Learning Łukasiewicz Logic Fragments by Quadratic Programming

Francesco Giannini, Michelangelo Diligenti, Marco Gori, and Marco Maggini

Department of Information Engineering and Mathematics
University of Siena, Via Roma 56, Siena, Italy
{fgiannini,diligmic,marco,maggini}@diism.unisi.it

Abstract. In this paper we provide a framework to embed logical constraints into the classical learning scheme of kernel machines, that gives rise to a learning algorithm based on a quadratic programming problem. In particular, we show that, once the constraints are expressed using a specific fragment from the Łukasiewicz logic, the learning objective turns out to be convex. We formulate the *primal* and *dual* forms of a general multi–task learning problem, where the functions to be determined are predicates (of any arity) defined on the feature space. The learning set contains both supervised examples for each predicate and unsupervised examples exploited to enforce the constraints. We give some properties of the solutions constructed by the framework along with promising experimental results.

Keywords: Learning from constraints, Kernel machines, First–order logic, Quadratic programming, Łukasiewicz Logic

1 Introduction

Learning from constraints is an extension of classical supervised learning in which the concept of *constraint* is used to refer to a more general class of knowledge than simple labelled examples. In particular a multi–task learning paradigm is assumed in which the functions to be learnt are subject to a set of relational constraints. An extensive investigation of different kinds of constraints can be found in [7], where variational calculus is exploited to generalize the Representation Theorem for kernel machines. In this paper, we focus on logical constraints, which provide a natural, expressive and formally well–defined representation for abstract knowledge as well as being supported by a strong mathematical theory.

In literature, logic is employed in a wide class of learning problems according to different approaches. Diligenti et al. bridge logic and kernel machines considering the logical constraints by means of a functional representation of logical formulas, extending the classical regularization scheme of kernel machines to incorporate logical constraints [4, 5]. However, the objective function, depending on the logic clauses, is not guaranteed to be convex in general. A different approach with kernel machines can be found in [3] where authors introduce a family of kernels parameterized by a Feature Description Language. Logic rules

are also combined with neural networks in different contexts. For instance, Hu et al. [8] propose an iterative procedure that transfers abstract knowledge encoded by the logic rules into the parameters of a deep neural network. Further, it is worth to mention some related works in *inductive logic programming* [11] and in *Markov logic networks* [13, 1] that derive probabilistic graphical models from the rule set. However, like other recent papers do [15], we decide to focus on logic and to not consider probabilistic aspects.

In this paper we provide a logical fragment of propositional Łukasiewicz Logic whose corresponding functional constraints turn out to be convex. It is worth to notice that formulas belonging to this fragment are referred as *simple Łukasiewicz clausal forms* in the literature and, among others, the satisfiability problem for these formulas has linear–time complexity [2]. Functions corresponding to formulas in the proposed fragment are concave (or convex for its dual) as also shown in [10], in which such formulas are used to get concave payoff functions. In our framework, formulas built up from the fragment derive convex functional constraints and we also prove that they coincide with the whole class of concave (convex) Łukasiewicz functions. In addition, thanks to McNaughton Theorem, the functions corresponding to Łukasiewicz formulas are continuous piecewise linear with integer coefficients. As a result, the learning task can be formulated as a quadratic programming problem.

The paper is organized as follows: in the next section we introduce and prove the main results about both concave and convex fragments of Łukasiewicz logic. In Section 3 we describe the different kinds of constraints we will deal with, paying particular attentions to logical constraints. The primal and dual forms, as well as the Lagrangian associated to the problem, are formulated in Section 4. Finally, in Section 5 we provide some preliminary experimental results and a toy example to illustrate the fundamental features of the theory.

2 Concave Fragment of Łukasiewicz Logic

Łukasiewicz Logic **Ł** is a suitable framework to express logical constraints for several reasons. First of all, there exist n-valued (for all finite n) as well as infinitely many–valued variants of this logic, thus we can deal with arbitrary degrees of truth. Among the three fundamental fuzzy logics given by a continuous t-norm (Łukasiewicz, Gödel and Product logic), **Ł** is the only one with an involutive negation and such that every formula has an equivalent *prenex normal form* (i.e. quantifiers followed by a quantifier–free part). Furthermore, for the propositional case, the algebra of formulas of **Ł** on n variables is isomorphic to the algebra of McNaughton functions defined on $[0, 1]^n$. As we will see, this is a crucial property and it allows us to get a substantial advantage with respect to the employment of other logics. In particular, we are interested in the fragment of **Ł** whose formulas correspond to concave functions.

We assume to deal with a knowledge base KB made of first order Łukasiewicz formulas. The syntax of formulas is defined in the usual way [12]. We recall some

basic notions of Łukasiewicz logic and some fundamental results for its algebraic semantics.

The propositional logic **Ł** is sound and complete with respect to the variety[1] of **MV**-algebras and in particular with respect to the standard algebra $[0,1]_{\text{Ł}}$ which generates the whole variety. $[0,1]_{\text{Ł}} = ([0,1], 0, 1, \neg, \wedge, \vee, \otimes, \oplus, \rightarrow)$[2], with operations defined as follows:

$$\neg x = 1 - x, \quad x \wedge y = \min\{x, y\}, \quad x \vee y = \max\{x, y\},$$

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

The operations $\wedge$ and $\otimes$ are the interpretations of *weak* and *strong* conjunction[3] of **Ł** respectively, while $\vee$ and $\oplus$ represent *weak* and *strong* disjunction. Further, $\otimes$ and $\rightarrow$ are Łukasiewicz t-norm and its residuum, where $x \rightarrow y$ is definable as $\neg x \oplus y$. In the paper, for short we write $\bigoplus_{i=1}^{n} x_i$ for $x_1 \oplus \ldots \oplus x_n$ and ax, with $a > 0$ for $\bigoplus_{i=1}^{a} x$, where $x, x_1, \ldots, x_n$ denote any literals.

