Artificial Intelligence 244 (2017) 143–165

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Semantic-based regularization for learning and inference

Michelangelo Diligenti *, Marco Gori, Claudio Saccà

Department of Information Engineering and Mathematics, University of Siena, Via Roma 56, Siena, Italy

ARTICLE INFO

Article history:
Received in revised form 13 August 2015
Accepted 26 August 2015
Available online 1 September 2015

Keywords:
Learning with constraints
Kernel machines
FOL

ABSTRACT

This paper proposes a unified approach to learning from constraints, which integrates the ability of classical machine learning techniques to learn from continuous feature-based representations with the ability of reasoning using higher-level semantic knowledge typical of Statistical Relational Learning. Learning tasks are modeled in the general framework of multi-objective optimization, where a set of constraints must be satisfied in addition to the traditional smoothness regularization term. The constraints translate First Order Logic formulas, which can express learning-from-example supervisions and general prior knowledge about the environment by using fuzzy logic. By enforcing the constraints also on the test set, this paper presents a natural extension of the framework to perform collective classification. Interestingly, the theory holds for both the case of data represented by feature vectors and the case of data simply expressed by pattern identifiers, thus extending classic kernel machines and graph regularization, respectively. This paper also proposes a probabilistic interpretation of the proposed learning scheme, and highlights intriguing connections with probabilistic approaches like Markov Logic Networks. Experimental results on classic benchmarks provide clear evidence of the remarkable improvements that are obtained with respect to related approaches.

1. Introduction

This paper presents Semantic Based Regularization (SBR), a unified framework for inference and learning that is centered around the notion of a constraint and of the parsimony principle. Semantic Based Regularization bridges the ability of machine learning techniques to learn from continuous feature-based representations with the ability of modeling arbitrary pattern relationships, typically used in Statistical Relational Learning (SRL) to model and learn from high-level semantic knowledge. In order to provide a unified context for manipulating perceptual data and prior knowledge, we propose to use the unifying concept of a *constraint*, which is sufficiently general to represent different kinds of sensorial data along with their relations, as well as to express abstract knowledge on the tasks. We unify continuous and discrete computational mechanisms, so as to accommodate in the same framework very different stimuli. In this paper, we focus on the kernel machine mathematical and algorithmic apparatus to learn from feature-based pattern representations and on constraints resulting from a fuzzy translation of First Order Logic (FOL) formulas, expressing the prior knowledge about the learning task at hand.

More specifically, SBR builds a multi-layer architecture having kernel machines at the input layer. The output of the kernel machines is fed to the higher layers implementing a fuzzy generalization of the FOL knowledge. Thanks to the

* Corresponding author.
E-mail addresses: diligmic@diism.unisi.it (M. Diligenti), marco@diism.unisi.it (M. Gori), sacc@unisi.it (C. Saccà).

http://dx.doi.org/10.1016/j.artint.2015.08.011
0004-3702/

basic properties of fuzzy FOL and kernel machines, the resulting model is continuous with respect to the feature values. Therefore, the high-level semantic inference provided by the logic can be back-propagated down to the kernel machines using any gradient-based schema. This process can be iterated during training until convergence. This is an extremely powerful technique to get advantage of the available unsupervised data, as the inference process performed on this data via the logic knowledge can be used to correct the output of the kernel machines.

We substantially extend earlier studies in Diligenti et al. [10] by showing that SBR enables new fundamental tasks of learning and inference that rely on the joint informative evidence coming from real-valued features and simple pattern identifiers, along with the corresponding relations. In particular, the paper gives the following new main results, which are of fundamental importance to gain an overall view of theory and, especially, to enable a large set of applications in statistical relational learning domains:

VARIABLE DIMENSION DOMAINS AND NULL INPUTS We extend the SBR framework [10] to truly hybrid domains, where real-valued feature pattern representations are integrated with pure symbolic entities (e.g. pattern identifiers). Indeed, in complex relational classification tasks, it is often the case that the entities are naturally representable by pattern spaces of different dimensions, including the remarkable case of "void patterns" in which only relational information is available.

COLLECTIVE CLASSIFICATION In this paper we propose a novel collective classification method to enforce the constraints on the test set, thus exploiting the full expressiveness of FOL, like in other statistical relational learning (SRL) approaches. Once again, the distinctive feature of the solution proposed in this paper arises when considering that the collective computational scheme also naturally exploits real-valued feature pattern representations.

PROBABILISTIC LINKS We extend studies on the probabilistic interpretation of regularization networks [38] to our case of learning from constraints. From one side, this highlights connections with Markov Logic Networks (MLNs) [40], while from the other side, this interpretation clearly shows the natural integration of real-valued features and object identifiers in SBR.

Furthermore, the paper presents how plain SVM, Transductive, and Laplacian SVMs can be derived as special cases of the proposed SBR framework. The paper also introduces new heuristics, connected to the ones employed in constraint satisfaction programming, to improve the quality of the found solutions. Finally, we present experimental results to show the effectiveness and generality of the approach.

The paper is organized as follows: in the next section previous work in the field is reviewed. Section 3 introduces First Order Logic and its fuzzy extensions, while Section 4 discusses learning from constraints with kernel machines. Section 5 presents how SBR generalizes several models commonly used in relational and transductive learning. Details on how training is performed in the SBR framework is presented in Section 6. In Section 7 a collective classification approach for SBR is presented and Section 8 presents connections between SBR and probabilistic models like Markov Logic Networks. The experimental evaluation of SBR is presented in Section 9 and, finally, Section 10 draws some conclusions.

2. Previous work

Statistical Relational Learning (SRL) combines robust parameter estimation in the presence of noise with learning complex relational structures. Probabilistic Relational Models (PRMs) [13] are an early SRL approach that learns a statistical model from a relational database. PRMs build a probability distribution over the attributes of the objects as an instance of a schema. A Bayesian network with one node for each attribute is built and parameters are estimated from the data. Relational Dependency Networks [34] learn a (local) conditional probability distribution for each node given its Markov blanket by using a conditional learner (like logistic regression or decision trees).

Markov Logic Networks (MLNs) [40] have received a lot of attention in the SRL community and have been extensively applied in many fields like bioinformatics [28] and computer vision [46]. Markov Logic Networks generalize and combine first-order logic and probabilistic graphical models. Thanks to their flexibility, MLNs have been used to tackle all the SRL main tasks: collective classification, link prediction, link-based clustering, social network modeling, and object identification. Many papers have also studied how to learn the structure of Markov Logic Networks from data without requiring an expert to express the structure in terms of prior knowledge [23,22]. Hybrid Markov Logic Networks (HMLNs) [49] extend MLNs to deal with continuous variables.

Probabilistic Soft Logic (PSL) [5] is another SRL approach, which relaxes MLNs to continuous fuzzy values in the $[0, 1]$ interval and restricts the considered FOL formulas to the ones with conjunctive body and a single literal head. PSL weight training can be solved via a convex optimization problem, but it can face only a small subset of the tasks that are potentially solved by a MLN.

One disadvantage of both MLNs and PSL in real-world applications is how they deal with entities that are associated to complex feature-based representations. Let's take as an example the common scenario of a multi-class classification task where the patterns are represented by large vectors of numeric features. In order to perform learning and inference in this domain using classical SRL techniques, different approaches are possible:

- the value of a feature can be correlated with one output class using one specific rule. For example, let x be a generic pattern in the domain and f be a binary feature, it is possible to express the rule $HasTrueValue(f, x) \wedge BelongsTo(x, c)$ for each category c. The training process will estimate a weight modeling the strength of this correlation. MLNs can capture a logistic regression model using this approach [11,12]. The advantage of this solution is that it employs a coherent framework for dealing with the pattern representations and the higher-level semantic knowledge. However, it requires to deal with a large number of weights and groundings. Furthermore, only very simple correlations between features and classes can be captured as more complex models would be too large to be tractable.
- Another approach is to pre-train an arbitrary classifier working on the low-level feature representations and let a SRL layer use the classifier as a prior for the assignments. The SRL layer will assign a proper weight to this prior during training and then the predictions of the base classifier can be refined during the prediction step. This approach uses standard machine learning techniques to learn from the low level data. Complex feature dependencies can be efficiently captured by, for example changing the employed kernel when using a kernel machine as the base classifier. The main disadvantage of this approach is that the two layers are sequentially trained in a greedy fashion. This means that the high level semantic inference is stacked up on top of a pre-trained frozen layer, which processes the pattern representations. This does not allow to use the output of the inference process performed at the higher level to improve the predictions of the base classifier. This is a huge limitation whenever a large number of unsupervised patterns is available and the decision made by the semantic layer could be used as additional supervisions at the lower level.

