal{G}(o_3)||} \approx \frac{13}{14.83} \approx 0.88 \\
\mathcal{G}(Sim(o_1, o_3) \vee Sim(o_2, o_3)) &= \mu_{max}(\mathcal{G}(Sim(o_1, o_3), \mathcal{G}(Sim(o_2, o_3)) \\
&\approx \max(0.86, 0.73) = 0.86
\end{aligned}$$

[3] Examples of t-norms which can be chosen here are Lukasiewicz, product, and Gödel. Lukasiewicz s-norm is defined as $\mu_{Luk}(x, y) = \min(x + y, 1)$; Product s-norm is defined as $\mu_{Pr}(x, y) = x + y - x \cdot y$; Gödel s-norm is defined as $\mu_{max}(x, y) = \max(x, y)$.

3 Learning as approximate satisfiability

We start by defining ground theory and their satisfiability.

Definition 2 (Satisfiability). *Let ϕ be a closed clause in $\mathcal{L}$, $\mathcal{G}$ a grounding, and $v \leq w \in [0,1]$. We say that $\mathcal{G}$ satisfies ϕ in the confidence interval $[v,w]$, written $\mathcal{G} \models_v^w \phi$, if $v \leq \mathcal{G}(\phi) \leq w$.*

A partial grounding, denoted by $\hat{\mathcal{G}}$, is a grounding that is defined on a subset of the signature of $\mathcal{L}$. A grounded theory is a set of clauses in the language of $\mathcal{L}$ and partial grounding $\hat{\mathcal{G}}$.

Definition 3 (Grounded Theory). *A* grounded theory (GT) *is a pair $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ where $\mathcal{K}$ is a set of pairs $\langle [v,w], \phi(\mathbf{x}) \rangle$, where $\phi(\mathbf{x})$ is a clause of $\mathcal{L}$ containing the set $\mathbf{x}$ of free variables, and $[v,w] \subseteq [0,1]$ is an interval contained in $[0,1]$, and $\hat{\mathcal{G}}$ is a partial grounding.*

Definition 4 (Satisfiability of a Grounded Theory). *A GT $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ is satisfiabile if there exists a grounding $\mathcal{G}$, which extends $\hat{\mathcal{G}}$ such that for all $\langle [v,w], \phi(\mathbf{x}) \rangle \in \mathcal{K}$ and any tuple $\mathbf{t}$ of closed terms, $\mathcal{G} \models_v^w \phi(\mathbf{t})$.*

From the previous definiiton it follows that checking if a GT $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ is satisfiable amounts to seaching for an extension of the partial grounding $\hat{\mathcal{G}}$ in the space of *all possible groundings*, such that *all* the instantiations of the clauses in $\mathcal{K}$ are satisfied w.r.t. the specified interval. Clearly this is unfeasible from a practical point of view. As is usual, we must restrict both the space of grounding and clause instantiations. Let us consider each in turn: To check satisfiability on a subset of all the functions on real numbers, recall that a grounding should capture a latent correlation between the quantitative attributes of an object and its relational properties[4]. In particular, we are interested in searching within a specific class of functions, in this paper based on tensor networks, although other family of functions can be considered. To limit the number of clause instantiations, which in general might be infinite since $\mathcal{L}$ admits function symbols, the usual approach is to consider the instantiations of each clause up to a certain depth [3].

When a grounded theory $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ is inconsitent, that is, there is no grounding $\mathcal{G}$ that satisfies it, we are interested in finding a grounding which satisfies *as much as possible* of $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$. For any $\langle [v,w], \phi \rangle \in \mathcal{K}$ we want to find a grounding $\mathcal{G}$ that minimizes the *satisfiability error*. An error occurs when a grounding $\mathcal{G}$ assigns a value $\mathcal{G}(\phi)$ to a clause ϕ which is outside the interval $[v,w]$ prescribed by $\mathcal{K}$. The measure of this error can be defined as the minimal distance between the points in the interval $[v,w]$ and $\mathcal{G}(\phi)$:

$$\text{Loss}(\mathcal{G}, \langle [v,w], \phi \rangle) = |x - \mathcal{G}(\phi)|, v \leq x \leq w \tag{1}$$

[4] For example, whether a document is classified as from the field of Artificial Intelligence (AI) depends on its bag-of-words grounding. If the language $\mathcal{L}$ contains the unary predicate $AI(x)$ standing for "x is a paper about AI" then the grounding of $AI(x)$, which is a function from bag-of-words vectors to [0,1], should assign values close to 1 to the vectors which are close semantically to AI. Furthermore, if two vectors are similar (e.g. according to the cosine similarity measure) then their grounding should be similar.