Some fundamental properties of these operations are:

$$\neg\neg x = x, \quad \neg(x \wedge y) = \neg x \vee \neg y, \quad \neg(x \otimes y) = \neg x \oplus \neg y, \quad x \otimes \neg x = 0, \quad x \oplus \neg x = 1,$$

$$x \otimes (y \wedge z) = (x \otimes y) \wedge (x \otimes z),\ x \otimes (y \vee z) = (x \otimes y) \vee (x \otimes z),\ x \oplus (y \wedge z) = (x \oplus y) \wedge (x \oplus z),$$

$$x \oplus (y \vee z) = (x \oplus y) \vee (x \oplus z),\ x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z),\ x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z).$$

The distributive property does not hold between strong conjunction and strong disjunction.

Since $[0,1]_{\text{Ł}}$ generates the whole algebraic 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 for every $(t_1, \ldots, t_n) \in [0,1]^n$:

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

$$(f \circ g)(t_1, \ldots, t_n) = f(t_1, \ldots, t_n) \circ g(t_1, \ldots, t_n), \qquad \text{with } \circ \in \{\wedge, \vee, \otimes, \oplus, \rightarrow\}.$$

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

---

[1] A variety is a class of algebras closed under the taking of homomorphic images, subalgebras and direct products.

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

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

Definition 1. *Let $f : [0,1]^n \to [0,1]$ 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 functions $p_1, \ldots, p_m$ with integer coefficients such that for all $(t_1, \ldots, t_n) \in [0,1]^n$, there exists $i \leq m$ such that*

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

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

As a consequence, for every **Ł**-formula $\varphi(x_1, \ldots, x_n)$ depending on propositional variables $x_1, \ldots, x_n$, we can consider its corresponding function $f_\varphi : (x_1, \ldots, x_n) \in [0,1]^n \mapsto f_\varphi(x_1, \ldots, x_n) \in [0,1]$, whose value on each point is exactly the evaluation of the formula with respect to the same variable assignment. For McNaughton Theorem f_φ is a McNaughton function.

2.1 Concave McNaughton Functions

In this framework the logic constraints will be expressed by McNaughton functions. Hence we are interested in operations (corresponding to logical connectives) that preserve concavity or convexity of McNaughton functions in order to establish a Łukasiewicz fragment in which concavity (or convexity) is guaranteed.

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.*

Proof. 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 $f \otimes g = \neg(\neg f \oplus \neg g)$.

As a result we get that, if f is a convex function and g is concave, then $f \to g = \neg f \oplus g$ is concave. This implies an immediate consequence.

Corollary 1. *Every Horn clause in Łukasiewicz logic, namely every formula of the form $(x_1 \otimes \ldots \otimes x_m) \to y$ with $x_1, \ldots, x_m, y$ propositional variables, has a corresponding concave McNaughton function.*

Proof. Projections functions $f_{x_1}, \ldots, f_{x_m}, f_y$ are both convex and concave functions. So from the fact that $(f_{x_1} \otimes \ldots \otimes f_{x_m}) \to f_y = \neg(f_{x_1} \otimes \ldots \otimes f_{x_m}) \oplus f_y = \neg f_{x_1} \oplus \ldots \oplus \neg f_{x_m} \oplus f_y$, for the previous lemma, to this formula corresponds a concave function.

Example 1. Horn clauses can be embedded in the $(\wedge, \oplus)^*$-fragment, but the converse does not hold, as shown by the following formulas:

$$x \wedge y, \qquad x \oplus (y \wedge z), \qquad x \wedge (x \to y), \qquad (x \otimes y) \to (x \wedge z).$$

Lemma 1 allows us to determine two Łukasiewicz fragments whose corresponding McNaughton functions are either concave or convex respectively.

Corollary 2. *Let $(\wedge, \oplus)^*$ and $(\otimes, \vee)^*$ be the smallest sets such that:*

- *if y is a propositional variable, then $y, \neg y \in (\wedge, \oplus)^*$ and $y, \neg y \in (\otimes, \vee)^*$;*
- *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 $\varphi_1 \otimes \varphi_2$, $\varphi_1 \vee \varphi_2 \in (\otimes, \vee)^*$;*
- *if $\varphi \in (\wedge, \oplus)^*$ ($\varphi \in (\otimes, \vee)^*$) and φ is equivalent to ψ, then $\psi \in (\wedge, \oplus)^*$ ($\psi \in (\otimes, \vee)^*$).*

Every **Ł***-formula $\varphi \in (\wedge, \oplus)^*$ ($\varphi \in (\otimes, \vee)^*$) corresponds to a concave (convex) McNaughton function f_φ.*

In the paper we will make use of the following result and we refer to Theorem 2.49 in [14] for a proof.

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 every McNaughton function $f : [0,1]^n \to [0,1]$, if f is convex then there exist $A_1, \ldots, A_k \in \mathbb{Z}^n$, $b_1, \ldots, b_k \in \mathbb{Z}$ such that:

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

On the other hand, every concave McNaughton function can be expressed as the minimum of a finite number of affine functions.

The coefficients of the linear functions are constructively determined by the shape of the considered formula. For instance, given any convex McNaughton function f_φ, such that $\varphi \in (\otimes, \vee)^*$, we can rewrite φ as a weak disjunction of strong conjunctions of literals according to the distributive laws. Finally, every disjointed term corresponds to a linear function or to the identically 0 function.

Example 2. Let $\varphi = (\neg x \oplus (y \wedge z)) \oplus \neg z$ be, then $\varphi \in (\wedge, \oplus)^*$ and it is equivalent to $(\neg x \oplus y \oplus \neg z) \wedge (\neg x \oplus z \oplus \neg z)$. From this latter expression, we get:

$$\text{for all } x, y, z \in [0,1]: \qquad f_\varphi(x, y, z) = \min\{1, -x + y - z + 2, -x + 2\}.$$

So far, we showed that $(\wedge, \oplus)^*$-formulas bring to concave corresponding McNaughton functions and, as we will see, to convex logical constraints. The next result shows that indeed $(\wedge, \oplus)^*$ contains all the concave ones.