SBR encodes a multi-layer architecture having kernel machines at the input layer, which provides the input to the higher-level layers implementing a fuzzy generalization of the FOL knowledge. Since the fuzzy FOL generalization of the knowledge is continuous with respect to the values coming from the input layer, information can flow in both directions in this architecture. This allows to back-propagate the information from the high-level inference performed by the logic knowledge down to the kernel machines using a simple gradient-based schema. This allows SBR to preserve the compactness and efficiency of kernel machines to deal with the feature space representations at the input level, while exploiting the full power of FOL to express the higher-level semantics.

As it will be discussed later in the paper, keeping two separate but communicating layers one dedicated to processing the feature space and one to performing the higher conceptual reasoning allows to devise more efficient training techniques breaking the complexity of learning. SBR optimization tasks are very similar to the nested optimization problems studied by bilevel programming [2]. Unfortunately, the results obtained in the context of bilevel programming are not easy to reuse for SBR training, as the outer SBR optimization problem is generally neither quadratic nor linear.

Other SRL approaches have focused on high level cognitive decision processes with no native integration with learning in the presence of continuous high dimensional data representations. Integration between logic and learning has been traditionally hard to achieve because of the barriers erected by the different mathematical models classically used to handle logical reasoning and learning from examples with real numbers. Connections between logic and kernel machines have been studied in the context of finding relationships between symbolic and sub-symbolic models in AI [21], with particular emphasis on hybrid models. When restricting to kernel machines, a rich analysis of the literature can be found in the survey [26], while a broader coverage of the field with emphasis on the connections with inductive logic programming can be found in DeRaedt at al. [39].

Convolution kernels in discrete structures [20] have been one of the main sources of inspiration for exploring connections of kernel machines with logic. Feature Description Logic (FDL) is introduced to support learning in relational domains [8,9]. 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. Muggleton et al. [33] presents support vector machines based on a kernel that is an inner product in the feature space spanned by a given set of first-order hypothesized formulas. The inductive logic programming system FOIL is combined with kernel methods in Landwehr et al. [24]. The resulting model (kFOIL) implements a dynamic propositionalization approach allowing to perform both classification and regression tasks. Landwehr et al. [25] presents a general theoretical framework for statistical logical learning based on dynamic propositionalization, 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. In the context of multi-task learning, functions can be coupled via dependencies induced by the structure of special multi-task kernels [6].

Quite a different approach is based on imposing constraints in the perceptual space [14,15,27], where the prior knowledge is incorporated in the form of constraints into a support vector machine classifier.

Most of the reviewed papers have already proposed different frameworks for incorporating prior knowledge expressed by logic formalisms into kernel machines. However, the integration seems to be shallow, being based on asking the kernel to play the additional role of incorporating logic structures, instead of primarily measuring the smoothness of the solution according to the Occam's razor. While this is a very interesting idea, the remarkable residual degree of freedom on the way the logic structures can be incorporated suggests that we are only partially addressing the inherent limitation of kernel methods.

In the presented framework, the adoption of T-norms allows to simply express most classic logic formalisms by constraints on real-valued functions. Therefore, the general and unified notion of constraint employed in this paper seems to be most natural and straightforward extension of the classic statistical framework of learning from examples, since they are just another special instance of a constraint.

3. First order fuzzy logic

The term *fuzzy logic* was firstly used by L.A. Zadeh in 1965 [51]. Classical logic works with variables assuming a binary truth value. Fuzzy Logic, instead, is a form of many-values logic or probabilistic logic. It deals with reasoning that is approximate rather than exact [18], where variables have a truth degree that ranges in $[0, 1]$: zero and one meaning that the variable is false and true with certainty, respectively. A fuzzy generalization of FOL has been first proposed by Novak [36].

T-norm and residuum A *t-norm* is a function $t : [0, 1] \times [0, 1] \rightarrow [0, 1]$, which is continuous, commutative, associative, monotone, and featuring a neutral element 1 (i.e. $t(a, 1) = a$). A strict t-norm is also strictly monotone. A *t-norm fuzzy logic* is defined by its t-norm $t(a_1, a_2)$ that models the logical AND. Given a variable $\bar{a}$ with continuous generalization a in $[0, 1]$, its negation $\neg\bar{a}$ corresponds to $1 - a$. Once the t-norm functions corresponding to the logical AND and NOT are defined, they can be composed to convert any arbitrary logic proposition into a continuous function. A t-norm expression behaves as classical logic when the variables assume the crisp values 0 (false) or 1 (true). Many different t-norm fuzzy logics have been proposed in the literature. For example, given two Boolean values $\bar{a}_1, \bar{a}_2$ and their continuous generalizations a_1, a_2 in the $[0, 1]$ range, the *product t-norm* is defined as:

$$(\bar{a}_1 \wedge \bar{a}_2) \quad \longrightarrow \quad t(a_1, a_2) = a_1 \cdot a_2$$

The $\vee$ operator is then consequently defined by using the De Morgan rule:

$$(\bar{a}_1 \vee \bar{a}_2) \equiv \neg(\neg\bar{a}_1 \wedge \neg\bar{a}_2) \quad \longrightarrow \quad t(a_1, a_2) = 1 - (1 - a_1) \cdot (1 - a_2)$$

Another commonly used t-norm is the *minimum t-norm* defined as:

$$(\bar{a}_1 \wedge \bar{a}_2) \quad \longrightarrow \quad t(a_1, a_2) = \min(a_1, a_2)$$

from which it follows:

$$(\bar{a}_1 \vee \bar{a}_2) \quad \longrightarrow \quad t(a_1, a_2) = \max(a_1, a_2)$$

The equivalence $\bar{a}_1 \Rightarrow \bar{a}_2 \equiv \neg\bar{a}_1 \vee \bar{a}_2$ is used in classic logic to represent implications (*modus ponens*). However, this equivalence is not appropriate to perform deductions with fuzzy variable values. Any t-norm has a corresponding binary operator $\Rightarrow$ called *residuum*, which is used in fuzzy logic to generalize implications when dealing with continuous variables. A t-norm residuum provides a natural way to express human fuzzy reasoning, while being equivalent to modus ponens when fuzzy variable values approach the extremes of the $[0, 1]$ range.

In particular, the residuum converting an implication for the minimum t-norm is defined as:

$$(\bar{a}_1 \Rightarrow \bar{a}_2) \quad \longrightarrow \quad t(a_1, a_2) = \begin{cases} 1 & a_1 \leq a_2 \\ a_2 & a_1 > a_2 \end{cases}$$

while for the product t-norm the residuum is defined as:

$$(\bar{a}_1 \Rightarrow \bar{a}_2) \quad \longrightarrow \quad t(a_1, a_2) = \begin{cases} 1 & a_1 \leq a_2 \\ \frac{a_2}{a_1} & a_1 > a_2 \end{cases}$$

The residuum allows to relax the condition of satisfaction for the implication, which is satisfied as soon as the t-norm expression of the head has a higher truth degree than the t-norm expression of the formula body.

Quantifiers With no loss of generality, we restrict our attention to FOL formulas in the Prenex Normal Form (PNF) form, where all the quantifiers $(\forall, \exists)$ and their associated quantified variables are placed at the beginning of the expression. For example:

$$\overbrace{\forall x_1 \forall x_2}^{\textit{Quantifiers}} \; \overbrace{A(x_1) \wedge B(x_2) \Rightarrow C(x_1)}^{\textit{Quantifier-free expression}} \tag{1}$$

Please note that, since we are dealing with variables ranging in $[0, 1]$, the quantifier-free part of the expression is equivalent to an assertion in fuzzy propositional logic once all the quantified variables are grounded. Hence, a t-norm fuzzy logic can be used to convert it into a continuous function.

