The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Characterization of the Convex Lukasiewicz Fragment for Learning from Constraints

Francesco Giannini, Michelangelo Diligenti, Marco Gori, Marco Maggini
Department of Information Engineering and Mathematical Sciences
University of Siena
Siena, via Roma 56, Italy
{fgiannini,diligmic,marco,maggini}@diism.unisi.it

Abstract

This paper provides a theoretical insight for the integration of logical constraints into a learning process. In particular it is proved that a fragment of the Łukasiewicz logic yields a set of convex constraints. The fragment is enough expressive to include many formulas of interest such as Horn clauses. Using the isomorphism of Łukasiewicz formulas and McNaughton functions, logical constraints are mapped to a set of linear constraints once the predicates are grounded on a given sample set. In this framework, it is shown how a collective classification scheme can be formulated as a quadratic programming problem, but the presented theory can be exploited in general to embed logical constraints into a learning process. The proposed approach is evaluated on a classification task to show how the use of the logical rules can be effective to improve the accuracy of a trained classifier.

1 Introduction

The concept of *constraint* has been considered in the definition of a general approach for learning that allows us to exploit different sources of knowledge for training an agent (Gnecco et al. 2015). In particular, in multi-task learning beside the supervised examples additional information on the task can be expressed as abstract knowledge by logical constraints. Some approaches to embed logical constraints in the learning procedure have been proposed, especially in the context of kernel machines (Diligenti et al. 2012; Diligenti, Gori, and Saccà 2015). These methods exploit the transcription of logical rules into real valued functions to define appropriate cost functions to be optimized in the learning process. Fuzzy logic is generally used in defining the mapping since it allows the use of continuous values in the unit interval (Diligenti et al. 2012; Serafini and d'Avila Garcez 2016). Related approaches combine logic rules with neural network learning (Hu et al. 2016) by an iterative procedure that transfers abstract knowledge encoded by the logic rules into the parameters of a deep neural network. It is also worth to mention the related methods in *statistical relational learning* (Getoor and Taskar 2007; De Raedt and Kersting 2008), like *Markov logic networks* (Richardson and Domingos 2006) that generate probabilistic graphical models from the rule set. Nevertheless, in such approaches logical formulas are assumed to be evaluated as true-false values, whereas a more general expressiveness is achieved in *Probabilistic Soft Logic* (PSL) where formulas can take any $[0,1]$ value. *PSL* (Bach 2015) is a probabilistic programming language that allows to consider disjunctive logic clauses (including Horn clauses as a special case) converted into real-valued functions by means of Łukasiewicz logic operators, where the resulting cost function is shown to be convex.

In this paper we provide a theoretical result stating that logic rules expressed in a specific propositional Łukasiewicz fragment correspond to convex logical constraints that can be exploited to train a learning agent. The fragment contains the *weak* conjunction and the *strong* disjunction applied to literals. By the McNaughton theorem, each constraint function corresponding to any formula in the fragment is a convex piecewise linear function with integer coefficients. Given the structure of the McNaughton functions for this specific fragment, we show that the constraint functions can be implemented as a set of linear constraints in the variable space. Given this result, we show how a collective classification task can be directly formulated as a quadratic programming problem. Finally, we provide an example to show the effect of the logical rules to improve the classification performance on a simple benchmark. It is interesting to notice that the theoretical result regarding the convex fragment is very general and it is not limited to the proposed application. For example, it can be used to extend the fragment of logic that can be represented in PSL, while preserving convexity.

The paper is organized as follows. Section 2 reports the theoretical results characterizing the Łukasiewicz fragment that yields convex logical constraints. Then in Section 3 it is shown how the theory can be applied to formulate the collective classification task as a quadratic programming problem. Section 4 provides an applicative example showing the effect of rules expressed by the proposed fragment in a transductive classification task. Finally, some conclusions are drawn in Section 5.

2 Convex Łukasiewicz Fragment

In a multi–task learning problem, beyond factual knowledge, we can express abstract relations among the objective functions by means of logical constraints. In principle they can be expressed in any logic, however we are especially interested in logics whose formulas have a suitable functional representation. By the McNaughton Theorem we know that, for the propositional case, the algebra of formulas of Łukasiewicz Logic **Ł** on n variables is isomorphic to the algebra of McNaughton functions defined on $[0,1]^n$. Moreover **Ł** has an involutive negation (i.e. $\neg\neg x = x$) and every **Ł**-formula has an equivalent *prenex normal form* (i.e. quantifiers followed by a quantifier–free part).

