Mach Learn (2012) 86:57–88
DOI 10.1007/s10994-011-5243-x

Bridging logic and kernel machines

Michelangelo Diligenti · Marco Gori · Marco Maggini · Leonardo Rigutini

Received: 28 July 2010 / Accepted: 27 February 2011 / Published online: 10 May 2011

Abstract We propose a general framework to incorporate first-order logic (FOL) clauses, that are thought of as an abstract and partial representation of the environment, into kernel machines that learn within a semi-supervised scheme. We rely on a multi-task learning scheme where each task is associated with a unary predicate defined on the feature space, while higher level abstract representations consist of FOL clauses made of those predicates. We re-use the kernel machine mathematical apparatus to solve the problem as primal optimization of a function composed of the loss on the supervised examples, the regularization term, and a penalty term deriving from forcing real-valued constraints deriving from the predicates. Unlike for classic kernel machines, however, depending on the logic clauses, the overall function to be optimized is not convex anymore. An important contribution is to show that while tackling the optimization by classic numerical schemes is likely to be hopeless, a stage-based learning scheme, in which we start learning the supervised examples until convergence is reached, and then continue by forcing the logic clauses is a viable direction to attack the problem. Some promising experimental results are given on artificial learning tasks and on the automatic tagging of bibtex entries to emphasize the comparison with plain kernel machines.

Keywords Kernel machines · First-order logic · Learning from constraints · Learning with prior knowledge · Multi-task learning · Semantic-based regularization

Editors: Paolo Frasconi and Francesca Lisi.

M. Diligenti · M. Gori (✉) · M. Maggini · L. Rigutini
Dipartimento di Ingegneria dell'Informazione, Università di Siena, Siena, Italy
e-mail: marco@dii.unisi.it

M. Diligenti
e-mail: michi@dii.unisi.it

M. Maggini
e-mail: maggini@dii.unisi.it

L. Rigutini
e-mail: rigutini@dii.unisi.it

1 Introduction

In this paper we propose a novel method to incorporate logic clauses, that are thought of as abstract and partial representations of the environment and are expected to dictate constraints on the development of an agent that learns from examples. We rely on a multi-task learning scheme where each task is associated with a unary predicate defined in the feature space, while higher level abstract representations consist of FOL clauses made of those task predicates. A proper re-use of the kernel machine mathematical apparatus makes it possible to cast the problem as primal optimization of a function composed of the loss on the supervised examples, the regularization term, and a penalty term deriving from forcing the constraints. This yields a coupling amongst the tasks that comes from the constraints, whereas in related studies (Caponnetto et al. 2008), the dependencies are induced by the structure of multi-task kernels. Based on these basic ideas, this paper proposes a fundamental re-thinking of learning in kernel machines which nicely bridges classic logic formalisms for knowledge representation so as to capture human cognitive abilities at the border between induction and deduction. The main results can be stated as follows

(i) *Learning from constraints in kernel machines*
We extend the learning framework of kernel machines to accommodate the new notion of constraint, and give a representer theorem which dictates the optimal solution of the problem as a kernel expansion. The theory is based on the assumption of using unsupervised data to assess the degree of satisfaction of the given constraints. Interestingly, this is a natural assumption that fits perfectly into the mathematical apparatus of kernel machines, since the sampling of the constraints corresponds somehow to the classic transit from functional to empirical risk. In a sense, the notion of constraint unifies the handling of supervised and unsupervised examples, and it gives rise to a methodology that well fits the increasing trend of emphasizing the role of unsupervised examples, that is common in relevant real-world problems.

(ii) *Bridging logic and kernel machines*
We use well-established results in the theory of T-norms for converting FOL logic clauses into real-valued functions, thus ending-up into a constrained multi-task learning problem. From one side, this natural incorporation of logic makes it possible to inject symbolic knowledge, so as to express logic connections between the tasks. From the other side, the machine operates inherently in the perceptual space with real numbers, thus offering a very natural bridge between symbolic and sub-symbolic information. As discussed in Sect. 2, this is remarkably different with respect to related approaches in the literature.

(iii) *Stage-based learning*
Unlike for classic kernel machines, however, depending on the logic clauses, because of the inherent complexity of the stated problem, the overall function to be optimized is not convex anymore, that makes it hopeless the adoption of classic optimization approaches. Following inspirations coming from the principles of cognitive development, that have been the subject of an in-depth analysis in children by Jean Piaget, we acquire experimental evidence on the crucial role of stage-based learning sketched in (Gori 2009), and reinforce the importance of insights on teaching issues like those pointed out by the notion of curriculum learning (Bengio 2009). In a first stage, the learning procedure considers only the supervised examples until convergence is reached, while the constraints defined by the logic clauses are incorporated in the second learning phase. Because of the coherence of supervised examples and logic clauses, the first stage facilitates significantly the minimization of the penalty term associated with the

constraints, since classic gradient descent heuristics are more likely to start into the basin of attraction of the global minimum than a random start.

(iv) *Experimental evidence of improvements w.r.t. kernel machines*
We show very promising experimental results to emphasize the comparison with plain kernel machines. In particular, we report the results of a massive experimentation on a set of artificial data based on a two-dimensional perceptual space. The performance turns out to be systematically better even in a such a "small space" in which relatively small collections of supervised examples are enough to attain remarkable performance with plain kernel machines. Basically, our massive experimental comparisons on artificial data has been carried out under disadvantaged conditions for the proposed constrained-based solution so as to better assess the potential power of the proposal. In order to show how a real-world problem can be properly approached by merging a knowledge-based with training data, we selected a problem of multi-label text classification for automated tag suggestion (Katakis et al. 2008) presented at the ECML/PKDD 2008 Discovery Challenge. We enriched the benchmark with a set of rules describing the semantic relations between the tags, so as to create the expected learning environment for the proposed model. The experiments that we have carried out on tagging does not only indicate improvements of the proposed model in the measure of precision, but they also clearly indicate that the attached tags are significantly more consistent with the knowledge base than in the case of plain kernel machines.

The paper is organized as follows: In the next section we discuss the related work, while in Sect. 3 we introduce the notion of learning from constraints with kernel machines. In Sect. 4 we present the basic idea of stage-based learning. The translation of FOL knowledge-based partial descriptions of the environment into real-valued constraints is described in Sect. 5, and some of our experimental results are reported in Sect. 6. Finally some conclusions and are drawn.

2 Related works

The connections between logic and kernel machines have been the subject of many investigations in the last few years. They have been mostly carried out within the framework of earlier studies on the relationships between symbolic and sub-symbolic models in AI (Hitzler et al. 2004), which are still addressing open problems that need to be solved for significant developments in both cognitive science and applied AI. Most emphasis has been on hybrid models, where perceptual and logic information is mostly handled separately in different modules, whereas a truly tight integration seems to be still hard to achieve because of the barriers erected by the different mathematical models classically used to handle logic reasoning and learning with real numbers. When restricting to kernel machines, a rich analysis of the literature can be found in the quite comprehensive survey (Laurer and Bloch 2009), while a broader coverage of the field with emphasis on the connections with inductive logic programming is in (Raedt et al. 2008). A related approach to combining first-order logic and probabilistic graphical models in a single representation has been proposed in (Richardson and Domingos 2006), by the Markov logic networks, which have received a lot of attention in the last few years.

The fundamental idea of convolution kernels in discrete structures (Haussler 1999) has been one of the main sources of inspiration for exploring the connections of kernel machines with logic. In (Cumby and Roth 2002, 2003), the Feature Description Logic (FDL) is introduced to support learning in domains that are relational, but where the amount of data and size of representation learned are very large. The paradigm provides a natural solution to the problem of learning and representing relational data and extends and unifies several lines of works in machine learning. An interesting related work on statistical learning for query answering is in (Fanizzi et al. 2008). In (Muggleton et al. 2005), support vector machines with a kernel that is an inner product in the feature space spanned by a given set of first-order hypothesized clauses is proposed, while in (Landwehr et al. 2006), the well-known inductive logic programming system FOIL is combined with kernel methods, by leveraging FOIL search for a set of relevant clauses. The model, referred to as kFOIL, implements a dynamic propositionalization approach and allows one to perform both classification and regression tasks. In (Landwehr et al. 2010), a general theoretical framework for statistical logical learning with kernels based on dynamic propositionalization is developed where structure learning corresponds to inferring a suitable kernel on logical objects, and parameter learning corresponds to function learning in the resulting reproducing kernel Hilbert space.

Quite a different approach, which is based on imposing constraints in the perceptual space, has been introduced in (Fung et al. 2002). An efficient procedure is proposed for incorporating prior knowledge in the form of convex constraints in the input space into a linear support vector machine classifier. A knowledge base in the form of propositional logic turns out to be naturally representable by the modeled constraints and leads to remarkable results in the breast cancer prognosis. An extension to the case of nonlinear kernel classifiers is given in (Fung et al. 2003), while in (Le et al. 2006) another extension is proposed, in which the separator is no longer a hyperplane, but the union of a half-space and the polyhedra associated with the knowledge base. Beginning from the same idea, in (Maclin et al. 2007), a limitation of the use of prior knowledge is introduced which naturally allows one to incorporate and refine incorrect knowledge.

While most of the reviewed papers have already proposed different frameworks for incorporating prior knowledge expressed by logic formalisms into kernel machines, one limitation seems to be that the integration does not reveal very tight connections. The incorporation of structures expressed by different formalisms into kernels yields intriguing, yet artificial, notions of similarities. The kernel, which is expected primarily to measure the smoothness of the solution according to the Occam's razor, is asked to play the additional role of incorporating logic structures. While this is a very interesting idea, which enriches significantly the role of the kernels, the remarkable residual degree of freedom on the way the same logic structures can be incorporated suggests that we are only partially addressing the inherent limitation of kernel methods and of most learning models, which do not fully take into account the constraints of the problem at hand. The only direct attempt to deal with the constraints of the environment, originated by the work in (Fung et al. 2002), focusses on the perceptual space. So far, there is no attempt to explore the consequence of learning into a multi-task environment, where the agent is expected to act consistently with a given set of constraints representing the background knowledge. The studies on convex constraints (Gori and Melacci 2010) and the coherent decisions of the classifiers acting on different views of the same pattern (Melacci et al. 2009) follow this research guideline, while some more tight connections come from the preliminary studies on FOL constraints and kernel machines given in (Diligenti et al. 2010a, 2010b). The idea of centering the theory around the general and unified notion of constraints turns out to be a very straightforward way of bridging logic and kernels, since the adoption of T-norms makes it possible to express most classic logic formalisms by constraints on real-valued functions. In a sense, the way we propose to bridge logic and kernel machines seems to be the most natural and straightforward extension of the classic statistical framework of learning from examples, since they are just a special instance of constraints. The theory behind our agents is founded on the replacement of supervised pairs with constraints, a notion which embraces logic descriptions. That replacement along with the systematic construction of a theory of learning from logic constraints is the main distinguishing feature of this paper. This is related with the interpretation given in (Gori 2009), where the classic concept of regularization, based on smoothness issues, is enriched with constraints. This view leads to think of a more powerful approach of facing the ill-position of learning from examples by *semantic-based regularization*, which is not covered in this paper.

3 Learning with constraints

We consider a multitask learning problem in which the input is a tuple $\mathcal{X} = \{x_j | x_j \in \mathcal{D}_j, j = 1, \ldots, n\}$, being $\mathcal{D}_j$ the domain of the values for the jth attribute. A set of multivariate functions $\{\tau_k(x_{j(1,k)}, \ldots, x_{j(n_k,k)}) | k = 1, \ldots, T,\ x_{j(l,k)} \in \mathcal{X},\ \tau_k \in \mathcal{T}_k\}$ are computed, such that each function is exploited to attach a specific feature to the tuple given a subset of its attributes. The ordered set of the function arguments is defined by the map $j(l,k)$ that yields the index j of the attribute in $\mathcal{X}$ used as the lth argument of function k, being n_k the number of its arguments. Further, we assumed that each of the T task functions belongs to an assigned functional space $\mathcal{T}_k$. For example a function depending on a single argument can determine whether the input belongs to a class, whereas a function defined on two arguments can predict if some given binary relationship holds between them. Some of the functions $\tau_k(x_{j(1,k)}, \ldots, x_{j(n_k,k)})$ may be known a priori whereas others must be inferred from examples. In general, we assume that the attributes in each domain $\mathcal{D}_j$, $j = 1, \ldots, n$ are described by a real valued vector of features that are relevant to solve the tasks at hand. Hence, it holds that $\mathcal{D}_j = \mathbb{R}^{d_j}$ and $\tau_k : \mathbb{R}^{d_{j(1,k)}} \times \cdots \times \mathbb{R}^{d_{j(n_k,k)}} \to \mathbb{R}$. For the sake of compactness, in the following we will indicate by $\boldsymbol{x}_k = [x'_{j(1,k)} \cdots x'_{j(n_k,k)}]' \in \mathbb{R}^{d_k}$, where $d_k = \sum_{l=1,\ldots,n_k} d_{j(l,k)}$, the input vector for the kth task. As an example, consider a multi-view image recognition system. In this case the input tuple is the set of different views acquired from the object and, once a proper processing is applied to extract relevant visual features, each of them can be represented by a real valued vector.

Further, we assume that the instances of the considered attributes x_j are generated from a probability distribution $p_{\mathcal{X}}(x_1, \ldots, x_n)$ that models the more general case in which there are dependencies among these variables. For instance, if the attributes represent different features extracted from the same object, like it happens for the single views in multi-view object recognition, their values are mutually dependent. These dependencies are assumed to be themselves an unknown property of the problem at hand that can be estimated from the training examples. The unknown joint probability distribution $p_{\mathcal{X}}(x_1, \ldots, x_n)$ allows us to model both the variabilities in the object space and the presence of noise in the feature extraction process. The probability distribution can be marginalized to obtain the distribution corresponding to a given subset of attributes. In particular in the following, we will denote by $p_{\boldsymbol{x}_k}(\boldsymbol{x}_k)$ the probability distribution of the arguments for the kth task function. If the attributes collected into $\boldsymbol{x}_k$ are mutually independent, then $p_{\boldsymbol{x}_k}(\boldsymbol{x}_k) = \prod_{l=1,\ldots,n_k} p_{x_{j(l,k)}}(x_{j(l,k)})$ holds, where $p_{x_j}(x_j)$ is the distribution of the attribute x_j in its domain $\mathcal{D}_j$.