Let's consider a FOL formula with variables $x_1, x_2, \ldots$ assuming values in the finite sets $\mathcal{X}_1, \mathcal{X}_2, \ldots$. $\mathcal{P} = \{p_1, p_2, \ldots\}$ is the vector of predicates, where the j-th n-ary predicate is grounded from $\mathcal{X}_j^\circ = \mathcal{X}_{j1} \times \mathcal{X}_{j2} \times \ldots$. In case of an unary predicate $\mathcal{X}_j^\circ = \mathcal{X}_{j1}$. Let $p_j(\mathcal{X}_j^\circ)$ indicate the set of possible groundings for the j-th predicate, and $\mathcal{P}(\mathcal{X})$ indicate all possible grounded predicates, such that $\mathcal{P}(\mathcal{X}) = p_1(\mathcal{X}_1^\circ) \cup p_2(\mathcal{X}_2^\circ) \cup \ldots$.

Assuming that the atoms $\mathcal{P}(\mathcal{X})$ are generalized to assume real values in $[0, 1]$, the degree of truth of a formula containing an expression E with a universally quantified variable x_i is defined as the *minimum* of $t_E(\cdot)$ obtained as the t-norm generalization of E when grounding x_i over $\mathcal{X}_i$:

$$\forall x_i \;\; E\big(\mathcal{P}(\mathcal{X})\big) \quad \longrightarrow \quad \Phi_\forall\big(\mathcal{P}(\mathcal{X})\big) = \min_{x_i \in \mathcal{X}_i} t_E\big(\mathcal{P}(\mathcal{X})\big)$$

For the existential quantifier, the truth degree is instead defined as the *maximum* of the t-norm expression over the domain of the quantified variable:

$$\exists x_i \; E\big(\mathcal{P}(\mathcal{X})\big) \quad \longrightarrow \quad \Phi_\exists\big(\mathcal{P}(\mathcal{X})\big) = \max_{x_i \in \mathcal{X}_i} t_E\big(\mathcal{P}(\mathcal{X})\big)$$

The two above generalizations correspond to the conjunction and disjunction of the minimum t-norm expressions of the propositional formula computed over all the groundings, respectively.

When multiple universally or existentially quantified variables are present, the conversion is performed recursively from the outer to the inner variables. Please note that any fuzzy FOL formula returns a value in the $[0, 1]$ range. The fuzzy formula expression is continuous and differentiable (except that over a set of points with null measure) with respect to the fuzzy value of a predicate.

A small modification to classic Fuzzy FOL has been used in this paper, in particular the universal quantifier is generalized as:

$$\forall x_i \;\; E\big(\mathcal{P}(\mathcal{X})\big) \quad \longrightarrow \quad \Phi_\forall\big(\mathcal{P}(\mathcal{X})\big) = \frac{1}{|\mathcal{X}_i|} \sum_{x_i \in \mathcal{X}_i} t_E\big(\mathcal{P}(\mathcal{X})\big)$$

where the min operator over the t-norm values has been replaced by the average over the set. This definition allows faster convergence during the training of the model: the classical Fuzzy FOL formulation directly depends only on one item over the set of groundings (the argmin element), whereas the average depends on all elements and allows parallel optimization during training. Obviously, the Fuzzy FOL expression in this new formulation is perfectly verified (e.g. it assumes the value 1) iff the same atom assignments would make the old formulation be verified, so the two formulations are consistent.

Example 3.1. The formula $\forall x_1 \exists x_2 \;\; A(x_1) \wedge B(x_2)$, using a product t-norm for the quantifier-free part conversion corresponds to the following continuous generalization:

$$\Phi(\mathcal{P}(\mathcal{X})) = \frac{1}{|\mathcal{X}_1|} \sum_{x_1 \in \mathcal{X}_1} \max_{x_2 \in \mathcal{X}_2} A(x_1) \cdot B(x_2)$$

where $A(x_1)$, $B(x_2)$ are the fuzzy truth values of the two predicates A and B grounded with x_1 and x_2.

As commonly done in description logic [1], a different operator called n-existential quantifier, indicated as $\exists_n$, can be defined to express a property that should hold true for at least n objects. The fuzzy generalization of this quantifier is defined as

$$\exists_n x_i \;\; E\big(\mathcal{P}(\mathcal{X})\big) \quad \longrightarrow \quad \Phi_{\exists_n}\big(\mathcal{P}(\mathcal{X})\big) = \frac{1}{n} \sum_{x_i \in \mathcal{M}_n(t_E, \mathcal{P}, \mathcal{X}_i, \mathcal{X})} t_E\big(\mathcal{P}(\mathcal{X})\big)$$

where $\mathcal{M}_n(t_E, \mathcal{P}, \mathcal{X}_i, \mathcal{X})$ is the set of groundings of $x_i \in \mathcal{X}_i$ corresponding to the n maximum values for $t_E\big(\mathcal{P}(\mathcal{X})\big)$. The conversion of this quantifier collapses into the universal quantifier conversion as $n \to |\mathcal{X}_i|$ and into the existential quantifier as $n \to 1$.

It is interesting to note that this translation is computationally more approachable than what can be done in most other SRL approaches. MLNs, for example, replace the existential quantifier by a disjunction over all the groundings. The $\exists_n$ quantifier can be expressed as n nested existential quantifiers in a MLN, but the number of groundings of the formula would scale as $|G|^n$, where $|G|$ is the number of groundings of the considered variable. In SBR, the $\exists_n$ corresponds to selecting the n groundings that are the best candidates to satisfy the formula. This candidate selection can be done with complexity $O(|G| \log n)$ by using a heap data structure.

4. Semantic based regularization

Consider a multi-task learning problem, where a set of T functions must be estimated (*query or unknown functions*) and another T' functions (*evidence or given functions*) are known a priori. Let $\boldsymbol{f} = \{f_1, \ldots, f_T, f_{T+1}, \ldots, f_{T+T'}\}$ indicate the vector of functions.

We assume that a set of H functional constraints in the form $1 - \Phi_h(\boldsymbol{f}) = 0, 0 \leq \Phi_h(\boldsymbol{f}) \leq 1, \;\; h = 1, \ldots, H$ are provided in order to describe how the query functions should behave. Functionals can express a property of a single function or correlate multiple functions, so that learning can be helped by exploiting these correlations.

Let $\mathcal{H}_k$ be the functional space where the k-th function lives. Following the classical penalty approach for constrained optimization, the constraint satisfaction can be enforced by adding a term that penalizes their violation. The resulting cost function to be minimized is:

$$C[\boldsymbol{f}] = \sum_{k=1}^{T} ||f_k||^2_{\mathcal{H}_k} + \sum_{h=1}^{H} \lambda_h \left(1 - \Phi_h(\boldsymbol{f})\right) ,$$

where the first term penalizes non-smooth solutions and λ_h is the weight for the h-th constraint. A higher value of λ_h makes it more costly not respecting the constraint, and the constraint becomes hard as $\lambda_h \to \infty$.

The learning problem is relaxed by assuming that the constraints are enforced only over a finite sample of the input patterns. Typically, each pattern is represented as a vector of real-valued features, but we will see also cases where the patterns are simply represented by a unique identifier. In particular, the j-th function is associated to a set $\boldsymbol{\mathcal{X}}_j^\circ$ of pattern representations $\boldsymbol{x}_j$. It is possible that multiple functions share the same sample of patterns (e.g. $\boldsymbol{\mathcal{X}}_j^\circ = \boldsymbol{\mathcal{X}}_i^\circ \ i \neq j$). Since some functions may express relations across multiple patterns, the pattern representations associated to a function can be generally expressed as the combination of the patterns from a set of finite domains: $\boldsymbol{\mathcal{X}}_j^\circ = \boldsymbol{\mathcal{X}}_{j1} \times \boldsymbol{\mathcal{X}}_{j2} \times \ldots$.

Let $f_k(\boldsymbol{\mathcal{X}}_k^\circ)$ indicate the vector of values obtained by applying the function f_k to the set of patterns $\boldsymbol{\mathcal{X}}_k^\circ$ and $\boldsymbol{f}(\boldsymbol{\mathcal{X}}) = f_1(\boldsymbol{\mathcal{X}}_1^\circ) \cup f_2(\boldsymbol{\mathcal{X}}_2^\circ) \cup \ldots$ collects the groundings for all functions. Enforcing the constraints only over the samples of data, yields the following cost function:

$$C_e[\boldsymbol{f}(\boldsymbol{\mathcal{X}})] = \sum_{k=1}^{T} ||f_k||^2_{\mathcal{H}_k} + \sum_{h=1}^{H} \lambda_h \left(1 - \Phi_h(\boldsymbol{f}(\boldsymbol{\mathcal{X}}))\right) . \tag{2}$$

Please note that for the sake of simplicity (and with abuse of notation), the same symbol Φ was used to refer to both the exact and empirical approximation functionals.