Notice that if $\mathcal{G}(\phi) \in [v, w]$, $\text{Loss}(\mathcal{G}, \phi) = 0$.

The above gives rise to the following definition of approximate satisfiability w.r.t. a family $\mathbb{G}$ of grounding functions on the language $\mathcal{L}$.

Definition 5 (Approximate satisfiability). *Let $\langle \mathcal{K}, \hat{\mathcal{G}} \rangle$ be a grounded theory and $\mathcal{K}_0$ a finite subset of the instantiations of the clauses in $\mathcal{K}$, i.e.*

$$K_0 \subseteq \{\langle [v, w], \phi(\mathbf{t}) \rangle\} \mid \langle [v, w], \phi(\mathbf{x}) \rangle \in \mathcal{K} \text{ and } \mathbf{t} \text{ is any } n\text{-tuple of closed terms.}\}$$

Let $\mathbb{G}$ be a family of grounding functions. We define the best satisfiability problem as the problem of finding an extensions $\mathcal{G}^*$ of $\hat{\mathcal{G}}$ in $\mathbb{G}$ that minimizes the satisfiability error on the set $\mathcal{K}_0$, that is:

$$\mathcal{G}^* = \underset{\hat{\mathcal{G}} \subseteq \mathcal{G} \in \mathbb{G}}{\text{argmin}} \sum_{\langle [v,w], \phi(\mathbf{t}) \rangle \in \mathcal{K}_0} \mathit{Loss}(\mathcal{G}, \langle [v, w], \phi(\mathbf{t}) \rangle)$$

4 Implementing Real Logic in Tensor Networks

Specific instances of Real Logic can be obtained by selectiong the space $\mathbb{G}$ of groundings and the specific s-norm for the interpretation of disjunction. In this section, we describe a realization of real logic where $\mathbb{G}$ is the space of real tensor transformations of order k (where k is a parameter). In this space, function symbols are interpreted as linear transformations. More precisely, if f is a function symbol of arity m and $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^n$ are real vectors corresponding to the grounding of m terms then $\mathcal{G}(f)(\mathbf{v}_1, \ldots, \mathbf{v}_m)$ can be written as:

$$\mathcal{G}(f)(\mathbf{v}_1, \ldots, \mathbf{v}_m) = M_f \mathbf{v} + N_f$$

for some $n \times mn$ matrix M_f and n-vector N_f, where $\mathbf{v} = \langle \mathbf{v}_1, \ldots, \mathbf{v}_n \rangle$.

The grounding of m-ary predicate P, $\mathcal{G}(P)$, is defined as a generalization of the neural tensor network [26] (which has been shown effective at knowledge compilation in the presence of simple logical constraints), as a function from $\mathbb{R}^{mn}$ to $[0, 1]$, as follows:

$$\mathcal{G}(P) = \sigma\left(u_P^T \tanh\left(\mathbf{v}^T W_P^{[1:k]} \mathbf{v} + V_P \mathbf{v} + B_P\right)\right) \tag{2}$$

where $W_P^{[1:k]}$ is a 3-D tensor in $\mathbb{R}^{mn \times mn \times k}$, V_P is a matrix in $\mathbb{R}^{k \times mn}$, and B_P is a vector in $\mathbb{R}^k$, and σ is the sigmoid function. With this encoding, the grounding (i.e. truth-value) of a clause can be determined by a neural network which first computes the grounding of the literals contained in the clause, and then combines them using the specific s-norm. An example of tensor network for $\neg P(x, y) \rightarrow A(y)$ is shown in Figure 1. This architecture is a generalization of the structure proposed in [26], that has been shown rather effective for the task of knowledge compilation, also in presence of simple logical constraints. In the above tensor network formulation, W_*, V_*, B_* and u_* with $* \in \{P, A\}$ are parameters to be learned by minimizing the loss function or, equivalently, to maximize the satisfiability of the clause $P(x, y) \rightarrow A(y)$.