We consider the case when the tasks functions τ_k have to meet a set of constraints that can be expressed by the functionals $\phi_h : \mathcal{T}_1 \times \cdots \times \mathcal{T}_T \to [0, +\infty)$ such that

$$\phi_h(\tau_1, \ldots, \tau_T) = 0 \quad h = 1, \ldots, H \tag{1}$$

must hold for any valid choice of $\tau_k \in \mathcal{T}_k$, $k = 1, \dots, T$. In particular, in the next section we will show how appropriate functionals can be defined to force the function values to meet a set of first-order logic constraints.

In order to define the learning task, we suppose that each task function τ_k can be approximated by a function f_k in an appropriate Reproducing Kernel Hilbert Space $\mathcal{H}_k$. Therefore, the learning procedure can be cast as a set of T optimization problems that aim at computing the optimal functions $f_1 \in \mathcal{H}_1, \dots, f_T \in \mathcal{H}_T$, where $f_k : \mathbb{R}^{d_{j(1,k)}} \times \cdots \times \mathbb{R}^{d_{j(n_k,k)}} \to \mathbb{R}$, $k = 1, \dots, T$. In the following, we will indicate by $\boldsymbol{f} = [f_1, \dots, f_T]'$ the vector collecting these functions. The function spaces $\mathcal{H}_k$ are specific for each function since the function domains are generally different from each other. Moreover, in general we may decide to approximate each task function in a different space due to some a priori knowledge on its properties (i.e. we may decide to exploit different kernels for the expansion).

We consider the classical learning formulation as a risk minimization problem. Assuming that the correlation among the input features $\boldsymbol{x}_k$ and the desired task function output y_k is modeled by a joint probability distribution $p_{(\boldsymbol{x}_k, y_k)}(\boldsymbol{x}_k, y_k)$, the risk associated to a hypothesis $\boldsymbol{f}$ is measured as,

$$R[\boldsymbol{f}] = \sum_{k=1}^{T} \lambda_k^{\tau} \cdot \int L_k^e\left(f_k(\boldsymbol{x}_k), y_k\right) p_{(\boldsymbol{x}_k, y_k)}(\boldsymbol{x}_k, y_k)\, d\boldsymbol{x}_k\, dy_k \tag{2}$$

where $\lambda_k^{\tau} > 0$ is the weight of the risk for the kth task and $L_k^e\left(f_k(\boldsymbol{x}_k), y_k\right)$ is a loss function that measures the fitting quality of $f_k(\boldsymbol{x}_k)$ with respect to the target y_k. Common choices for the loss function are the quadratic function especially for regression tasks, and the hinge function for binary classification tasks. In the considered multitask problem a different loss function can be exploited for each task function f_k, especially in the case if both classification and regression tasks are mixed together.

As for the regularization term, unlike the general setting of multi-task kernels (Caponnetto et al. 2008), we simply consider scalar kernels that do not yield interactions amongst the different tasks,[1] that is

$$N[\boldsymbol{f}] = \sum_{k=1}^{T} \lambda_k^{r} \cdot \|f_k\|^2_{\mathcal{H}_k}, \tag{3}$$

where $\lambda_k^{r} > 0$ can be used to weight of the regularization contribution for the kth task.

Clearly, if the tasks are uncorrelated, the optimization of the objective function $R[\boldsymbol{f}] + N[\boldsymbol{f}]$ with respect to the T functions $f_k \in \mathcal{H}_k$ is equivalent to T stand-alone optimization problems for each function with objectives $\lambda_k^{\tau} \cdot R_k[f_k] + \lambda_k^{r} \cdot \|f_k\|^2_{\mathcal{H}_k}$, $k = 1, \dots, T$. However, if we consider a problem for which some correlations among the tasks are known a priori and coded as rules, we can enforce also these constraints in the learning procedure. Following the classical penalty approach for constrained optimization, we can embed the constraints by adding a term that penalizes their violation. Since the functionals $\phi_h(\boldsymbol{f})$ are strictly positive when the related constraint is violated and zero otherwise, the overall degree of constraint violation of the current hypothesis $\boldsymbol{f}$ can be measured as

$$V[\boldsymbol{f}] = \sum_{h=1}^{H} \lambda_h^{v} \cdot \phi_h(\boldsymbol{f}), \tag{4}$$

[1] It is worth mending that this choice is simply dictated by simplicity and to emphasize the role of learning under constraints. However, the essence of what is proposed could be directly extended to the general case of multi-task kernels.

where the parameters $\lambda_h^v > 0$ allow us to weight the contribution of each constraint. It should be noticed that, differently from the previous terms considered in the optimization objective, the penalty term involves all the functions and, thus, explicitly introduces a correlation among the tasks in the learning statement. Finally, we can add together all the contributions yielding the objective $E[\boldsymbol{f}] = R[\boldsymbol{f}] + N[\boldsymbol{f}] + V[\boldsymbol{f}]$.

Since the distributions $p_{(\boldsymbol{x}_k, y_k)}(\boldsymbol{x}_k, y_k)$, $k = 1, \ldots, T$ needed to determine $R[\boldsymbol{f}]$ are usually not known and their estimation is equivalent to the learning task at hand, we apply the common assumption to approximate them through their empirical realizations. This requires to have a set of examples drawn from these unknown distributions. Basically, the learning set will contain a set of labeled examples for each task k such as,

$$\mathcal{L}_k = \left\{\left(\boldsymbol{x}_k^i, y_k^i\right) \mid i = 1, \ldots, \ell_k\right\}. \tag{5}$$

The unsupervised examples are collected in $\mathcal{U}_k = \{\boldsymbol{x}_k^i | i = 1, \ldots, u_k\}$, while $\mathcal{S}_k^L = \{\boldsymbol{x}_k | (\boldsymbol{x}_k, y_k) \in \mathcal{L}_k\}$ collects the sample points that are in the supervised set for the kth task. The set of the supervised and unsupervised points for the kth task is $\mathcal{S}_k = \mathcal{S}_k^L \cup \mathcal{U}_k$.

In this formulation there is no a priori bias for the selection of the samples for the $\mathcal{L}_k$ and $\mathcal{U}_k$ sets. Hence, given an input object we can assume that also a partial labeling can be provided, i.e. it is not required to specify the targets for all the considered tasks for each sample corresponding to the ith instance $\mathcal{X}^i$ of the input tuple. Furthermore, the constraint penalty term generally considers only those examples that are partially supervised or completely unsupervised (i.e. samples derived from input tuples $\mathcal{X}^i$ that are not contained at least in one of the supervised set $\mathcal{L}_k$). In fact, the completely supervised examples are likely to carry little information since the task constraints are already expressed in the provided supervisions that are supposed to be consistent with the given rules. However, the use of the completely labeled examples in the set of points exploited to enforce the constraints may yield some benefits when the labels are affected by noise, but the analysis of this effect is out the scope of this paper. In the following we will refer to the unsupervised set $\mathcal{U} = \{\mathcal{X}^i | \exists k \, : \, \boldsymbol{x}_k^i \in \mathcal{U}_k\}$.

In general, the functionals $\phi_h(\boldsymbol{f})$ implementing the constraints involve all the values computed by the functions in $\boldsymbol{f}$ on their whole domains and it may be complex to provide a closed form that can be efficiently dealt with in the training process. Hence, as in the case of the risk, we assume that these functionals can be conveniently approximated by considering an appropriate sampling in the function domains. In particular, the exact constraint functionals will be replaced by their approximations $\hat{\phi}_h(\mathcal{U}, \boldsymbol{f})$ that exploit only the values of the unknown functions $\boldsymbol{f}$ computed for the points in $\mathcal{U}$. In the next section, we will address the benefits and limitations of this approximation in the case of logic constraints. Therefore, $\phi_h(\boldsymbol{f}) \approx \hat{\phi}_h(\mathcal{U}, \boldsymbol{f})$.

Thus, the given learning problem is cast in a semi-supervised framework where it is assumed that a set of (partially) labeled examples is exploited together with an usually larger set of unlabeled examples. In particular, the choice of the unsupervised examples can be optimized in order to maximize the information available in the joint knowledge of the a priori rules and the labeled examples. Given the available supervised examples in $\mathcal{L}_k$ and an unsupervised sample $\mathcal{U}_k$, $k = 1, \ldots, T$, the objective function considering the empirical risk and the empirical penalty is,

$$E_{emp}[\mathbf{f}] = \sum_{k=1}^{T} \frac{\lambda_k^\tau}{|\mathcal{L}_k|} \sum_{(\boldsymbol{x}_k^j, y_k^j) \in \mathcal{L}_k} L_k^e\left(f_k(\boldsymbol{x}_k^j), y_k^j\right) + \sum_{k=1}^{T} \lambda_k^r \cdot \|f_k\|_{\mathcal{H}_k}^2 + \sum_{h=1}^{H} \lambda_h^v \cdot \hat{\phi}_h(\mathcal{U}, \boldsymbol{f}). \tag{6}$$

The solution to the optimization task defined by the objective function of (6) can be computed by considering the following extension of the Representer Theorem.

Theorem 1 *Given a multitask learning problem for which the task functions* $f_1, \ldots, f_T$, $f_k : \mathbb{R}^{d_k} \to \mathbb{R}$, $k = 1, \ldots, T$, *are assumed to belong to the Reproducing Kernel Hilbert Spaces* $\mathcal{H}_1, \ldots, \mathcal{H}_T$, *respectively, and the problem is formulated as*

$$[f_1^*, \ldots, f_T^*] = \operatorname*{argmin}_{f_1 \in \mathcal{H}_1, \ldots, f_T \in \mathcal{H}_T} E_{emp}[f_1, \ldots, f_T]$$

where $E_{emp}[f_1, \ldots, f_T]$ *is defined as in* (6), *then each function* f_k^* *in the solution can be expressed as*

$$f_k^*(\boldsymbol{x}_k) = \sum_{\boldsymbol{x}_k^i \in \mathcal{S}_k} w_{k,i}^* K_k(\boldsymbol{x}_k^i, \boldsymbol{x}_k)$$

where $K_k(\boldsymbol{x}'_k, \boldsymbol{x}_k)$ *is the reproducing kernel associated to the space* $\mathcal{H}_k$, *and* $\mathcal{S}_k = \mathcal{S}_k^L \cup \mathcal{U}_k$ *is the set of the available samples for the kth task function.*

Proof The proof follows the same scheme as the one of the classic Representer Theorem (Scholkopf and Smola 2001). In fact each function $f_k \in \mathcal{H}_k$ can be decomposed into two components: the projection of f_k in the space spanned by the functions $K_k(\boldsymbol{x}_k^i, \boldsymbol{x}_k)$, $\boldsymbol{x}_k^i \in \mathcal{S}_k$, and the component $v_k(\boldsymbol{x}_k) \in \mathcal{H}_k$ orthogonal to the previous space, that is $\langle K_k(\boldsymbol{x}_k^i, \cdot), v_k(\cdot)\rangle = 0$, $\forall \boldsymbol{x}_k^i \in \mathcal{S}_k$. Using the reproducing property of the kernel $K_k(\boldsymbol{x}'_k, \boldsymbol{x}_k)$, the value of the function in a training point $\boldsymbol{x}_k^j \in \mathcal{S}_k$ can be computed as

$$\begin{aligned} f_k(\boldsymbol{x}_k^j) &= \left\langle \sum_{\boldsymbol{x}_k^i \in \mathcal{S}_k} w_{k,i} K_k(\boldsymbol{x}_k^i, \cdot) + v_k(\cdot), K_k(\boldsymbol{x}_k^j, \cdot) \right\rangle \\ &= \sum_{\boldsymbol{x}_k^i \in \mathcal{S}_k} w_{k,i} \langle K_k(\boldsymbol{x}_k^i, \cdot), K_k(\boldsymbol{x}_k^j, \cdot)\rangle = \sum_{\boldsymbol{x}_k^i \in \mathcal{S}_k} w_{k,i} K_k(\boldsymbol{x}_k^i, \boldsymbol{x}_k^j), \end{aligned}$$

showing that the value of the function computed on the training points depends only on the first component and is independent on $v_k(\cdot)$. Hence the terms in the cost function $E_{emp}[\boldsymbol{f}]$ of (6), corresponding to the empirical risk and constraint contributions, that exploit only the values of the functions f_k computed in the training points $\boldsymbol{x}_k^j \in \mathcal{S}_k$, do not depend on the components $v_k(\cdot)$. Thus the only term affected by these functions is the regularization term whose elements can be written as,

$$\|f_k\|_{\mathcal{H}_k}^2 = \langle f_k, f_k\rangle_{\mathcal{H}_k} = \left\| \sum_{\boldsymbol{x}_k^i \in \mathcal{S}_k} w_{k,i} K_k(\boldsymbol{x}_k^i, \cdot) \right\|_{\mathcal{H}_k}^2 + \|v_k(\cdot)\|_{\mathcal{H}_k}^2,$$

where we exploited the fact that the two components are orthogonal. Hence the only term in the function $E_{emp}[\boldsymbol{f}]$ that depends on the components $v_k(\cdot)$ is $\|v_k(\cdot)\|_{\mathcal{H}_k}^2$, which is clearly minimized by the constant function $v_k(\cdot) = 0$. □

4 Stage-based learning

The optimization of the overall error function is performed in the primal space using gradient descent (Chapelle 2007). However, the objective function $E_{em}[\boldsymbol{f}]$ is non-convex in most interesting problems due to the constraint term, whereas, in case of positive kernel, the strict convexity is guaranteed when restricting the learning to the supervised examples only.

In order to face the problems connected with the emergence of sub-optimal solutions, we propose a solution that is based the following two stages:

1. *Piagetian initialization*: During this phase, we only enforce a regularized fitting of the supervised examples, by setting $\lambda_h^v = 0, h = 1, \ldots, H$, and $\lambda_k^\tau = \lambda^\tau, \lambda_k^r = \lambda^r, k = 1, \ldots, T$, where λ^τ and λ^r are positive constants. This phase terminates according to standard stopping criteria adopted for plain kernel machines.
2. *Abstraction*: During this phase, we start enforcing the constraints by setting $\lambda_h^v = \lambda^v, h = 1, \ldots, H$, where λ^v is a positive constant, whereas λ^τ and λ^r are left unchanged.