Syntax of propositional **Ł**-formulas is defined in the usual way (see i.e. (Novák, Perfilieva, and Močkoř)). We recall that, for the propositional case, **Ł** is sound and complete with respect to the variety of **MV**-algebras that is generated by the standard algebra $[0,1]_{\text{Ł}} = ([0,1], 0, 1, \neg, \wedge, \vee, \otimes, \oplus, \rightarrow)$[1] whose operations, for every $x, y \in [0,1]$, are defined as:

$$x \wedge y = \min\{x, y\}, \quad x \otimes y = \max\{0, x + y - 1\},$$
$$x \vee y = \max\{x, y\}, \quad x \oplus y = \min\{1, x + y\},$$
$$\neg x = 1 - x, \quad x \rightarrow y = \min\{1, 1 - x + y\}.$$

The operations $\wedge$ and $\otimes$ are the interpretations of the *weak* and *strong* conjunction[2] of **Ł** respectively, while $\vee$ and $\oplus$ represent the *weak* and *strong* disjunction. Further, $\otimes$ and $\rightarrow$ are the Łukasiewicz t-norm and its residuum. We also note that $x \rightarrow y$ is definable as $\neg x \oplus y$ and we will write $\bar{0}$ ($\bar{1}$) for the formula corresponding to the strong conjunction (disjunction) of any literal with its negation.

Some fundamental properties, holding among **MV**-algebras operations, are the *De Morgan laws* between both weak and strong operations, as well as the *Distributive laws* of strong over weak operations. We only notice that the distributive property does not hold between strong conjunction and strong disjunction.

Since $[0,1]_{\text{Ł}}$ generates the whole **MV**-variety, the algebra of **Ł**-formulas on n variables is isomorphic to the algebra of functions from $[0,1]^n$ to $[0,1]$ constructible from the projections by means of pointwise defined operations. In particular, the zero of the algebra is the constant function equal to 0 and, given $\circ \in \{\wedge, \vee, \otimes, \oplus, \rightarrow\}$, for every $(t_1, \ldots, t_n) \in [0,1]^n$:

$$\neg f(t_1, \ldots, t_n) = 1 - f(t_1, \ldots, t_n),$$
$$(f \circ g)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n) \circ g(t_1, \ldots, t_n).$$

In addition, we know exactly which kind of functions corresponds to Łukasiewicz formulas. The result is expressed by the the well–known McNaughton Theorem.

Definition 1. *Let $f : [0,1]^n \rightarrow [0,1]$ be a continuous function with $n \geq 0$, f is called a McNaughton function if it is piecewise linear with integer coefficients, that is, there exists a finite set of linear polynomials $p_1, \ldots, p_m$ with integer coefficients such that for all $(t_1, \ldots, t_n) \in [0,1]^n$, there exists $i \leq m$ such that*

$$f(t_1, \ldots, t_n) = p_i(t_1, \ldots, t_n).$$

[1]Through the paper we use the same symbols for connectives and algebraic operations.

[2]The difference can be explained as follows: $\alpha \wedge \beta$ gives us the availability of both α and β but we can peak just one of them; $\alpha \otimes \beta$ forces us to peak both formulas in the pair (α, β).

Theorem 1 (McNaughton Theorem)**.** *For each $n \geq 0$, the class of $[0,1]$-valued functions defined on $[0,1]^n$ that correspond to formulas of propositional Łukasiewicz logic coincides with the class of McNaughton functions defined on $[0,1]^n$ and equipped with pointwise defined operations.*

As a consequence, for every **Ł**-formula φ depending on n propositional variables, we can consider its corresponding function $f_\varphi : [0,1]^n \rightarrow [0,1]$, whose value on each point is exactly the evaluation of the formula with respect to the same variable assignment. For the McNaughton Theorem f_φ is a McNaughton function.

About Convexity

Because of the functional representation of formulas, we have a natural way to consider logical constraints in a learning setting. Nevertheless, in general such constraints are not convex, hence we are interested in operations (corresponding to logical connectives) that preserve concavity or convexity.

Lemma 1. *Let f, g be two $[0,1]$-valued functions defined on $[0,1]^n$, then*

1. *f is convex if and only if the function $\neg f$ is concave;*
2. *if f, g are concave then the functions $f \wedge g$ and $f \oplus g$ are concave;*
3. *if f, g are convex then the functions $f \vee g$ and $f \otimes g$ are convex.*

As a result we get that, if f is a convex function and g is concave, then $f \rightarrow g = \neg f \oplus g$ is concave.

Let us consider the two different Łukasiewicz fragments $(\wedge, \oplus)^*$ and $(\otimes, \vee)^*$ of concave and convex McNaughton functions respectively, they are defined as follows.