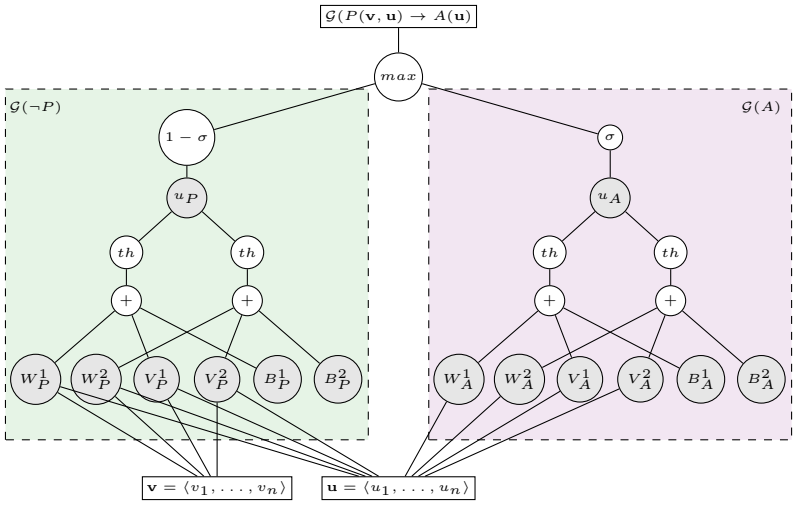

Fig. 1. Tensor net for $P(x, y) \rightarrow A(y)$, with $\mathcal{G}(x) = \mathbf{v}$ and $\mathcal{G}(y) = \mathbf{u}$ and $k = 2$.

5 An Example of Knowledge Completion

Logic Tensor Networks have been implemented as a Python library called `ltn` using Google's TENSORFLOW™ . To test our idea, in this section we use the well-known *friends and smokers*[5] example [24] to illustrate the task of knowledge completion in `ltn`. There are 14 people divided into two groups $\{a, b, \ldots, h\}$ and $\{i, j, \ldots, n\}$. Within each group of people we have complete knowledge of their smoking habits. In the first group, we have complete knowledge of who has and does not have cancer. In the second group, this is not known for any of the persons. Knowledge about the friendship relation is complete within each group only if symmetry of friendship is assumed. Otherwise, it is imcomplete in that it may be known that, e.g., a is a friend of b, but not known whether b is a friend of a. Finally, there is also general knowledge about smoking, friendship and cancer, namely, that smoking causes cancer, friendship is normally a symmetric and anti-reflexive relation, everyone has a friend, and that smoking propagates (either actively or passively) among friends. All this knowledge can be represented by the knowledge-bases shown in Figure 2.

The facts contained in the knowledge-bases should have different degrees of truth, and this is not known. Otherwise, the combined knowledge-base would be inconsistent (it would deduce e.g. $S(b)$ and $\neg S(b)$). Our main task is to complete the knowledge-base (KB), that is: (i) find the degree of truth of the facts contained in KB, (ii) find a truth-value for all the missing facts, e.g. $C(i)$, (iii) find the grounding of each constant symbol $a, ..., n$.[6] To answer (i)-(iii), we use `ltn` to find a grounding that best

[5] Normally, a probabilistic approach is taken to solve this problem, and one that requires instantiating all clauses to remove variables, essentially turning the problem into a propositional one; `ltn` takes a different approach.

[6] Notice how no grounding is provided about the signature of the knowledge-base.

