Applied Intelligence (2023) 53:18775–18789
https://doi.org/10.1007/s10489-022-04383-6

T-norms driven loss functions for machine learning

Francesco Giannini[1] · Michelangelo Diligenti[2] · Marco Maggini[2] · Marco Gori[2,3] · Giuseppe Marra[4]

Accepted: 5 December 2022 / Published online: 9 February 2023

Abstract

Injecting prior knowledge into the learning process of a neural architecture is one of the main challenges currently faced by the artificial intelligence community, which also motivated the emergence of neural-symbolic models. One of the main advantages of these approaches is their capacity to learn competitive solutions with a significant reduction of the amount of supervised data. In this regard, a commonly adopted solution consists of representing the prior knowledge via first-order logic formulas, then relaxing the formulas into a set of differentiable constraints by using a t-norm fuzzy logic. This paper shows that this relaxation, together with the choice of the penalty terms enforcing the constraint satisfaction, can be unambiguously determined by the selection of a t-norm generator, providing numerical simplification properties and a tighter integration between the logic knowledge and the learning objective. When restricted to supervised learning, the presented theoretical framework provides a straight derivation of the popular cross-entropy loss, which has been shown to provide faster convergence and to reduce the vanishing gradient problem in very deep structures. However, the proposed learning formulation extends the advantages of the cross-entropy loss to the general knowledge that can be represented by neural-symbolic methods. In addition, the presented methodology allows the development of novel classes of loss functions, which are shown in the experimental results to lead to faster convergence rates than the approaches previously proposed in the literature.

Keywords Learning from constraints · T-norm generators · Loss functions · Integration of logic and learning · Neural-symbolic integration

1 Introduction

Deep Neural Networks [1] have been a break-through for several classification problems involving sequential or high-dimensional data. However, deep neural architectures strongly rely on a large amount of labeled data to develop powerful feature representations. Unfortunately, it is difficult and labor intensive to annotate such large collections of data. In this regard, prior knowledge expressed by First-Order Logic (FOL) rules represents a natural solution to make learning efficient when the training data is scarce and some domain expert knowledge is available. The integration of logic inference with learning could also overcome another limitation of deep architectures, namely that they mainly act as black-boxes from a human perspective, making their usage difficult in safety critical applications, like in health or car industry applications [2]. For these reasons, Neural-Symbolic (NeSy) approaches [3, 4] integrating logic and learning have become one of the fundamental research lines for the machine learning and artificial intelligence communities. One of the most common approaches to exploit logic knowledge to train a deep neural learner relies on mapping the FOL knowledge into differentiable constraints using t-norms. Then, the constraints can be enforced using gradient-based optimization techniques, like done in [5, 6]. Most work in this area approached the problem of translating logic rules into a differentiable form by defining a collection of heuristics that often lack semantic consistency and have no clear motivation from a theoretical point of view. For instance, there is no agreement on the relation between the selected t-norm and the aggregation function corresponding to the logic

Francesco Giannini, Michelangelo Diligenti and Giuseppe Marra are contributed equally to this work.

✉ Francesco Giannini
francesco.giannini@unisi.it

Extended author information available on the last page of the article.

quantifiers, nor even on the chosen loss to enforce the constraints.

This paper first traces back the properties of t-norm fuzzy logic operators down to the selection of a generator function. Then, we show that the loss function of a learning problem accounting for both supervised data and logic constraints can also be determined by the single choice of the *t-norm generator*. The generator determines the fuzzy relaxation of connectives and quantifiers occurring in the logic rules. As a result, a simplified and semantically consistent optimization problem can be formulated. In this framework, the classical fitting of supervised training data can be enforced by atomic logic constraints. Since the careful choice of loss functions has been crucial to the success of deep learning, this paper also investigates the relation between supervised training losses and generator choices. As a special case, we get a novel justification for the popular cross-entropy loss [7], that has been shown to provide faster convergence and to reduce the vanishing gradient problem in very deep structures.

Contributions This paper introduces a theoretical framework centered around the notion of t-norm generator, unifying the choice of the logic semantics and of the loss function in neural-symbolic learners. In particular, we extend the preliminary formalization sketched in [8], together with a more comprehensive experimental validation. This unification results in a simplified learning objective that is shown to be numerically more stable, while retaining the flexibility to customize the learning process on the considered applications.

The paper is organized as follows: Section 2 presents some prior work on the integration of learning and logic inference, Section 3 presents the basic concepts about t-norms, generators and aggregator functions and Section 4 introduces a general neural-symbolic framework used to extend supervised learning with logic rules. Section 5 presents the main results of the paper, showing the link between t-norm generators and loss functions and how these can be exploited in neural-symbolic approaches. Section 6 presents the experimental results and a discussion on the presented methodology is provided in Section 7. Finally, Section 8 draws some conclusions.

2 Related works

Neural-symbolic approaches [9, 10] aim at combining symbolic reasoning with (deep) neural networks, e.g. by exploiting additional logic knowledge when available. This knowledge can be either injected into the learner internal structure (e.g. by constraining the network architecture) or enforced on the learner outputs (e.g. by adding new loss terms). In this context, First-Order Logic is commonly chosen as the declarative framework to represent the knowledge because of its flexibility and expressive power. NeSy methodologies are rooted in previous work on Statistical Relational Learning (SRL) [3, 11], which developed frameworks for performing logic inference in presence of uncertainty. For instance, Markov Logic Networks (MLN) [12] and Probabilistic Soft Logic (PSL) [13] integrate FOL and probabilistic graphical models by using the logic rules as potential functions defining a probability distribution. MLNs have received a lot of attention by the SRL community [14–16] and have been widely used in different tasks like information extraction, entity resolution and text mining [17, 18]. More recently, MLNs have also been extended to work with neural potential functions in [19], showing impressive results e.g. in generating molecular data. PSL can be considered a fuzzy extension of MLNs, as it exploits a fuzzy relaxation of the logic potentials by using Łukasiewicz Logic. The framework proposed in this paper builds upon t-norm fuzzy logics, however it is not limited to any specific t-norm. Hence it could be also adopted to define alternative logic potential functions for PSL.

A common solution to integrate logic reasoning and deep learning relies on using deep neural networks to approximate the truth values (i.e. fuzzy semantics) or the probabilities (i.e. probabilistic semantics) of certain target predicates, and then apply logic or probabilistic inference on the network outputs [20]. In the former case, the logic rules can be relaxed according to a differentiable fuzzy logic and then the overall architecture can be optimized end-to-end. This approach is followed with minor variants by Semantic-Based Regularization (SBR) [5], Lyrics [21] and Logic Tensor Networks (LTN) [6], especially for classification problems. On the other hand, some examples of NeSy approaches based on probabilistic logic are given by Semantic Loss [22], Differentiable Reasoning [23], Deep Logic Models [24], Relational Neural Machines [25] and DeepProbLog [26]. Similarly, Lifted Relational Neural Networks [27] and Neural Theorem Provers [28, 29] realize a soft forward or backward chaining via an end-to-end gradient based scheme. This paper investigates the bound between the selected logic semantics to represent the knowledge and the loss function in the learning task. This is a common problem for all NeSy approaches, that encode the logic knowledge into differentiable constraints used by a deep learner.

Learning with fuzzy logic constraints In general, if some FOL knowledge is available for a learning problem, this is expressed in Boolean form. To define a differentiable learning objective is then fundamental to establish a mapping to relax the logic formulas into differentiable functional constraints by means of an appropriate fuzzy logic. For instance, Serafini et al. [30] introduces a learning framework where the formulas are converted according to the t-norm and t-conorm of Łukasiewicz logic. Giannini et al. [31] also proposes to convert the formulas according to Łukasiewicz logic, however they exploit the weak conjunction in place of the t-norm, thus guaranteeing convex functional constraints. A more empirical approach has been considered in SBR, where all the fundamental t-norms have been evaluated on different learning settings to select the best t-norm on the single tasks [5]. More recent studies on the learning properties of different fuzzy logic operators have also been proposed by Van Krieken et al. [32, 33]. By combining different logic semantics for the connectives, the authors achieved the most significant performance improvement, but the dependence between the connectives is no longer obeying any specific logic theory.

The relaxation of logic quantifiers has also been the subject of a wide range of studies. On the performance side, different quantifier conversions have been taken into account and validated. For instance, in Diligenti et al. [5] the arithmetic mean and the maximum operator have been used to convert the universal and existential quantifiers, respectively. Different possibilities have been considered for the universal quantifier in Donadello et al. [34], while the existential quantifier depends on this choice via the application of the strong negation using the DeMorgan law. However, the arithmetic mean operator has been shown to achieve better performances in the conversion of the universal quantifier [34], with the existential quantifier implemented by Skolemization. In spite of improving the performances, the universal and existential quantifiers should be thought of as a generalized AND and OR, respectively. Therefore, converting these quantifiers using a mean operator has no direct justification inside a logic theory, and spoil the original semantics.