Interestingly enough, the two stages herein proposed turn out to be a viable way for tackling complexity issues and suggest a gradual process in which the higher abstraction required to incorporate constraints must follow the classic induction step based on supervised examples. This solution is somewhat related to issues of developmental psychology, since it is well-known that many animals and, especially humans, experiment stage-based learning. According to Piaget (Inhelder and Piaget 1958; Piaget 1961), we can identify four major stages or "periods of development" in child learning, where each stage is self-contained and builds upon the preceding stage. In addition, children seem to proceed through these stages in a universal, fixed order. Even though the four stages described for humans are collapsed to two different stages only, there is an intriguing analogy that mainly involves the distinction between sub-symbolic and symbolic processes. When restricting the learning protocol to examples, simple teaching plans have been proposed (see e.g. Gorse et al. 2004, 1997). Recently, the research in deep learning has shifted the attention on teaching plans in a more systematic way (see e.g. Bengio 2009). However, the two stages involve the structural difference between examples, that involve sub-symbolic learning, and predicates, that are related to symbolic processing and, therefore, to more abstract representations. The notion of developmental intelligent agents has been also the subject of recent explorations in cognitive science (see e.g. Guerin and McKenzie 2008; Guerin 2008; Sloman 2009), where some interesting philosophic foundations have been emerging that could be of interest for further improvement of the simple proposed stage-based developmental scheme. The current solution that has been massively used in our experiments is based on carrying out the global optimization of function (6) using gradient descent. The stage-based solution consists of starting with the first two terms of the function and, later on, to continue by incorporating the penalty term associated with the constraints only after having reached a satisfactory learning performance on the basis of supervised examples only. The idea was preliminarily suggested in (Gori 2009), where parsimonious agents capable of dealing with constraints were introduced using arguments from variational calculus.

Interestingly, the switch from the first to the second stage need not to take place abruptly; alternatively one could use numerical solutions based on continuation methods (Allgower and Georg 2003) that transform the objective function gradually from the initial one to the final target function to be optimized. Of course, the study of the role of the ordering, that in this paper is limited to supervised examples and logic constraints only, is likely to disclose interesting issues on the importance of the ordering of different constraints. Another relevant issue is the purposely selected unsupervised examples for checking the degree of satisfaction of the constraints. All the collection of unsupervised data is used to construct the penalty term, but in real-world problem one might benefit from a careful gradual selection of unsupervised examples. This requires somehow to perform active learning, by selecting those unsupervised examples that are more useful to incorporate the constraints.

While one might provide plenty of arguments from developmental psychology to support stage-based learning by pointing out the inspiration from biology, the most sound analyses to motivate the scheme proposed come from optimization and complexity issues. As already put forward, unlike classic kernel machines, the overall function to be optimized (6) is not convex anymore. In a sense, the first stage, that is based on learning from supervised examples only, with the correspondent guarantee of convergence to an optimal solution, makes it possible to approach the global basin of attraction, while the second stage, in which the constraints are involved, performs a refinement of learning beginning from a good initialization. It is worth mentioning that the constructive interaction between the two learning stages is made possible by the coherence of the supervised pairs with the knowledge represented by the constraints. This qualitative complexity issue suggests that stage-based learning, as discussed in developmental psychology, might not be the outcome of biology, but it could be instead the consequence of optimization principles and complexity issues that hold regardless of the body.

5 Translation of first-order logic clauses into real-valued constraints

When a partial description of the environment is given in terms of logic constraints, in order to follow the approach of learning from constraints of the previous section, one needs to devise a conversion process to translate logic formalisms into real-valued functions. We focus the attention on knowledge-based descriptions given by first-order logic (FOL–KB). The formulas in a KB can be implicitly conjoined, and thus a KB can be viewed as a single large formula. In the following, we indicate by $\mathcal{V} = \{v_1, \ldots, v_N\}$ the set of the variables used in the KB, with $v_s \in \mathcal{D}_s$. Given the set of predicates used in the KB

$$\mathcal{P} = \{p_k | p_k : \mathcal{D}_{s(1,k)} \times \cdots \times \mathcal{D}_{s(n_k,k)} \to \{\mathit{true}, \mathit{false}\},\ k = 1, \ldots, T\},$$

the clauses will be built from the set of atoms

$$\mathcal{A} = \left\{p_{k(i)}(v_{s(1,k(i))}, \ldots, v_{s(n_{k(i)},k(i))}) | i = 1, \ldots, m,\ p_{k(i)} \in \mathcal{P},\ v_{s(j,k(i))} \in \mathcal{V}\right\},$$

where the ith atom is an instance of the $k(i)$th predicate for which the jth argument is assigned to the variable $v_{s(j,k(i))} \in \mathcal{D}_{s(j,k(i))}$. In the following, for the sake of compactness, we will indicate by $\boldsymbol{v}_{a_i} = [v_{s(1,k(i))}, \ldots, v_{s(n_{k(i)},k(i))}]$ the argument list of the atom $a_i \in \mathcal{A}$.

Any FOL clause has an equivalent version in *Prenex Normal form* (PNF), that has all the quantifiers ($\forall, \exists$) and their associated quantified variables at the beginning of the clause. Standard methods exist to convert a generic FOL clause into its corresponding PNF and the conversion can be easily automated. Therefore, with no loss of generality, we restrict our attention to FOL clauses in the PNF form. The quantifier-free part of the expression is equivalent to an assertion in propositional logic for any given assignment of the quantified variables. Since any propositional expression can be written in *Conjunctive Normal Form* (CNF), we can assume that the given PNF-CNF FOL expression is available in the following canonical form

$$\overbrace{[\forall\exists]v_{s(1)} \ldots [\forall\exists]v_{s(Q)}}^{\text{Quantified Portion}} \overbrace{\bigwedge_{c=1,\ldots,C} \left(\bigvee_{d=1,\ldots,D_c} [\neg]\, a_{i(c,d)}(\mathbf{v}_{a_{i(c,d)}}) \right)}^{\text{Quantifier-free CNF expression } E_0(\mathbf{v}_{E_0}, \mathcal{P})} \tag{7}$$

where $a_{i(c,d)} \in \mathcal{A}$ is an atom and the variables $v_{s(q)} \in \mathcal{V}$, $q = 1, \ldots, Q$ constitute the set of the quantified variables. The quantifier-free expression $E_0(\boldsymbol{v}_{E_0}, \mathcal{P})$ depends on the list of arguments $\boldsymbol{v}_{E_0} = [v_{s(1,E_0)}, \dots, v_{s(n_{E_0},E_0)}]$ corresponding to the variables used in all the atoms $a_{i(c,d)}$, i.e. $v_{s(j,E_0)} \in \{v_q \in \mathcal{V} | \exists c, d \; v_q \in \text{args}(a_{i(c,d)})\}$ where $\text{args}(a_{i(c,d)})$ is the set of the variables $\boldsymbol{v}_{a_{i(c,d)}}$ used as arguments in the atom $a_{i(c,d)}$. When all the variables appearing in $\boldsymbol{v}_{E_0}$ are quantified, the resulting expression is a constant that must evaluate to *true*.

We assume that the task functions f_k are exploited to implement the predicates in $\mathcal{P}$ and that the variables in $\mathcal{V}$ correspond to the attributes defining the tuple $\mathcal{X}$ on which the functions f_k are defined. The mapping between $\mathcal{V}$ and the attributes in $\mathcal{X}$ is defined such that $v_s \to x_{j(s)}$ and when the same attribute is referred to by different variables the corresponding instances are assumed to be independent on each other. In this framework, the predicates yield a continuous real value that can be interpreted as a *truth degree*. As we will show in the following, the output values of the functions f_k can be mapped into the interval $[0, 1]$, such that the value 0 is associated with *false* and 1 with *true*.

The FOL–KB will contain a set of clauses corresponding to expressions with no free variables (i.e. all the variables appearing in the expression are quantified) that are assumed to be *true* in the considered domain. These clauses can be converted into a set of constraints as in (1) that can be enforced during the kernel based learning process. The conversion process of a clause into a constraint functional consists of the following three steps:

(I) *Predicate substitution*: substitution in (7) of the predicates with their continuous implementation realized by the functions $\boldsymbol{f}$ composed with a squash function, mapping the output values into the interval $[0, 1]$. In particular, the atom $a_i(\boldsymbol{v}_{a_i})$ is mapped to $\sigma(f_{k(i)}(\boldsymbol{v}_{a_i}))$, where $\sigma : \mathbb{R} \to [0, 1]$ is a monotonically increasing squashing function. A natural choice for the squash function is the piecewise linear mapping $\sigma(y) = \min(1, \max(y, 0))$, this is indeed the function that was employed in the experimental setting.

(II) *Conversion of the Propositional Expression*: conversion of the quantifier-free expression using t-norms as detailed in Sect. 5.1.

(III) *Quantifier conversion*: conversion of the universal and existential quantifiers as shown in Sect. 5.2.

5.1 Logic expressions and their T-norm representation

Any quantifier-free expression defined over the set of atoms $\mathcal{A}$ is equivalent to a sentence in propositional logic, once its variables are assigned to some given value. The expression can be mapped to a function processing real values by relying on the classic association from Boolean expressions to real-valued functions as defined by the *t-norms* (triangular norms) (Klement et al. 2000), commonly used in fuzzy logic (Klir and Yuan 1995).

A t-norm is a function $T : [0, 1] \times [0, 1] \to \mathbb{R}$, that is commutative (i.e. $T(x, y) = T(y, x)$), associative (i.e. $T(x, T(y, z)) = T(T(x, y), z)$), monotonic (i.e. $y \le z \Rightarrow T(x, y) \le T(x, z)$), and featuring a neutral element 1 (i.e. $T(x, 1) = x$). A *t-norm fuzzy logic* is defined by its t-norm $T(x, y)$ that models the logic AND, while the negation of a variable $\neg x$ is computed as $1 - x$. The *t-conorm*, modeling the logical OR, is defined as $1 - T((1 - x), (1 - y))$, as a generalization of the De Morgan's law ($x \vee y = \neg(\neg x \wedge \neg y)$). A t-norm is continuous if $T(x, y)$ is continuous. Many different t-norm logics have been proposed in the literature. In the following we will mainly focus on the *product t-norm* $T(x, y) = x \cdot y$, for which the t-conorm is computed as $1 - (1 - x)(1 - y) = x + y - xy$. Another commonly used t-norm is the *minimum t-norm* defined as $T(x, y) = \min(x, y)$. In this case, the t-conorm corresponds to the function $\max(x, y)$. It is clear from their definitions that both the product and minimum t-norms are continuous. Once defined the t-norm functions corresponding to the logical AND, OR and NOT, these functions can be composed to convert any arbitrary logic proposition. Please note that when using a continuous t-norm, any proposition is converted into a continuous function.

Since any proposition can be transformed into an equivalent CNF form, with no loss of generality, this section will detail the conversion of a clause written in its CNF form. A CNF formula is a collection of maxterms connected by conjunctions (AND). Each maxterm is composed of a set of terms connected by disjunctions (OR). The terms in the maxterms are the predicates appearing either asserted or negated. We will show the conversion using a product t-norm, but it is trivial to derive a similar result for the other t-norms.

Given the disjunction of a set of atomic terms appearing in the mth maxterm of the quantifier-free proposition $E_0(\boldsymbol{v}_{E_0}, \mathcal{P})$ of a clause in the KB, it is possible to express the maxterm as

$$\bigvee_{q \in P^+_{(m,E_0)}} a_q(\boldsymbol{v}_{a_q}) \vee \bigvee_{r \in P^-_{(m,E_0)}} \neg a_r(\boldsymbol{v}_{a_r}) = \neg \left(\bigwedge_{q \in P^+_{(m,E_0)}} \neg a_q(\boldsymbol{v}_{a_q}) \wedge \bigwedge_{r \in P^-_{(m,E_0)}} a_r(\boldsymbol{v}_{a_r}) \right),$$

where $P^+_{(m,E_0)}$ and $P^-_{(m,E_0)}$ are the sets of the indexes of the asserted and negated literals (atoms in $\mathcal{A}$) that appear in this maxterm of the clause. Using the product t-norm, this expression is converted into the function

$$t_{(m,E_0)}(\boldsymbol{v}_{t_{(m,E_0)}}, \boldsymbol{f}) = 1 - \prod_{r \in P^-_{(m,E_0)}} \sigma\left(f_{k(r)}(\boldsymbol{v}_{a_r})\right) \cdot \prod_{q \in P^+_{(m,E_0)}} \left(1 - \sigma\left(f_{k(q)}(\boldsymbol{v}_{a_q})\right)\right),$$

where the squashing function σ must be introduced in the computations since t-norms are only defined for input variables in [0, 1], whereas the functions f_k can yield any real value. The resulting expression depends on all the variables appearing as arguments in the atoms used in the expression, i.e. $\boldsymbol{v}_{t_{(m,E_0)}} = [v_{s(1,t_{(m,E_0)})}, \ldots, v_{s(n_{(m,E_0)},t_{(m,E_0)})}]$, where $v_{s(j,t_{(m,E_0)})} \in \{v_s \in \mathcal{V} | \exists q \in P^+_{(m,E_0)} \cup P^-_{(m,E_0)},\ v_s \in \text{args}(a_q(\boldsymbol{v}_{a_q}))\}$.

The conjunction of the maxterms forming the entire CNF proposition is obtained by multiplying the associated t-norm expressions,

$$t_{E_0}(\boldsymbol{v}_{t_{E_0}}, \boldsymbol{f}) = \prod_{m=1,\ldots,M_{E_0}} t_{(m,E_0)}(\boldsymbol{v}_{t_{(m,E_0)}}, \boldsymbol{f}),$$

where M_{E_0} is the number of maxterms in the CNF expression $E_0(\boldsymbol{v}_{E_0}, \mathcal{P})$ and $\boldsymbol{v}_{t_{E_0}}$ is the argument list containing all the variables in the argument lists $\boldsymbol{v}_{t_{(m,E_0)}}$, $m =, \ldots, M_{E_0}$. Please note that if we require $1 - t_{E_0}(\boldsymbol{v}_{t_{E_0}}, \boldsymbol{f}) = 0$, then each term of the conjunction must be equal to 1 (i.e. the corresponding maxterm needs to be *true*).

Once the logic quantifier-free expression is written using a t-norm, the constraint $1 - t_{E_0}(\boldsymbol{v}^i_{t_{E_0}}, \boldsymbol{f}) = 0$ expresses the fact that the expression must be verified for a given input variable configuration $\boldsymbol{v}^i_{t_{E_0}}$. Hence, a generic CNF logic clause can be enforced by the correspondent functional constraint $\varphi_{E_0}(\boldsymbol{v}^i_{t_{E_0}}, \boldsymbol{f})$ defined as

$$\varphi_{E_0}(\boldsymbol{v}^i_{t_{E_0}}, \boldsymbol{f}) = \left(1 - t_{E_0}(\boldsymbol{v}^i_{t_{E_0}}, \boldsymbol{f})\right) = 0. \tag{8}$$

When the constraint is not verified φ_{E_0} is strictly positive and it can be interpreted as the degree of not satisfaction of the clause of the KB for the given variable configuration $\boldsymbol{v}^i_{t_{E_0}}$.