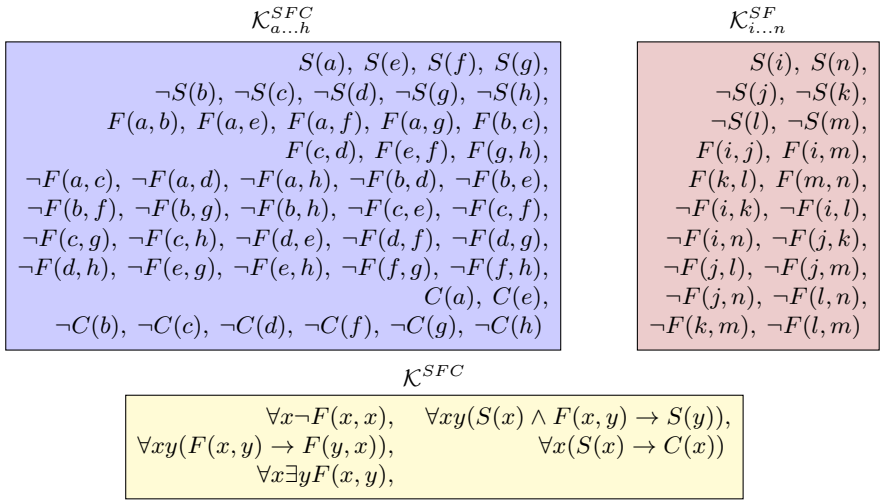

Fig. 2. Knowledge-bases for the friends-and-smokers example.

approximates the complete KB. We start by assuming that all the facts contained in the knowledge-base are true (i.e. have degree of truth 1). To show the role of background knolwedge in the learning-inference process, we run two experiments. In the first ($exp1$), we seek to complete a KB consisting of only factual knowledge: $\mathcal{K}_{exp1} = \mathcal{K}^{SFC}_{a\ldots h} \cup \mathcal{K}^{SF}_{i\ldots n}$. In the second ($exp1$), we also include background knowledge, that is: $\mathcal{K}_{exp2} = \mathcal{K}_{exp1} \cup \mathcal{K}^{SFC}$.

We confgure the network as follows: each constant (i.e. person) can have up to 30 real-valued features. We set the number of layers k in the tensor network to 10, and the regularization parameter[7] $\lambda = 1^{-10}$. For the purpose of illustration, we use the Lukasiewicz t-norm with s-norm $\mu(a,b) = \min(1, a+b)$, and use the harmonic mean as aggregation operator. An estimation of the optimal grounding is obtained after 5,000 runs of the RMSProp learning algorithm [27] available in TensorFlow™ .

The results of the two experiments are reported in Table 1. For readability, we use boldface for truth-values greater than 0.5. The truth-values of the facts listed in a knowledge-base are highlighted with the same background color of the knowledge-base in Figure 2. The values with white background are the result of the knowledge completion produced by the LTN learning-inference procedure. To evaluate the quality of the results, one has to check whether (i) the truth-values of the facts listed in a KB are indeed close to 1.0, and (ii) the truth-values associated with knowledge completion correspond to expectation. An initial analysis shows that the LTN associated with $\mathcal{K}_{exp1}$ produces the same facts as $\mathcal{K}_{exp1}$ itself. In other words, the LTN fits the data. However, the LTN also learns to infer additional positive and negative facts about F and C not derivable from $\mathcal{K}_{exp1}$ by pure logical reasoning; for example: $F(c,b)$, $F(g,b)$ and $\neg F(b,a)$. These facts are derived by exploiting similarities between the groundings of

[7] A smoothing factor $\lambda||\mathbf{\Omega}||_2^2$ is added to the loss function to create a preference for learned parameters with a lower absolute value.

	S	C	F							
			a	b	c	d	e	f	g	h
a	**1.00**	**1.00**	0.00	**1.00**	0.00	0.00	**1.00**	**1.00**	**1.00**	0.00
b	0.00	0.00	0.00	0.00	**1.00**	0.00	0.00	0.00	0.00	0.00
c	0.00	0.00	0.00	**0.82**	0.00	**1.00**	0.00	0.00	0.00	0.00
d	0.00	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.00	0.00
e	**1.00**	**1.00**	0.00	0.33	0.21	0.00	0.00	**1.00**	0.00	0.00
f	**1.00**	0.00	0.00	0.00	0.05	0.00	0.00	0.00	0.00	0.00
g	**1.00**	0.00	0.03	**1.00**	**1.00**	**1.00**	0.11	**1.00**	0.00	**1.00**
h	0.00	0.00	0.00	0.23	0.01	0.14	0.00	0.02	0.00	0.00