There have been a few attempts in the literature to address the problem of choosing semantically driven loss functions to enforce the satisfaction of the logic constraints. However, these works are generally not fully semantically coherent or too specific. A unified principle to select a suitable loss function that can be logically interpreted according to the adopted fuzzy logic semantics is still missing. For instance, both SBR [5] and LTN [30] rely on minimizing the strong negation of each logic constraint, whereas Lyrics [21] also allows the usage of the negative logarithm. A different perspective is considered in Semantic Loss [22], where the authors propose a new loss function that is very close to the negative logarithm one and that is able to achieve (near) state-of-the-art performances on semi-supervised learning tasks, by combining neural networks and logic constraints. In this paper, we show that these loss functions (and infinitely many more) are special cases of t-norm generators that can be uniquely determined by the choice of a fuzzy logic relaxation.

3 Background on t-norm fuzzy logic

Many-valued logics have been introduced in order to extend the admissible set of truth values from *true* (1) and *false* (0) to a scale of truth-degree having *absolutely true* and *absolutely false* as boundary cases. A fuzzy logic is a many-valued logic, whose set of truth values coincides with the real unit interval $[0, 1]$. This section introduces the basic notions of fuzzy logic together with some illustrative examples.

T-norms [35] are a special kind of binary operations on the real unit interval $[0, 1]$, representing an extension of the Boolean conjunction.

Definition 1 $T : [0, 1]^2 \to [0, 1]$ is a t-norm if and only if for every $x, y, z \in [0, 1]$:

$$T(x, y) = T(y, x), \quad T(x, T(y, z)) = T(T(x, y), z),$$
$$T(x, 1) = x, \quad T(x, 0) = 0, \quad \text{if } x \le y \text{ then } T(x, z) \le T(y, z)\,.$$

T is a continuous t-norm if it is a continuous function in $[0, 1]$.

A fuzzy logic can be uniquely defined according to the choice of a certain t-norm T [36]. A wide variety of operations corresponding to different fuzzy logic connectives are defined starting from T and the strong negation "$\neg$", and their notation is introduced in Definition 2. Table 1 reports the algebraic semantics of these connectives for Gödel, Łukasiewicz and Product logics, which are referred as the fundamental fuzzy logics, because all the continuous t-norms can be obtained from them by ordinal sums [37].

Definition 2

(t-norm)	$x \otimes y = T(x, y)$
(residuum)	$x \Rightarrow y = \max\{z : x \otimes z \le y\}$
(bi-residuum)	$x \Leftrightarrow y = (x \Rightarrow y) \otimes (y \Rightarrow x)$
(weak conjunction)	$x \wedge y = x \otimes (x \Rightarrow y)$
(weak disjunction)	$x \vee y = ((x \Rightarrow y) \Rightarrow y) \otimes ((y \Rightarrow x) \Rightarrow x)$
(residual negation)	$\sim x = x \Rightarrow 0$
(strong negation)	$\neg x = 1 - x$
(t-conorm)	$x \oplus y = \neg(\neg x \otimes \neg y)$
(material implication)	$x \rightarrow y = \neg x \oplus y$

3.1 Archimedean t-norms

Continuous Archimedean t-norms [35] are special t-norms that can be constructed by means of unary monotone functions, called *generators*.

Table 1 The truth functions for the t-norm, residuum, bi-residuum, weak conjunction, weak disjunction, residual negation, strong negation, t-conorm and material implication of the fundamental fuzzy logics

	Gödel	Łukasiewicz	Product
$x \otimes y$	$\min\{x, y\}$	$\max\{0, x + y - 1\}$	$x \cdot y$
$x \Rightarrow y$	$x \leq y?1 : y$	$\min\{1, 1 - x + y\}$	$\min\{1, \frac{y}{x}\}$
$x \Leftrightarrow y$	$x \leq y?x : y$	$1- \mid x - y \mid$	$x = y?1 : \min\left\{\frac{x}{y}, \frac{y}{x}\right\}$
$x \wedge y$	$\min\{x, y\}$	$\min\{x, y\}$	$\min\{x, y\}$
$x \vee y$	$\max\{x, y\}$	$\max\{x, y\}$	$\max\{x, y\}$
$\sim x$	$x = 0?1 : 0$	$1 - x$	$x = 0?1 : 0$
$\neg x$	$1 - x$	$1 - x$	$1 - x$
$x \oplus y$	$\max\{x, y\}$	$\min\{1, x + y\}$	$x + y - x \cdot y$
$x \rightarrow y$	$\max\{1 - x, y\}$	$\min\{1, 1 - x + y\}$	$1 - x + x \cdot y$

Definition 3 A t-norm T is Archimedean if for every $x \in (0, 1)$ it holds $T(x, x) < x$. T is said to be strict if for all $x \in (0, 1)$ we have $0 < T(x, x) < x$, otherwise it is said to be nilpotent.

For example, Łukasiewicz (T_L) and Product (T_P) t-norms are nilpotent and strict respectively, while Gödel (T_G) t-norm is idempotent (i.e. $\forall x : T_G(x, x) = x$) and hence not even Archimedean. In addition, all the nilpotent and strict t-norms can be related to the Łukasiewicz and Product t-norms as follows.

Theorem 1 ([35]) *Any nilpotent t-norm is isomorphic to T_L and any strict t-norm is isomorphic to T_P.*

The next theorem shows how to construct t-norms by *additive*[1] *generators* [35].

Theorem 2 *Let $g : [0, 1] \rightarrow [0, +\infty]$ be a strictly decreasing function with $g(1) = 0$ and $g(x) + g(y) \in Range(g) \cup \{g(0^+), +\infty\}$ for all x, y in $[0, 1]$, and $g^{(-1)}$ its pseudo-inverse. Then the function $T : [0, 1]^2 \rightarrow [0, 1]$ defined as*

$$T(x, y) = g^{-1}\left(\min\{g(0^+), g(x) + g(y)\}\right) \tag{1}$$

is a t-norm and g is said an additive generator for T. Moreover, T is strict if $g(0^+) = +\infty$, otherwise T is nilpotent.

Example 1 If we take $g(x) = 1 - x$, we get the Łukasiewicz t-norm T_L.

$$T(x, y) = 1 - \min\{1, 1 - x + 1 - y\} = \max\{0, x + y - 1\}$$

Example 2 If we take $g(x) = -\log(x)$, we get the Product t-norm T_P.

$$T(x, y) = \exp\left(-(\min\{+\infty, -\log(x) - \log(y)\})\right) = x \cdot y$$

According to (1), the other fuzzy logic connectives deriving from the t-norm can be expressed with respect to the generator. For instance:

$$\begin{aligned} x \Rightarrow y &= g^{-1}(\max\{0, g(y) - g(x)\}) \\ x \Leftrightarrow y &= g^{-1}(\mid g(x) - g(y) \mid) \\ x \oplus y &= 1 - g^{-1}\left(\min\{g(0^+), g(1 - x) + g(1 - y)\}\right) \end{aligned} \tag{2}$$

3.2 Parameterized classes of t-norms

T-norm generators can also depend on a parameter, by consequently defining a parameterized class of t-norms. For instance, given a generator g of a t-norm T and $\lambda > 0$, then T^λ denotes a class of increasing t-norms that correspond to the generator function $g^\lambda(x) = (g(x))^\lambda$. In addition, let T_D and T_G denote the Drastic ($T_D(x, y) = (x = y = 1)?1 : 0$) and Gödel t-norms respectively, we get:

$$\lim_{\lambda \to 0^+} T^\lambda = T_D \quad \text{and} \quad \lim_{\lambda \to \infty} T^\lambda = T_M$$

Over the years, several parameterized families of t-norms have been introduced and studied in the literature [35, 38]. In the following, we recall some prominent examples that we will exploit in the experimental evaluation.

Definition 4 (The Schweizer-Sklar family) For $\lambda \in (-\infty, +\infty)$, consider:

$$g_\lambda^{SS}(x) = \begin{cases} -\log(x) & \text{if } \lambda = 0 \\ \frac{1 - x^\lambda}{\lambda} & \text{otherwise} \end{cases}$$

The t-norms corresponding to this generator are called Schweizer-Sklar t-norms, and they are defined according to:

$$T_\lambda^{SS}(x, y) = \begin{cases} T_G(x, y) & \text{if } \lambda = -\infty \\ (x^\lambda + y^\lambda - 1)^{\frac{1}{\lambda}} & \text{if } -\infty < \lambda < 0 \\ T_P(x, y) & \text{if } \lambda = 0 \\ \max\{0, x^\lambda + y^\lambda - 1\}^{\frac{1}{\lambda}} & \text{if } 0 < \lambda < +\infty \\ T_D(x, y) & \text{if } \lambda = +\infty \end{cases}$$

The Schweizer-Sklar t-norm T_λ^{SS} is Archimedean if and only if $\lambda > -\infty$, continuous if and only if $\lambda < +\infty$, strict

[1] Since here we only deal with additive generators, we will drop the term "additive" for simplicity.

if and only if $-\infty < \lambda \leq 0$ and nilpotent if and only if $0 < \lambda < +\infty$. This t-norm family is strictly decreasing for $\lambda \geq 0$ and continuous with respect to $\lambda \in [-\infty, +\infty]$, in addition $T_1^{SS} = T_L$.