Finally, it is worth mentioning that each constraint can be satisfied for different configurations of the predicate values that make true the corresponding logic proposition. For instance, if we consider the proposition $a \wedge b \Rightarrow c$, that is always true except for the configuration ($a = \mathit{true}, b = \mathit{true}, c = \mathit{false}$), its implementation using the product t-norm is $1 - ab(1 - c)$ and the corresponding constraint is $abc - ab \geq 0$. This constraint is satisfied, when $a = 0, b, c \in [0, 1]$ or $b = 0, a, c \in [0, 1]$ or $c = 1, a, b \in [0, 1]$.

T-norms allow the mapping of any arbitrary quantifier-free expression $E(\boldsymbol{v}_E, \mathcal{P})$ to a functional constraint $\varphi_E(\boldsymbol{v}_E, \boldsymbol{f}) = 0$, depending on all the variables collected in the argument list $\boldsymbol{v}_E = [v_{s(1,E)}, \ldots, v_{s(n_E,E)}]$ and on the predicates implemented by the functions $\boldsymbol{f}$.

5.2 Quantifier conversion

The quantified portion of the expression is processed recursively by moving backward from the inner quantifier in the PNF expansion.

Let us consider the universal quantifier first. The universal quantifier expresses the fact that the expression must hold for any realization of the quantified variable v_q. When considering the real-valued mapping of the original boolean expression, the universal quantifier can be naturally converted measuring the degree of non-satisfaction of the expression over the domain $\mathcal{D}_{j(q)}$ where the feature vector $x_{j(q)}$, corresponding to the variable v_q, ranges. This measure can be implemented by computing the overall distance of $\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})$, that is the degree of violation associated to the quantified expression, from the constant function equal to 0 (this is the only value for which the constraint is always verified), over the domain $\mathcal{D}_{j(q)}$. Measuring the distance using the infinity norm yields

$$\begin{aligned}&\forall v_q\ E(\mathbf{v}_E, \mathcal{P}) \to \|\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})\|_\infty \\ &\quad = \lim_{p\to\infty} \left(\int_{v_q \in \mathcal{D}_{j(q)}} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})|^p \, p_{x_{j(q)}}(v_q) dv_q \right)^{\frac{1}{p}} = \sup_{v_q \in \mathcal{D}_{j(q)}} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})|, \qquad (9)\end{aligned}$$

where the resulting expression depends on all the variables in $\boldsymbol{v}_E$ except v_q. Hence, the result of the conversion applied to the expression $E_q(\boldsymbol{v}_{E_q}, \mathcal{P}) = \forall v_q\ E(\boldsymbol{v}_E, \mathcal{P})$ is a functional $\varphi_{E_q}(\boldsymbol{v}_{E_q}, \boldsymbol{f})$, assuming values in $[0, 1]$ and depending on the set of variables $\boldsymbol{v}_{E_q} = [v_{s(1,E_q)}, \ldots, v_{s(n_{E_q},E_q)}]$, such that $n_{E_q} = n_E - 1$ and $v_{s(j,E_q)} \in \{v_r \in \mathcal{V} | \exists i\ v_r = v_{s(i,E)},\ v_r \neq v_q\}$. The variables in $\boldsymbol{v}_{E_q}$ need to be quantified or assigned a specific value in order to obtain a constraint functional depending only on the functions $\boldsymbol{f}$.

Theorem 2 *Let $E(v, \mathcal{P})$ be an FOL expression with no quantifiers depending on the variable v. Let $t_E(v, \boldsymbol{f})$ be the t-norm representation of E, obtained using a continuous t-norm, where f_k corresponds to p_k, $k = 1, \ldots, T$. If $f_k \in \mathbb{C}^0$, $k = 1, \ldots, T$, then $\|1 - t_E(v, \boldsymbol{f})\|_p = 0$ iff $\forall v\ E(v, \mathcal{P})$ is true.*

Proof $\Rightarrow$. $t_E(v, \boldsymbol{f}) \in \mathbb{C}^0$, since it is obtained by composing continuous functions. $t_E(v, \boldsymbol{f})$ is also non-negative being a t-norm. Now, suppose that $\|1 - t_E(v, \boldsymbol{f})\|_p = 0$ but it does not hold that $\forall v\ E(v, \mathcal{P})$ is *true*. If $\forall v\ E(v, \mathcal{P})$ is *false* than its negation must be *true*: $\neg\forall v\ E(v, \mathcal{P}) \equiv \exists v\ \neg E(v, \mathcal{P})$. This means that it must exist at least one instance v' such that $E(v', \mathcal{P})$ is *false*. If $E(v', \mathcal{P})$ is *false* then $t_E(v', \boldsymbol{f}) = 0$. Since $t_E(v, \boldsymbol{f}) \in \mathbb{C}^0$, for any ϵ such that $0 < \epsilon < 1$, it is possible to find a $\delta > 0$ such that $\forall \Delta v :\ \|\Delta v\| < \delta,\ 1 - t_E(v' + \Delta v, \boldsymbol{f}) > 1 - \epsilon > 0$. Since t_E is non-negative, this implies that, when computing the distance of the constraint from the target constant value 0 using any p-norm, $\|1 - t_E(v, \boldsymbol{f})\|_p \geq \Omega(\delta, p)(1 - \epsilon) > 0$, where $\Omega(\delta, p)$ is a positive value depending on the measure of the region where the constraint is violated. Hence, the assumption leads to a contradiction.

$\Leftarrow$. If $\forall v\ E(v, \mathcal{P})$ holds, $t_E(v, \boldsymbol{f})$ is a constant function equal to 1 for each v, and $\|1 - t_E(v, \boldsymbol{f})\| = 0$ holds for any functional norm. □

Theorem 2 shows that there is a duality between an universally quantified expression and its continuous generalization. It is therefore possible to test whether the expression holds by checking the value of the converted expression. If we consider the conversion of the PNF representing a FOL constraint without free variables, the variables are recursively quantified until the set of the free variables is empty. In the case of the universal quantifier we apply again the mapping described previously. The existential quantifier can be realized by enforcing the De Morgan law to hold also in the continuous mapped domain. The De Morgan law states that

$$\exists v_q\ E(\boldsymbol{v}_E, \mathcal{P}) \Longleftrightarrow \neg\forall v_q\ \neg E(\boldsymbol{v}_E, \mathcal{P}).$$

Using the conversion of the universal quantifier defined in (9), we obtain the following conversion for the existential quantifier

$$\exists v_q\ E(\boldsymbol{v}_E, \mathcal{P}) \rightarrow \inf\nolimits_{v_q \in \mathcal{D}_{j(q)}} \varphi_E(\boldsymbol{v}_E, \boldsymbol{f}).$$

Example 1 Let $a(\cdot), b(\cdot)$ be two unary predicates, implemented by the functions $f_a(\cdot), f_b(\cdot)$. The clause $\forall v_1\ \forall v_2\ a(v_1) \vee b(v_2)$ is converted in three steps as follows.

I. Conversion of the atoms $a(v_1)$ and $a(b_2)$.

$$a(v_1) \rightarrow \sigma(f_a(v_1)), \qquad b(v_2) \rightarrow \sigma(f_b(v_2)).$$

II. Conversion of the quantifier free expression $E_0([v_1, v_2], \{a(\cdot), b(\cdot)\}) = a(v_1) \vee b(v_2)$ using T-norms.

$$t_{E_0}([v_1, v_2], [f_a, f_b]) = \sigma(f_a(v_1)) + \sigma(f_a(v_2)) - \sigma(f_a(v_1))\sigma(f_b(v_2)).$$

III. Conversion of the universal quantifiers for the variables v_1 and v_2. First the quantifier free expression $E_0([v_1, v_2], \{a(\cdot), b(\cdot)\})$ is converted into the distance measure

$$\varphi_{E_0}([v_1, v_2], [f_a, f_b]) = 1 - \sigma(f_a(v_1)) - \sigma(f_a(v_2)) + \sigma(f_a(v_1))\sigma(f_b(v_2)).$$

Then, the two universal quantifiers are converted using the infinity norm, yielding the constraint

$$\varphi_E([], [f_a, f_b]) = \sup_{v_1 \in \mathcal{D}_1} \sup_{v_2 \in \mathcal{D}_2} \big(1 - \sigma(f_a(v_1)) - \sigma(f_a(v_2)) + \sigma(f_a(v_1))\sigma(f_b(v_2))\big) = 0.$$

Using the same procedure it is easy to show that the clause $\forall v_1\ \exists v_2\ a(v_1) \vee b(v_2)$ is mapped to the constraint

$$\sup_{v_1 \in \mathcal{D}_1} \inf_{v_2 \in \mathcal{D}_2} \big(1 - \sigma(f_a(v_1)) - \sigma(f_b(v_2)) + \sigma(f_a(v_1))\sigma(f_b(v_2))\big) = 0.$$

Example 2 We consider the case when the same predicate is used in different atoms. Let us consider the clause

$$\forall v_1 \forall v_2\ r(v_1, v_2) \Rightarrow (a(v_1) \wedge a(v_2)) \vee (\neg a(v_1) \wedge \neg a(v_2)),$$

that states that when the relationship $r(v_1, v_2)$ holds between two items, then the predicate $a(v)$ should yield the same value for both of them. In this case $\mathcal{P} = \{r(\cdot,\cdot), a(\cdot)\}$ and $\mathcal{A} = \{r(v_1, v_2), a(v_1), a(v_2)\}$. The first conversion step yields

$$a(v_1) \to \sigma(f_a(v_1)), \qquad a(v_2) \to \sigma(f_a(v_1)), \qquad r(v_1, v_2) \to \sigma(f_r(v_1, v_2)).$$

The quantifier free expression $E_0([v_1, v_2], \{a(\cdot), r(\cdot,\cdot)\})$ corresponds to the T-norm function

$$\begin{aligned} t_{E_0}([v_1, v_2], [f_a, f_r]) = 1 - \sigma(f_r(v_1, v_2))(1 - \sigma(f_a(v_1))\sigma(f_a(v_2))) \\ \times (\sigma(f_a(v_1)) + \sigma(f_a(v_2)) - \sigma(f_a(v_1))\sigma(f_a(v_2))). \end{aligned}$$

Finally the quantification of the two variables v_1 and v_2 ranging in the same domain $\mathcal{D}$ yields the constraint

$$\begin{aligned} \sup_{v_1 \in \mathcal{D}} \sup_{v_2 \in \mathcal{D}} (\sigma(f_r(v_1, v_2))(1 - \sigma(f_a(v_1))\sigma(f_a(v_2))) \\ \times (\sigma(f_a(v_1)) + \sigma(f_a(v_2)) - \sigma(f_a(v_1))\sigma(f_a(v_2)))) = 0. \end{aligned}$$

Unfortunately, it is generally complex to compute the exact expression for the functionals since the conversion of the quantifiers requires to extend the computation on the whole domain of the quantified variables, considering the feature distributions $p_{x_j}(x_j)$. Hence, we assume that the computation can be approximated by exploiting the available empirical realizations of the feature vectors. If we consider the examples available for training, both supervised and unsupervised, we can extract the empirical distribution $\mathcal{S}_{x_j}$ for the feature x_j by considering all the instances of the tuples $\mathcal{X}^i$. Hence, the universal quantifier exploiting the infinity norm is approximated as

$$\forall v_q \; E(\boldsymbol{v}_E, \mathcal{P}) \to \max_{v_q \in \mathcal{S}_{x_{j(q)}}} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})|.$$

Similarly, for the existential quantifier it holds

$$\exists v_q \; E(\boldsymbol{v}_E, \mathcal{P}) \to \min_{v_q \in \mathcal{S}_{x_{j(q)}}} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})|.$$

It is interesting to note that the $\|\cdot\|_\infty$ norm in the empirical case defines the universal quantifier as the Minimum T-norm representation of the conjunction of the values of the loss of the expression evaluated over each point of the sample set. The existential quantifier instead corresponds to the Minimum T-norm representation of the disjunction of the loss of the expression evaluated over each point of the sample set.

Table 1 highlights the conversion rules that allow the mapping of any FOL clause into a continuous constraint.

It is also possible to select a different functional norm to convert the universal quantifier. However, these alternative norms are not consistent with the DeMorgan law even if they feature nice averaging properties, which make them a preferable choice when the resulting constraint must be integrated into a cost function to be optimized (e.g. soft enforcing of the constraint). For example, when using the $\|\cdot\|_1$ norm, the universal quantifier is implemented as

$$\forall v_q \; E(\boldsymbol{v}_E, \mathcal{P}) \to \|\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})\|_1 = \int_{v_q \in \mathcal{D}_{j(q)}} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})| \, p_{x_{j(q)}}(v_q) \, dv_q. \qquad (10)$$

Table 1 Rules for the three step conversion of a PNF clause. After the conversion of atoms in step I, the rules of step II are applied recursively to convert a quantifier free expression $E_0(\boldsymbol{v}_{E_0}, \mathcal{P})$ into its T-norm implementation $t_{E_0}(\boldsymbol{v}_{E_0}, \boldsymbol{f})$. Finally, the rules in step III allow the recursive definition of the final constraint $\varphi_E([], \boldsymbol{f}) = 0$, where $\varphi_E([], \boldsymbol{f})$ is the functional obtained after the quantification of all the variables in the argument list $\boldsymbol{v}_{E_0}$

Step	Expression	Mapping
I	$a_i(\boldsymbol{v}_{a_i})$	$\sigma(f_{k(i)}(\boldsymbol{v}_{a_i}))$
II	$\neg E(\boldsymbol{v}_E, \mathcal{P})$	$1 - t_E(\boldsymbol{v}_E, \boldsymbol{f})$
	$E_1(\boldsymbol{v}_{E_1}, \mathcal{P}) \wedge E_2(\boldsymbol{v}_{E_2}, \mathcal{P})$	$t_{E_1}(\boldsymbol{v}_{E_1}, \boldsymbol{f}) \cdot t_{E_2}(\boldsymbol{v}_{E_2}, \boldsymbol{f})$
	$E_1(\boldsymbol{v}_{E_1}, \mathcal{P}) \vee E_2(\boldsymbol{v}_{E_2}, \mathcal{P})$	$1 - (1 - t_{E_1}(\boldsymbol{v}_{E_1}, \boldsymbol{f}))(1 - t_{E_2}(\boldsymbol{v}_{E_2}, \boldsymbol{f}))$
	$E_1(\boldsymbol{v}_{E_1}, \mathcal{P}) \Rightarrow E_2(\boldsymbol{v}_{E_2}, \mathcal{P})$	$1 - t_{E_1}(\boldsymbol{v}_{E_1}, \boldsymbol{f})(1 - t_{E_2}(\boldsymbol{v}_{E_2}, \boldsymbol{f}))$
III	$E_0(\boldsymbol{v}_{E_0}, \mathcal{P})$	$\varphi_{E_0}(\boldsymbol{v}_{E_0}, \boldsymbol{f}) = 1 - t_{E_0}(\boldsymbol{v}_{E_0}, \boldsymbol{f}))$
	$E_q(\boldsymbol{v}_{E_q}, \mathcal{P}) = \forall v_q \ E_1(\boldsymbol{v}_{E_1}, \mathcal{P})$	$\varphi_{E_q}(\boldsymbol{v}_{E_q}, \boldsymbol{f}) = \max_{v_q \in \mathcal{S}_{x_{j(q)}}} \varphi_{E_1}(\boldsymbol{v}_{E_1}, \boldsymbol{f})$
	$E_q(\boldsymbol{v}_{E_q}, \mathcal{P}) = \exists v_q \ E_1(\boldsymbol{v}_{E_1}, \mathcal{P})$	$\varphi_{E_q}(\boldsymbol{v}_{E_q}, \boldsymbol{f}) = \min_{v_q \in \mathcal{S}_{x_{j(q)}}} \varphi_{E_1}(\boldsymbol{v}_{E_1}, \boldsymbol{f})$