	S	C	F					
			i	j	k	l	m	n
i	**1.00**	0.00	0.00	**1.00**	0.00	0.00	**1.00**	0.00
j	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
k	0.00	0.00	0.10	**1.00**	0.00	**1.00**	0.00	0.00
l	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.00
m	0.00	0.03	**1.00**	**1.00**	0.12	**1.00**	0.00	**1.00**
n	**1.00**	0.01	0.00	0.98	0.00	0.01	0.02	0.00

Learning and reasoning on $\mathcal{K}_{exp1} = \mathcal{K}^{SFC}_{a...h} \cup \mathcal{K}^{SF}_{i...n}$

	S	C	F							
			a	b	c	d	e	f	g	h
a	**0.84**	**0.87**	0.02	**0.95**	0.01	0.03	**0.93**	**0.97**	**0.98**	0.01
b	0.13	0.16	0.45	0.01	**0.97**	0.04	0.02	0.03	0.06	0.03
c	0.13	0.15	0.02	**0.94**	0.11	**0.99**	0.03	0.16	0.15	0.15
d	0.14	0.15	0.01	0.06	**0.88**	0.08	0.01	0.03	0.07	0.02
e	**0.84**	**0.85**	0.32	0.06	0.05	0.03	0.04	**0.97**	0.07	0.06
f	**0.81**	0.19	0.34	0.11	0.08	0.04	0.42	0.08	0.06	0.05
g	**0.82**	0.19	**0.81**	0.26	0.19	0.30	0.06	0.28	0.00	**0.94**
h	0.14	0.17	0.05	0.25	0.26	0.16	0.20	0.14	**0.72**	0.01

	S	C	F							
			i	j	k	l	m	n		
i	**0.83**	**0.86**	0.02	**0.91**	0.01	0.03	**0.97**	0.01		
j	0.19	0.22	**0.73**	0.03	0.00	0.04	0.02	0.05		
k	0.14	0.34	0.17	0.07	0.04	**0.97**	0.04	0.02		
l	0.16	0.19	0.11	0.12	0.15	0.06	0.05	0.03		
m	0.14	0.17	**0.96**	0.07	0.02	0.11	0.00	**0.92**		
n	**0.84**	**0.86**	0.13	0.28	0.01	0.24	**0.69**	0.02		

	$a, \ldots, h, i, \ldots, n$
$\forall x \neg F(x, x)$	**0.98**
$\forall xy(F(x, y) \rightarrow F(y, x))$	**0.90** **0.90**
$\forall x(S(x) \rightarrow C(x))$	**0.77**
$\forall x(S(x) \wedge F(x, y) \rightarrow S(y))$	**0.96** **0.92**
$\forall x \exists y(F(x, y))$	**1.0**

Learning and reasoning on $\mathcal{K}_{exp2} = \mathcal{K}^{SFC}_{a...h} \cup \mathcal{K}^{SF}_{i...n} \cup \mathcal{K}^{SFC}$

Table 1.

the constants generated by the LTN. For instance, $\mathcal{G}(c)$ and $\mathcal{G}(g)$ happen to present a high cosine similarity measure. As a result, facts about the friendship relations of c affect the friendship relations of g and vice-versa, for instance $F(c, b)$ and $F(g, b)$. The level of satisfiability associated with $\mathcal{K}_{exp1} \approx 1$, which indicates that $\mathcal{K}_{exp1}$ is classically satisfiable.

The results of the second experiment show that more facts can be learned with the inclusion of background knowledge. For example, the LTN now predicts that $C(i)$ and $C(n)$ are true. Similarly, from the symmetry of the friendship relation, the LTN concludes that m is a friend of i, as expected. In fact, all the axioms in the generic background knowledge $\mathcal{K}^{SFC}$ are satisfied with a degree of satisfiability higher than 90%, apart from the *smoking causes cancer* axiom - which is responsible for the classical inconsistency since in the data f and g smoke and do not have cancer -, which has a degree of satisfiability of 77%.