Proposition 1. *Let $f_\varphi : [0,1]^n \to [0,1]$ be a concave McNaughton function, then $\varphi \in (\wedge, \oplus)^*$.*

Proof. 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 $q_i = (\neg x \oplus x) \oplus \ldots \oplus (\neg x \oplus x)$, q_i times. 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 $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_m$.

This result allows us to conclude that we are taking into account the largest fragment of Łukasiewicz Logic whose McNaughton functions are concave.

Corollary 3. *$(\otimes, \vee)^*$ is the fragment coinciding with the class of all convex McNaughton functions.*

3 Constraints

We consider a learning problem in which each learnable task function is a predicate on a given domain. The considered constraints belong to the following three categories.

1. *Pointwise constraints* are given as supervised examples in training set.
2. *Logical constraints* establish some relations that must hold between predicates. They can be expressed as first order formulas in any logic. In this paper we restrict our attention to Łukasiewicz logic. They are enforced on an unsupervised training set.
3. *Consistency Constraints* derive from the need to limit the possible values of predicates into the real unit interval to guarantee the consistency in the definition of the logical constraints.

Let $KB = \{\varphi_h : h \in \mathbb{N}_H{}^4\}$ be a knowledge base of first order Łukasiewicz formulas. We assume KB contains predicates from the set $\mathbf{P} = \{p_j^{(a_j)} : j \in \mathbb{N}_J\}$ (with their own fixed arity) each one defined on a subset of a power of $\mathbb{R}$. More specifically, for every j we have $p_j^{(a_j)} : \mathcal{R}_{j_1} \times \ldots \times \mathcal{R}_{j_{a_j}} = \mathcal{R}_{n_j} \to \mathbb{R}$, where for $k = 1, \ldots, a_j$, $\mathcal{R}_{j_k} \subseteq \mathbb{R}^{j_k}$ and so $\mathcal{R}_{n_j} \subseteq \mathbb{R}^{n_j}$. However, throughout the paper, we will always evaluate predicates in finite, already fixed, training sets. It is worth noting that, different predicates can share some domains of their arguments. For instance, if $\mathcal{R}_{j_k} = \mathcal{R}_{i_l}$ (for some j, k, i, l), this means the k-th argument of predicate p_j is the same as the l-th argument of predicate p_i. For short, we will write $\mathbf{p} = (p_1, \ldots, p_J) : \mathcal{R}^n \to \mathbb{R}^J$, where $n = n_1 + \ldots + n_J$, for the overall task function.

3.1 Pointwise Constraints

We can formulate the learning task as a *classification* or a *regression* problem. However, even if regression would exploit fuzzy predicates in a narrow sense (allowing label in the whole unit interval), for the moment we only consider the classification case.

For $j = 1, \ldots, J$, we consider a supervised training set for the j-th task p_j:

$$\mathscr{L}_j = \{(\mathbf{x}_l^j, y_l^j) : l \in \mathbb{N}_{l_j},\, \mathbf{x}_l^j \in \mathbb{R}^{n_j}, y_l^j \in \{-1, 1\}\}.$$

In the following, we will omit the superscript j in the case it is clear from the context. For instance, we will write $p_j(\mathbf{x}_l)$ instead of $p_j(\mathbf{x}_l^j)$, because we assume the j-th task applies only to example points from j-th training set. If $y_l = 1$ then $\mathbf{x}_l$ belongs to the true–class, so we would like to have $p_j(\mathbf{x}_l) \geq 1$, whereas if $y_l = -1$ then $\mathbf{x}_l$ belongs to the false–class and we would like to have $p_j(\mathbf{x}_l) \leq 0$. Therefore, supervisions correspond to the following hard–constraints[5]:

$$y_l(2p_j(\mathbf{x}_l) - 1) \geq 1 - 2\xi_{j_l} \text{ with } \xi_{j_l} \geq 0 \qquad \text{for } j = 1, \ldots, J \text{ and } l = 1, \ldots, l_j.$$

where we introduced a slack variable ξ for each point in the training sets to allow soft violations.

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

[5] If we consider the usual 0-1 logic values as labels, we can consider the condition $(2y_l - 1)p_j(\mathbf{x}_l) \geq y_l$.

3.2 Logical Constraints

Logical constraints arise from the Knowledge Base KB that is any collection of *first order* formulas. For Łukasiewicz logic, we can assume without loss of generality that formulas are in *prenex normal form*. We suppose that each variable occurring in predicates takes its values from a fixed set $\mathscr{U}_{j_k}$. The Cartesian product of the sets $\mathscr{U}_{j_k}$ constitutes the unsupervised training set $\mathscr{U}_j$ for the j-th task,

$$\mathscr{U}_j = \{\mathbf{x}_u^j = (\mathbf{x}_1^j, \ldots, \mathbf{x}_{a_j}^j) \in \mathbb{R}^{n_j} : \mathbf{x}_k^j \in \mathscr{U}_{j_k},\, u \in \mathbb{N}_{u_j}\},$$

whose elements will be used to evaluate p_j in logical constraints. In the following we will omit the superscript j when it is clear from the context. Finally we set $U = u_1 + \ldots + u_J$.

Assuming a finite domain for variables, each quantified formula can be replaced with a propositional one applying the following equivalence once per quantifier:

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

where x is the k-th argument of a certain predicate p_j occurring in the formula ψ and $[\mathbf{x}_k/x]$ represents the substitution of the element $\mathbf{x}_k$ 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 such predicates.

By applying (2), we get a new set of formulas KB', where the propositional variables are exactly all possible groundings of predicates occurring in formulas. Each grounding of a predicate returns a value in the unit interval and since predicate functions have to be determined, these values can be thought of as (propositional) variables.