Definition 5 (Frank t-norms) For $\lambda \in [0, +\infty]$, consider:

$$g_\lambda^F(x) = \begin{cases} -\log(x) & \text{if}\lambda = 1 \\ 1 - x & \text{if}\lambda = +\infty \\ \log\left(\frac{\lambda - 1}{\lambda^x - 1}\right) & \text{otherwise} \end{cases}$$

The t-norms corresponding to this generator are called Frank t-norms and they are strict if $\lambda < +\infty$. The overall class of Frank t-norms is decreasing and continuous.

$$T_\lambda^F(x, y) = \begin{cases} T_G & \text{if } \lambda = 0 \\ T_P & \text{if } \lambda = 1 \\ T_L & \text{if } \lambda = +\infty \\ \log_\lambda\left(1 + \frac{(\lambda^x - 1)(\lambda^y - 1)}{\lambda - 1}\right) & \text{otherwise} \end{cases}$$

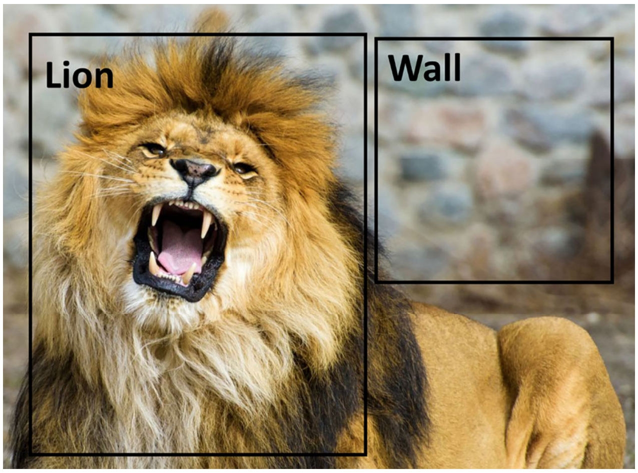


Fig. 1 Image labeled with the presence of a lion and a wall. Classification tasks performed by sub-symbolic models can benefit from logic inference on additional symbolic knowledge

4 Background on the integration of learning and logic reasoning

According to the *learning from logical constraints* paradigm [20], the available prior knowledge is represented by a set of logic rules. which are relaxed into continuous and differentiable constraints over the task functions (implementing FOL predicates). Positive and negative supervised samples can also be seen as atomic constraints, and the learning process corresponds to finding the task functions that best satisfy the constraints.

Example 3 Let us assume that the prior knowledge for an image classification task is expressed by the following sentences "lions live in savanna or in zoos" and "there are no walls in the savanna" (see Fig. 1). This domain knowledge can be represented in FOL as "$\forall x\ Lion(x) \rightarrow LiveIn(x, savanna) \vee LiveIn(x, zoo)$" and "$\forall x\ Wall(x) \rightarrow \neg LiveIn(x, savanna)$", being $Lion$, $Wall$ two unary predicates, $LiveIn$ a binary predicate and $savanna$, zoo two constants. If a neural classifier is able to correctly detect the presence of a lion and a wall in Fig. 1, it is also able to establish that the lion is living in a zoo by exploiting the symbolic knowledge.

In the following, we introduce more formally the framework where our work takes place. Let us consider a multi-task learning problem where $\mathbf{B}_P = (P_1, \ldots, P_J)$ denotes the vector of real-valued functions (task functions) to be determined. Given the set $\mathcal{X} \subseteq \mathbb{R}^n$ of available data, a supervised learning problem can be generally formulated as $\min_{\mathbf{B}_P} \mathcal{L}(\mathcal{X}, \mathbf{B}_P)$ where $\mathcal{L}$ is a positive-valued functional denoting a certain loss. In our framework, we assume that the task functions are FOL predicates and all the available knowledge about these predicates, including supervisions, is collected into a knowledge base $KB = \{\psi_1, \ldots, \psi_H\}$ of FOL formulas. The learning task is then expressed as:

$$\min_{\mathbf{B}_P} \mathcal{L}(\mathcal{X}, KB, \mathbf{B}_P)$$

The link between FOL knowledge and learning was also presented e.g. in [21] and it can be summarized as follows.

- Each *Individual* is an element of a specific domain, which can be used to ground the predicates defined on such a domain. Any replacement of variables with individuals for a certain predicate is called *grounding*.
- *Predicates* express the truth degree of some property for an individual (unary predicate) or group of individuals (n-ary predicate). In particular, this paper will focus on learnable predicate functions implemented by (deep) neural networks, but other models can also be used. FOL *functions* can be included and learned in a similar fashion [39]. However, in this presentation, function-free FOL is used to keep the notation simpler.
- *Knowledge Base* (KB) is a collection of FOL formulas expressing the learning task. The integration of learning and logical reasoning is achieved by compiling the logical rules into continuous real-valued constraints correlating all the defined elements and enforcing some expected behavior on them.

Given any rule in KB, individuals, predicates, logical connectives and quantifiers can all be seen as nodes of an *expression tree* [40]. Then, the translation into a functional constraint corresponds to a post-fix visit of the expression tree, consisting of the following steps:

- visiting a *variable* substitutes the variable with the corresponding feature representation of the individual to which the variable is currently assigned;
- visiting a *predicate* computes the output of the predicate with the current input groundings;
- visiting a *connective* combines the grounded predicate values by means of the real-valued operation associated to the connective;
- visiting a *quantifier* aggregates the outputs of the expressions obtained for the single individuals (variable groundings).

Thus, the compilation of the expression tree allows us to convert a formula into a real-valued function, represented by a computational graph. The different functions corresponding to predicates are composed (i.e. aggregated) by means of the truth-functions corresponding to connectives and quantifiers. Given a formula φ, we denote by f_φ its corresponding real-valued functional representation. f_φ tightly depends on the chosen t-norm driving the fuzzy relaxation. The expression tree corresponding to the FOL formula $\forall x\ Wall(x) \rightarrow \neg LiveIn(x, savanna)$ is reported in Fig. 2 as an example.

Example 4 Given two predicates P_1, P_2 and the formula $\varphi(x) = P_1(x) \Rightarrow P_2(x)$, the functional representation of φ is given by $f_\varphi(x, \mathbf{B}_P) = \min\{1, 1 - P_1(x) + P_2(x)\}$ and $f_\varphi(x, \mathbf{B}_P) = \min\{1, P_2(x)/P_1(x)\}$ in the Łukasiewicz and Product logics, respectively.

A special note concerns quantifiers. They aggregate the truth-values of predicates over their corresponding domains. For instance, according to [41], that first proposed a fuzzy generalization of FOL, the universal and existential quantifiers may be converted as the infimum and supremum over a domain variable (coinciding with minimum and maximum when dealing with finite domains). In particular, given a formula $\varphi(x)$ depending on a certain variable $x \in \mathcal{X}$, where $\mathcal{X}$ denotes the finite set of available samples for one of the involved predicates in φ, the fuzzy semantics of the quantifiers is given by:

$$\psi = \forall x\, \varphi(x) \longrightarrow f_\psi(\mathcal{X}, \mathbf{B}_P) = \min_{x\in\mathcal{X}} f_\varphi(x, \mathbf{B}_P)$$
$$\psi = \exists x\, \varphi(x) \longrightarrow f_\psi(\mathcal{X}, \mathbf{B}_P) = \max_{x\in\mathcal{X}} f_\varphi(x, \mathbf{B}_P)$$

As shown in the next section, this quantifier relaxation is not convenient for all the t-norms and we propose a more principled approach for the translation.

Once all the formulas in KB are converted into real-valued functions, their distance from satisfaction (i.e. distance from 1-evaluation) can be computed according to a certain decreasing mapping L expressing the penalty for the violation of any constraint. In order to satisfy all the constraints, the learning problem can be formulated as the joint minimization over the single rules using the following loss function factorization:

$$\mathcal{L}(\mathcal{X}, KB, \mathbf{B}_P) = \sum_{\psi\in KB} \beta_\psi L\big(f_\psi(\mathcal{X}, \mathbf{B}_P)\big) \tag{3}$$

Here any β_ψ denotes the weight for the logical constraint ψ in the KB, which can be selected via cross-validation or jointly learned [24, 42], f_ψ is the functional representation of the formula ψ according to a certain t-norm fuzzy logic and L is a decreasing function denoting the penalty associated to the distance from satisfaction of formulas, so that $L(1) = 0$.