The solution to the optimization task defined by the objective function in Equation (2) is a kernel expansion in the form:

$$f_k^*(\boldsymbol{x}) = \beta_k + \sum_{i=1}^{|\boldsymbol{\mathcal{X}}_k^\circ|} w_{ki}^\star K_k(\boldsymbol{x}_{ki}, \boldsymbol{x}) , \tag{3}$$

where β_k is a bias and $K_k(\cdot,\cdot)$ is the reproducing kernel associated to the space $\mathcal{H}_k$. The proof [10] is a straightforward extension of the Representer Theorem for plain kernel machines [42]. The bias β_k is added to specify the default value of a function. The weights of the k-th function are indicated as $\boldsymbol{w}_k = \{w_{k1}, \ldots, w_{k|\boldsymbol{\mathcal{X}}_k^\circ|}\}$. Plugging Equation (3) into (2), we get an expression that can be minimized by optimizing the $\boldsymbol{w}_k$ by gradient descent:

$$C_e[\boldsymbol{f}(\boldsymbol{\mathcal{X}})] = \sum_{k=1}^{T} \boldsymbol{w}_k^T \boldsymbol{G}_k \boldsymbol{w}_k + \sum_{h=1}^{H} \lambda_h \Big(1 - \Phi_h\big(\boldsymbol{f}(\boldsymbol{\mathcal{X}})\big)\Big) \tag{4}$$

where $\boldsymbol{f}(\boldsymbol{\mathcal{X}}) = \{\boldsymbol{G}_1\boldsymbol{w}_1, \ldots, \boldsymbol{G}_T\boldsymbol{w}_T, f_{T+1}(\boldsymbol{\mathcal{X}}_{T+1}^\circ), \ldots, f_{T+T'}(\boldsymbol{\mathcal{X}}_{T+T'}^\circ)\}$ and $\boldsymbol{G}_k$ is the Gram matrix for the k-th function, whose (i, j) element is equal to $K_k(\boldsymbol{x}_{ki}, \boldsymbol{x}_{kj})$. For the sake of simplicity, we have not explicitly added the vector of bias values as input to $\Phi_h(\boldsymbol{f}(\boldsymbol{\mathcal{X}}))$.

4.1. Logic and constraints

Let us assume that a knowledge base *KB*, consisting of a set of FOL formulas and a set of groundings for the variables, is provided to express some domain knowledge. Some of the predicates are unknown and must be estimated. The j-th unknown predicate is approximated by the function $\sigma(f_j(\cdot))$ where f_j is the function that must be learned, and $\sigma(\cdot)$ is a sigmoidal function mapping the functions' outputs onto the $[0, 1]$ range, on which the t-norms are defined. In the remainder of the paper we drop the σ to not overload the notation. The variables in the *KB* that are input to any f_j are replaced with the feature-based representation of the object grounded by the variables. We will indicate as $\boldsymbol{x}_i$ the representation of the object grounded by x_i, where $\rho(\boldsymbol{x}_i) = x_i$ is a function mapping a pattern representation into its object identifier. The groundings $\mathcal{X}_i$ of the i-th variable are therefore replaced by the set $\boldsymbol{\mathcal{X}}_i$, indicating the set of feature-based representations of the groundings. One constraint $1 - \Phi_i(\cdot) = 0$ for each formula F_i in the knowledge base is built by taking the fuzzy FOL generalization of the formula $\Phi_i(\cdot)$, where the unknown predicates are replaced by the learned functions, and the variables input to the learned functions are replaced by their duals iterating over the feature-based representations of the groundings.

Example 4.1. Let's suppose to have a text categorization task, where the documents must be assigned to the categories A, B, C, D. Let d be a generic document to be classified and $A(d), B(d), C(d), D(d)$ be the unknown indicator functions for the classes. We indicate with $\boldsymbol{d}$ the feature representation for d, which would typically be its bag-of-word TF or TF-IDF representation. The functions $f_A(\boldsymbol{d})$, $f_B(\boldsymbol{d})$, $f_C(\boldsymbol{d})$, $f_D(\boldsymbol{d})$ must be estimated to approximate respectively the unknown $A(d)$, $B(d)$, $C(d)$, $D(d)$. We will assume that the variable d will be grounded with two documents $\{d_1, d_2\}$, having as bag-of-word representations: $\mathcal{D} = \{\boldsymbol{d}_1, \boldsymbol{d}_2\}$, respectively. We will use this simple learning task as running example for the rest of the paper.

The fuzzy FOL generalization of the knowledge base provides a continuous and differentiable surrogate of the initial logic formulas such that $0 \leq \Phi_i(\cdot) \leq 1$. Therefore, the resulting constraints $1 - \Phi_i(\cdot) = 0$ are continuous and differentiable as well, and this will allow to train the query functions using any gradient-based learning schema.

The form of the constraints depends on which t-norm has been used to generalize the FOL rules. Whereas it is always true that all t-norms are consistent with classical logic when variables assume crisp values $(0, 1)$, the behavior of the various t-norms differs for the intermediate values, ultimately leading to different constraints. Similarly to the selection of the right kernel for kernel machines, the choice of a t-norm is always a matter of context, and it depends on the problem which is modeled [17]. For example, the minimum t-norm can be a better choice than a product t-norm when performing inference in presence of long conjunctive chains, where the value of the conjunction would vanish exponentially in the number of terms when using a product t-norm. The product t-norm is correct in this case as well, since it behaves consistently to its probabilistic-independence assumption, but training may be too slow. The t-norm selection should not have a strong impact in most applications, and it did not provide any significant change in all the experiments presented in this paper.

Example 4.2. Continuing the text categorization example, we can assume to have some external knowledge about the categories: $\forall d \ A(d) \vee B(d)$ expressing that any document must belong to class A or B. The constraint resulting from the fuzzy FOL generalization of the formula after substituting the query predicates with the unknown functions and the two document representations is:

$$\begin{aligned} 1 - \Phi(\{f_A(\mathcal{D}), f_B(\mathcal{D})\}) &= 1 - \frac{1}{2} \sum_{\boldsymbol{d} \in \mathcal{D}} \overbrace{\max\left(f_A(\boldsymbol{d}), f_B(\boldsymbol{d})\right)}^{t(f_A(\boldsymbol{d}), f_B(\boldsymbol{d}))} \\ &= 1 - \frac{1}{2}\left(\max\left(f_A(\boldsymbol{d}_1), f_B(\boldsymbol{d}_1)\right) + \max\left(f_A(\boldsymbol{d}_2), f_B(\boldsymbol{d}_2)\right)\right) \\ &= 0 \end{aligned}$$

where $t(\cdot)$ is the minimum t-norm representation for the propositional part of the formula. This constraint can be directly plugged into Equation (4) for optimization.

4.2. Nodes with no feature vector representation

It is common to have query predicates for which the input arguments (their groundings) have no associated feature representation. Most Statistical Relational Learning approaches are indeed assuming this case. In this case, the inputs can simply be listed as a sequence of unique identifiers. Equation (4) can still be used as generic expression of the cost function by setting the Gram matrix to be the identity matrix for all predicates that have no vectorial input e.g. $\boldsymbol{G}_k = \boldsymbol{I}$. In this case the vector of weights becomes the vector of values of the function over the inputs as $f_k(\boldsymbol{\mathcal{X}}_k^o) = \boldsymbol{G}_k \boldsymbol{w}_k = \boldsymbol{I}\boldsymbol{w}_k = \boldsymbol{w}_k$. These unknown values must be directly estimated without generalizing over the input space (which is null).

Obviously, an interesting case is when some functions have inputs represented as vectors of features and others don't. The experimental section will present some experiments in this mixed setting.