Using the empirical distribution for the feature x_j, the integral can be approximated as a sum over the set $\mathcal{S}_{x_j}$, yielding the conversion rule

$$\forall v_q \ E(\boldsymbol{v}_E, \mathcal{P}) \to \frac{1}{|\mathcal{S}_{x_{j(q)}}|} \sum_{v_q \in \mathcal{S}_{x_{j(q)}}} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{f})|.$$

Please note that $\varphi_E([], \boldsymbol{f})$ will always reduce to the following form, when computed for an empirical distribution of data for any selected functional norm,

$$\varphi_E([], \boldsymbol{f}) = \mathrm{O}_{v_{s(1)} \in \mathcal{S}_{x_{j(s(1))}}} \cdots \mathrm{O}_{v_{s(Q)} \in \mathcal{S}_{x_{j(s(Q))}}} t_{E_0}(\boldsymbol{v}_{E_0}, \boldsymbol{f}), \tag{11}$$

where $\mathrm{O}_{v_q \in \mathcal{S}_{x_{j(q)}}}$ specifies the aggregation operator to be computed on the sample set $\mathcal{S}_{x_{j(q)}}$ for each quantified variable v_q. In the case of the infinity norm, $\mathrm{O}_{v_q \in \mathcal{S}_{x_{j(q)}}}$ is either the minimum or maximum operator over the set $\mathcal{S}_{x_{j(q)}}$. Therefore, the presented conversion procedure implements the logical constraint depending on the realizations of the functions over the data point samples. For this class of constraints, Theorem 1 holds and the optimal solution can be expressed as a kernel expansion over the data points. In fact, since the constraint is represented by $\varphi_E([], \boldsymbol{f}) = 0$ in the definition of the learning objective function of (6) we can substitute $\hat{\phi}(\mathcal{U}, \boldsymbol{f}) = \varphi_E([], \boldsymbol{f})$.

When using the minimum and/or maximum operators for defining $\hat{\phi}(\mathcal{U}, \boldsymbol{f})$, the resulting objective function is continuous with respect to the parameters $w_{k,i}$ defining the RKHS expansion, since it is obtained by combining continuous functions. However, in general, its derivatives are no more continuous. In practice, this is not a problem for gradient descent based optimization algorithms once appropriate stopping criteria are applied. In particular, the optimal minima can be located also in configurations corresponding to discontinuities in the gradient values, i.e. when a maximum or minimum operator switches its choice among two different points in the dataset. Given the current configuration of the parameters $w_{k,i}$ in order to compute the gradient, first the variable configuration $[v^*_{s(1)}, \ldots, v^*_{s(Q)}]$ is computed for the minimum/maximum operators by using the current estimate for the functions $\boldsymbol{f}$. Then, using this configuration the gradient of $\hat{\phi}(\mathcal{U}, \boldsymbol{f})$ is computed by considering the function $t_{E_0}([v^*_{s(1)}, \ldots, v^*_{s(Q)}], \boldsymbol{f})$.

5.3 Complexity issues

The computation of the functional implementing a FOL clause needs to perform a linear scan over all the realizations of each variable. Since the variables are nested, all possible combinations of the variables must be generated. Let b_i be the number of realizations of the ith variable, the total number of combinations that are generated to evaluate the satisfaction of a constraint is equal to $\prod_{i=1}^{n} b_i$, where n is the number of variables in the FOL clause. It is clear that this process can quickly become intractable, as soon as a clause involves a significant number of input variables or the samples are large. This is a direct effect of the fact that FOL does not assume any a-priori correlation among the variables, and this forces to verify a clause over all the possible combinations of the inputs.

However, there are cases where the some of the variables are correlated and this is modeled by the unknown joint distribution $p_{\mathcal{X}}(x_1, \ldots, x_n)$. Hence, there exist some configurations of the quantified variables that are not allowed or are very unlikely to appear. The complexity could be significantly reduced by exploiting the correlations in the evaluation of the operators associated to the quantifiers. In FOL, this happens when a clause is in the following form

$$\forall v_{s(1)} \ldots \forall v_{s(m)} [\forall\exists] v_{s(m+1)} \ldots [\forall\exists] v_{s(Q)}\ r(v_{s(1)}, \ldots, v_{s(m)}) \Rightarrow E(\boldsymbol{v}_E, \mathcal{P}), \tag{12}$$

where $r(\cdot)$ is a given (not to be estimated) predicate modeling the strength of the correlation among a subset of universally quantified variables and $E(\boldsymbol{v}_E, \mathcal{P})$ is a generic quantifier-free logic expression. If $t_E(\boldsymbol{v}_E, \boldsymbol{f})$ is the T-norm representation of $E(\boldsymbol{v}_E, \mathcal{P})$, the expression is converted into

$$\max_{v_{s(1)}} \ldots \max_{v_{s(m)}} [\max_{v_{s(m+1)}} \min] \ldots [\max_{v_{s(Q)}} \min]\, \tau_r(v_{s(1)}, \ldots, v_{s(m)})(1 - t_E(\boldsymbol{v}_E, \boldsymbol{f})),$$

where $\tau_r(v_{s(1)}, \ldots, v_{s(m)})$ is the task function used to implement the predicate $r(\cdot)$. When the relation does not hold for a given configuration of the variables, $\tau_r(\cdot)$ is equal to zero and it gives no contribution to the expression. Therefore, only the variable configurations for which the relation holds, i.e. $\tau_r(v_{s(1)}, \ldots, v_{s(m)}) > 0$, can be considered and the converted expression can be evaluated as

$$\max_{[v_{s(1)}, \ldots, v_{s(m)}] \in \mathcal{R}} [\max_{v_{s(m+1)}} \min] \ldots [\max_{v_{s(Q)}} \min]\, \tau_r(v_{s(1)}, \ldots, v_{s(Q)})(1 - t_E(\boldsymbol{v}_E, \boldsymbol{f})),$$

where $\mathcal{R}$ is an observed sample from the distribution of the variable configuration for which the relation $r(\cdot)$ holds. In this special case, the computation is efficient as only the configurations for which the relation holds are considered instead of all the possible combinations of the values for all the input variables.

Indeed, when the clause is in the form of (12), it is possible to trivially exploit the fact that the variables are not independent as in the most general setting. Since, the correlation among the variables is expressed by the relation, it becomes possible to sample from the joint distribution instead from the single distributions of the variables. This keeps the complexity linear in the size of the sample for the set of variables $[v_{s(1)}, \ldots, v_{s(m)}]$ involved in the relation.

6 Experimental results

The experimental analysis has been carried out on artificial benchmarks properly created to emphasize the comparisons with plain kernel machines. The generated training datasets contain a set of partially labeled examples, a set of unsupervised examples and a test set with 100 patterns per class. For each task, some prior knowledge in the form of FOL logic clauses is assumed to be available to partially describe the classification problem. In the experiments the targets $\{1, 0\}$ were used to represent the $\{true, false\}$ values for the supervised examples. This setting biases the solution towards the *false* value since the regularization term, that depends on the RKHS function norm, tends to favor a constant solution equal to 0. This may be an useful property for those cases in which the negative class is not well described by the given examples, as it happens for instance in verification tasks. However, it is straightforward to redefine the task in an unbiased setting by mapping the logic values to $\{-1, 1\}$ as it is usually done in classification tasks.

6.1 Benchmark 1

The dataset used in the first synthetic experiment is composed by patterns in $\mathbb{R}^2$ and belonging to three classes, A, B, C, where the patterns of class A, B, C are uniformly distributed over the rectangles $\{(x, y) : x \in [0, 2], y \in [0, 1]\}$, $\{(x, y) : x \in [1, 3], y \in [0, 1]\}$, $\{(x, y) : x \in [1, 2], y \in [0, 2]\}$, respectively. For such a task, we suppose that the following clauses are known to hold a-priori:

– CLAUSE 1 states that the intersection between the classes A and B is contained into the boundaries of class C. This statement can be written in FOL as $\forall \mathbf{x}\ a(\mathbf{x}) \wedge b(\mathbf{x}) \Rightarrow c(\mathbf{x})$, where $a(\mathbf{x}), b(\mathbf{x}), c(\mathbf{x})$ are three predicates indicating whether the pattern $\mathbf{x}$ belongs to the classes A, B, C, respectively.
– CLAUSE 2 expresses the fact that any pattern must belong to at least one class, i.e. $\forall \mathbf{x} \neg a(\mathbf{x}) \wedge \neg b(\mathbf{x}) \Rightarrow c(\mathbf{x}) \equiv a(\mathbf{x}) \vee b(\mathbf{x}) \vee c(\mathbf{x})$.

Using and not-using prior knowledge in learning This experiment compares the classification accuracy obtained when integrating or not integrating the logic clauses into the learning process.

The parameters λ^r, λ^τ and λ^v have been set to 0.25, 1 and 1.5, respectively. We exploited a Gaussian kernel with variance equal to 0.1. The values for these parameters have been determined via exhaustive search during a validation procedure. During different runs, the training set size has been increased from 6 to 132 examples. Similarly, the unsupervised data was varied between 0 and 300 patterns.

Figure 1 reports the classification accuracy, averaged over 5 random generations of the train and test patterns. The improvement is particularly significant when the constraints are enforced also on the unsupervised data, since the additional points allow a more precise definition of the class (see the plots in Fig. 2). However, we can notice a performance increase even when applying the constraints only on the partially labeled examples, i.e. no completely unsupervised examples are used. A one-tailed t-test showed that the gains are statistically significant (at least 95% confidence).

Adding more prior knowledge This experiment investigates the effect of adding additional prior knowledge. Starting from the same setting of the previous experiment, we considered two additional clauses:

– CLAUSE 3 states that any pattern of class A and C belongs to class B, i.e. $\forall \mathbf{x}\ a(\mathbf{x}) \wedge c(\mathbf{x}) \Rightarrow b(\mathbf{x}) \equiv \forall \mathbf{x}\ \neg a(\mathbf{x}) \vee b(\mathbf{x}) \vee \neg c(\mathbf{x})$.
– CLAUSE 4 states that any pattern of class B and C belongs to class A, i.e. $\forall \mathbf{x}\ b(\mathbf{x}) \wedge c(\mathbf{x}) \Rightarrow a(\mathbf{x}) \equiv \forall \mathbf{x}\ a(\mathbf{x}) \vee \neg b(\mathbf{x}) \vee \neg c(\mathbf{x})$.

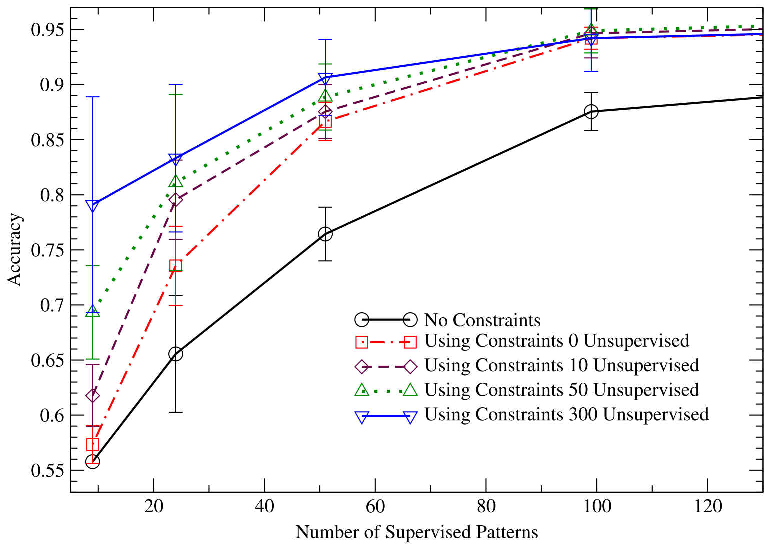


Fig. 1 Benchmark 1. Classification accuracy for different labeled and unlabeled datasets when using or not using the constraints in training

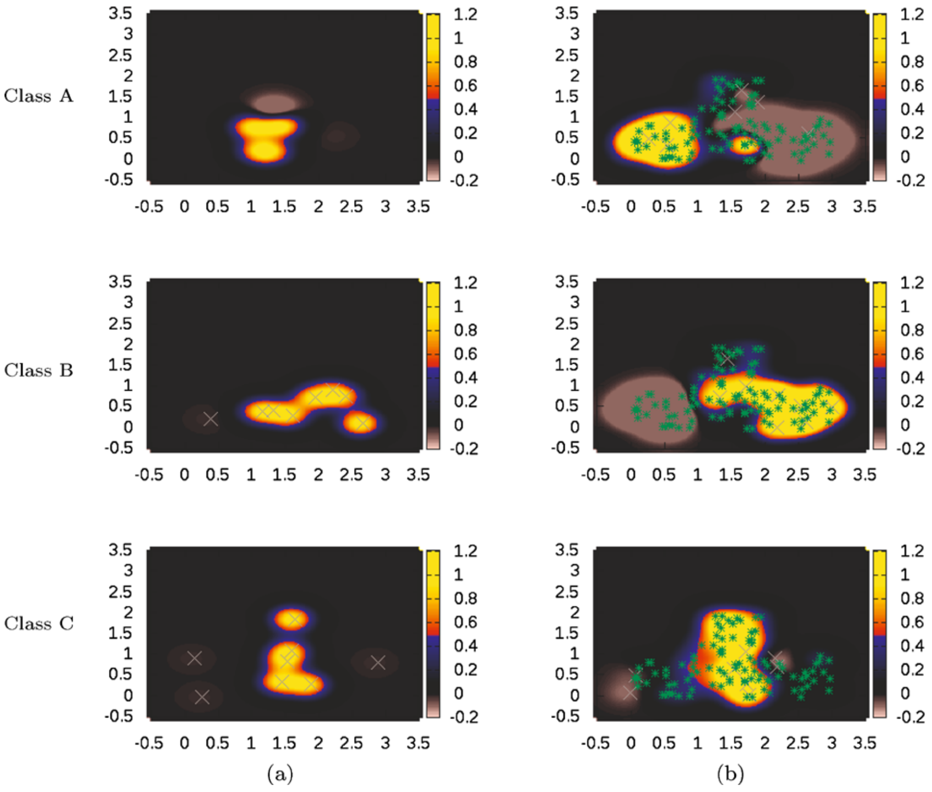


Fig. 2 Benchmark 1. Activation maps for classes A, B, C when (**a**) no constraints, and (**b**) 100 unsupervised patterns are used. The classifier was trained using 25 supervised patterns