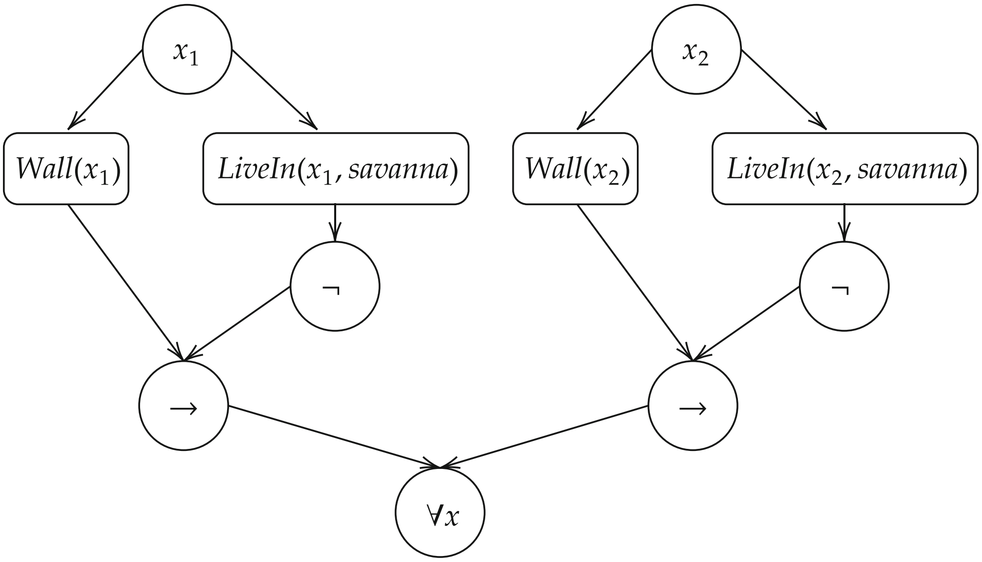


Fig. 2 The expression tree corresponding to $\forall x\ Wall(x) \rightarrow \neg LiveIn(x, savanna)$ for the domain of constants $\mathcal{X} = \{x_1, x_2\}$

As described in Section 2, in this neural-symbolic scenario all the steps involved in the translation of FOL formulas into a loss function are treated separately, involving very heterogeneous choices. In the next section, we show instead that these steps are intrinsically connected and they can be uniformly derived from a unique global choice: the selection of a t-norm generator.

5 Loss functions by t-norms generators

This section presents a generalization of the approach introduced in [8], which was limited to supervised learning. In this paper, we present a unified principle to translate the fuzzy relaxation of FOL formulas into the loss function of general machine learning tasks. In particular, we study the mapping of FOL formulas into functional constraints by means of continuous Archimedean t-norm fuzzy logics. We adopt the t-norm generator to penalize the violation of the constraints, i.e. we take $L = g$. Moreover, since the quantifiers can be seen as generalized AND and OR over the grounded expressions (see Remark 1), we show that by adopting the same fuzzy conversion for connectives and quantifiers, the overall loss function expressed in (3) only depends on the chosen t-norm generator g.

Remark 1 Given a formula $\varphi(x)$ defined on the available set of samples $\mathcal{X} = \{x_1, \ldots, x_N\}$, the roles of the quantifiers have to be interpreted as follows:

$$\begin{aligned} \forall x\, \varphi(x) \;&\simeq\; \varphi(x_1) \text{ AND } \ldots \text{ AND } \varphi(x_N) \\ \exists x\, \varphi(x) \;&\simeq\; \varphi(x_1) \text{ OR } \ldots \text{ OR } \varphi(x_N) \end{aligned}$$

5.1 General formulas

Given a certain formula $\varphi(x)$ depending on a variable x that ranges in the set $\mathcal{X}$ and its corresponding functional representation $f_\varphi(x, \mathbf{B}_P)$, the conversion of any universal quantifier may be carried out by means of an Archimedean t-norm T, while the existential quantifier by a t-conorm. For instance, given the formula $\psi = \forall x\, \varphi(x)$, we have:

$$f_\psi(\mathcal{X}, \mathbf{B}_P) = g^{-1}\left(\min\left\{g(0^+), \sum_{x\in\mathcal{X}} g\big(f_\varphi(x, \mathbf{B}_P)\big)\right\}\right) \tag{4}$$

where g is a generator of the t-norm T.

Since any generator function g is decreasing and $g(1) = 0$, a generator is a suitable choice to map the fuzzy conversion of a formula into a constraint loss to be minimized. By exploiting the same generator of T as loss function (i.e. taking $L = g$) for $\psi = \forall x\, \varphi(x)$ expressed by (4), we get the following term $L\big(f_\psi(\mathcal{X}, \mathbf{B}_P)\big)$ to be minimized:

$$L\big(f_\psi(\mathcal{X}, \mathbf{B}_P)\big) = \begin{cases} \min\left\{g(0^+), \sum\limits_{x\in\mathcal{X}} g(f_\varphi(x, \mathbf{B}_P))\right\} & \text{if } T \text{ is nilpotent} \\ \sum\limits_{x\in\mathcal{X}} g(f_\varphi(x, \mathbf{B}_P)) & \text{if } T \text{ is strict} \end{cases} \tag{5}$$

As a consequence, the following result can be provided with respect to the convexity of the loss $L\big(f_\psi(\mathcal{X}, \mathbf{B}_P)\big)$.

Proposition 1 *If g is a linear function and f_ψ is concave then $L\big(f_\psi(\mathcal{X}, \mathbf{B}_P)\big)$ is convex. If g is a convex function and f_ψ is linear then $L\big(f_\psi(\mathcal{X}, \mathbf{B}_P)\big)$ is convex.*

Proof Both the arguments follow since, if f_ψ is concave (we recall that a linear function is both concave and convex) and g is a convex non-increasing function defined over a univariate domain, then $g \circ f_\psi$ is convex. □

Proposition 1 establishes a general criterion to define convex constraints according to a certain generator depending on the fuzzy conversion f_ψ and, in turn, by the logical expression ψ. In the following of this section, we show some application cases of this proposition.

So far, we did not make any hypothesis on the formula φ. In the following, different cases of interest for the main connective of φ are reported. Given an additive generator g for a t-norm T, additional connectives may be expressed with respect to g, as reported by (2). If P_1, P_2 are two unary predicate functions sharing the same input domain $\mathcal{X}$, the following formulas yield the following penalty terms, where we supposed T strict for simplicity:

$$\begin{aligned} \forall x\; P_1(x) \;&\longrightarrow\; \sum_{x\in\mathcal{X}} g(P_1(x)) \\ \forall x\; P_1(x) \Rightarrow P_2(x) \;&\longrightarrow\; \sum_{x\in\mathcal{X}} \max\{0, g(P_2(x)) - g(P_1(x))\} \\ \forall x\; P_1(x) \Leftrightarrow P_2(x) \;&\longrightarrow\; \sum_{x\in\mathcal{X}} \mid g(P_1(x)) - g(P_2(x)) \mid \end{aligned}$$

Examples of derived losses According to the selection of the generator, the same FOL formula can be mapped to different loss functions. This enables us to design customized losses that are more suitable for a specific learning problem, or to provide a theoretical justification to the losses that are already commonly utilized by the machine learning community. Examples 5–8 show some application cases. In particular, also the cross-entropy loss (see Example 6) can be justified under the same logical perspective.

Example 5 If $g(x) = 1 - x$ we get the Łukasiewicz t-norm, that is nilpotent. Hence, from (5) we get:

$$L\big(f_\psi(\mathcal{X}, \mathbf{B}_P)\big) = \min\left\{1, \sum_{x \in \mathcal{X}} (1 - (f_\varphi(x, \mathbf{B}_P)))\right\} .$$

In case f_ψ is concave (e.g. if ψ belongs to the concave fragment of Łukasiewicz logic [31]), this function is convex.

Example 6 If $g(x) = -\log(x)$ we get the Product t-norm, that is strict. From (5) we get a generalization of the cross-entropy loss:

$$L\big(f_\psi(\mathcal{X}, \mathbf{B}_P)\big) = -\sum_{x \in \mathcal{X}} \log(f_\varphi(x)) .$$

In case $f_\psi(x)$ is linear (e.g. a literal), this function is convex.