4.3. SBR as a multi-layer architecture

Given an arbitrary KB, the procedure described in the previous section can be generally encoded by a multi-layer network with the following structure:

- *input layer*: computation of the atoms, this is performed by computing the query and evidence function values for all possible groundings;
- *propositional layer*: the value of the t-norm expression of the propositional part of each formula is computed for each compatible combination of atoms;
- *quantifier layers*: the t-norm values computed at the previous layer are aggregated by the average or max operator for the universal and existential quantifiers, respectively. Please note that the number of quantifier layers is not fixed, as the aggregation of the outputs is recursively nested according to the number of quantifiers in the FOL formula. The output of this layer is a score in $[0, 1]$ expressing how strongly the FOL formula is respected.
- *output layer*: accumulation by summation of the contributions coming from the single formulas.

Learning and inference are intractable in large domains, because of the exponential growth of the number of groundings with respect to the number of nested quantifiers. However, in most applications, a large portion of the ground clauses are trivially satisfied or not satisfied by the known evidence atoms regardless of unknown assignments. These groundings can be discarded without introducing any approximation in the training or inference process. In the context of MLNs, many heuristics and algorithms like FROG preprocessing [43], Tuffy [35] or LazySAT [44] have been proposed to detect and discard these non-informative groundings. Since the grounding process of MLNs and SBR is essentially the same, these algorithms

can be directly reused in SBR. In particular, FROG preprocessing has been used in the presented experiments to prune the encoded network in the input and propositional layers, whenever the grounding of the evidence predicates makes the formula always holding true or false, independently on the values assumed by the query predicates. In this case the contribution to the cost function of the grounding is constant and it has no effect on the learning process. Therefore, it is useless to instantiate this grounding in the encoded network. A common case where the pruning process can be applied is when dealing with a FOL formula in the following form: $R(x) \Rightarrow A(x)$ where $R(x)$ is an evidence predicate. When it holds that $R(p) = \mathit{false}$ given the grounding $x = p$, the rule is verified regardless of the value of the query predicate $A(p)$. Therefore the grounded predicates $R(p), A(p)$ can be safely discarded from the encoding network during training, unless appearing in another FOL formula.

To summarize, the network encoded by SBR for a given KB can be defined as in the following.

Definition 4.3 *(SBR network).* Assume a FOL KB composed by rules and predicates, which are grounded by a set of constants. Let $\boldsymbol{x}$ represent the feature vector associated to a grounding x. We indicate with $f_i(\cdot)$ the function implemented by a Kernel Machine, approximating the i-th unknown predicate p_i, where $f_i(\cdot)$ takes as input the feature vector representations of the constants grounding the predicate. SBR builds a multi-layer network $\mathcal{N}$ computing the fuzzy FOL approximation of the KB, where the value $f_i(\boldsymbol{x})$ replaces a grounded unknown predicate $p_i(x)$.

Some examples of KB and relative network encodings will be shown in the remainder of the paper.

5. Some special cases of SBR

This section will show how SBR can reproduce standard and Transductive SVMs, manifold and graph regularization. At the same time all these learning schemas can be mixed and extended arbitrarily with the full expressiveness of FOL.

5.1. Case 1: SVMs

Let $\mathcal{X}_k^+, \mathcal{X}_k^-$ be the sets of positive and negative examples for the k-th unknown predicate p_k, ($\mathcal{X}_k^\circ = \mathcal{X}_k^+ \cup \mathcal{X}_k^-$). The following logic formula expresses the fact that p_k is constrained on the values assumed over the supervised data:

$$\forall x \left(P_k(x) \wedge p_k(x)\right) \vee \left(\neg P_k(x) \wedge \neg p_k(x)\right)$$

where $x \in \mathcal{X}_k$ and the predicate $P_k(x)$ is an evidence function holding true iff x is a positive example for the query predicate p_k (e.g. $x \in \mathcal{X}_k^+$). Using the minimum t-norm and replacing p_k with its approximation f_k, this corresponds to the following constraint:

$$\begin{aligned}
1 - \Phi\left(\{P_k(\mathcal{X}_k^\circ), f_k(\boldsymbol{\mathcal{X}}_k^\circ)\}\right) &= 1 - \frac{1}{|\boldsymbol{\mathcal{X}}_k^\circ|} \sum_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}_k^\circ} \max\Big(\min\left(P_k(\rho(\boldsymbol{x})), f_k(\boldsymbol{x})\right), \min\left(1 - P_k(\rho(\boldsymbol{x})), 1 - f_k(\boldsymbol{x})\right)\Big) \\
&= 1 - \frac{1}{|\boldsymbol{\mathcal{X}}_k^\circ|}\Big(\sum_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}_k^+} \min\left(1, f_k(\boldsymbol{x})\right) + \sum_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}_k^-} \min\left(1, 1 - f_k(\boldsymbol{x})\right)\Big) \\
&= \frac{1}{|\boldsymbol{\mathcal{X}}_k^\circ|}\Big(\sum_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}_k^+} \max\left(0, 1 - f_k(\boldsymbol{x})\right) + \sum_{\boldsymbol{x} \in \boldsymbol{\mathcal{X}}_k^-} \max\left(0, f_k(\boldsymbol{x})\right)\Big) \\
&= 0
\end{aligned}$$

where $\max(0, 1 - f_k(\boldsymbol{x}))$ is the hinge loss used by regular SVM. The same hinge loss would appear for negative supervisions if using the -1 value for negative supervisions instead of the 0 used here. Therefore, plugging the previous constraint into Equation (4) shows that classical SVMs are a special case in our framework, when expressing the fitting of the supervised data as logic knowledge. Please note that the constraints resulting from the supervised data fitting are a special case of a constraint that is convex and relatively easy to optimize. We will discuss this point into more details in the following section.

When unsupervised data is also available, it is possible to express the supervisions using two evidence predicates. In particular, let $\boldsymbol{\mathcal{X}}_k^+$, $\boldsymbol{\mathcal{X}}_k^-$, $\boldsymbol{\mathcal{X}}_k^u$ be the available positive, negative and unsupervised examples for task k (e.g. $\boldsymbol{\mathcal{X}}_k^\circ = \boldsymbol{\mathcal{X}}_k^+ \cup \boldsymbol{\mathcal{X}}_k^- \cup \boldsymbol{\mathcal{X}}_k^u$). The learned function is constrained in terms of the assumed values on the supervised data by specifying the following logic formulas F_p, F_n:

$$\begin{aligned}
F_p: &\ \forall x\, P_k(x) &\Rightarrow\ & p_k(x) \\
F_n: &\ \forall x\, N_k(x) &\Rightarrow\ & \neg p_k(x)
\end{aligned}$$

where $P_k(x)$ and $N_k(x)$ are evidence functions holding true iff x is a positive or negative example for the query predicate p_k, respectively. Also this formulation corresponds to the learning task solved by standard SVM training (patterns that are

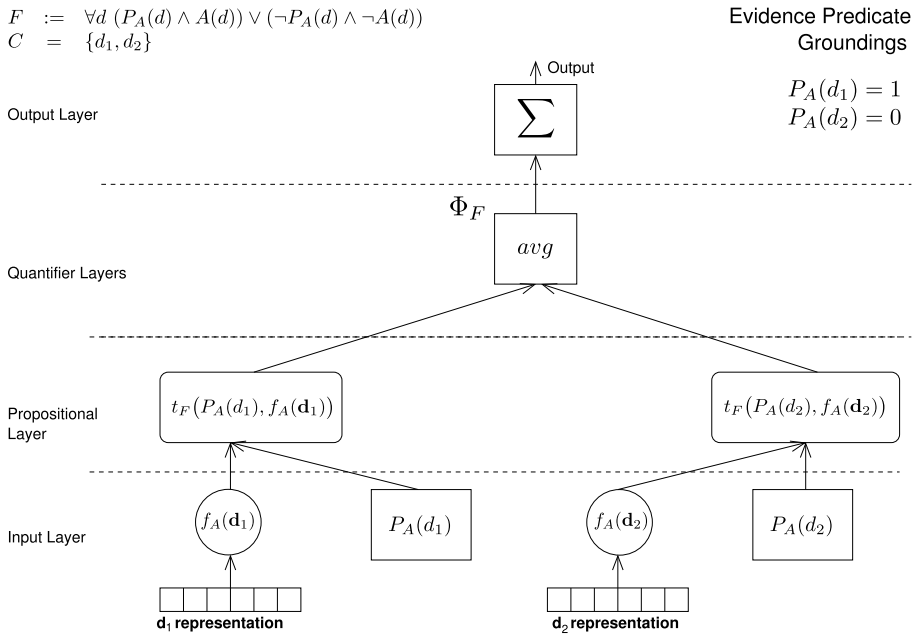

Fig. 1. Network encoded from a knowledge base containing the F formula as see in the figure, which implements the SVM supervised training, and that has been grounded with the constants (patterns) $C = \{d_1, d_2\}$.

neither positive nor negative supervised do not contribute to the cost function) and it will be used as base step for solving more complicated learning tasks, where unsupervised data plays a role, in the remainder of the paper.