Definition 2. *Let $(\wedge, \oplus)^*$ and $(\otimes, \vee)^*$ be the smallest sets of formulas (up to equivalence) such that:*

- *if y is a propositional variable, then $\bar{0}, y, \neg y \in (\wedge, \oplus)^*$ and $\bar{1}, y, \neg y \in (\otimes, \vee)^*$;*
- *if $\varphi_1, \varphi_2 \in (\wedge, \oplus)^*$, then $\varphi_1 \wedge \varphi_2$, $\varphi_1 \oplus \varphi_2 \in (\wedge, \oplus)^*$;*
- *if $\varphi_1, \varphi_2 \in (\otimes, \vee)^*$, then $\varphi_1 \otimes \varphi_2$, $\varphi_1 \vee \varphi_2 \in (\otimes, \vee)^*$.*

As a consequence of Lemma 1 and the previous definition, the set of Horn clauses, namely the set of formulas of the form $(x_1 \otimes \ldots \otimes x_m) \rightarrow y$ with $x_1, \ldots, x_m, y$ propositional variables, is (properly) contained in $(\wedge, \oplus)^*$.

Since we are interested in convex McNaughton functions, which are among others piecewise linear functions, the following result (see i.e. (Rockafellar and Wets 2009)) turns out to be fundamental for our investigation.

Theorem 2. *Any convex piecewise linear function on $\mathbb{R}^n$ can be expressed as a max of a finite number of affine functions.*

This means that, for each n-ary convex McNaughton function f there exist $A_1, \ldots, A_k \in \mathbb{Z}^n$, $b_1, \ldots, b_k \in \mathbb{Z}$ such that:

$$\text{for all } x \in [0,1]^n \qquad f(x) = \max_{i=1}^{k} A_i' \cdot x + b_i. \qquad (1)$$

On the other hand, every concave McNaughton function can be expressed as the minimum of a finite number of affine functions. We only mention that the coefficients of the affine functions are constructively determined by the shape of the considered formula.

Example. Let us consider $\varphi = ((x \wedge y) \oplus \neg y \oplus z) \wedge \neg z$, then $\varphi \in (\wedge, \oplus)^*$ and it is equivalent to $(x \oplus \neg y \oplus z) \wedge (y \oplus \neg y \oplus z) \wedge \neg z$. From this latter expression, we get:

$$f_\varphi(x, y, z) = \min\{1, x - y + z + 1, 1 - z\}.$$

Finally, exploiting (1) and thanks to Lemma 1, we can prove that the defined fragments of Łukasiewicz logic coincide with the sets of all **Ł**-formulas whose corresponding McNaughton functions are either concave or convex.

Proposition 1. *Let $f_\varphi : [0,1]^n \to [0,1]$ be a McNaughton function. Then f_φ is concave if and only if $\varphi \in (\wedge, \oplus)^*$ and f_φ is convex if and only if $\varphi \in (\otimes, \vee)^*$.*

This result allows us to conclude that, in each case, we are taking into account the largest fragment of Łukasiewicz logic whose McNaughton functions are either concave or convex.

3 Collective Classification with Logical Constraints

We consider a learning problem in which we consider a set of learnable task functions corresponding to predicates in $\mathbf{P} = \{p_j^{(a_j)} : j \in \mathbb{N}_J\}$[3], each with its own fixed arity a_j. In general, we may assume that each predicate is defined on a subset of a power of $\mathbb{R}$, such that $p_j^{(a_j)} : \mathcal{R}_{j_1} \times \ldots \times \mathcal{R}_{j_{a_j}} = \mathcal{R}_{n_j} \to \mathbb{R}$, where for $k = 1, \ldots, a_j$, $\mathcal{R}_{j_k} \subseteq \mathbb{R}^{j_k}$ and so $\mathcal{R}_{n_j} \subseteq \mathbb{R}^{n_j}$, $n_j = n_{j_1} + \ldots + n_{j_{a_j}}$. However, we assume that the predicates are evaluated on finite, already fixed, sets of sample points. In other words, we consider a *collective* learning setting in which we are only interested to compute the values of the predicates on the available discrete set of points. We may assume that an appropriate model (f.i. a neural network or a kernel machine) has already been trained[4] to compute the prior values of $p_j(\mathbf{x}_l^j)$ for a given input $\mathbf{x}_l^j \in \mathbb{R}^{n_j}$. If we suppose that each variable occurring in predicates takes its values from a discrete set $\mathscr{S}_{j_k}$, in general we can consider the Cartesian product of the sets $\mathscr{S}_{j_k}$ as the evaluation set $\mathscr{S}_j$ for the j-th task,