Example 7 If $g(x) = \frac{1}{x} - 1$, with corresponding strict t-norm $T(x, y) = \frac{xy}{x+y-xy}$, the penalty term that is obtained applying g to the formula $\psi = \forall x\ P_1(x) \Rightarrow P_2(x)$ is given by

$$L\big(f_\psi(\mathcal{X}, \mathbf{B}_P)\big) = \sum_{x \in \mathcal{X}} \max\left\{0, \frac{1}{P_2(x)} - \frac{1}{P_1(x)}\right\} .$$

Example 8 If $g(x) = 1 - x^2$, with corresponding nilpotent t-norm $T(x, y) = \min\{1, 2 - x^2 - y^2\}$, we get for $\psi = \forall x\ P_1(x) \Rightarrow P_2(x)$

$$L\big(f_\psi(\mathcal{X}, \mathbf{B}_P)\big) = \min\left\{1, \sum_{x \in \mathcal{X}} \max\left\{0, (P_1(x))^2 - (P_2(x))^2\right\}\right\} .$$

5.2 Simplication property

An interesting property of the presented formulation consists in the fact that, in case of compound formulas, several occurrences of the generator may be simplified. For instance, the conversion $f_\psi(\mathcal{X}, \mathbf{B}_P)$ of the formula $\psi = \forall x\ P_1(x) \otimes P_2(x) \Rightarrow P_3(x)$ with respect to the selection of a strict t-norm generator g becomes:

$$\overbrace{g^{-1}\left(\sum_x g\left(\overbrace{g^{-1}\left(\max\left\{0, g(P_3(x)) - g\left(\overbrace{g^{-1}(g(P_1(x))+g(P_2(x)))}^{conjunction}\right)\right\}\right)}^{implication}\right)\right)}^{quantifier}$$
$$= g^{-1}\left(\textstyle\sum_x \max\{0, g(P_3(x)) - g(P_1(x)) - g(P_2(x))\}\right)$$

The simplification expressed on the lower side is general and can be applied to a wide range of logical operators, reducing the required number of applications of g^{-1} to just the one in front of the expression. In these cases, by applying $L = g$, the overall penalty of the formula can be determined by just evaluating g on the predicate functions and without applying g^{-1}. Since g and g^{-1} can be in general affected by numerical issues (e.g. $g = -\log$), this property may allow the implementation of more numerically stable loss functions, totally preserving the initial semantics of the formula.

However, this property does not hold for all the connectives that are definable upon a certain generated t-norm (see Definition 2). For instance, $\forall x\ \ P_1(x) \oplus P_2(x)$ becomes:

$$g^{-1}\Big(\sum_x g(1 - g^{-1}(g(1 - P_1(x)) + g(1 - P_2(x))))\Big)$$

This suggests to identify the connectives that allow, on one hand the simplification of any occurrence of g^{-1} in $L\big(f_\psi(\mathcal{X}, \mathbf{B}_P)\big)$, and on the other hand the evaluation of g only on grounded predicates. For short, in the following we say that the formulas built upon such connectives have the *simplification property*.

Lemma 1 *Any formula φ whose connectives are restricted to $\{\wedge, \vee, \otimes, \Rightarrow, \sim, \Leftrightarrow\}$ has the simplification property.*

Proof The proof is by induction with respect to the number $l \geq 0$ of connectives occurring in φ.

- If $l = 0$, i.e. $\varphi = P_j(x_i)$ for a certain $j \leq J$ and $x_i \in \mathcal{X}$, then $g(f_\varphi) = g(P_j(x_i))$. Hence φ has the simplification property.
- If $l = k + 1$ then $\varphi = (\alpha \circ \beta)$ for $\circ \in \{\wedge, \vee, \otimes, \Rightarrow, \sim, \Leftrightarrow\}$ and we have the following cases.
  - If $\varphi = (\alpha \wedge \beta)$ then we get $g(\min\{f_\alpha, f_\beta\}) = \max\{g(f_\alpha), g(f_\beta)\}$. The claim follows by an inductive hypothesis on α, β whose number of involved connectives is less or equal than k. The argument still holds replacing $\wedge$ with $\vee$ and min with max.
  - If $\varphi = (\alpha \otimes \beta)$ then we get
  $$g(g^{-1}(\min\{g(0^+), g(f_\alpha) + g(f_\beta)\})) = \min\{g(0^+), g(f_\alpha) + g(f_\beta)\} .$$
  As in the previous case, the claim follows by inductive hypothesis on α, β.
  - The remaining of the cases can be treated in the same way and noting that $\sim \alpha = \alpha \Rightarrow 0$.

□

The simplification property provides several advantages from an implementation point of view. First, it allows the evaluation of the generator function only on grounded predicate expressions and avoids an explicit computation of the pseudo-inverse g^{-1}. Second, this property provides a general method to implement n-ary t-norms, of which universal quantifiers can be seen as a special case since we only deal with finite domains (see Section 7). Moreover, it is worth to notice that this property does not rely on specific assumptions on the neural models adopted to implement the predicate functions nor on the chosen fuzzy logic exploited for the relaxation. As a result, Lemma 1 can be applied in a wide range of cases.

Finally, the simplification property yields an interesting analogy between truth-functions and loss functions. In logic, the truth degree of a formula is obtained by combining the truth degree of its sub-formulas by means of connectives and quantifiers. In the same way, the loss corresponding to a formula that satisfies the simplification property is obtained by combining the losses corresponding to its sub-formulas, while connectives and quantifiers combine losses rather than truth degrees.

5.3 Manifold regularization: an example

Let us consider a simple multi-task classification problem where two objects A, B must be detected in a set of input images $\mathcal{I}$, represented as a set of features. The learning task consists in determining the predicates $P_A(i)$, $P_B(i)$, which return true if and only if the input image i is predicted to contain the object A, B, respectively. The positive supervised examples are provided as two sets (or equivalently their membership functions) $\mathcal{P}_A \subset \mathcal{I}$, $\mathcal{P}_B \subset \mathcal{I}$ with the images known to contain the object A, B, respectively. The negative supervised examples for A, B are instead provided as two sets $\mathcal{N}_A \subset \mathcal{I}$, $\mathcal{N}_B \subset \mathcal{I}$. Furthermore, the location where the images have been taken is assumed to be known, and a predicate $SameLoc(i_1, i_2)$ is used to express whether two images i_1, i_2 have been taken in the same location. Finally, we assume that two images taken in the same location are likely to contain the same object. This knowledge about the environment can be enforced via *Manifold Regularization*, which regularizes the classifier outputs over the manifold built by the image co-location defined via the $SameLoc$ predicate.

The overall knowledge on this learning task can be expressed using FOL via the statement declarations shown in Table 2, where it was assumed that images i_23, i_60 have been taken in the same location and it holds that $\mathcal{P}_A = \{i\_10, i\_101\}$, $\mathcal{P}_B = \{i\_103\}$, $\mathcal{N}_A = \{i\_11\}$ and $\mathcal{N}_B = \emptyset$. The statements define the constraints that the learners must respect on all the available samples, expressed as FOL rules. Please note that also the fitting of the supervisions on specific input images are expressed as constraints.

Table 2 Example of a learning task expressed using FOL

$\forall i_1, i_2 : SameLoc(i_1, i_2) \Rightarrow (P_A(i_1) \Leftrightarrow P_A(i_2))$
$\forall i_1, i_2 : SameLoc(i_1, i_2) \Rightarrow (P_B(i_1) \Leftrightarrow P_B(i_2))$
$\forall i : (\mathcal{P}_A(i) \Rightarrow P_A(i)) \wedge (\mathcal{N}_A(i) \Rightarrow \neg P_A(i))$
$\forall i : (\mathcal{P}_B(i) \Rightarrow P_B(i)) \wedge (\mathcal{N}_B(i) \Rightarrow \neg P_B(i))$
$\mathcal{P}_A(i\_10) = 1,\ \ \mathcal{P}_A(i\_101) = 1,\ \ \mathcal{N}_A(i\_11) = 1,\ \ \mathcal{P}_B(i\_103) = 1$
$SameLoc(i\_23, i\_60) = 1$

Given the selection of a strict generator g and a set of images $I \subseteq \mathcal{I}$, the FOL knowledge in Table 2 is compiled into the following optimization task:

$$\begin{aligned}\arg\min_{\mathbf{B}_P}\ & \beta_1 \sum_{i \in \mathcal{P}_A} g(P_A(i)) + \beta_2 \sum_{i \in \mathcal{N}_A} g(1 - P_A(i)) \\ & + \beta_3 \sum_{i \in \mathcal{P}_B} g(P_B(i)) + \beta_4 \sum_{i \in \mathcal{N}_B} g(1 - P_B(i)) \\ & + \beta_5 \sum_{(i_1, i_2) \in I_{sl}} \mid g(P_A(i_1)) - g(P_A(i_2)) \mid \\ & + \beta_6 \sum_{(i_1, i_2) \in I_{sl}} \mid g(P_B(i_1)) - g(P_B(i_2)) \mid\end{aligned}$$

where $\mathbf{B}_P = \{P_A, P_B\}$, each β_i is a meta-parameter deciding how strongly the i-th contribution should be weighted, I_{sl} is the set of image pairs having the same location $I_{sl} = \{(i_1, i_2) : SameLoc(i_1, i_2)\}$. The first four elements of the cost function express the fitting of the supervised data, while the latter two express manifold regularization over co-located images.

6 Experimental results

The experimental results have been carried out using the *Deep Fuzzy Logic* (DFL) software[2] which allows us to inject prior knowledge in form of a set of FOL formulas into a machine learning task. The formulas are compiled into differentiable constraints using the theory of generators as described in the previous sections. The learning task is then cast into an optimization problem like shown in Section 5.3 and, finally, optimized using the TensorFlow (TF) environment[3] [43]. In the following section, it is assumed that each FOL constant corresponds to a tensor storing its feature representation. Predicates are mapped to generic functions in the TF computational graph. If the function does not contain any learnable parameter in the graph, it is said to be *given*, otherwise the function/predicate is said to be *learnable*, and its parameters will be optimized to maximize the constraints satisfaction. Please note that any learner expressed as a TF computational graph can be transparently incorporated into DFL.