6 Related work

In his recent note, [15], Guha advocates the need for a new model theory for distributed representations (such as those based on embeddings). The note sketches a proposal, where terms and (binary) predicates are all interpreted as points/vectors in an n-dimensional real space. The computation of the truth-value of the atomic formulae $P(t_1, \ldots, t_n)$ is obtained by comparing the projections of the vector associated to each

t_i with that associated to P_i. Real logic shares with [15] the idea that terms must be interpreted in a geometric space. It has, however, a different (and more general) interpretation of functions and predicate symbols. Real logic is more general because the semantics proposed in [15] can be implemented within an `ltn` with a single layer ($k = 1$), since the operation of projection and comparison necessary to compute the truth-value of $P(t_1, \ldots, t_m)$ can be encoded within an $nm \times nm$ matrix W with the constraint that $\langle \mathcal{G}(t_1), \ldots, \mathcal{G}(t_n) \rangle^T W \langle \mathcal{G}(t_1), \ldots, \mathcal{G}(t_n) \rangle \leq \delta$, which can be encoded easily in `ltn`.

Real logic is orthogonal to the approach taken by (Hybrid) Markov Logic Networks (MLNs) and its variations [24, 29, 22]. In MLNs, the level of truth of a formula is determined by the number of models that satisfy the formula: the more models, the higher the degree of truth. Hybrid MLNs introduce a dependency from the real features associated to constants, which is given, and not learned. In real logic, instead, the level of truth of a complex formula is determined by (fuzzy) logical reasoning, and the relations between the features of different objects is learned through error minimization. Another difference is that MLNs work under the *closed world assumption*, while Real Logic is open domain. Much work has been done also on neuro-fuzzy approaches [19]. These are essentially propositional while real logic is first-order.

Bayesian logic (BLOG) [20] is open domain, and in this respect similar to real logic and LTNs. But, instead of taking an explicit probabilistic approach, LTNs draw from the efficient approach used by tensor networks for knowledge graphs, as already discussed. LTNs can have a probabilistic interpretation but this is not a requirement. Other statistical AI and probabilistic approaches such as lifted inference fall into this category, including probabilistic variations of inductive logic programming (ILP) [23], which are normally restricted to Horn clauses. Metainterpretive ILP [21], together with BLOG, seem closer to LTNs in what concerns the knowledge representation language, but do not explore the benefits of tensor networks for computational efficiency.

An approach for embedding logical knowledge onto data for the purpose of relational learning, similar to Real Logic, is presented in [25]. Real Logic and [25] share the idea of interpreting a logical alphabet in an n-dimensional real space. Terminologically, the term "grounding" in Real Logic corresponds to "embeddings" in [25]. However, there are several differences. First, [25] uses function-free langauges, while we provide also groundings for functional symbols. Second, the model used to compute the truth-values of atomic formulas adopted in [25] is a special case of the more general model proposed in this paper (as described in Eq. (2)). Finally, the semantics of the universal and existential quantifiers adopted in [25] is based on the closed-world assumption (CWA), i.e. universally (respectively, existentially) quantified formulas are reduced to the finite conjunctions (respectively, disjunctions) of all of their possible instantiations; Real Logic does not make the CWA. Furthermore, Real Logic does not assume a specific t-norm.

As in [11], LTN is a framework for learning in the presence of logical constraints. LTNs share with [11] the idea that logical constraints and training examples can be treated uniformly as supervisions of a learning algorithm. LTN introduces two novelties: first, in LTN existential quantifiers are not grounded into a finite disjunction, but are *scolemized*. In other words, CWA is not required, and existentially quantified formu-

las can be satisfied by "new individuals". Second, LTN allows one to generate data for prediction. For instance, if a grounded theory contains the formula $\forall x \exists y R(x, y)$, LTN generates a real function (corresponding to the grounding of the Skolem function introduced by the formula) which for every vector $\mathbf{v}$ returns the feature vector $f(\mathbf{v})$, which can be intuitively interpreted as being the set of features of a *typical* object which takes part in relation R with the object having features equal to $\mathbf{v}$.