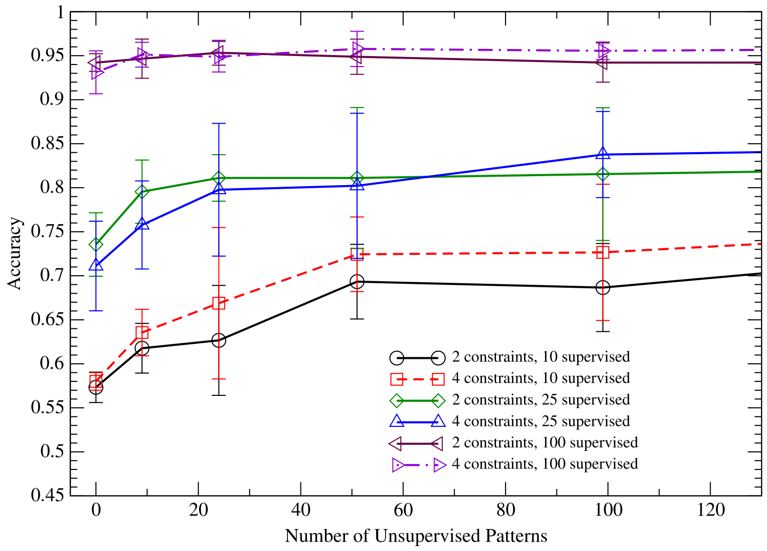


Fig. 3 Benchmark 1. Accuracy of the classifier for different numbers of supervised and unsupervised patterns when using CLAUSE 1, 2 or 1, 2, 3, 4

Figure 3 reports the accuracy values for different numbers of supervised and unsupervised patterns, obtained by integrating only CLAUSE 1 and 2 in the training process or all the available prior knowledge (CLAUSE 1, 2, 3 and 4). The reported results are an average over 5 random generations of the train and test patterns and show that the classification performances increase when more a priori knowledge is added. A one-tailed t-test showed that the gains are statistically significant with over 95% probability.

Adding an existential quantified clause This experiment aims at testing the effect of existentially quantified clauses. The setting is the same of the previous experiments with the addition of the following FOL constraint (CLAUSE 5):

$$\forall \mathbf{x}\ a(\mathbf{x}) \wedge b(\mathbf{x}) \Rightarrow \exists \mathbf{y}\ r(\mathbf{x}, \mathbf{y}) \wedge c(\mathbf{y}),$$

where $r(\mathbf{x}, \mathbf{y})$ is a known relationship between $\mathbf{x}$ and $\mathbf{y}$, such that $r(\mathbf{x}, \mathbf{y}) = \mathit{TRUE} \iff \mathbf{y} = \mathbf{x} + [0, 1]'$. This rule describes the fact that for each pattern $\mathbf{x}$ laying in the intersection of the classes A and B, a pattern $\mathbf{y}$ exists which is related to $\mathbf{x}$ according to $r(\mathbf{x}, \mathbf{y})$ and which belongs to the class C. This rule can be used by itself or added to a KB including other clauses. In particular, we tested the effect of the rule together with CLAUSE 1 and 2 as defined in the previous settings. Please note that CLAUSE 1 and 5 together allow to perfectly determine the boundaries for class C, provided that a sufficient number of examples of class A and B is provided. Therefore, unlike in the previous experiments, no supervised examples for class C have been provided to emphasize the effect of the rules.

Figure 4 shows the accuracy that can be obtained by adding CLAUSE 5.

Using a polynomial kernel The previous experiments have been carried out using a Gaussian kernel. This experiment is based on the same experimental dataset and setting as the previous experiments but focuses on the effect of the constraints when a polynomial kernel

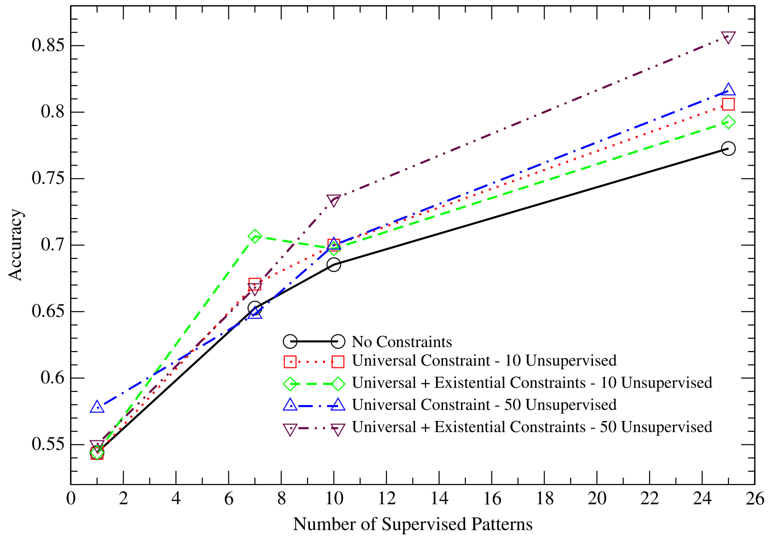


Fig. 4 Benchmark 1. Accuracy when using or not using the existentially quantified rule 5 together with the universally quantified clauses 1 and 2

Table 2 Classification accuracy on the benchmark 1 when using a polynomial kernel

Supervised	Polynomial			
	Unsupervised			
	No	50	300	1000
50	0.899	0.9	0.905	0.909
100	0.909	0.912	0.92	0.926
300	0.935	0.936	0.942	0.949
1000	0.955	0.953	0.953	0.954

is selected. Table 2 shows the classification accuracy, confirming that the accuracy is positively affected by the embedding of the constraints. The gains are particularly significant when a small number of supervised examples is used. As a general trend, the accuracy tends to increase as more unsupervised data is used in learning.

Increasing the input space dimensionality This experiment studies how the dimensionality of the input feature space affects the classification performances. A set of artificial datasets was created by using the same data distributions as in the previous experiments but adding a variable number of new dimensions, according to the following geometry:

$$A = \{\mathbf{x} : 0 \le x_1 \le 2,\ 0 \le x_2 \le 2,\ 0 \le x_3 \le 1, \ldots, 0 \le x_n \le 1\}$$

$$B = \{\mathbf{x} : 1 \le x_1 \le 3,\ 0 \le x_2 \le 2,\ 0 \le x_3 \le 1, \ldots, 0 \le x_n \le 1\}$$

$$C = \{\mathbf{x} : 1 \le x_1 \le 2,\ 0 \le x_2 \le 2,\ 0 \le x_3 \le 1, \ldots, 0 \le x_n \le 1\}$$

In particular, the patterns have been generated in $\mathbb{R}^{10}$, $\mathbb{R}^{50}$ and $\mathbb{R}^{100}$. Given an uniform sampling over the hyper-rectangles, a higher dimensional input space corresponds to sparser

Table 3 Classification accuracy for benchmark 1 for patterns in $\mathbb{R}^{10}$, $\mathbb{R}^{50}$ and $\mathbb{R}^{100}$ and using Gaussian (*left*) and polynomial (*right*) kernels. In bold the cases reporting statistically significant improvements

Dim.	Supervised	Gaussian				Polynomial			
		Unsupervised				Unsupervised			
		No	50	300	1000	No	50	300	1000
$\mathbb{R}^{10}$	50	0.884	**0.897**	**0.903**	**0.91**	0.829	**0.84**	**0.844**	**0.851**
	100	0.91	0.92	**0.928**	**0.934**	0.864	0.871	**0.882**	**0.881**
	300	0.958	0.955	0.956	0.956	0.931	0.922	0.925	0.927
	1000	0.975	0.97	0.969	0.971	0.954	0.944	0.945	0.946
$\mathbb{R}^{50}$	50	0.5	**0.559**	0.512	**0.517**	0.7	**0.732**	**0.73**	**0.756**
	100	0.501	**0.675**	**0.634**	**0.637**	0.75	0.76	**0.776**	**0.78**
	300	0.727	**0.834**	**0.817**	**0.789**	0.8	**0.819**	**0.82**	**0.818**
	1000	0.887	0.884	0.879	0.868	0.86	**0.888**	**0.883**	**0.89**
$\mathbb{R}^{100}$	50	0.504	**0.703**	**0.675**	**0.681**	0.72	**0.786**	**0.784**	**0.772**
	100	0.598	**0.806**	**0.805**	**0.787**	0.76	**0.813**	**0.803**	**0.8**
	300	0.848	**0.876**	**0.875**	**0.86**	0.843	0.859	**0.867**	**0.865**
	1000	0.925	0.916	0.914	0.912	0.923	0.922	0.926	0.924

training data for a fixed number of labeled patterns. This is an effect of the well known *curse-of-dimensionality*, making generalization more difficult in high dimensional input spaces.

The classification accuracy has been evaluated when employing either a Gaussian or a polynomial kernel. Table 3 reports the obtained results, averaged over 5 different instances of the supervised, unsupervised and test sets. The table shows that the joint employment of the constraints and the unlabeled data improves the classification accuracy in all the settings for both kernels. However, the accuracy gains for the Gaussian kernel are in general more significant than when a polynomial kernel is used. This is due to the fact that a kernel with limited support (as the Gaussian) requires a large number of points to create appropriate decision boundaries in high dimensional spaces. It can therefore benefit more from the availability of a large sample of unsupervised data.

6.2 Benchmark 2: 7 classes, 4 clauses

This experiment validates the effectiveness of the two-stage learning process to optimize the cost function. The multi-task classification problem is based on 7 different classes (A, B, C, D, E, F, G), whose indicator functions are realized by the predicates $a(\mathbf{x}), b(\mathbf{x}), c(\mathbf{x}), d(\mathbf{x}), e(\mathbf{x}), f(\mathbf{x}), g(\mathbf{x})$. The classes are known to be arranged according to a hierarchy defined by the following clauses: $\forall \mathbf{x}\, a(\mathbf{x}) \wedge b(\mathbf{x}) \Rightarrow c(\mathbf{x})$, $\forall \mathbf{x}\, d(\mathbf{x}) \wedge e(\mathbf{x}) \Rightarrow f(\mathbf{x})$, $\forall \mathbf{x}\, c(\mathbf{x}) \wedge f(\mathbf{x}) \Rightarrow g(\mathbf{x})$ and $\forall \mathbf{x}\, a(\mathbf{x}) \vee b(\mathbf{x}) \vee c(\mathbf{x}) \vee d(\mathbf{x}) \vee e(\mathbf{x}) \vee f(\mathbf{x}) \vee g(\mathbf{x})$. The patterns for each class are uniformly distributed over the following rectangles:

$$A = \{(x, y) : 0 \le x \le 2, 0 \le y \le 2\}$$

$$B = \{(x, y) : 1 \le x \le 3, 0 \le y \le 2\}$$

$$C = \{(x, y) : 1 \le x \le 2, 0 \le y \le 2\}$$

$$D = \{(x, y) : 0 \le x \le 2, 0 \le y \le 1\}$$

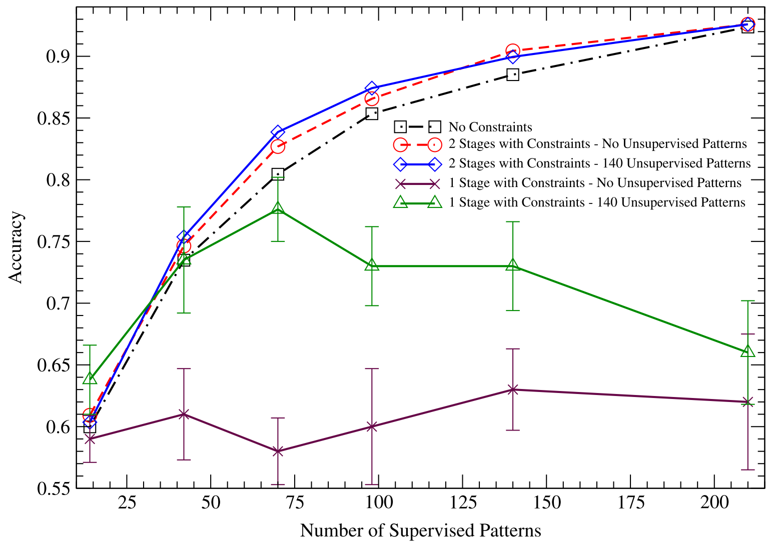


Fig. 5 Benchmark 2. The effect of the two-stage training process

$$E = \{(x, y) : 1 \le x \le 3, 0 \le y \le 1\}$$

$$F = \{(x, y) : 1 \le x \le 2, 0 \le y \le 1\}$$

$$G = \{(x, y) : 1 \le x \le 2, 0 \le y \le 1\}$$

The training set size has been increased from 14 to 203 examples during different runs of the experiment. Similarly, the unsupervised data was varied between 0 and 140 patterns. A Gaussian kernel with variance equal to 0.16 has been used in this experiment.

In a first set of trials, the kernel machine weights are optimized using a cost function including the constraint part ($\lambda^v > 0$) since the first iteration. In a second set of trials, the learning process takes place in two phases: a *Piagetian initialization* stage, where the cost function does not include the constraint part, and the subsequent *abstraction* phase where constraints are taken into account. In order to factor out the sampling noise, the accuracy numbers have been averaged over 20 different samples of the supervised, unsupervised and test patterns. Figure 5 compares the classification accuracy obtained in the two sets of experiments. The two-stage training process significantly improves the one-stage training. Indeed, the cost function resulting from the introduction of the constraints is plagued by many local minima and the good starting point provided by the Piagetian initialization phase is fundamental to discover a close-to-optimal solution.