[2] http://sailab.diism.unisi.it/deep-logic-framework/

[3] https://www.tensorflow.org/

6.1 The learning task

The CiteSeer dataset [44] consists of 3312 scientific papers, each one assigned to one of six classes: Agents, AI, DB, IR, ML and HCI. The papers are not independent as they are connected by a citation network with 4732 links. This dataset defines a relational learning benchmark, where it is assumed that the representation of an input document is not sufficient for its classification without exploiting the citation network. The citation network can be used to inject useful information into the learning task, as it is often true that two papers connected by a citation belong to the same category.

This knowledge can be expressed by providing a general rule of the form: $\forall x\ \forall y\ Cite(x, y) \Rightarrow \big(P(x) \iff P(y)\big)$, where $Cite$ is a binary predicate encoding the fact that x is citing y and P is a task function implementing the membership function of one of the six considered categories. This logical formula expresses a form of manifold regularization, which often emerges in relational learning tasks. Indeed, by linking the prediction of two distinct documents, the behavior of the underlying task functions is regularized enforcing smooth transition over the manifold induced by the $Cite$ relation.

Each paper is represented via its bag-of-words, which is a vector having the same size of the vocabulary with the i-th element having a value equal to 1 or 0, depending on whether the i-th word in the vocabulary is present or absent in the document, respectively. In particular, the dictionary in this task consists of 3703 unique words. The set of input document representations is indicated by X, which is split into a train and test set X_{tr} and X_{te}, respectively. The percentage of documents in the two splits is varied across the different experiments. The six task functions P_i with $i \in \{Agents, AI, DB, IR, ML, HCI\}$ are bound to the six outputs of a Multi-Layer-Perceptron (MLP) implemented in TF. The neural architecture has 3 hidden layers, with 100 ReLU units each, and softmax activation on the output. Therefore, the task functions share the weights of the hidden layers in such a way that all of them can exploit a common hidden representation. The $Cite$ predicate is a given function, which outputs 1 if the document passed as first argument cites the document passed as second argument, otherwise it outputs 0. Furthermore, an additional given predicate $\mathcal{P}_i$ is defined for each P_i, such that it outputs 1 if and only if x is a positive example for the category i (i.e. it belongs to that category). $\mathcal{P}_i$ is a supervision predicate, which easily allows us to introduce a supervised signal using FOL (Section 5.1). A manifold regularization learning problem [46] can be defined by providing, $\forall i \in \{Agents, AI, DB, IR, ML, HCI\}$, the following two FOL formulas:

$$\forall x\ \forall y \quad Cite(x, y) \Rightarrow \big(P_i(x) \Leftrightarrow P_i(y)\big) \tag{6}$$

$$\forall x \quad \mathcal{P}_i(x) \Rightarrow P_i(x) \tag{7}$$

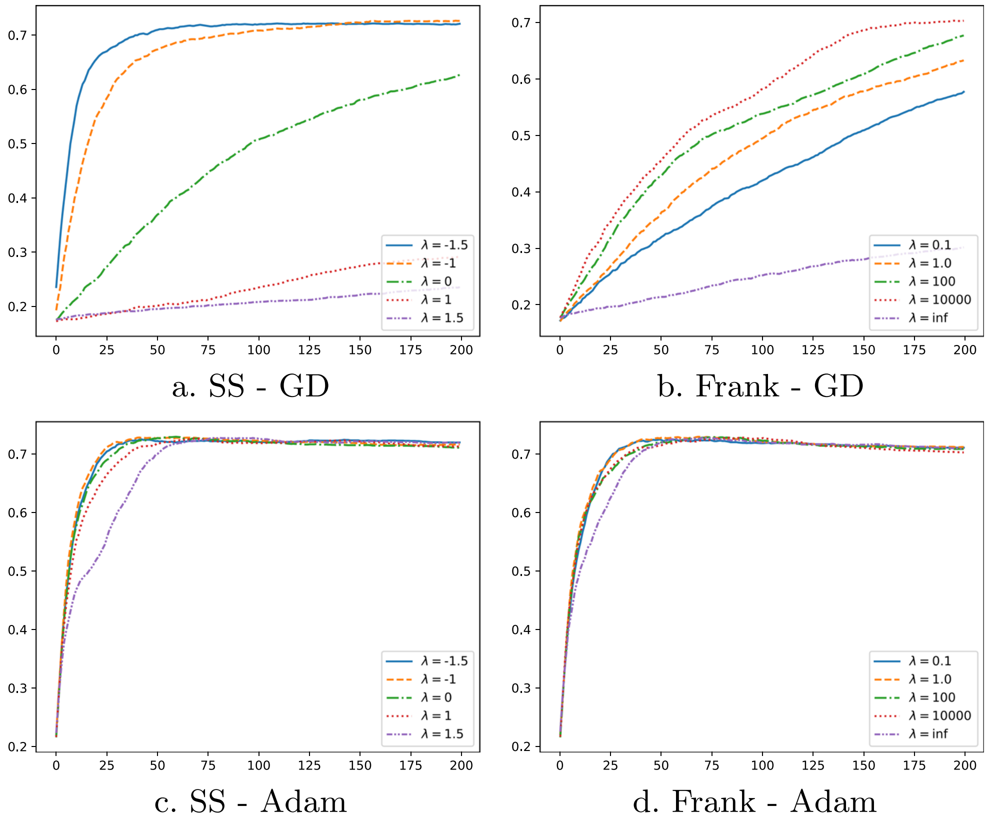
where only positive supervisions have been provided because the trained networks for this task employ a softmax activation function on the output layer, which has the effect of imposing mutually exclusivity among the task functions, reinforcing the positive class and discouraging all the others.

DFL allows the user to specify the weights of the formulas, which are treated as hyperparameters. Since we use two formulas per predicate, the weight of the formula expressing the fitting of the supervisions (7) is set to a fixed value equal to 1, while the weight of the manifold regularization rule (6) is cross-validated from the grid of values $\{0.1, 0.01, 0.006, 0.003, 0.001, 0.0001\}$.

6.2 Results

The experimental results measure different aspects of the integration of the prior logic knowledge into a supervised learning task. In particular, different experiments have been designed to track the speed at which the training process converges to the best solution, and how the classification accuracy changes with a variable amount of training data.

Training convergence rate This experimental setup aims at verifying the relation between the choice of the generator and the speed of convergence of the training process. In particular, a simple supervised learning setup is assumed for this experiment, where the learning task enforces the fitting of the supervised examples as defined by (7). The training and test sets are composed of 90% and 10% of the total number of papers, respectively. Two parameterized families of t-norms have been considered: the SS family (Definition 4) and the Frank family (Definition 5). Their parameter λ was varied to construct classical t-norms for some special values of the parameter but also to evaluate some intermediate ones. In order to keep a clear intuition behind the results, optimization was initially carried out using simply a Gradient Descent schema with a fixed learning rate equal to $\eta = 10^{-5}$. Results are shown in Fig. 3(-a) and (-b): it is evident that strict t-norms tend to learn faster than nilpotent ones by penalizing more strongly highly unsatisfied ground formulas. This difference is significant, although slightly reduced, when leveraging the state-of-the-art dynamic learning rate optimization algorithm Adam [45] as shown in Fig. 3-c and -d. This finding is consistent with the empirically well known fact



a. SS - GD

b. Frank - GD

c. SS - Adam

d. Frank - Adam

Fig. 3 Learning dynamics in terms of test accuracy on a supervised task when choosing different t-norms generated by the parameterized SS and Frank families: (a.) and (b.) are learning processes optimized with standard gradient descent, while (c.) and (d.) are optimized with Adam [45]

that the cross-entropy loss performs well in supervised learning tasks for deep architectures, because it is effective in avoiding gradient vanishing in deep architectures. The cross-entropy loss corresponds to a strict generator with $\lambda = 0$ and $\lambda = 1$ in the SS and Frank families, respectively. This selection corresponds to a fast and stable converging solution when paired with Adam, while there are faster converging solutions when using a fixed learning rate.

Classification accuracy The effect of the selection of the generator on the classification accuracy is tested on a classification problem with manifold regularization. This learning task works in a transductive setting, where all the data is available at training time, even if only the training set supervisions are used during learning. In particular, the data is split into different datasets, where $\{10\%, 25\%, 50\%, 75\%, 90\%\}$ of the available data is used as a test set, while the remaining data is used as training set. The fitting of the supervised data defined by (7) is enforced for the training data during the learning process, whereas manifold regularization (6) can be enforced on all the available data. The Adam optimizer and the SS family of parametric t-norms have been employed in this experiment. Table 3 shows the average test accuracy and its standard deviation over 10 different samples of the train/test splits. As expected, all generator selections improve the final accuracy over what obtained by pure supervised learning, as manifold regularization brings relevant information to the learner.