$$\mathscr{S}_j = \{\mathbf{x}_s^j = (\mathbf{x}_1^j, \ldots, \mathbf{x}_{a_j}^j) \in \mathbb{R}^{n_j} : \mathbf{x}_k^j \in \mathscr{S}_{j_k}, s \in \mathbb{N}_{s_j}\},$$

whose elements will be used to evaluate p_j. The value of the predicate on the s-th sample point in $\mathscr{S}_j$ will be denoted as $p_{js} = p_j(\mathbf{x}_s^j)$, and the values will be collected into the vector $\mathbf{p}_j = [p_{j1}, \ldots, p_{js_j}] \in \mathbb{R}^{s_j}$. Finally, in the following we will indicate as $\hat{\mathbf{p}}_j$ as the available vector of priors for the predicate values.

Let $KB = \{\varphi_h : h \in \mathbb{N}_H\}$ be a knowledge base of first order Łukasiewicz formulas defined on the set of predicates $\mathbf{P}$. KB represents a set of *logical constraints* that establish some relations that must hold between the predicates. Since we are dealing with a finite domain for variables, each quantified formula can be replaced with a propositional one applying the following equivalence once per quantifier:

$$\forall x\, \psi(x) \simeq \bigwedge_{\mathbf{x}_k^j \in \mathscr{S}_{j_k}} \psi[\mathbf{x}_k^j/x],\ \exists x\, \psi(x) \simeq \bigvee_{\mathbf{x}_k^j \in \mathscr{S}_{j_k}} \psi[\mathbf{x}_k^j/x], \tag{2}$$

where x is the k-th argument of a certain predicate p_j occurring in the formula ψ and $[\mathbf{x}_k^j/x]$ represents the substitution of the element $\mathbf{x}_k^j$ in place of x in ψ. If two or more predicates share a variable, in possibly different arguments, then that variable will be evaluated on the same set of values for those predicates. By applying (2) to remove all the quantifiers, we obtain a set of formulas KB', where the propositional variables correspond to the possible groundings of predicates occurring in formulas. Each grounding of the j-th predicate is a value corresponding to a specific entry in the vector $\mathbf{p}_j$. Hence, in this setting, these values can be thought of as (propositional) variables. Clearly, when two (or more) predicates p_{j_1} and p_{j_2} share the same quantified variable as argument, the variables $p_{j_1 s_1}$ and $p_{j_2 s_2}$ corresponding to their groundings have to be selected accordingly to the shared values in their arguments.

Example. Let us consider $\mathbf{P} = \{p_1^{(1)}, p_2^{(2)}\}$ with the logical constraint

$$\varphi : \forall x \exists y (p_1(x) \to p_2(x, y)).$$

Now let us suppose $\mathscr{S}_{1_1} = \mathscr{S}_{2_1} = \{a, b\}$ and $\mathscr{S}_{2_2} = \{c, d\}$, then φ' corresponds to

$$[(p_1(a) \to p_2(a, c)) \vee (p_1(a) \to p_2(a, d))] \wedge$$
$$\wedge [(p_1(b) \to p_2(b, c)) \vee (p_1(b) \to p_2(b, d))]$$

If we consider the grounding vectors for the two predicates

$$\mathbf{p}_1 = [p_1(a), p_1(b)],$$
$$\mathbf{p}_2 = [p_2(a, c), p_2(a, d), p_2(b, c), p_2(b, d)]$$

φ' can be rewritten as

$$[(p_{11} \to p_{21}) \vee (p_{11} \to p_{22})] \wedge [(p_{12} \to p_{23}) \vee (p_{12} \to p_{24})].$$

Every $n-$ary formula φ in KB' is expressed in the context of propositional Łukasiewicz Logic **Ł** and, hence, it is isomorphic to a McNaughton function $f_\varphi : [0,1]^n \to [0,1]$. For the sake of simplicity, we write f_h for the function corresponding to the formula φ'_h. Moreover, each formula φ'_h in KB' actually depends only on the groundings of the occurring predicates, but to use a uniform notation, we will write every formula as depending on all groundings of all predicates, i.e. $f_{\varphi'}(\mathbf{p})$ being $\mathbf{p} = [\mathbf{p}_1, \ldots, \mathbf{p}_J] \in \mathbb{R}^S$, where $S = s_1 + \ldots + s_J$. We can enforce the satisfaction of logical constraints in KB' by requiring $1 - f_h(\mathbf{p}) = 0$, for all $h = 1, \ldots, H$. In order to allow soft violations of these constraints, we can introduce a set of slack variables obtaining the following contraints:

$$1 - f_h(\mathbf{p}) \leq \xi_h \text{ with } \xi_h \geq 0 \quad \text{for } h \in \mathbb{N}_H. \tag{3}$$

[3]Where, for every $j \in \mathbb{N}$, $\mathbb{N}_j = \{n \in \mathbb{N} : n \leq j\}$.