Example 5.1. In our text categorization example, let us assume that d_1 belongs to class A and d_2 does not. This can be expressed by stating that d_1 is a positive supervised example: $P_A(d_1) = true$, while $P_A(d_2) = false$. We can then express the rule: $\forall d\ (P_A(d) \wedge A(d)) \vee (\neg P_A(d) \wedge \neg A(d))$ to incorporate the labeled data into the learning task. The following constraint results from the fuzzy FOL generalization of the formulas using the minimum t-norm and after substituting the document identifiers with their feature representations and the query predicates with the corresponding functions to estimate:

$$1 - \Phi\big(f_A(\mathcal{D})\big) = \frac{1}{2}\big(\max(0, 1 - f_A(\boldsymbol{d}_1)) + \max(0, f_A(\boldsymbol{d}_2))\big) = 0\ ,$$

Fig. 1 shows the network which is encoded when performing this simple learning task.

5.2. Case 2: manifold regularization

Manifold Regularization [3] assumes that the learned function should be regular over the input manifold, which is represented as a graph, whose edges connect the input patterns. The graph can be directly built over the input data distribution, or built from external knowledge like html hyperlinks in a web page classification problem. Laplacian SVM (LSVM) [31] is an effective semi-supervised approach to train SVMs under the manifold regularization assumption.

Manifold Regularization is a special case of SBR, where there are rules forcing the fitting of the supervised examples, as previously described, and an additional rule expressing the manifold assumption. In particular, let $R(x, y)$ be a given (evidence) relation expressing whether x, y are connected on the manifold. The manifold assumption in a logic setting for a predicate p is expressed by the following FOL formula, asserting that two connected points should either be both true or both false:

$$\forall x\, \forall y\ R(x, y) \Rightarrow (p(x) \wedge p(y)) \vee (\neg p(x) \wedge \neg p(y))\ .$$

Example 5.2. Let us assume that, in the previously considered text categorization task, d_1 links to d_2 via a hyperlink $R(d_1, d_2) = 1$, while no links are available between the other documents. Manifold regularization in this domain is expressed for the category A by a formula F_R, whose fuzzy FOL generalization Φ_{F_R} is obtained by substituting the query predicate A with the unknown function f_A and using the minimum t-norm. This yields the following constraint:

$$1 - \Phi_{F_R}\big(f_A(\mathcal{D})\big) = 1 - \frac{1}{4}\Big(3 + \max\big(\min(f_A(\boldsymbol{d}_1), f_A(\boldsymbol{d}_2)), \min(1 - f_A(\boldsymbol{d}_1), 1 - f_A(\boldsymbol{d}_2))\big)\Big) = 0$$

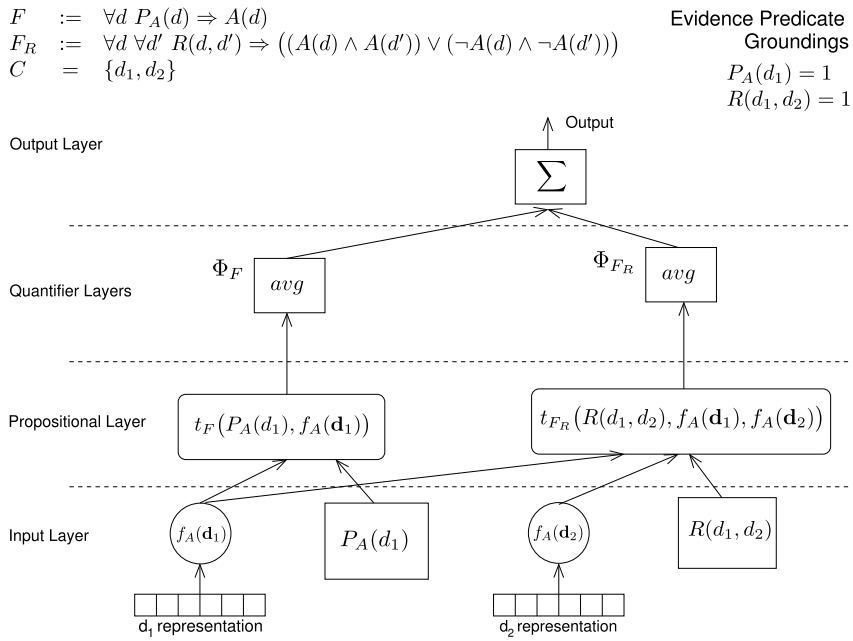

Fig. 2. Multi-layer network that is encoded by SBR for the $KB = \{F, F_R\}$ where F_R implements manifold regularization and F supervised learning for the positive supervisions. The KB has been grounded with the constants (patterns) $C = \{d_1, d_2\}$.

where the first contribution comes from the 3 groundings for which the R predicate is false and the rule trivially verified. This contribution does not affect the training process and can be dropped. This constraint can be plugged into Equation (4) to get the cost function to optimize. Fig. 2 shows the encoded network after FROG-preprocessing for this learning task with also added one positive supervision expressed by a formula indicated as F. This rule is in the form that can also account for any unsupervised data as previously described. FROG-preprocessing keeps in the network only the groundings for which either P_A or R are true, as the rules F and F_R are always verified based on the evidence predicates otherwise. Similar rules could be added to express manifold regularization for the other classes B, C, D.

5.3. Case 3: hierarchical classification

Complex classification tasks often involve a large number of classes organized into a hierarchy. Typically, a hierarchy can be represented as a Directed Ordered Acyclic Graph (DOAG), where each node corresponds to a class. A single root node is provided as starting point of the classification process, from where all other nodes can be reached. The classification process explores a set of paths on the graph, where each path ends with a leaf node. A two level-hierarchical classification with n classes at the first level can be expressed by the rules:

$$\forall x\ p_1(x) \vee \ldots \vee p_n(x)$$
$$\forall x\ p_i(x) \Rightarrow p^c_{i1}(x) \vee \ldots \vee p^c_{in_i}(x) \quad i = 1, \ldots, n$$

where $p_i,\ i = 1, \ldots, n$ are the father classes at the first level of the hierarchy, n_i is the number of child classes of class p_i and $p^c_{ij},\ j = 1, \ldots, n_i$ are the child classes of p_i. Class priors for each category can also be expressed via the rules:

$$\exists_{m_i} x \quad p_i(x) \quad i = 1, \ldots, n$$
$$\exists_{m_{ij}} x \quad p^c_{ij}(x) \quad i = 1, \ldots, n \ \ j = 1, \ldots, n_i$$

where m_i and m_{ij} can estimated from the supervised data. This schema can be recursively generalized to taxonomies of arbitrary depth.

Example 5.3. For the text categorization example, we assume that C is the only child class of A in the taxonomy. Therefore, the formula $F_T := \forall d\ A(d) \Rightarrow C(d)$ expresses the taxonomical information that any document belonging to class A belongs also to class C. The resulting constraint obtained from the fuzzy FOL generalization Φ_{F_T} of the formula over the set $\mathcal{D}$ of available documents is:

$$1-\Phi_{F_T}\left(\{f_A(\mathcal{D}),f_C(\mathcal{D})\}\right)=1-\frac{1}{2}\sum_{\boldsymbol{d}\in\mathcal{D}}\overbrace{\begin{cases}1 & f_C(\boldsymbol{d})\geq f_A(\boldsymbol{d})\\ f_C(\boldsymbol{d}) & f_C(\boldsymbol{d})<f_A(\boldsymbol{d})\end{cases}}^{t_T(f_A(\boldsymbol{d}),f_C(\boldsymbol{d}))}=0$$

where t_T is the fuzzy generalization of the formula F_T using the residuum of the minimum t-norm.

5.4. Case 4: transductive SVMs

Transductive SVMs (TSVMs) [48] extend SVMs by finding a hyperplane with maximum separation margin for the labeled data and the labeling of the unsupervised data induced by the hyperplane itself. Therefore, Transductive SVMs tend to place the separating hyperplane on low density regions of the input space.