Table 3 also shows the test accuracy when the parameter λ of the SS parametric family is selected from the grid $\{-1.5, -1, 0, 1, 1.5\}$, where values of $\lambda \leq 0$ move across strict t-norms (with $\lambda = 0$ being the product t-norm), and values greater than 0 move across nilpotent t-norms (with $\lambda = 1$ being the Łukasiewicz t-norm). Strict t-norms seem to provide slightly better performances than nilpotent ones on supervised tasks for the vast majority of the splits. However, this does not hold in manifold regularization learning tasks and a limited number of supervisions, where nilpotent t-norms perform better. An explanation of this behavior can be found in the different nature of the two constraints. Indeed, while supervisions provide hard constraints that need to be strongly satisfied, manifold regularization is a general soft rule, which should allow exceptions. When the number of supervision is small and manifold regularization drives the learning process, the milder behavior of nilpotent t-norms performs better, as it more closely models the semantics of the prior knowledge. Finally, it is worth noticing that very strict t-norms (e.g. $\lambda = -1.5$ in the considered experiment) provide higher standard deviations compared to other t-norms, especially in the manifold regularization setup. This provides some evidence of a

Table 3 Test accuracy of collective classification in a transductive setting on the Citeseer dataset for different percentages of available training data and different selections of the parameter λ of the SS generator family

% Test	λ	Supervised		Manifold	
		Avg Accuracy	Stddev	Avg Accuracy	Stddev
10%	−1.5	72.44	0.8	79.07	1.07
	−1.0	72.26	0.96	79.37	0.68
	0.0	71.63	0.74	79.37	0.84
	1.0	71.57	0.88	78.58	0.69
	1.5	71.93	1.11	77.77	0.89
25%	−1.5	72.22	0.46	77.17	0.70
	−1.0	72.02	0.52	77.51	0.72
	0.0	71.35	0.56	77.39	0.50
	1.0	71.22	0.47	77.36	0.64
	1.5	71.51	0.77	76.41	0.57
50%	−1.5	70.94	0.56	75.52	0.46
	−1.0	70.98	0.51	76.16	0.32
	0.0	70.49	0.52	75.71	0.39
	1.0	70.07	1.71	76.39	0.46
	1.5	70.09	0.47	75.97	0.55
75%	−1.5	67.06	0.58	72.25	0.50
	−1.0	66.96	0.44	72.48	0.50
	0.0	67.02	0.54	72.73	0.61
	1.0	66.34	0.29	73.77	0.34
	1.5	65.93	0.64	73.37	0.37
90%	−1.5	61.09	0.78	66.02	2.51
	−1.0	61.59	0.44	67.24	1.72
	0.0	61.52	0.33	68.60	0.75
	1.0	61.31	0.52	70.69	0.52
	1.5	61.17	0.84	70.32	0.89

Table 4 Comparison of the test accuracy on the Citeseer dataset obtained by content based and relational classifiers against supervised and relational learning expressed using DFL

Method	Classification Accuracy
Naive Bayes	74.87
ICA Naive Bayes	76.83
GS Naive Bayes	76.80
Logistic Regression	73.21
ICA Logistic Regression	77.32
GS Logistic Regression	76.99
Loopy Belief Propagation	77.59
Mean Field	77.32
NN	72.26
DFL	**79.37**

All reported results are computed as average over 10 random splits of the train and test data. The bold number indicates the best performer and a statistically significant improvement over the competitors

trade-off between the improved learning speed provided by strict t-norms and the introduced training instability due to their extremely non-linear behavior.

Competitive evaluation Table 4 compares the accuracy of the selected neural model (NN) trained only with the supervised constraint against other two content-based classifiers, namely logistic regression (LR) and Naive Bayes (NB). These baseline classifiers have been compared against collective classification approaches using the citation network data: Iterative Classification Algorithm (ICA) [47] and Gibbs Sampling (GS) [48] applied on top of the output of the LR and NB content-based classifiers.

Furthermore, the results are compared against the two top performers on this task: Loopy Belief Propagation (LBP) [49] and Relaxation Labeling through Mean-Field Approach (MF) [49]. Finally, the results of DFL were built by training the same neural network with both supervision and manifold regularization constraints, for which it was used a generator from the SS family with $\lambda = -1$. The accuracy values are obtained as an average over 10-folds created by random splits of 90% and 10% of the data for the train and test sets, respectively. Unlike the other relational approaches that can only be executed at inference time (collective classification), DFL can distill the knowledge in the weights of the neural network. The accuracy results are the highest among all the tested methodologies, in spite of the fact that the neural network trained only on the supervisions performs slightly worse than the other content-based competitors.

7 Discussion and practical implications

The presented framework can be contextualized among a new class of learning frameworks, which exploits the continuous relaxation of FOL to integrate logic knowledge in the learning process [5, 6, 21, 33].

Ease of design and numerical stability Previous frameworks in this class require an a-priori definition of the operators of a given t-norm fuzzy logic. On the other hand, the presented framework requires only the generator to be defined. This provides two main advantages: a minimum *design effort* and an improved *numerical stability*. Indeed, it is possible to apply the generator only on grounded atoms by exploiting the simplification property to apply the penalty function (generator) to the atoms, whereas all compositions are performed via stable operators (e.g. min,max,sum). On the contrary, the previous FOL relaxations correspond to an arbitrary mix of non-linear operators, which can potentially lead to numerically unstable implementations.

Tensor-based integration The presented framework provides a fundamental advantage in the integration with tensor-based machine learning frameworks like TensorFlow [43] or PyTorch [50]. Modern deep learning architectures can be effectively trained by leveraging tensor operations performed via Graphics Processing Units (GPU). However, this ability is conditioned on the possibility of concisely express the operators in terms of parallelizable operations like sums or products over n arguments, which are often implemented as atomic operations in GPU computing frameworks, without requiring to resort to slow iterative procedures. Fuzzy logic operators can not be easily generalized to their n-ary form. For example, the Łukasiewicz conjunction $T_L(x, y) = \max\{0, x + y - 1\}$ can be generalized to n-ary form as $T_L(x_1, x_2, \ldots, x_n) = \max\{0, \sum_{i=1}^{n}(x_i) - n + 1\}$. On the other hand, the general SS t-norm $T_\lambda^{SS}(x, y) = (x^\lambda + y^\lambda - 1)^{\frac{1}{\lambda}}$, with $-\infty < \lambda < 0$, does not have any (similarly simple) generalization and the implementation of the n-ary form must resort to an iterative application of the binary form, which is very inefficient in tensor-based computations. Previous frameworks like LTN and SBR had to limit the form of the formulas that can be expressed, or carefully select the t-norms in order to provide efficient n-ary implementations. However, the presented framework can express operators in n-ary form in terms of the generators. Thanks to the simplification property, n-ary operators for any continuous Archimedean t-norm can always be expressed as $T(x_1, x_2, \ldots, x_n) = g^{-1}(\min\{g(0^+), \sum_{i=1}^{n} g(x_i)\})$ in general, and $T(x_1, x_2, \ldots, x_n) = g^{-1}(\sum_{i=1}^{n} g(x_i))$ if T is strict.

Limitations Linking the loss function to the desired fuzzy semantics via the single choice of the t-norm generator guarantees logic coherence and simplification properties, but does not guarantee to achieve the highest accuracy for a given task. Another limitation of this approach is that it may not be directly applicable to neural-symbolic models not relaxing the Boolean formulas using t-norm fuzzy logic operators.

8 Conclusions

This paper presents a framework to embed prior knowledge expressed as logic statements into a learning task yielding several important contributions. First, we showed how human knowledge in the form of logical rules can be translated into differentiable loss functions used during learning. A critical aspect of our approach is that the translation from logic formulas to loss functions is uniquely defined by the choice of a unique operator, i.e. the generator of the corresponding t-norm. This feature clearly distinguishes our approach from the majority of related methods, which are often based on multiple specific choices for each of the fuzzy operators. Second, we have shown that the classical loss functions for supervised learning are naturally recovered within the theory, and that the use of parametric t-norm generators allows the definition of entire classes of loss functions with different convergence properties. The choice of the parameter can therefore be guided by the requirements of the specific applications. Third, the presented theory has driven to the implementation of a general software simulator, called Deep Fuzzy Logic (DFL), which bridges logic reasoning and deep learning using the unifying concept of t-norm generator, as general abstraction to translate any FOL declarative knowledge into an optimization problem solved in TensorFlow. Finally, we designed and implemented multiple experiments in DFL which show how the proposed method allows the definition of new loss functions with better performances both in terms of accuracy and training efficiency. Furthermore, by being able to incorporate logical knowledge seamlessly, our method outperforms several related works on the task of document classification in citation networks.

As future work, we plan to extend the method by allowing the learning of the parameters of the t-norm generator from data. In this regard, casting what presented in this paper within a Bayesian framework [24] is likely a promising direction. Furthermore, we plan to expand the range of applications of DFL to domains like visual question answering [51] and structure learning [3].

Funding This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 825619. This work was also supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215. Giuseppe Marra is funded by Research Foundation-Flanders (FWO-Vlaanderen, 1239422N). This project has received funding from the European Union's Horizon-MSCA-2021 research and innovation program under grant agreement No 101073307.