Example 3. Let us consider $\mathbf{P} = \{p_1^{(1)}, p_2^{(2)}\}$ and $KB = \{\varphi_1, \varphi_2\}$ with

$$\varphi_1 : \forall x \exists y (p_1(x) \to p_2(x, y)), \qquad \varphi_2 : \forall x (p_1(x)).$$

Now let us suppose $\mathscr{U}_{1_1} = \mathscr{U}_{2_1} = \{a, b\}$ and $\mathscr{U}_{2_2} = \{a, c\}$, then $KB' = \{\varphi_1', \varphi_2'\}$ with φ_1' and φ_2' respectively:

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

$$p_1(a) \wedge p_1(b).$$

We can simplify the notation introducing new variables, for instance with:

$$y_1 = p_1(a),\, y_2 = p_1(b),\, y_3 = p_2(a,a),\, y_4 = p_2(a,c),\, y_5 = p_2(b,a),\, y_6 = p_2(b,c),$$

$$\varphi_1' : [(y_1 \to y_3) \vee (y_1 \to y_4)] \wedge [(y_2 \to y_5) \vee (y_2 \to y_6)], \qquad \varphi_2' : y_1 \wedge y_2.$$

Formulas in KB' are expressed in the context of propositional Łukasiewicz Logic **Ł** and so we can use all the results reported in the previous section. In particular, every $n-$ary formula φ 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 φ'_h formula.

Every formula in KB' depends on all possible groundings of their occurring predicates, so it is useful for $j = 1, \ldots, J$ to indicate with $\bar{\mathbf{p}}_j$ the vector of all groundings of p_j, namely $\bar{\mathbf{p}}_j = (p_j(\mathbf{x}_1), \ldots, p_j(\mathbf{x}_{u_j}))$. To make all uniform, with a little abuse of notation, we will write every formula $\varphi' \in KB'$ as depending on all groundings of all predicates, $f_{\varphi'} = f_{\varphi'}(\bar{\mathbf{p}})$ where $\bar{\mathbf{p}} = (\bar{\mathbf{p}}_1, \ldots, \bar{\mathbf{p}}_J) \in \mathbb{R}^U$.

Formulas in KB' depend on the parameters which determine each predicate function. Let $p_j \in \mathbf{P}$ be, such that $p_j : \mathcal{R}_{n_j} \subseteq \mathbb{R}^{n_j} \to \mathbb{R}$. We assume that there exists a feature map ϕ_j[6] $: \mathbb{R}^{n_j} \to \mathbb{R}^{N_j}$ (where it may be $n_j << N_j$), such that $p_j(\mathbf{x}) = \omega'_j \cdot \phi_j(\mathbf{x}) + b_j$ with $\omega_j \in \mathbb{R}^{N_j}, b_j \in \mathbb{R}$. If we set $\hat{\omega}_j = (\omega'_j, b_j)', \hat{\phi}_j = (\phi'_j, 1)'$ we have

$$p_j(\mathbf{x}) = \hat{\omega}_j \cdot \hat{\phi}_j(\mathbf{x}). \tag{3}$$

On a fixed training set, the values of predicates are totally determined by the matrices $\hat{\omega}_1, \ldots, \hat{\omega}_J$. This entails that in the weight space formulas will be evaluated by composition as functions on $\mathbb{R}^{N+J}$ where $N = N_1 + \ldots + N_J$. Hence we need to guarantee the convexity of McNaughton functions corresponding to formulas in the weight spaces.

Lemma 2. *Let $f : Y \subseteq \mathbb{R}^m \to \mathbb{R}$ be a convex or concave function and $g : X \subseteq \mathbb{R}^d \to Y$ such that $g(x) = Ax + b$ with $A \in \mathbb{R}^{m,d}, b \in \mathbb{R}^m$. Then the function $h : X \to \mathbb{R}$ defined by $h = f \circ g$ is convex or concave in X.*

Corollary 4. *If a logical constraint is concave in its propositional variables space then it is also concave if expressed in the weight space of the predicates.*

Finally, given $KB' = \{f_1, \ldots, f_H\}$ we enforce the satisfaction of logical constraints by requiring $1 - f_h(\bar{\mathbf{p}}) = 0$, for all $h = 1, \ldots, H$. In order to allow soft violations of these constraints, we introduce new slack variables rewriting the constraints as:

$$1 - f_h(\bar{\mathbf{p}}) \leq \xi_h \text{ with } \xi_h \geq 0 \quad \text{for } h \in \mathbb{N}_H. \tag{4}$$

If f_h is a concave function, then $g_h = 1 - f_h$ is a convex one and we note g_h is the function corresponding to the formula $\neg\varphi'_h$. This means $g_h(\bar{\mathbf{p}}) \leq \xi_h$ is a convex constraint. This formulation brings us to formulate our learning problem as a *convex* mathematical program.

However, in the specific case of Łukasiewicz logic, each formula corresponds to a McNaughton function. Furthermore, we consider only concave functions which

[6] For every $j = 1, \ldots, J$, ϕ_j is determined by the j-th kernel function k_j of the RKHS $\mathcal{H}_j$ and it is such that $k_j(\mathbf{x}, \mathbf{y}) = \langle \phi_j(\mathbf{x}), \phi_j(\mathbf{y}) \rangle_{\mathbb{R}^{N_j}}$.

derive convex constraints like (4). Since every convex McNaughton function can be written according to (1) as the maximum of affine functions, we have

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

where $M_i^h \in \mathbb{R}^{1,U}$ and $q_i^h \in \mathbb{R}$ are integer coefficients determined by the shape of the formula $\neg\varphi'_h$.