[4]Even if in the present work we restrict the attention on the collective classification setting, the proposed framework can be exploited also to define a learning scheme for the predicate function themselves.

If f_h is a concave function, then $g_h = 1 - f_h$ is a convex one that corresponds to the formula $\neg\varphi'_h$. Hence in this case, $g_h(\mathbf{p}) \leq \xi_h$ is a convex constraint. When we restrict to formulas in the concave Łukasiewicz fragment defined in the previous section, we obtain constraints defined by convex McNaughton functions that can be written according to (1) as the maximum of affine functions:

$$g_h(\mathbf{p}) = 1 - f_h(\mathbf{p}) = \max_{i \in \mathbb{N}_{I_h}} M_i^h \cdot \mathbf{p} + q_i^h \leq \xi_h \quad \text{for } h \in \mathbb{N}_H,$$

where $M_i^h \in \mathbb{R}^{1,S}$ and $q_i^h \in \mathbb{R}$ are integer coefficients determined by the shape of the formula $\neg\varphi'_h$. Now, since $g_h(\mathbf{p})$ is a *max* of terms, we have $g_h(\mathbf{p}) \leq \xi_h$ if and only if

$$M_i^h \cdot \mathbf{p} + q_i^h \leq \xi_h \quad \text{for all } i \in \mathbb{N}_{I_h}. \qquad (4)$$

This means for every $h \in \mathbb{N}_H$, we can consider the I_h linear constraints expressed by (4) in place of (3). Even if the number of constraints is increased, each of them is now an affine function, so we can deal with a quadratic programming problem.

In order to guarantee the *consistency* in the definition of the logical constraints, we need to limit the values of the predicate groundings to be into the interval $[0, 1]$. This simply requires to add the following hard constraints:

$$0 \leq p_{js} \leq 1, \quad \text{for } j \in \mathbb{N}_J,\ s \in \mathbb{N}_{s_j}\ . \qquad (5)$$

Given this setting, the considered multi–task learning problem is formulated as

$$\min_{\mathbf{p}} \sum_{j \in \mathbb{N}_J} ||\mathbf{p}_j - \hat{\mathbf{p}}_j||^2 + C_1 \sum_{h \in \mathbb{N}_H} \xi_h \qquad \text{subject to}$$

$$\begin{array}{llll} 1 - f_h(\mathbf{p}) & \leq \xi_h & \xi_h \geq 0 & h \in \mathbb{N}_H \\ 0 \leq p_{js} & \leq 1 & & j \in \mathbb{N}_J,\ s \in \mathbb{N}_{s_j}\ , \end{array}$$

where $C_1 > 0$ is used to weigh the degree of satisfaction of the constraints. This optimization problem aims at finding the grounding for the predicates closest to the given priors and yielding the minimal violation of the logical constraints in KB. When dealing with formulas in the considered Łukasiewicz fragment, by using the rewriting of equation (4), we obtain a quadratic programming problem that can be efficiently solved.

Among the considered logical constraints, we can include formulas implementing a manifold regularization effect in the original feature space (Diligenti, Gori, and Saccà 2015). In fact, the topological properties of the original domains of the predicates are not explicitly represented in the considered setting, a part from the values assigned for the given priors. If we consider the predicate $p_j(\mathbf{x}^j)$ and we know that two groundings $p_{js_1} = p_j(\mathbf{x}^j_{s_1})$ and $p_{js_2} = p_j(\mathbf{x}^j_{s_2})$ are computed on two samples belonging to a given manifold, then we can express this fact by a given predicate $M_{js_1s_2}$ whose value can range in $[0, 1]$ to weigh the membership of the two points to the same manifold. For instance, when dealing with a spatial regularization manifold we can exploit the common definition $M_{js_1s_2} = e^{-\frac{||\mathbf{x}^j_{s_1} - \mathbf{x}^j_{s_2}||^2}{\sigma^2}}$, for a given value for the neighborhood width parameter σ. The logical constraint that imposes the coherency of the predicate evaluations on the manifold is expressed by

$$M_{js_1s_2} \rightarrow ((p_{js_1} \rightarrow p_{js_2}) \wedge (p_{js_2} \rightarrow p_{js_1}))$$

that can be rewritten as

$$\neg M_{js_1s_2} \oplus [(\neg p_{js_1} \oplus p_{js_2}) \wedge (\neg p_{js_2} \oplus p_{js_1})]\ .$$