6.3 Benchmark 3: 11 classes and 45 clauses

This synthetic experiment tackles a more difficult multi-task classification problem, where both the number of classes and of FOL clauses is higher. In particular, we assume that there are 11 different classes: A, B, C, D, E, F, G, H, I, L, M. Each class is associated to an indicator function (predicate), which is indicated with the corresponding lower-case letter. The patterns for each class are assumed to be uniformly distributed on a rectangle, as shown in Table 4.

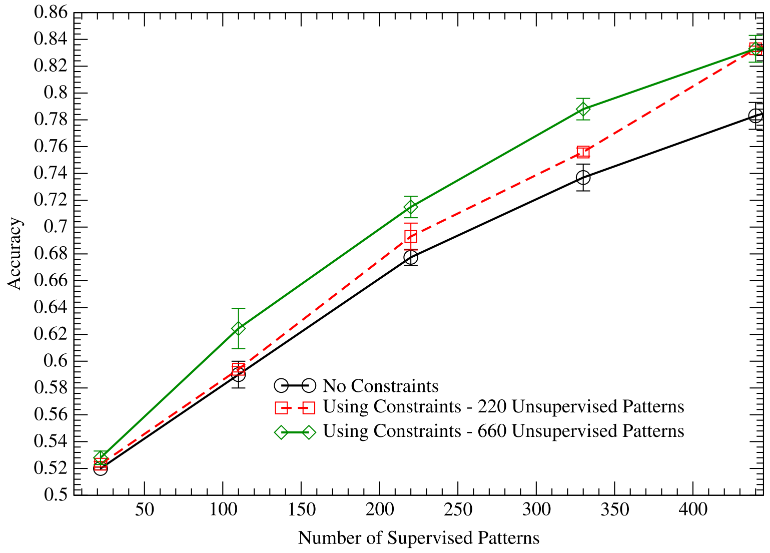


Fig. 6 Benchmark 3. Classification accuracy for different labeled and unlabeled datasets when using or not using the constraints in training

Let us assume to have available some a-priori knowledge, expressing some geometrical properties of the class regions. In particular,

– 27 clauses model the disjunction of the areas covered by pairs of classes. Two examples of clauses belonging to this category are: $\forall \mathbf{x}\ \neg a(\mathbf{x}) \vee \neg g(\mathbf{x})$ and $\forall \mathbf{x}\ \neg b(\mathbf{x}) \vee \neg g(\mathbf{x})$.
– Another set of 17 clauses models the complete inclusion of the area covered by one class within the areas covered by the union of a set of other classes, e.g.: $\forall \mathbf{x} g(\mathbf{x}) \Rightarrow c(\mathbf{x}) \vee d(\mathbf{x})$ (the union of C and D contains G), $\forall \mathbf{x}\ g(\mathbf{x}) \Rightarrow c(\mathbf{x}) \vee e(\mathbf{x})$, $\forall \mathbf{x}\ a(\mathbf{x}) \Rightarrow f(\mathbf{x}) \vee h(\mathbf{x}) \vee l(\mathbf{x})$, etc.
– the closed-world assumption clause was added to state that each pattern must belong to at least one class: $\forall \mathbf{x}\ a(\mathbf{x}) \vee b(\mathbf{x}) \vee c(\mathbf{x}) \vee d(\mathbf{x}) \vee e(\mathbf{x}) \vee f(\mathbf{x}) \vee g(\mathbf{x}) \vee h(\mathbf{x}) \vee i(\mathbf{x}) \vee l(\mathbf{x}) \vee m(\mathbf{x})$.

The two-stage learning algorithm described in Sect. 3 is exploited in this experiment. Figure 6 reports the obtained results, averaged over 10 different generations of the training and test sets. The introduction of the constraints is beneficial with an improvement in the classification accuracy between 2% and 5%. This experiment confirms the importance of the unsupervised data in the learning process: increasing the amount of available unsupervised patterns also significantly increases the classification accuracy.

6.4 Manifold regularization in a logic setting

In this benchmark, we assume to have patterns laying in a $\mathbb{R}^2$ feature space and belonging to two classes A, B, according to the well-known two moon-like shaped distributions. Figure 7 shows a random sample of patterns from the two distributions, where points represented as crosses correspond to patterns of class A and circles correspond to patterns of class B. The unknown predicate f must be learned to approximate the indicator function of class A. This predicate should output a TRUE value (1) for all patterns if class A and FALSE for all

Fig. 7 A sample of the input patterns for class A and B used in the manifold regularization experiment

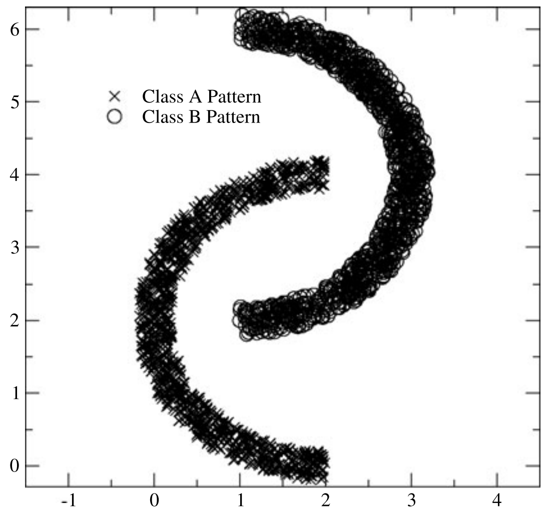


Table 4 Benchmark 3. Rectangles in $\mathbb{R}^2$ over which the patterns for each class are uniformly distributed

$A = \{(x, y) : 0 \leq x \leq 6, 4 \leq y \leq 6\}$
$B = \{(x, y) : 0 \leq x \leq 6, 3 \leq y \leq 5\}$
$C = \{(x, y) : 0 \leq x \leq 6, 2 \leq y \leq 4\}$
$D = \{(x, y) : 0 \leq x \leq 6, 1 \leq y \leq 3\}$
$E = \{(x, y) : 0 \leq x \leq 6, 0 \leq y \leq 2\}$
$F = \{(x, y) : 0 \leq x \leq 2, 3 \leq y \leq 6\}$
$G = \{(x, y) : 0 \leq x \leq 2, 0 \leq y \leq 3\}$
$H = \{(x, y) : 2 \leq x \leq 4, 3 \leq y \leq 6\}$
$I = \{(x, y) : 2 \leq x \leq 4, 0 \leq y \leq 3\}$
$L = \{(x, y) : 4 \leq x \leq 6, 3 \leq y \leq 6\}$
$M = \{(x, y) : 4 \leq x \leq 6, 0 \leq y \leq 3\}$

patterns of class B. We assume to be assigned a binary predicate which models a similarity relation $r(\mathbf{x}, \mathbf{y})$ between a pair of patterns $(\mathbf{x}, \mathbf{y})$. The semantic meaning of the relation r can differ in different applications. For example, it could be used to represent the hyperlink connections between documents in Web retrieval tasks, or the co-citations among authors, etc. In this experiment, we assume that r models the geometric proximity of the patterns in the feature space.

The following FOL clause is used to express the knowledge that any pair of patterns which are similar according to the relation should yield the same predicate output:

$$\forall \mathbf{x} \forall \mathbf{y}\ r(\mathbf{x}, \mathbf{y}) \Rightarrow (f(\mathbf{x}) \wedge f(\mathbf{y})) \vee (\neg f(\mathbf{x}) \wedge \neg f(\mathbf{y})). \tag{13}$$

This is a reformulation of the well known assumption made in manifold regularization (Belkin et al. 2006) in a continuous logic setting. This assumption expresses the fact that the input patterns are distributed along a manifold, over which the functions to be learned should be smooth, e.g. connected inputs on the manifold should tend to correspond to similar function outputs. Please note that this assumption is very general and it can be applied wherever the input patterns lay in a metric space.

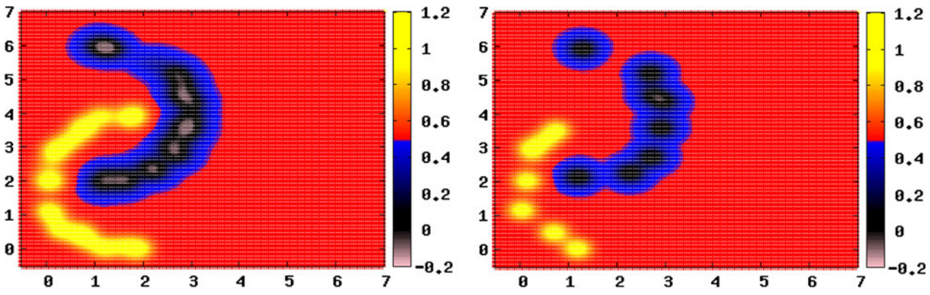


Fig. 8 Manifold Regularization: predicate output when using 16 labeled examples and using or not using the FOL clause on the left and right sides, respectively

Table 5 Moon benchmark: classification accuracy on the test set obtained with and without using the manifold regularization expressed in FOL form. Boldface values indicate gains that are statistically significant

	Num labeled patterns		
	4	8	12
With FOL knowledge	**59.6%**	**68.5%**	72.3%
Without FOL knowledge	40.4%	53.5%	71.2%

The FOL clause in (13) is equivalent to, $\forall \mathbf{x} \forall \mathbf{y}\ \neg(r(\mathbf{x},\mathbf{y}) \wedge \neg(f(\mathbf{x}) \wedge f(\mathbf{y})) \wedge \neg(\neg f(\mathbf{x}) \wedge \neg f(\mathbf{y})))$. As described in Sect. 5.3, this FOL clause has a continuous equivalent which can be efficiently computed. In particular, using the product t-norm and the mapping to a continuous cost function as explained in Sect. 3, we obtain the following constraint term for the cost function,

$$V(f) = \sum_{(\mathbf{x},\mathbf{y}):\mathbf{x},\mathbf{y}\in\mathcal{S},r(\mathbf{x},\mathbf{y})\neq 0} r(\mathbf{x},\mathbf{y})(1 - f(\mathbf{x})f(\mathbf{y}))(1-(1-f(\mathbf{x}))(1-f(\mathbf{y}))).$$

In our experimental setting, each pattern $\mathbf{x}$ is connected to the 5 closest patterns with a continuous strength of the relation computed as $r(\mathbf{x},\mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|/\sigma_d}$, where $\sigma_d = 2/3$. The constraint part is then plugged into the cost function and optimized by gradient descent using the two-stage learning process. Figure 8 plots the output map of the learned indicator function f. The prior knowledge expressed by the FOL clause smoothes the predicate output value over the supervised and unsupervised data. This allows the estimated indicator function to cover regions where scarce labeled data is available. Indeed, the activation map perfectly reconstructs the boundaries of the regions where the input patterns are distributed for the two classes.

Table 5 reports the classification accuracy for different numbers of the labeled patterns, averaged over 10 different random generations of the training and test data. 100 unlabeled patterns have been also used when learning using the FOL prior knowledge. The accuracy gain is very significant when little labeled data is available. A one-tailed t-test confirms that, when learning from 4 and 8 labeled patterns, the accuracy improvements are statistically significant with over 95% confidence.

Table 6 A sample of the semantic rules used in training the kernel machines in the bibtex tagging experiment

$\forall x$ phase(x) $\wedge$ transition(x) $\Rightarrow$ physics(x)
$\forall x$ chemistry(x) $\Rightarrow$ science(x)
$\forall x$ immunoelectrode(x) $\Rightarrow$ physics(x) $\vee$ biology(x)
$\forall x$ semantic(x) $\wedge$ web20(x) $\Rightarrow$ knowledgemanagement(x)
$\forall x$ rdf(x) $\Rightarrow$ semanticweb(x)
$\forall x$ software(x) $\wedge$ visualization(x) $\Rightarrow$ engineering(x)
$\forall x$ folksonomy(x) $\Rightarrow$ social(x)
$\forall x$ mining(x) $\wedge$ web(x) $\Rightarrow$ informationretrieval(x)
$\forall x$ mining(x) $\wedge$ information(x) $\Rightarrow$ datamining(x)
$\forall x$ computer(x) $\wedge$ science(x) $\Rightarrow$ engineering(x)

6.5 Automatic tagging of bibtex entries

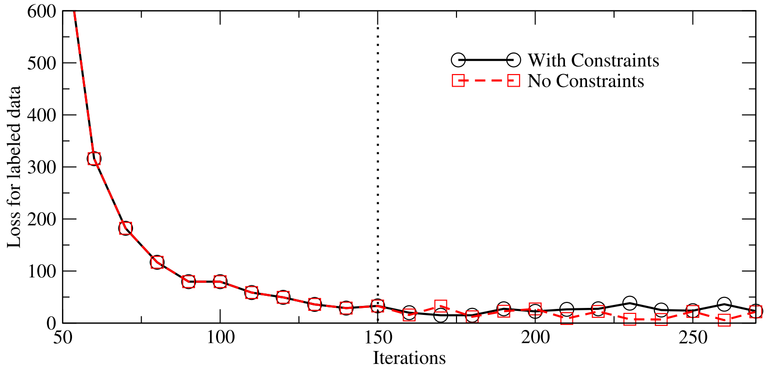
Text tagging associates a document with a set of tags, which usually summarize the semantic content of the text. Text tagging is often manually performed in the context of social networks, or directories organizing Web resources. Having the documents tagged with high consistency and precision would allow us to develop more sophisticated information retrieval mechanisms that the ones typically provided in search-by-keyword applications. However, a manual collective tagging process has many limitations. First, it is not suited for very large collection of documents (like the Web) or very highly dynamic collections, where the response time is crucial. Furthermore, the collective tagging process does not provide any guarantee of consistency of the tags across documents, creating many issues for the subsequent consumption of the tags. Automatic text tagging is regarded as a way to address, at least partially, these limitations. Text tagging can be typically seen as a classical text categorization task (Sebastiani 2002), where each tag corresponds with a different category. Differently to many categorization tasks explored in the literature, the number of tags is typically in the order of hundreds to thousands, and the tags are not mutually exclusive, thus yielding a multi label classification task.

In this section, we consider a dataset collecting 7395 bibtex entries that have been tagged by users of a social network[2] using 159 tags. This dataset has been used in previous literature like (Katakis et al. 2008). Each bibtex entry contains a small set of textual elements representing the author, the title, and the conference or journal name. The text is represented as a bag-of-words, thus yielding a feature space with dimensionality equal to 1836. The training set was obtained by sampling 10% of the entries, leaving the remaining for the test set. Previous studies in the literature employed the F1 score to establish the prediction accuracy of the employed classifier on this task.