TSVMs can be expressed in SBR by adding a FOL formula forcing all pattern classifications to be either true of false:

$$\forall x\ p_k(x)\vee\neg p_k(x)$$

where $p_k(x)$ is the k-th query predicate that returns true iff a pattern x belongs to a given class. While the above formula is always trivially verified in standard FOL, the same does not apply to the fuzzy generalizations of FOL, where the classification scores can be anywhere in the $[0,1]$ range.

As in Transductive SVMs, trivial solutions are avoided by forcing the balancing between the number of unlabeled patterns to be positively or negatively classified using a prior determined over the supervised data:

$$\begin{aligned}&\exists_{n_k}x\ \ p_k(x)\\ &\exists_{m_k}x\ \neg p_k(x)\ ,\end{aligned}$$

where, given the sample $\mathcal{X}_k$ of the variable x, n_k, m_k ($n_k+m_k=|\mathcal{X}_k|$) are the expected numbers of patterns in the sample for which the unknown predicate should hold true and false, respectively.

5.5. Case 5: graph regularization

Graph regularization [52] is a transductive learning task where all generalization happens during training. All the data patterns are arranged as nodes of a graph, whose edges are associated to the weights expressing the degree of similarity of the connected patterns. Some graph nodes are supervised and, therefore, associated with a target value. The learning task consists in assigning a value to all the nodes in the graph, while being smooth over similar (connected) nodes. No feature representation is available in any node and generalization happens at a purely topological level.

In SBR, the supervisions for a Graph Regularization task are expressed using the same logic formulas described in Section 5.1. Like for manifold regularization, let $R(x,y)$ be a known relation expressing whether two patterns are connected, then the following rule can be used to express the smoothness over the graph connections: $\forall x\ \forall y\ R(x,y)\Rightarrow(p_k(x)\wedge p_k(y))\vee(\neg p_k(x)\wedge\neg p_k(y))$, for the k-th query predicate p_k. The encoded network has the same structure as the example presented in Fig. 2, but with void input representations in the first layer and functions implemented via the special kernel discussed in Section 4.2.

6. Training

Training in SBR means to determine the weights of the kernel machines in the input layer or, directly, the outputs for the functions with null feature-based inputs. The weights are optimized via gradient descent using a back-propagation schema, where the output layer computes the derivative with respect to each constraint: $\frac{\partial C_e}{\partial \Phi_k}$. In the quantifier layers, the derivative of a constraint with respect to each predicate grounding is computed: $\frac{\partial \Phi_k}{\partial t_{\Phi_k}}$. At the propositional level the derivatives with respect to the single functions are computed: $\frac{\partial t_{\Phi_k}}{\partial f_i}$. At the input level the derivatives with respect to the single parameters are computed: $\frac{\partial f_i}{\partial w_{ij}}=K_i(\boldsymbol{x}_j,\cdot)$.

The overall derivative of the cost function with respect to the j-th weight of the i-th function w_{ij} is:

$$\frac{\partial C_e}{\partial w_{ij}}=\sum_k\frac{\partial C_e}{\partial \Phi_k}\cdot\frac{\partial \Phi_k}{\partial w_{ij}}=\sum_k\frac{\partial C_e}{\partial \Phi_k}\cdot\left(\sum_{t_{\Phi_k}}\frac{\partial \Phi_k}{\partial t_{\Phi_k}}\cdot\frac{\partial t_{\Phi_k}}{\partial f_i}\cdot\frac{\partial f_i}{\partial w_{ij}}\right)\ . \tag{5}$$

When no input feature representation is provided, the weight is the function value itself $w_{ij}=f_{ij}$, and no back-propagation over the function weights is needed. Resilient gradient descent using a custom learning rate for each parameter was empirically found to converge very quickly and was therefore used in all the presented experiments.

6.1. Logic formulas and complexity in optimization

Not all FOL formulas are translated into a constraint which is well suited for optimization. Indeed, this section will show that the general intractability of FOL inference is directly translated into a SBR cost function that is plagued by many local minima.

Let's consider universally quantified FOL formulas in DNF form:

$$\forall x_1 \ldots \forall x_n \overbrace{\left(n_{11} P_1(x_1) \wedge \ldots \wedge n_{1n} P_n(x_n)\right)}^{minterm\ 1} \vee \ldots \vee \overbrace{\left(n_{k1} P_1(x_1) \wedge \ldots \wedge n_{kn} P_n(x_n)\right)}^{minterm\ k}$$

where n_{ij} determines whether the j-th variable in the i-th minterm is negated or not. Applying a double negation and using the DeMorgan rule, yields the following expression for each grounding:

$$\neg\Big(\neg\big(n_{11} P_1(x_1) \wedge \ldots \wedge n_{1n} P_n(x_n)\big) \wedge \ldots \wedge \neg\big(n_{k1} P_1(x_1) \wedge \ldots \wedge n_{kn} P_n(x_n)\big)\Big)$$

Now converting the above propositional expression using the product t-norm, and assuming to generalize the atoms in the $[0, 1]$ range using the unknown function approximations, we get the constraint:

$$1 - \Phi\big(\boldsymbol{f}(\boldsymbol{\mathcal{X}})\big) = \frac{1}{\prod_{i=1}^{n} |\boldsymbol{\mathcal{X}}_i|} \sum_{\boldsymbol{x}_1} \ldots \sum_{\boldsymbol{x}_{n-1}} \prod_{r=1}^{k} \Big(1 - \prod_{i \in A_r^p} f_i(\boldsymbol{x}_i) \prod_{j \in A_r^n} (1 - f_j(\boldsymbol{x}_j))\Big) = 0$$

where A_r^p and A_1^n are the set of non-negated and negated atoms in the r-th minterm and, as in the rest of the paper, we have omitted the squashing function $\sigma(\cdot)$ in front of each $f_i(\cdot)$ which keeps the values in the $[0, 1]$ range. An assignment verifying the r-th minterm will result in $\prod_{i \in A_r^p} f_i(\boldsymbol{x}_i) \prod_{j \in A_r^n} (1 - f_j(\boldsymbol{x}_j)) = 1$. Therefore, a null contribution will result from any assignment verifying one minterm when summing over all the groundings. Since all minterms are by construction different and the polynomial equation is continuous and assuming values greater or equal to zero as guaranteed by any t-norm, the resulting expression has as many local minima as the number of true configurations in the truth table for the grounded propositional formula (e.g. this number is by construction equal to the number of minterms in the initial DNF). Therefore, there is a perfect duality between the number of possible assignments of the atoms verifying the FOL formula for a given grounding of the variables, and the number of local minima introduced into the constraint resulting from generalizing the formula to a continuous domain. Not surprisingly, this shows that optimization in the continuous domain is as hard as finding the correct assignments in the original FOL formulation. It is clear that formulas with a single minterm correspond to a convex constraint and have a single minimum, we will see examples of this special case in the following paragraph. While the product t-norm was used in this paragraph, because it is simple to study the solutions of the resulting polynomial, this result can be extended to any strict t-norm, since all strict t-norms are isomorphic to the product t-norm [45].

6.2. Stage-based learning

As shown in the previous sections, the cost function is plagued by many local minima when dealing with a non-trivial knowledge base. This section will discuss a heuristic, which has been experimentally proved to allow to successfully train models on large scale datasets.

In constraint satisfaction programming [41], picking up the variable with the smallest number of admissible values remaining in its domain is one of the most commonly used heuristics to select the next variable to assign during search [16]. As explained by Haralick and Elliot [19], this heuristic minimizes the depth of the search tree. Following the standard notation used in Rossi et al. [41], we will refer to this heuristic as *dom*. In the context of SBR, the *dom* heuristic corresponds to force earlier the formulas with a lower number of possible valid (e.g. verifying the formula) assignments to the atoms to be learned, once the evidence predicates have been grounded. As explained in the previous section, these formulas introduce a lower number of local minima into the cost function.