This formula corresponds to the following linear constraints for each grounding:

$$\begin{array}{ll} M_{js_1s_2} + p_{js_1} - p_{js_2} - 1 & \leq \xi_{h_1}, \\ M_{js_1s_2} - p_{js_1} + p_{js_2} - 1 & \leq \xi_{h_2}. \end{array}$$

4 Experimental Results

The experimental analysis was carried out on an image classification task. The classification problem is based on the original benchmark proposed by P.Winston (Winston and Horn 1986), which was initially designed to show the ability of logic programming to determine the class of an animal from some partial clues. The dataset used in this paper is composed by 3325 images, taken from ImageNet database equally divided into 7 categories, each one representing an animal class: albatross, cheetah, giraffe, ostrich, penguin, tiger, zebra. The images of each animal present in the dataset occur at different distances, angles and poses. All images used in these experiments have size 32×32 pixels. The vector of numbers used to represent each image is composed by two parts: one representing the colors in the image, and one representing its shape. In particular, the feature representation contains a 12-dimension normalized color histogram for each channel in the RGB color space. Furthermore, the SIFT descriptors (Lowe 1999) have been built by sampling a set of images from the dataset and then detecting all the SIFT representations present in at least one of the sampled images. Finally, the SIFT representations have been clustered into 600 *visual words*. The final representation of an image contains 600 values, where the i-th element represents the normalized count of the i-th visual word for the given image (bag-of-descriptors). As previously done in Diligenti et al.(Diligenti, Gori, and Scoca 2016), the test phase does not get as input a sufficient set of clues to perform classification, but the image representations are used by the learning framework to develop the intermediate clues over which to perform inference.

Assuming to be in a transductive context, all images are available at training time but only a subset of the supervisions are available. In particular, in the experiments the amount of training supervisions is varied between 10% and 90%.

The knowledge domain is expressed in terms of first order logic rules as shown in Table 1, where 7 of the predicates in the KB correspond to the final animal classes, and the others are intermediate predicates that help in determining the final classes during the inference process performed by collective classification. For example, *hair*, *mammal*, *milk*, *carnivore*, etc. are intermediate predicates whereas *cheetah* is a final predicate representing a final class in the multi-task classification.

A feedforward neural network having one single output neuron and a single hidden layer containing 30 neurons was

(carnivore(x) ∧ tawny(x) ∧ blackstripes(x)) → tiger(x)
(mammal(x) ∧ cud(x)) → ungulate(x)
(bird(x) ∧ swim(x) ∧ blackwhite(x)) → penguin(x)
layeggs(x) → bird(x)
(mammal(x) ∧ pointedteeth(x) ∧ claws(x) ∧ forwardeyes(x)) → carnivore(x)
(mammal(x) ∧ meat(x)) → carnivore(x)
(ungulate(x) ∧ white(x) ∧ blackstripes(x)) → zebra(x)
(mammal(x) ∧ hoofs(x)) → ungulate(x)
(carnivore(x) ∧ tawny(x) ∧ darkspots(x)) → cheetah(x)
hair(x) → mammal(x)
(ungulate(x) ∧ longlegs(x) ∧ longneck(x) ∧ tawny(x) ∧ darkspots(x)) → giraffe(x)
milk(x) → mammal(x)
(bird(x) ∧ longlegs(x) ∧ longneck(x) ∧ black(x)) → ostrich(x)
feather(x) → bird(x)
(bird(x) ∧ goodflier(x)) → albatross(x)
mammal(x) ⊕ bird(x)
cheetah(x)⊕tiger(x)⊕giraffe(x)⊕zebra(x)⊕ostrich(x)⊕penguin(x)⊕albatross(x)
hair(x) ⊕ feather(x)
darkspots(x) → ¬ blackstripes(x)
blackstripes(x) → ¬ darkspots(x)
tawny(x) → (¬ black(x) ∧¬ white(x))
black(x) → (¬ tawny(x) ∧¬ white(x))
white(x) → (¬ black(x) ∧¬ tawny(x))
black(x) → ¬ white(x)
black(x) → ¬ tawny(x)
white(x) → ¬ black(x)
white(x) → ¬ tawny(x)
tawny(x) → ¬ white(x)
tawny(x) → ¬ black(x)

Table 1: Examples of the rules in the KB used for training the models in the Winston image classification benchmark.

trained for each final or intermediate predicate in the KB. The single output neuron used a sigmoidal activation function, while the hidden neurons used a rectified linear activation function. The networks are trained against the training set labels using a quadratic cost function on the output. Resilient backpropagation (Riedmiller and Braun 1993; Mosca and Magoulas 2015) was used to accelerate the convergence of the training process, which was executed for 500 full-batch iterations and using 0.0001 as initial learning rate for all the weights. The network outputs are used to initialize the values of the grounded predicates for the subsequent collective classification step.