Finally, related work in the domain of neural-symbolic computing and neural network fibring [10] has sought to combine neural networks with ILP to gain efficiency [14] and other forms of knowledge representation, such as propositional modal logic and logic programming. The above are more tightly-coupled approaches. In contrast, LTNs use a richer FOL language, exploit the benefits of knowledge compilation and tensor networks within a more loosely- coupled approach, and might even offer an adequate representation of equality in logic. Experimental evaluations and comparison with other neural-symbolic approaches are desirable though, including the latest developments in the field, a good snapshot of which can be found in [1].

7 Conclusion and future work

We have proposed *Real Logic*: a uniform framework for learning and reasoning. Approximate satisfiability is defined as a learning task with both knowledge and data being mapped onto real-valued vectors. With an inference-as-learning approach, relational knowledge constraints and state-of-the-art data-driven approaches can be integrated. We showed how real logic can be implemented in deep tensor networks, which we call Logic Tensor Networks (LTNs), and applied efficiently to knowledge completion and data prediction tasks. As future work, we will make the implementation of LTN available in TENSORFLOW™ and apply it to large-scale experiments and relational learning benchmarks for comparison with statistical relational learning, neural-symbolic computing, and (probabilistic) inductive logic programming approaches.

References

1. *Cognitive Computation: Integrating Neural and Symbolic Approaches*, Workshop at NIPS 2015, Montreal, Canada, April 2016. CEUR-WS 1583.
2. *Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches*, AAAI Spring Symposium, Stanford University, CA, USA, March 2015.
3. Dimitris Achlioptas. Random satisfiability. In *Handbook of Satisfiability*, pages 245–270. 2009.
4. Leon Barrett, Jerome Feldman, and Liam MacDermed. A (somewhat) new solution to the variable binding problem. *Neural Computation*, 20(9):2361–2378, 2008.
5. Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.
6. M. Bergmann. *An Introduction to Many-Valued and Fuzzy Logic: Semantics, Algebras, and Derivation Systems*. Cambridge University Press, 2008.
7. David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003.
8. Léon Bottou. From machine learning to machine reasoning. Technical report, arXiv.1102.1808, February 2011.
9. Artur S. d'Avila Garcez, Marco Gori, Pascal Hitzler, and Luís C. Lamb. Neural-symbolic learning and reasoning (dagstuhl seminar 14381). *Dagstuhl Reports*, 4(9):50–84, 2014.

10. Artur S. d'Avila Garcez, Luís C. Lamb, and Dov M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Cognitive Technologies. Springer, 2009.
11. Michelangelo Diligenti, Marco Gori, Marco Maggini, and Leonardo Rigutini. Bridging logic and kernel machines. *Machine Learning*, 86(1):57–88, 2012.
12. David Silver et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.
13. Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
14. Manoel V. M. França, Gerson Zaverucha, and Artur S. d'Avila Garcez. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine Learning*, 94(1):81–104, 2014.
15. Ramanathan Guha. Towards a model theory for distributed representations. In *2015 AAAI Spring Symposium Series*, 2015.
16. Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning About Systems*. Cambridge University Press, New York, NY, USA, 2004.
17. Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, January 2003.
18. Douwe Kiela and Léon Bottou. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of EMNLP 2014*, Doha, Qatar, 2014.
19. Bart Kosko. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.
20. Brian Milch, Bhaskara Marthi, Stuart J. Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: probabilistic models with unknown objects. In *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005*, pages 1352–1359, 2005.
21. Stephen H. Muggleton, Dianhuan Lin, and Alireza Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. *Machine Learning*, 100(1):49–73, 2015.
22. Aniruddh Nath and Pedro M. Domingos. Learning relational sum-product networks. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2878–2886, 2015.
23. Luc De Raedt, Kristian Kersting, Sriraam Natarajan, and David Poole. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2016.
24. Matthew Richardson and Pedro Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, February 2006.
25. Tim Rocktaschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, June 2015.
26. Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*. 2013.
27. T. Tieleman and G. Hinton. Lecture 6.5 - RMSProp, COURSERA: Neural networks for machine learning. Technical report, 2012.
28. Leslie G. Valiant. Robust logics. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, STOC '99, pages 642–651, New York, NY, USA, 1999. ACM.
29. Jue Wang and Pedro M. Domingos. Hybrid markov logic networks. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1106–1111, 2008.