Remark 1. Let f_φ be the convex McNaughton function of a Łukasiewicz formula φ, then $\varphi \in (\otimes, \vee)^*$ and we can write

$$\varphi(y_{1_1}, \ldots, y_{I_{k_I}}) = \bigvee_{i=1}^{I} \bigotimes_{k=1}^{k_i} (\neg) y_{i_k}.$$

Therefore for all i, each term $((\neg)y_{i_1} \otimes \ldots \otimes (\neg)y_{i_{k_i}})$ is related to an affine function of $y_{i_1}, \ldots, y_{i_{k_i}}$ corresponding to f_φ in a certain piece of its domain. For instance,

$$\text{if } \varphi = (y_1 \otimes y_1 \otimes \neg y_2) \vee (y_1 \otimes \neg y_2 \otimes \neg y_2), \text{ then } f_\varphi = \max\{2y_1 - y_2 - 1, y_1 - 2y_2, 0\}.$$

Now, since for every h, $g_h(\bar{\mathbf{p}})$ is a *max* of terms, we have $g_h(\bar{\mathbf{p}}) \leq \xi_h$ if and only if

$$M_i^h \cdot \bar{\mathbf{p}} + q_i^h \leq \xi_h \quad \text{for all } i \in \mathbb{N}_{I_h}. \tag{5}$$

This means for every $h \in \mathbb{N}_H$, we can consider the I_h logical constraints expressed by (5) in place of (4). 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.

3.3 Consistency Constraints

The function $\mathbf{p}$ represents a tuple of logical predicates. In principle, when we write any p_j according to (3), we have no guarantees p_j is evaluated in $[0, 1]$. This is the reason why we need to add the following hard constraints:

$$0 \leq p_j(\mathbf{x}_s^j) \leq 1, \quad \text{for } j \in \mathbb{N}_J, \mathbf{x}_s^j \in \mathscr{S}_j = \mathscr{U}_j \cup \mathscr{L}'_j, \tag{6}$$

with $\mathscr{L}'_j = \{\mathbf{x}_l^j : (\mathbf{x}_l^j, y_l^j) \in \mathscr{L}_j\}$. In the following, we indicate with s_j the cardinality of $\mathscr{S}_j$ and $S = s_1 + \ldots + s_J$.

4 Learning by Quadratic Programming

We can formulate the multi–task learning problem as an optimization problem both in the primal and in the dual space. In the primal space, for the overall predicate $\mathbf{p}$ exploiting (3) for every $j = 1, \ldots, J$, we get:

Primal Problem (7)

$$\min \frac{1}{2}\sum_{j\in\mathbb{N}_J}||\omega_j||^2 + C_1\sum_{\substack{j\in\mathbb{N}_J\\ l\in\mathbb{N}_{l_j}}}\xi_{j_l} + C_2\sum_{h\in\mathbb{N}_H}\xi_h \qquad \text{subject to}$$

$$\begin{array}{ccc} y_l(2p_j(\mathbf{x}_l)-1)\geq 1-2\xi_{j_l} & \xi_{j_l}\geq 0 & j\in\mathbb{N}_J, l\in\mathbb{N}_{l_j}, (\mathbf{x}_l,y_l)\in\mathscr{L}_j \\ 1-f_h(\bar{\mathbf{p}})\leq\xi_h & \xi_h\geq 0 & h\in\mathbb{N}_H \\ 0\leq p_j(\mathbf{x}_s)\leq 1 & & s\in\mathbb{N}_{s_j}, \mathbf{x}_s\in\mathscr{S}_j. \end{array}$$

Here $C_1, C_2 > 0$ express the degree of satisfaction of the constraints.

The primal problem (7) is a convex optimization problem, since we have to minimize a convex function subject to convex or affine constraints.

We reformulate the above optimization problem as the minimization of a Lagrangian function. In order to get a quadratic optimization problem, we consider the constraints (5) with $\xi_h \geq 0$ for all $h \in \mathbb{N}_H$ in place of (4), obtaining

$$\mathcal{L}(\hat{\omega},\xi,\lambda,\mu,\eta) = \frac{1}{2}\sum_{j=1}^{J}||\omega_j||^2 + C_1\sum_{j=1}^{J}\sum_{l=1}^{l_j}\xi_{j_l} + C_2\sum_{h=1}^{H}\xi_h - \sum_{j=1}^{J}\sum_{l=1}^{l_j}\mu_{j_l}\xi_{j_l} +$$

$$-\sum_{j=1}^{J}\sum_{l=1}^{l_j}\lambda_{j_l}(y_l(2p_j(\mathbf{x}_l)-1)-1+2\xi_{j_l}) - \sum_{h=1}^{H}\sum_{i=1}^{I_h}\lambda_{h_i}(\xi_h - M_i^h\cdot\bar{\mathbf{p}} - q_i^h) - \sum_{h=1}^{H}\mu_h\xi_h +$$

$$-\sum_{j=1}^{J}\sum_{s=1}^{s_j}\eta_{j_s}p_j(\mathbf{x}_s) - \sum_{j=1}^{J}\sum_{s=1}^{s_j}\bar{\eta}_{j_s}(1-p_j(\mathbf{x}_s)). \tag{8}$$

with the **KKT** conditions, for all $j\in\mathbb{N}_J, l\in\mathbb{N}_{l_j}, h\in\mathbb{N}_H, i\in\mathbb{N}_{I_h}, s\in\mathbb{N}_{s_j}$:

$$\begin{array}{c} \xi_{j_l}\geq 0,\ \xi_h\geq 0,\ \lambda_{j_l}\geq 0,\ \mu_{j_l}\geq 0,\ \lambda_{h_i}\geq 0,\ \mu_h\geq 0,\ \eta_{j_s}\geq 0,\ \bar{\eta}_{j_s}\geq 0, \\ \lambda_{j_l}(y_l(2p_j(\mathbf{x}_l)-1)-1+2\xi_{j_l})=0,\ \mu_{j_l}\xi_{j_l}=0,\ \lambda_{h_i}(\xi_h - M_i^h\cdot\bar{\mathbf{p}} - q_i^h)=0, \\ \mu_h\xi_h=0,\ \eta_{j_s}p_j(\mathbf{x}_s)=0,\ \bar{\eta}_{j_s}(1-p_j(\mathbf{x}_s))=0,\ y_l(2p_j(\mathbf{x}_l)-1)-1+2\xi_{j_l}\geq 0, \\ \xi_h - M_i^h\cdot\bar{\mathbf{p}} - q_i^h\geq 0,\ p_j(\mathbf{x}_s)\geq 0,\ 1-p_j(\mathbf{x}_s)\geq 0. \end{array}$$