Three settings are compared in the experiments. In the first setting, the neural networks classify directly the images based on their input feature based representation without any logic knowledge. In the second, the proposed collective classification, based on the given KB, is applied after initializing the grounded predicate values using only the supervisions available in the train set, whereas setting the other values to 0.5. Finally, in the third setting, the collective classification is performed after the initialization of each grounded predicate with the output of the corresponding neural network for that grounding. Basically, this last classifier in general provides a better prior also for those nodes that are not supervised.

Figure 4 reports the accuracy on the test labels for different percentages of the available supervisions for training. The accuracy is computed on the 7 exclusive final animal classes, where the class assignment for a pattern is performed via an argmax of the output classification values.

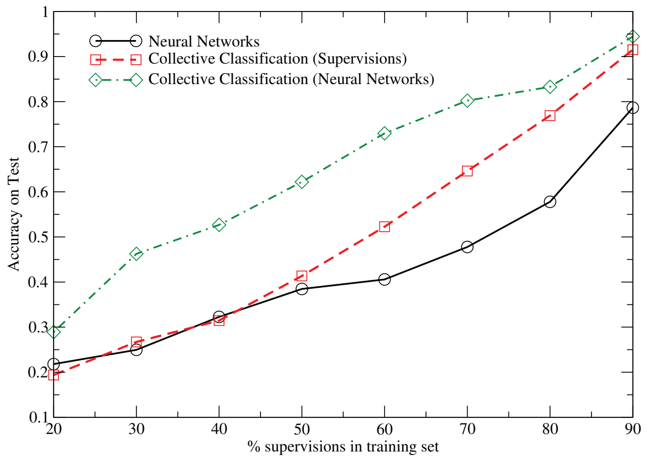

Figure 1: Accuracy of the classifiers varying the amount of available training supervisions.

Collective classification exploiting the KB significantly improves the performances of the neural network classifiers. In particular, the inference step consolidates the final assignments by fixing the inconsistencies of the classifier outputs using the prior knowledge. Among the two collective classification settings, the best performance is obtained by exploiting the outputs of the neural networks as priors. This is because the neural networks not only fit well the available supervisions, but also provide a better prior for the unsupervised grounded predicates. When many supervisions are available, the inference process tends to be fully specified (the benchmark assumes that it is always possible to uniquely identify an animal given a complete set of clues, e.g. the values for the intermediate predicated are known), and the gap in the performances of the two collective classifiers becomes smaller.

5 Conclusions

The paper provides a theoretical result that defines a fragment in the Łukasiewicz logic that yields convex constraints that can be embedded in the training procedure for a learning agent. Convexity in general allows the definition of more efficient methods for the optimization process at the basis of the learning task. For instance, convexity guarantees the unicity of the solution and it is one of the keys of the success of kernel machines. As an example, we showed that the proposed result can be exploited to define collective classification from a knowledge base as a quadratic programming problem.

Future work will aim at analyzing the implications of the specific choice of the fragment connectives with respect to the fuzzy interpretation of the rules. Moreover, the extension of the results to other contexts beside collective classification is under investigation.

Appendix

Proof of Lemma 1. 1. This point is obvious remembering that for all x, $\neg f(x) := 1 - f(x)$.

2. If f, g are concave then for all $x, y \in [0,1]^n$, $\lambda \in [0,1]$, $(f \wedge g)(\lambda x + (1-\lambda)y) = \min\{f(\lambda x + (1-\lambda)y), g(\lambda x + (1-\lambda)y)\} \geq \min\{\lambda f(x) + (1-\lambda)f(y), \lambda g(x) + (1-\lambda)g(y)\} \geq \{\lambda(f \wedge g)(x) + (1-\lambda)(f \wedge g)(y)\}$.
   Moreover, by definition $(f \oplus g)(x) = \min\{1, f(x) + g(x)\}$ thus if $(f \oplus g)(\lambda x + (1-\lambda)y) = 1$ then obviously it is greater or equal than $\lambda(f \oplus g)(x) + (1-\lambda)(f \oplus g)(y)$. Otherwise $f \oplus g = f + g$ and sum preserves concavity (and it preserves convexity too) so the thesis easily follows.

3. This point follows from 1. and 2. plus recalling that $f \vee g = \neg(\neg f \wedge \neg g)$ and $f \otimes g = \neg(\neg f \oplus \neg g)$.