Declarations

Conflict of Interests The authors declare that they have no competing interest.

References

1. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436
2. Selbst A, Powles J (2018) meaningful information and the right to explanation. In: Conference on fairness, accountability and transparency. PMLR, pp 48–48
3. De Raedt L, Dumančić S, Manhaeve R, Marra G (2021) From statistical relational to neural-symbolic artificial intelligence. In: Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence, pp 4943–4950
4. Garcez A, Gori M, Lamb L, Serafini L, Spranger M, Tran S (2019) Neural-symbolic computing: an effective methodology for principled integration of machine learning and reasoning. Journal of Applied Logics 6(4):611–631
5. Diligenti M, Gori M, Sacca C (2017) Semantic-based regularization for learning and inference. Artif Intell 244:143–165
6. Badreddine S, Garcez AD, Serafini L, Spranger M (2022) Logic tensor networks. Artif Intell 303:103649
7. Goodfellow I, Bengio Y, Courville A (2016) Deep learning
8. Giannini F, Marra G, Diligenti M, Maggini M, Gori M (2019) On the relation between loss functions and t-norms. In: Proceedings of the conference on inductive logic programming (ILP)
9. Garcez AD, Bader S, Bowman H, Lamb LC, De Penning L, Illuminoo B, Poon H, Gerson Zaverucha C (2022) Neural-symbolic learning and reasoning: a survey and interpretation. Neuro-Symbolic Artificial Intelligence: The State of the Art 342:1
10. Hitzler P (2022) Neuro-symbolic artificial intelligence: the state of the art
11. Raedt LD, Kersting K, Natarajan S, Poole D (2016) Statistical relational artificial intelligence: logic, probability, and computation. Synthesis Lectures on Artificial Intelligence and Machine Learning 10(2):1–189
12. Richardson M, Domingos P (2006) Markov logic networks. Mach Learn 62(1):107–136
13. Bach SH, Broecheler M, Huang B, Getoor L (2017) Hinge-loss markov random fields and probabilistic soft logic. J Mach Learn Res 18:1–67
14. Niu F, Ré C, Doan A, Shavlik J (2011) Tuffy: scaling up statistical inference in markov logic networks using an rdbms. Proceedings of the VLDB Endowment 4(6)
15. Chekol MW, Huber J, Meilicke C, Stuckenschmidt H (2016) Markov logic networks with numerical constraints. In: Proceedings of the twenty-second european conference on artificial intelligence, pp 1017–1025
16. Qu M, Bengio Y, Tang J (2019) Gmnn: graph markov neural networks. In: International conference on machine learning. PMLR, pp 5241–5250
17. Khot T, Balasubramanian N, Gribkoff E, Sabharwal A, Clark P, Etzioni O (2015) Exploring markov logic networks for question answering. In: Proceedings of the 2015 conference on empirical methods in natural language processing, pp 685–694
18. Gayathri K, Easwarakumar K, Elias S (2017) Probabilistic ontology based activity recognition in smart homes using markov logic network. Knowl-Based Syst 121:173–184
19. Marra G, Kuželka O (2021) Neural markov logic networks. In: Uncertainty in artificial intelligence. PMLR, pp 908–917
20. Diligenti M, Giannini F, Gori M, Maggini M, Marra G (2021) A constraint-based approach to learning and reasoning. In: Neuro-symbolic artificial intelligence: the state of the art, pp 192–213
21. Marra G, Giannini F, Diligenti M, Gori M (2019) Lyrics: a general interface layer to integrate logic inference and deep learning. In: Proceedings of the joint european conference on machine learning and knowledge discovery in databases (ECML/PKDD)
22. Xu J, Zhang Z, Friedman T, Liang Y, Broeck G (2018) A semantic loss function for deep learning with symbolic knowledge. In: International conference on machine learning. PMLR, pp 5502–5511
23. van Krieken E, Acar E, van Harmelen F (2019) Semi-supervised learning using differentiable reasoning. Journal of Applied Logics—IfCoLog Journal of Logics and their Applications 6(4)
24. Marra G, Giannini F, Diligenti M, Gori M (2019) Integrating learning and reasoning with deep logic models. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 517–532
25. Marra G, Diligenti M, Giannini F, Gori M, Maggini M (2020) Relational neural machines. In: Proceedings of the European conference on artificial intelligence (ECAI)
26. Manhaeve R, Dumancic S, Kimmig A, Demeester T, De Raedt L (2018) Deepproblog: neural probabilistic logic programming. Adv Neural Inf Process Syst 31
27. Sourek G, Aschenbrenner V, Zelezny F, Schockaert S, Kuzelka O (2018) Lifted relational neural networks: efficient learning of latent relational structures. J Artif Intell Res 62:69–100
28. Rocktäschel T, Riedel S (2017) End-to-end differentiable proving. In: Advances in neural information processing systems, pp 3788–3800
29. Minervini P, Riedel S, Stenetorp P, Grefenstette E, Rocktäschel T (2020) Learning reasoning strategies in end-to-end differentiable proving. In: ICML
30. Serafini L, Donadello I, Garcez AD (2017) Learning and reasoning in logic tensor networks: theory and application to semantic image interpretation. In: Proceedings of the symposium on applied computing. ACM, pp 125–130
31. Giannini F, Diligenti M, Gori M, Maggini M (2018) On a convex logic fragment for learning and reasoning. IEEE Transactions on Fuzzy Systems
32. van Krieken E, Acar E, van Harmelen F (2020) Analyzing differentiable fuzzy implications. In: KR2020: Proceedings of the 17th Conference on Principles of Knowledge Representation and Reasoning. Rhodes, Greece. September 12–18, 2020. IJCAI Organization, pp 893–903
33. van Krieken E, Acar E, van Harmelen F (2022) Analyzing differentiable fuzzy logic operators. Artif Intell 302:103602
34. Donadello I, Serafini L, d'Avila Garcez A (2017) Logic tensor networks for semantic image interpretation. In: IJCAI International joint conference on artificial intelligence, pp 1596–1602
35. Klement EP, Mesiar R, Pap E (2013) Triangular norms 8
36. Hájek P. (2013) Metamathematics of Fuzzy Logic 4
37. Jenei S (2002) A note on the ordinal sum theorem and its consequence for the construction of triangular norms. Fuzzy Sets Syst 126(2):199–205
38. Mizumoto M (1989) Pictorial representations of fuzzy connectives, part i: cases of t-norms, t-conorms and averaging operators. Fuzzy Sets Syst 31(2):217–242
39. Marra G, Giannini F, Diligenti M, Gori M (2019) Constraint-based visual generation. In: International conference on artificial neural networks. Springer, pp 565–577
40. Diligenti M, Roychowdhury S, Gori M (2018) Image classification using deep learning and prior knowledge. In: Proceedings of third international workshop on declarative learning based programming (DeLBP)
41. Novák V., Perfilieva I, Mockor J (2012) Mathematical Principles of Fuzzy Logic 517
42. Kolb S, Teso S, Passerini A, De Raedt L (2018) Learning smt (lra) constraints using smt solvers. In: IJCAI, pp 2333–2340

43. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al (2016) Tensorflow: a system for large-scale machine learning. In: OSDI, vol 16, pp 265–283
44. Fakhraei S, Foulds J, Shashanka M, Getoor L (2015) Collective spammer detection in evolving multi-relational social networks. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '15, pp 1769–1778, https://doi.org/10.1145/2783258.2788606
45. Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. arXiv:1412.6980
46. Belkin M, Niyogi P, Sindhwani V (2006) Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. J Mach Learn Res 7(Nov):2399–2434
47. Neville J, Jensen D (2000) Iterative classification in relational data. In: Proc. AAAI-2000 workshop on learning statistical models from relational data, pp 13–20
48. Lu Q, Getoor L (2003) Link-based classification. In: Proceedings of the 20th international conference on machine learning (ICML-03), pp 496–503
49. Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93
50. Ketkar N (2017) Introduction to pytorch. In: Deep learning with python, pp 195–208
51. Yi K, Wu J, Gan C, Torralba A, Kohli P, Tenenbaum JB (2018) Neural-Symbolic VQA: disentangling reasoning from vision and language understanding. In: Advances in neural information processing systems (NIPS)

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Francesco Giannini[1] · **Michelangelo Diligenti**[2] · **Marco Maggini**[2] · **Marco Gori**[2,3] · **Giuseppe Marra**[4]

Michelangelo Diligenti
michelangelo.diligenti@unisi.it

Marco Maggini
marco.maggini@unisi.it

Marco Gori
marco.gori@unisi.it

Giuseppe Marra
giuseppe.marra@kuleuven.be

[1] Consorzio Interuniversitario Nazionale per l'Informatica, CINI, Roma (Rome), Italy

[2] Department of Information Engineering and Science, University of Siena, Siena, Italy

[3] Maasai, Inria, I3S, CNRS, Universitê Côte d'Azur, Nice, France

[4] Department of Computer Science, KU Leuven, Leuven, Belgium