To derive the dual space formulation we apply the null gradient condition to the derivatives of $\mathcal{L}$ with respect to every $\omega_j, b_j, \xi_{j_l}, \xi_h$.

$$\nabla_{\omega_j}\mathcal{L} = \omega_j - 2\sum_l \lambda_{j_l}y_l\phi_j(\mathbf{x}_l) + \sum_{h,i}\lambda_{h_i}\sum_u M_{i,u}^h\phi_j(\mathbf{x}_u) - \sum_s(\eta_{j_s}-\bar{\eta}_{j_s})\phi_j(\mathbf{x}_s) = 0;$$

$$\frac{\partial\mathcal{L}}{\partial b_j} = -2\sum_l\lambda_{j_l}y_l + \sum_{h,i}\lambda_{h_i}\sum_u M_{i,u}^h - \sum_s(\eta_{j_s}-\bar{\eta}_{j_s}) = 0;$$

$$\frac{\partial\mathcal{L}}{\partial\xi_{j_l}} = C_1 - 2\lambda_{j_l} - \mu_{j_l} = 0;$$

$$\frac{\partial \mathcal{L}}{\partial \xi_h} = C_2 - \sum_i \lambda_{h_i} - \mu_h = 0.$$

Hence if we substitute, we get

$$\theta(\lambda,\eta) = \mathcal{L}(\hat{\omega}^*, \xi^*, \lambda, \mu, \eta) = -\frac{1}{2}\sum_j [4\sum_{l,l'} \lambda_{j_l}\lambda_{j_{l'}} y_l y_{l'} k_j(\mathbf{x}_l, \mathbf{x}_{l'}) +$$

$$+ \sum_{\substack{h,i \\ h',i'}} \lambda_{h_i}\lambda_{h'_{i'}} \sum_{u,u'} M^h_{i,u} M^{h'}_{i',u'} k_j(\mathbf{x}_u, \mathbf{x}_{u'}) + \sum_{s,s'} (\eta_{j_s} - \bar{\eta}_{j_s})(\eta_{j_{s'}} - \bar{\eta}_{j_{s'}}) k_j(\mathbf{x}_s, \mathbf{x}_{s'}) +$$

$$-4\sum_{l,h,i} \lambda_{j_l}\lambda_{h_i} y_l \sum_u M^h_{i,u} k_j(\mathbf{x}_l, \mathbf{x}_u) + 4\sum_{l,s} \lambda_{j_l} y_l (\eta_{j_s} - \bar{\eta}_{j_s}) k_j(\mathbf{x}_l, \mathbf{x}_s) +$$

$$-2\sum_{h,i,s} \lambda_{h_i}(\eta_{j_s} - \bar{\eta}_{j_s}) \sum_u M^h_{i,u} k_j(\mathbf{x}_u, \mathbf{x}_s)] + \sum_{j,l} \lambda_{j_l} + \sum_{h,i} \lambda_{h_i}(\frac{1}{2}\sum_u M^h_{i,u} + q^h_i) +$$

$$-\frac{1}{2}\sum_{j,s} (\eta_{j_s} + \bar{\eta}_{j_s}).$$

Finally, we can formulate the dual problem as:

Dual Problem (9)

$$\max\ \theta(\lambda,\eta) \qquad \text{subject to}$$

$$\sum_{h=1}^{H}\sum_{i=1}^{I_h} \lambda_{h_i} \sum_{u=1}^{u_j} M^h_{i,u} = 2\sum_{l=1}^{l_j} \lambda_{j_l} y_l + \sum_{s=1}^{s_j} (\eta_{j_s} - \bar{\eta}_{j_s}) \qquad j \in \mathbb{N}_J$$

$$0 \le \lambda_{j_l} \le C_1 \qquad j \in \mathbb{N}_J,\ l \in \mathbb{N}_{l_j}$$

$$0 \le \lambda_{h_i} \le C_2 \qquad h \in \mathbb{N}_H,\ i \in \mathbb{N}_{I_h}$$

$$\eta_{j_s} \ge 0,\ \bar{\eta}_{j_s} \ge 0 \qquad j \in \mathbb{N}_J,\ s \in \mathbb{N}_{s_j}.$$

From (9) we can find the optimal values $\lambda^*_{j_l}, \lambda^*_{h_i}, \eta^*_{j_s}, \bar{\eta}^*_{j_s}$ for all j, l, h, i, s. Then, for all $j = 1, \ldots, J$, we get the formulation of the j-th task in the dual space as:

$$p_j(\mathbf{x}) = \omega^*_j \cdot \phi_j(\mathbf{x}) + b^*_j = 2\sum_{l=1}^{l_j} \lambda^*_{j_l} y_l k_j(\mathbf{x}_l, \mathbf{x}) +$$

$$-\sum_{h=1}^{H}\sum_{i=1}^{I_h} \lambda^*_{h_i} \sum_{u=1}^{u_j} M^h_{i,u} k_j(\mathbf{x}_u, \mathbf{x}) + \sum_{s=1}^{s_j} (\eta^*_{j_s} - \bar{\eta}^*_{j_s}) k_j(\mathbf{x}_s, \mathbf{x}) + b^*_j. \qquad (10)$$

5 Experimental Results

In this section, we report the experimental evaluation of the proposed method on two datasets. The first one is a toy problem that allows us to enlighten how logical constraints contribute to the solution of a given problem. The second one is based on the Winston benchmark for image classification.