□

Proof of Proposition 1. By hypothesis f_φ is a concave piecewise linear function hence there exist some elements $a_{i_j}, b_i \in \mathbb{Z}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$, such that:

$$f_\varphi(x) = \min_{i=1}^{m} a_{i_1}x_1 + \ldots + a_{i_n}x_n + b_i, \qquad x \in [0,1]^n.$$

If we set $p_i(x) = a_{i_1}x_1 + \ldots + a_{i_n}x_n + b_i$ for $i = 1, \ldots, m$, our claim follows provided every p_i corresponds to a formula in $(\wedge, \oplus)^*$, indeed the operation of minimum is exactly performed by the connective $\wedge$. Let us fix $i \in \{1, \ldots, m\}$, then we can write

$$p_i(x) = \sum_{j \in P_i} a_{i_j}x_j + \sum_{j \in N_i} a_{i_j}x_j + b_i,$$

where $P_i = \{j \leq n : a_{i_j} > 0\}$ and $N_i = \{j \leq n : a_{i_j} < 0\}$. It is worth noticing that in general p_i will assume values out of the unit interval. However $p_i(x) \geq 0$ for all $x \in [0,1]^n$ and the values greater than 1 obviously do not contribute to f_φ. Thanks to this last remark, we can take into account a formula whose corresponding McNaughton function corresponds to p_i truncated to 1 and restricted in $[0,1]^n$. Let us consider the formula

$$\varphi_i = \bigoplus_{j \in P_i} |a_{i_j}|x_j \oplus \bigoplus_{j \in N_i} |a_{i_j}|\neg x_j \oplus q_i,$$

where $|a_{i_j}|x_j = x_j \oplus \ldots \oplus x_j$, $|a_{i_j}|$ times and $q_i = \bar{1} \oplus \ldots \oplus \bar{1}$, q_i times if it is greater than 0, or $\bar{0}$ otherwise. Indeed the first strong disjunction corresponds to all the positive monomials of p_i. The second one corresponds to all the negative monomials of p_i, but it also introduces the quantity $\sum_{j \in N_i} |a_{i_j}|$. Finally $q_i = b_i - \sum_{j \in N_i} |a_{i_j}|$, with $q_i \geq 0$ since $p_i(x) \geq 0$ for all $x \in [0,1]^n$ and in particular $p_i(\bar{x}) = b_i - \sum_{j \in N_i} |a_{i_j}| \geq 0$ where $\bar{x}$ is the vector with 0 in positive and 1 in negative monomial positions respectively. The overall formula can be written as $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_m$. □

References

Bach, S. H. 2015. *Hinge-loss Markov random fields and probabilistic soft logic: A scalable approach to structured prediction*. Ph.D. Dissertation, University of Maryland, College Park.

De Raedt, L., and Kersting, K. 2008. Probabilistic inductive logic programming. In *Probabilistic Inductive Logic Programming*. Springer. 1–27.

Diligenti, M.; Gori, M.; Maggini, M.; and Rigutini, L. 2012. Bridging logic and kernel machines. *Machine learning* 86(1):57–88.

Diligenti, M.; Gori, M.; and Saccà, C. 2015. Semantic-based regularization for learning and inference. *Artificial Intelligence*.

Diligenti, M.; Gori, M.; and Scoca, V. 2016. *Learning Efficiently in Semantic Based Regularization*. Cham: Springer International Publishing. 33–46.

Getoor, L., and Taskar, B. 2007. *Introduction to statistical relational learning*. MIT press.

Gnecco, G.; Gori, M.; Melacci, S.; and Sanguineti, M. 2015. Foundations of support constraint machines. *Neural computation* 27(2):388–480.

Hu, Z.; Ma, X.; Liu, Z.; Hovy, E.; and Xing, E. 2016. Harnessing deep neural networks with logic rules. *arXiv preprint arXiv:1603.06318*.

Lowe, D. G. 1999. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, 1150–1157. IEEE.

Mosca, A., and Magoulas, G. D. 2015. Adapting resilient propagation for deep learning. *CoRR* abs/1509.04612.

Novák, V.; Perfilieva, I.; and Močkoř, J. Mathematical principles of fuzzy logic. 1999.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine learning* 62(1):107–136.

Riedmiller, M., and Braun, H. 1993. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE INTERNATIONAL CONFERENCE ON NEURAL NETWORKS*, 586–591.

Rockafellar, R. T., and Wets, R. J.-B. 2009. *Variational analysis*, volume 317. Springer Science & Business Media.

Serafini, L., and d'Avila Garcez, A. S. 2016. *Learning and Reasoning with Logic Tensor Networks*. Cham: Springer International Publishing. 334–348.

Winston, P. H., and Horn, B. K. 1986. *Lisp*. Addison Wesley Pub., Reading, MA.