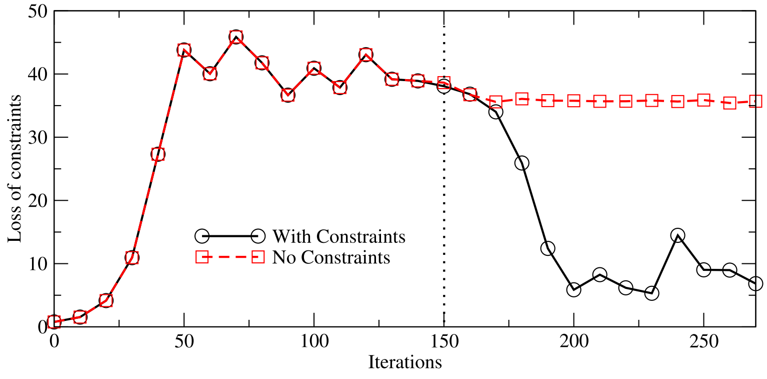
The performed experiments tested the prediction capabilities of the classifiers, when considering the 25 and 100 most popular tags in the dataset as output categories. A knowledge base containing a set of 115 rules, expressed by FOL, has been collected by the authors as to express semantic relationships between the categories. Table 6 shows some of the rules inserted in the knowledge base. The rules correlate the tags and, after their conversion into the continuous form, they have been used to train the kernel machines according to the procedure described in the previous sections.

Figures 9 and 10 display the loss on the labeled data and on the constraints for the test set (generalization) at the different iterations of the training for the 25 and 100 tag classifiers, respectively. The training of the classifiers with no constraints was performed until the

[2]The dataset can be downloaded from http://mulan.sourceforge.net/datasets.html.



(9.1) 25 tags - Loss for labeled data



(9.2) 25 tags - Constraint error

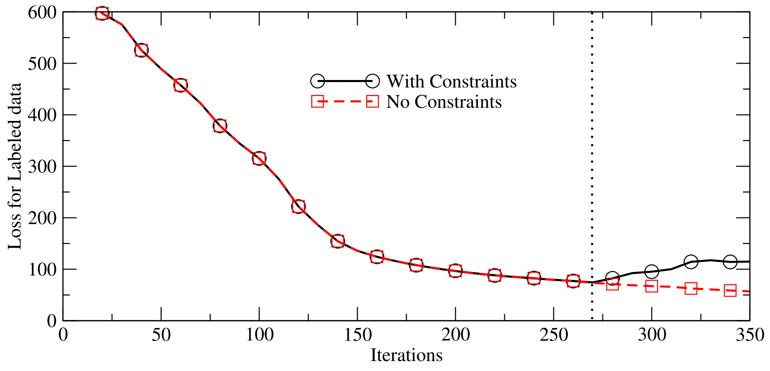
Fig. 9 Loss term on labeled data and on constraints deriving from the rules over the 25 tag test set

gradient reached a threshold chosen so as to get close to the global minimum. Then, the constraints were introduced in the overall cost function. The figures show how the introduction of the constraints does not change significantly the loss on the labeled data, whereas the constraint loss is strongly reduced, that leading to a solution that fits much better the prior knowledge on the task.

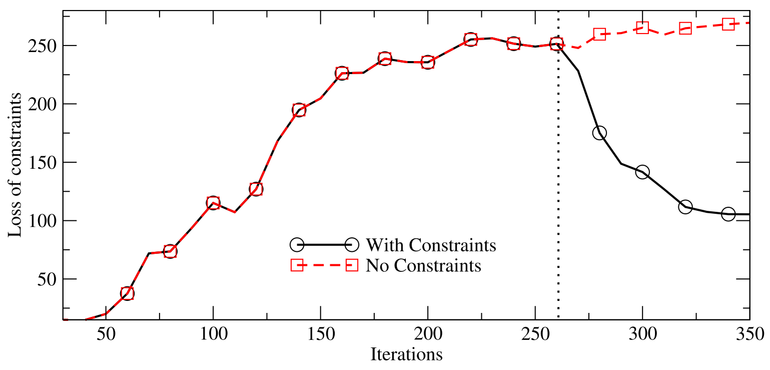
It turned out that enforcing the constraints during learning also led to improve the accuracy of the prediction of the tags with respect to a kernel machine learning only from labeled data. In particular, the macro and micro F1 scores of the 100 tag predictor, computed over the test set, were increased from 0.042 to 0.055 and from 0.140 to 0.155, respectively.

7 Conclusions and future work

In this paper we propose a solution for bridging logic and kernel machines by extending the general framework of regularization to learning from constraints. Like for kernel machines, we introduce parsimonious agents that find simple explanations of data coming from the environment. However, while kernel machines restrict the communication protocol to deal with pairs of supervised examples, in this paper we deal with a general multi-task environment in which a rich collection of constraints on the image of the functions may reduce



(10.1) 100 tags - Loss for labeled data



(10.2) 100 tags - Constraint error

Fig. 10 Loss term on labeled data and on constraints deriving from the rules over the 100 tag test set

significantly the hypothesis space. It is proven that once the constraint satisfaction is relaxed to hold on a finite sample of examples, a representation theorem holds which dictates the optimal solution of the problem as a kernel expansion.

This makes it possible to use a semi-supervised scheme in which the unsupervised examples play a crucial role for the incorporation of the constraints. While the optimization of the error functions deriving from the proposed formulation is typically hopeless because of the presence of local minima, we claim that a proper introduction of stage-based learning, somehow inspired to developmental psychology, offers a viable solution to tackle the problem. This reinforces the related belief on the importance of the gradual presentation of examples (Bengio 2009), and might contribute to a systematic treatment of emerging fields like developmental robotics (Weng 2004). While the methodology proposed in the paper holds in general for learning from constraints, the focus is on the case of constraints given in terms of first-order logic, that are properly compiled into real-valued constraints required by the proposed kernel-based approach. The theory is validated by a number of artificial experiments that clearly show the improvements with respect to plain kernel machines, even in problems of low dimension in which they already achieve top level performance. A remarkable improvement has also been found from experiments on a problem of multi-label text classification for automated tag suggestion, where in addition to the better classification results with respect to plain kernel machines, there is also clear evidence that the attached tags are significantly more consistent with the knowledge base.

When comparing with most of the related studies, we can early realize that the distinguishing feature of the proposed approach consists of centering the studies on logic and learning around the unifying notion of constraint. While this direction had already been followed (see e.g. Fung et al. 2002, 2003; Le et al. 2006; Maclin et al. 2007), with remarkable results, this paper goes beyond the idea of imposing constraints into the perceptual space by considering multi-task environments. In so doing the background knowledge involves abstract categories more than the identification of input sets. Most interestingly, the way the constraints are processed naturally extends the notion of functional risk for supervised learning by introducing the sampling on the unsupervised set that is somehow dual with respect to the sampling which gives rise to the empirical risk. It is the well-established connection with T-norms that makes the model very well suited for connections with logic. This paper contains the basic results that might open the doors to a new distinctive approach to kernel machines, in which the empirical risk is replaced by a penalty coming from a set of constraints. Interestingly, while in classic statistical learning theory, there is only access to a limited set of supervised examples, the construction of penalties only requires unsupervised data. The focus on constraints also suggests the adoption of prior knowledge that does not necessarily come from formal logic. In addition, even the representer theorem given in this paper, which comes from the assumption of sampling the constraints, might be extended so as to incorporate the truly nature of specific constraints. A parsimonious agent could be devised which exhibits a smooth behavior and is consistent with the constraints. When involving abstract categories and quantifiers, it become very important also to model strong consistency with the constraints. Hence, one might want to go beyond the softness which has been inherently associated with the penalty term in this paper and impose the hard fulfillment of some clauses. Interestingly, in principle, the constraints can either be softly or hardly imposed, and in the first case one might also take advantage from some knowledge on their membership function. Hence, in general we can deal with fuzzy constraints, while the results found in (Poggio and Girosi 1989) clearly suggests also the probabilistic interpretation of the learned tasks. We are currently investigating the construction of a unified theory of learning from constraints using the well developed mathematical apparatus of constrained variational calculus (Giaquinta and Hildebrand 1996a,1996b), that makes it possible to study parsimonious agents, whose behavior needs to be somehow consistent with the environmental constraints. We are also systematically studying how the primary role of constraints can extend the classic regularization theory to a sort of *semantic-based regularization machines*, where the kernels turns out to be dependent on smoothness requirements as well as on the given constraints.

Acknowledgements We thank Volha Bryl, Ernesto De Vito, Luciano Serafini and Alessandro Verri for insightful discussions and comments on a earlier draft of this paper.

References

Allgower, E., & Georg, K. (2003). Introduction to numerical continuation methods. In *Society for industrial mathematics* (p. 2003).

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, *7*, 2434.

Bengio, Y. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning* (pp. 41–48).

Caponnetto, A., Micchelli, C., Pontil, M., & Ying, Y. (2008). Universal kernels for multi-task learning. *Journal of Machine Learning Research*.

Chapelle, O. (2007). Training a support vector machine in the primal. *Neural Computation*, *19*(5), 1155–1178.

Cumby, C., & Roth, D. (2002). Learning with feature description logics. In *Proceedings of the 12th international conference on inductive logic programming*.

Cumby, C., & Roth, D. (2003). On kernel methods for relational learning. In *Proceedings of the twentieth international conference on machine learning (ICML-2003)*, Washington DC, 2003.

Diligenti, M., Gori, M., Maggini, M., & Rigutini, L. (2010a). Multitask kernel-based learning with first-order logic constraints. In *The 20th international conference on inductive logic programming*.

Diligenti, M., Gori, M., Maggini, M., & Rigutini, L. (2010b). Multitask kernel-based learning with logic constraints. In *The 19th European conference on artificial intelligence*.

Fanizzi, N., D'Amato, C., & Esposito, F. (2008). Statistical learning for inductive query answering on owl ontologies. In *THE SEMANTIC WEB—ISWC* (pp. 195–212).

Fung, G., Mangasarian, O., & Shavlik, J. (2002). Knowledgebased support vector machine classifiers. In *Proceedings of sixteenth conference on neural information processing systems (NIPS)*, Vancouver, Canada.

Fung, G., Mangasarian, O., & Shavlik, J. (2003). Knowledgebased nonlinear kernel classifiers. In *International conference on learning theory—COLT*, Washington D.C.

Giaquinta, M., & Hildebrand, S. (1996a). *Calculus of variations I* (Vol. 1). Berlin: Springer.

Giaquinta, M., & Hildebrand, S. (1996b). *Calculus of variations II* (Vol. 2). Berlin: Springer.

Gori, M. (2009). Semantic-based regularization and Piaget's cognitive stages. *Neural Networks*, *22*(7), 1035–1036.

Gori, M., & Melacci, S. (2010). Learning with convex constraints. In *20th International conference on artificial neural networks*.

Gorse, D., Shepherd, A. J., & Taylor, J. (1997). The new era in supervised learning. *Neural Networks*, *10*(2), 343–352.

Gorse, D., Sherpard, A. J., & Taylor, J. (2004). A classical algorithm for avoiding local minima. In *Proceedings of WCCI-2004*.

Guerin, F. (2008). Constructivism in ai: Prospects, progress and challenges. In *Proceedings of the AISB convention 2008*, Aberdeen, Scotland, 1–4 April, 2008, (pp. 20–27).

Guerin, F., & McKenzie, D. (2008). A Piagetian model of early sensorimotor development. In *Proceedings of the eighth international conference on epigenetic robotics*, University of Sussex, 30–31 July 2008.

Haussler, D. (1999). Convolution kernels on discrete structures, *Tech. rep.*, Department of Computer Science, University of California at Santa Cruz.

Hitzler, P., Holldobler, S., & Sedab, A. K. (2004). Logic programs and connectionist networks. *Journal of Applied Logic*, *2*(3), 245–272.

Inhelder, B., & Piaget, J. (1958). *The growth of logical thinking from childhood to adolescence*. New York: Basic Books.

Katakis, I., Tsoumakas, G., & Vlahavas, I. (2008). Multilabel text classification for automated tag suggestion. *ECML PKDD Discovery Challenge, 75*.

Klement, E., Mesiar, R., & Pap, E. (2000). *Triangular norms*. Norwell: Kluwer Academic.

Klir, G., & Yuan, B. (1995). *Fuzzy sets and fuzzy logic: theory and applications*. New York: Prentice Hall.

Landwehr, N., Passerini, A., Raedt, L. D., & Frasconi, P. (2006). Kfoil: learning simple relational kernels. In *Proceeding of the AAAI-2006*.

Landwehr, N., Passerini, A., Raedt, L., & Frasconi, P. (2010). Fast learning of relational kernels. *Machine Learning*.

Laurer, F., & Bloch, G. (2009). Incorporating prior knowledge in support vector machines for classification: a review. *Neurocomputing*, *71*(7–9), 1578–1594.

Le, Q., Smola, A., & Gartner, T. (2006). Simpler knowledge-based support vector machines. In *Proceedings of the 23rd international conference on machine learning*.

Maclin, R., Wild, E., Shavlik, J., Torrey, L., & Walker, T. (2007). Refining rules incorporated into knowledge-based support vector learners via successive linear programming. In A. Press (Ed.), *AAAI conference on artificial intelligence*, Vancouver, British Columbia, Canada, pp. 584–589.

Melacci, S., Maggini, M., & Gori, M. (2009). Semi-supervised learning with constraints for multi-view object recognition. In *Proceedings of the 19th international conference on artificial neural networks* (pp. 653–662). Berlin: Springer.

Muggleton, S.L.H., Amini, A., & Sternberg, M., (2005). In A. Hoffmann, H. Motoda, & T. Scheffer (Eds.), *Support vector inductive logic programming* (pp. 163–175). San Mateo: Kaufmann.

Piaget, J. (1961). *La psychologie de l'intelligence*. Paris: Armand Colin.

Poggio, T., & Girosi, F. (1989). A theory of networks for approximation and learning. *Tech. rep.*, MIT, 1989.

Raedt, L. D., Frasconi, P., Kersting, K., & Muggleton, S. (Eds.). (2008). Probabilistic inductive logic programming (Vol. 4911). *Lecture notes in artificial intelligence*. Berlin: Springer.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*(1–2), 107–136.

Scholkopf, B., & Smola, A. J. (2001). *Learning with Kernels*. Cambridge: MIT Press.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, *34*(1), 1–47.

Sloman, A. (2009). Ontologies for baby animals and robots, *Tech. rep.*, Talks 68.

Weng, J. (2004). Developmental robotics: Theory and experiments. *International Journal of Humanoid Robotics*, *1*, 199–236.