5.1 Toy problem

Let us consider the following multi–task problem, where we want to determine three predicate functions p_1, p_2, p_3 defined on $\mathbb{R}^2$ and let L_1, L_2, L_3 be such that $L_1 = \{(0.1, 0.5, -1), (0.4, 0.4, -1), (0.3, 0.8, 1), (0.9, 0.7, 1)\}$, $L_2 = \{(0.1, 0.3, -1), (0.6, 0.4, -1), (0.2, 0.8, 1), (0.7, 0.6, 1)\}$ and $L_3 = \{(0.4, 0.2, -1), (0.9, 0.3, -1), (0.2, 0.6, 1), (0.5, 0.7, 1)\}$ the sets of supervised examples. In Fig. 1, we show the (unique) solution for the standard kernel machine scheme in which we only take into account pointwise constraints and we set $C_1 = 15$.

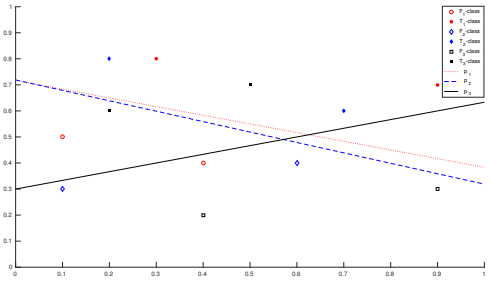

Fig. 1. The dotted, dashed and solid lines represent the task functions p_1, p_2 and p_3 respectively. Data points with different shape relate to different predicates and while empty figures correspond to the false–class, the filled ones correspond to the true–class.

Now let us suppose to know, apart from supervisions, some additional relational informations about task functions, expressed by the logical clauses

$$\varphi_1 = \forall x(p_1(x) \rightarrow p_2(x)), \qquad \varphi_2 = \forall x(p_2(x) \rightarrow p_3(x)),$$

whose intuitive meaning is: the p_1 true–class is contained in the p_2 true–class which in turn is contained in the p_3 true–class. To evaluate logical constraints we fix a few unsupervised examples. For instance, in Fig. 2 we make a comparison between the effect of a single point $P\,(0.8, 0.3)$ (in which logical constraints were violated) and the effect of a larger unsupervised training set $U = \{(0.1, 0.5), (0.3, 0.7), (0.5, 0.4), (0.8, 0.3), (0.9, 0.2), (1, 0.5)\}$. The last plot shows that with just few unsupervised points the boundaries of the predicates are correctly placed in the (unique) solution in order to satisfy the given logic constraints in the considered domain.

5.2 Winston benchmark

The second experimental analysis is based on the animal identification problem originally proposed by P. Winston [16]. This benchmark was initially designed

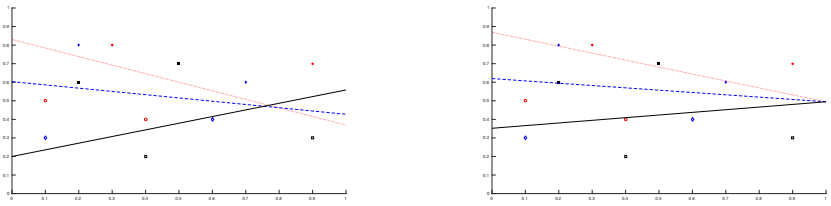

Fig. 2. The pictures show, respectively from left to right, the effect of considering as unsupervised examples the single point P and the set U. In both cases we set $C_1 = C_2 = 2.5$.

to show the ability of logic programming to guess the animal type from some initial clues.

The dataset is composed of 5605 images, taken from the *ImageNet*[7] database, equally divided into 7 classes, each one representing one animal category: albatross, cheetah, giraffe, ostrich, penguin, tiger and zebra. 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 [9] 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. [6], 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[8].

The images have been split into two initial sets: the first one is composed of 2100 images utilized for building the visual vocabulary, while the second set is composed of 3505 images used in the learning process. The experimental analysis has been carried out using by randomly sampling from the overall set of the supervisions the labels to keep as training, validation and test set, randomly sampling 50%, 25%, 25% of the supervisions, respectively.

The knowledge about the classification task is expressed by a set of FOL rules. A total of 33 classes are referenced by the KB, the 7 animal classes plus other intermediate classes each of which is either representing a subset of animals in a taxonomy (like the classes *mammal* or *bird*) or indicating some specific feature of the animals (like *hair* or *darkspots*). Table 1 shows the set of rules used

[7] http://www.image-net.org

[8] The dataset and code to reproduce the results can be downloaded from https://github.com/diligmic/ECML2017_1

Rule
$\text{hair}(x) \rightarrow \text{mammal}(x)$
$\text{milk}(x) \rightarrow \text{mammal}(x)$
$\text{feather}(x) \rightarrow \text{bird}(x)$
$\text{layeggs}(x) \rightarrow \text{bird}(x)$
$\text{mammal}(x) \wedge \text{meat}(x) \rightarrow \text{carnivore}(x)$
$\text{mammal}(x) \wedge \text{pointedteeth}(x) \wedge \text{claws}(x) \wedge \text{forwardeyes}(x) \rightarrow \text{carnivore}(x)$
$\text{mammal}(x) \wedge \text{hoofs}(x) \rightarrow \text{ungulate}(x)$
$\text{mammal}(x) \wedge \text{cud}(x) \rightarrow \text{ungulate}(x)$
$\text{carnivore}(x) \wedge \text{tawny}(x) \wedge \text{darkspots}(x) \rightarrow \text{cheetah}(x)$
$\text{carnivore}(x) \wedge \text{tawny}(x) \wedge \text{blackstripes}(x) \rightarrow \text{tiger}(x)$
$\text{ungulate}(x) \wedge \text{longlegs}(x) \wedge \text{longneck}(x) \wedge \text{tawny}(x) \wedge \text{darkspots}(x) \rightarrow \text{giraffe}(x)$
$\text{ungulate}(x) \wedge \text{white}(x) \wedge \text{blackstripes}(x) \rightarrow \text{zebra}(x)$
$\text{bird}(x) \wedge \text{longlegs}(x) \wedge \text{longneck}(x) \wedge \text{black}(x) \rightarrow \text{ostrich}(x)$
$\text{bird}(x) \wedge \text{swim}(x) \wedge \text{blackwhite}(x) \rightarrow \text{penguin}(x)$
$\text{bird}(x) \wedge \text{goodflier}(x) \rightarrow \text{albatross}(x)$
$\text{cheetah}(x) \oplus \text{tiger}(x) \oplus \text{giraffe}(x) \oplus \text{zebra}(x) \oplus \text{ostrich}(x) \oplus \text{penguin}(x) \oplus \text{albatross}(x)$
$\text{mammal}(x) \oplus \text{bird}(x)$
$\text{hair}(x) \oplus \text{feather}(x)$
$(\text{darkspots}(x)) \rightarrow \neg\ \text{blackstripes}(x)$
$(\text{blackstripes}(x)) \rightarrow \neg\ \text{darkspots}(x)$
$\text{tawny}(x) \rightarrow \neg\ \text{black}(x)) \wedge \neg\ \text{white}(x)$
$\text{black}(x) \rightarrow \neg\ \text{tawny}(x)) \wedge \neg\ \text{white}(x)$
$\text{white}(x) \rightarrow \neg\ \text{black}(x)) \wedge \neg\ \text{tawny}(x)$
$\text{black}(x) \rightarrow \neg\ \text{white}(x)$
$\text{black}(x) \rightarrow \neg\ \text{tawny}(x)$
$\text{white}(x)) \rightarrow \neg\ \text{black}(x)$
$\text{white}(x) \rightarrow \neg\ \text{tawny}(x)$
$\text{tawny}(x) \rightarrow \neg\ \text{white}(x)$
$\text{tawny}(x)) \rightarrow \neg\ \text{black}(x)$

Table 1. The KB used for training the models in the Winston image classification benchmark.

in this task. The first 15 rules are the same as stated in the original problem definition by Winston. The fact that each image shows one and only one animal classification is expressed by another rule stating that each pattern should belong to only one of the classes representing an animal. Another set of rules forces the semantic consistency among the intermediate classes.

All the images are available at training time, but only the training sample of the labels is made available during training. For each reported experiment, one Kernel Machine is trained for each of the 33 predicates in the KB. Gaussian kernels have been selected to be used in the experiments as they were clearly outperforming both the linear kernel and all the tested variations of the polynomial kernel.

	Baseline	Łukasiewicz	Goedel	Product	Convex Łukasiewicz
F1	0.45	0.53	0.52	0.55	**0.56**

Table 2. F1 values for the baseline (no logic knowledge) and using the KB integrated into the learning process using different t-norms.

Table 2 reports the summary of the results. Learning with the logic knowledge is significantly improving the results. The convex Łukasiewicz fragment yields the highest F1 metric on this benchmark.

6 Conclusions

In this paper we proposed a theoretical framework that allows the formulation of learning with logical constraints as a quadratic problem for kernel machines. This is guaranteed when logical constraints are expressed by means of $(\wedge, \oplus)^*$-formulas of Łukasiewicz Logic. As a result, the optimal solution is unique. We evaluated the learning scheme on one image classification benchmark. Future developments include the inference of new logical statements from examples. This is straight connected with the development of a *parsimonious agent* able to learn from the environment and to propose new abstract knowledge concerning its tasks.

References

1. Bach, S.H., Broecheler, M., Huang, B., Getoor, L.: Hinge-loss markov random fields and probabilistic soft logic. arXiv preprint arXiv:1505.04406 (2015)
2. Bofill, M., Manya, F., Vidal, A., Villaret, M.: Finding hard instances of satisfiability in lukasiewicz logics. In: Multiple-Valued Logic (ISMVL), 2015 IEEE International Symposium on. pp. 30–35. IEEE (2015)
3. Cumby, C.M., Roth, D.: On kernel methods for relational learning. In: Proceedings of the 20th International Conference on Machine Learning (ICML-03). pp. 107–114 (2003)
4. Diligenti, M., Gori, M., Maggini, M., Rigutini, L.: Bridging logic and kernel machines. Machine learning 86(1), 57–88 (2012)
5. Diligenti, M., Gori, M., Saccà, C.: Semantic-based regularization for learning and inference. Artificial Intelligence (2015)
6. Diligenti, M., Gori, M., Scoca, V.: Learning efficiently in semantic based regularization. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 33–46. Springer (2016)
7. Gnecco, G., Gori, M., Melacci, S., Sanguineti, M.: Foundations of support constraint machines. Neural computation 27(2), 388–480 (2015)
8. Hu, Z., Ma, X., Liu, Z., Hovy, E., Xing, E.: Harnessing deep neural networks with logic rules. arXiv preprint arXiv:1603.06318 (2016)
9. Lowe, D.G.: Object recognition from local scale-invariant features. In: Computer vision, 1999. The proceedings of the seventh IEEE international conference on. vol. 2, pp. 1150–1157. IEEE (1999)
10. Marchioni, E., Wooldridge, M.: Łukasiewicz games: A logic-based approach to quantitative strategic interactions. ACM Transactions on Computational Logic (TOCL) 16(4), 33 (2015)
11. Muggleton, S., Lodhi, H., Amini, A., Sternberg, M.J.: Support vector inductive logic programming. In: Discovery science. vol. 3735, pp. 163–175. Springer (2005)
12. Novák, V., Perfilieva, I., Močkoř, J.: Mathematical principles of fuzzy logic. 1999
13. Richardson, M., Domingos, P.: Markov logic networks. Machine learning 62(1), 107–136 (2006)
14. Rockafellar, R.T., Wets, R.J.B.: Variational analysis, vol. 317. Springer Science & Business Media (2009)
15. Serafini, L., Garcez, A.d.: Logic tensor networks: Deep learning and logical reasoning from data and knowledge. arXiv preprint arXiv:1606.04422 (2016)
16. Winston, P.H., Horn, B.K.: Lisp. Addison Wesley Pub., Reading, MA (1986)