For example, let's consider the following KB:

1 $\forall x\ (P_1(x) \wedge p_1(x)) \vee (\neg P_1(x) \wedge \neg p_1(x))$

2 $\forall x\ P_1(x) \Rightarrow p_1(x) \vee p_2(x)$

3 $\forall x\ p_1(x) \vee p_2(x) \vee p_3(x)$

where the predicates p_1, p_2, p_3 must be learned. The first rule has the same form of the one presented in Section 5.1 to express the fitting of the supervised examples. This rule has the minimum possible degree of freedom as there is only one possible assignment to the atom $p_1(x)$ for each evidence grounding of $P_1(x)$. Therefore, this rule should always be the first to be used in training according to the *dom* heuristic. Not surprisingly, the previous section showed that rules with one single degree of freedom are convex when translated into a continuous constraint using t-norms. Rule 2 has

three and four possible assignments when the evidence grounded predicate $P_1(x)$ holds true and false, respectively. Rule 3 has seven possible different assignments of the query predicates. Therefore, the rule 2 should be added to the training process before than the last one, as it has less degrees of freedom. Since the assignments verifying a rule depend on the evidence predicate assignments, we usually use the average over all possible evidence groundings for a rule (in the example the average is 3.5 for rule 2). We leave for further studies how to improve this simple schema. For example, it would be possible to compute the expected number of possible assignments of the query predicates from the distribution of the observed evidence groundings in the training data.

The *dom* heuristic applied to SBR has connections with research in deep learning that has shifted the attention on teaching plans in a more systematic way [4], but also at a more generic level with studies in the field of developmental psychology, since it is well-known that many animals experiment stage-based learning [37]. The experimental results will show that it is indeed beneficial to split the optimization problem into multiple stages, where the constraints are introduced in order of complexity as dictated by *dom*.

7. Collective classification

Collective Classification is the task of performing inference over a set of instances that are connected among each other via a set of relationships. Collective classification is often an easier task than independent pattern classification, because the relationships can be exploited to enforce classification consistency.

Collective classification in SBR assumes to have available some FOL knowledge, which is converted into a set of constraints using the previously described methodology. Given a test set composed of a set of groundings, the collective classification process will force the test set classification assignments to respect the constraints.

In particular, let $f_k(\mathcal{X}'_k)$ indicate the vector of values obtained by evaluating the kernel machine function f_k over the data points of the test set $\mathcal{X}'_k$. The set of vectors will be compactly referred to as: $\boldsymbol{f}(\mathcal{X}') = f_1(\mathcal{X}'_1) \cup \ldots \cup f_T(\mathcal{X}'_T)$. If no kernel machine has been trained for f_k (no examples or no feature representations were available during training), $f_k(\mathcal{X}'_k)$ is assumed to be just filled with default values equal to 0.5.

Collective classification searches for the values $\bar{\boldsymbol{f}}(\mathcal{X}') = \bar{f}_1(\mathcal{X}'_1) \cup \ldots \cup \bar{f_T}(\mathcal{X}'_T)$ respecting the FOL formulas on the test data, while being close to the prior values established by the kernel machines over the test data:

$$C_{cc}[\bar{\boldsymbol{f}}(\mathcal{X}'), \boldsymbol{f}(\mathcal{X}')] = \frac{1}{2}\sum_{k=1}^{T} |\bar{f}_k(\mathcal{X}'_k) - f_k(\mathcal{X}'_k)|^2 + \sum_h \Big(1 - \Phi_h\big(\bar{\boldsymbol{f}}(\mathcal{X}')\big)\Big)$$

Optimization can be performed via gradient descend by computing the derivative with respect to the function values:

$$\frac{\partial C_{cc}[\bar{\boldsymbol{f}}(\mathcal{X}'), \boldsymbol{f}(\mathcal{X}')]}{\partial \bar{f}_k(\mathcal{X}')} = \bar{f}_k(\mathcal{X}') - f_k(\mathcal{X}') - \sum_h \left(\frac{\partial \Phi_h\big(\bar{\boldsymbol{f}}(\mathcal{X}')\big)}{\partial \bar{f}_k(\mathcal{X}')}\right) \tag{6}$$

As shown in Equation (6), SBR collective classification reuses the same schema to compute the gradients of the constraints $\frac{\partial \Phi_h}{\partial \bar{f}_k}$ as shown in Equation (5). However, whereas the gradient was computed with respect to the functions' weights in the training phase, only the first terms of the chain rule have to be taken into account during collective classification:

$$\frac{\partial \Phi_h}{\partial \bar{f}_k} = \sum_{t_{\Phi_h}} \frac{\partial \Phi_k}{\partial t_{\Phi_h}} \cdot \frac{\partial t_{\Phi_h}}{\partial \bar{f}_k} \,.$$

Indeed, since the input weights are now fixed, no back-propagation of the derivative of the error down to the weights is needed. This provides an elegant solution to collective classification, which employs the same back-propagation routines used in training with no additional complexity in the implementation.

8. SBR as a probabilistic model

In this section we highlight some probabilistic interpretations of the solutions found by SBR. Given the constraints $\Phi = \{\Phi_1, \ldots, \Phi_H\}$ computed over the data $\mathcal{X}$, the probability distribution $P(V = \boldsymbol{f}|\mathcal{X}, \Phi)$ of the possible assignments to the functions $\boldsymbol{f}$ is assumed to follow an exponential model as:

$$\begin{aligned} P(V = \boldsymbol{f}|\mathcal{X}, \Phi) &= \frac{1}{Z} \cdot \phi_{pr}(\boldsymbol{f}) \cdot \phi\big(\boldsymbol{f}(\mathcal{X}), \Phi\big) \\ &= \frac{1}{Z} \exp\left(-\sum_k ||\boldsymbol{f}_k||^2 + \sum_h \lambda_h \Phi_h\big(\boldsymbol{f}(\mathcal{X})\big)\right) , \end{aligned}$$

where Z is a normalization factor, $\phi\big(\boldsymbol{f}(\mathcal{X}), \Phi\big)$ measures how well $\boldsymbol{f}$ respects the constraints and $\phi_{pr}(\boldsymbol{f})$ penalizes irregular solutions. The prior is the expression of the classical Tikhonov regularization in terms of prior probabilities [50] when assuming a Gaussian prior: $P_{pr}(\boldsymbol{f}) \propto \phi_{pr}(\boldsymbol{f}) = \exp(-\sum_k ||\boldsymbol{f}_k||^2)$. Therefore,

$$\log P(V = \boldsymbol{f}|\boldsymbol{\mathcal{X}}, \Phi) = -\log Z - \sum_k ||f_k||^2 + \sum_h \lambda_h \Phi_h\big(\boldsymbol{f}(\boldsymbol{\mathcal{X}})\big)\,. \tag{7}$$

This result provides a probabilistic interpretation of the SBR formulation given in Equation (2) (the change of sign is due to the fact that the probability is maximized, where in Equation (2) the cost is minimized). In the special case of a constraint representing a formula with only universal quantifiers, it holds that:

$$P(V = \boldsymbol{f}|\boldsymbol{\mathcal{X}}, \Phi) = \frac{1}{Z}\exp\left(-\sum_k ||f_k||^2 + \sum_h \lambda'_h \sum_{i_h} t_{\Phi_h}\big(\boldsymbol{f}(\boldsymbol{x}_{i_h})\big)\right)\,, \tag{8}$$

where i_h iterates over the G_h possible groundings of the h formula F_h, having $t_{\Phi_h}(\cdot)$ as its the t-norm conversion, $\boldsymbol{x}_{i_h}$ is the set of pattern representations of the i-th grounding of F_h, $\boldsymbol{f}(\boldsymbol{x}_{i_h})$ indicates the set of values returned by the functions when computed on $\boldsymbol{x}_{i_h}$ and $\lambda'_h = \frac{\lambda_h}{|G_h|}$.

8.1. SBR and Markov logic networks

A Markov network is a model for the joint distribution of a set of variables V and it is composed of an undirected graph expressing the variable dependencies and a set of potential functions. The graph has a node for each variable, and the model has a potential function for each clique in the graph. A potential function is a non-negative function of the state of the corresponding clique. The joint distribution represented by a Markov network is given by

$$P(V = v) = \frac{1}{Z}\prod_k \Phi(v_{\{k\}})$$

where $v_{\{k\}}$ is the state of the variables that appear in that clique and Z is the partition function. Markov networks are often represented as log-linear models, where each clique potential is replaced by an exponentiated weighted sum of features of the state:

$$P(V = v) = \frac{1}{Z}\exp\sum_j w_j f_j(v)$$

A Markov Logic Network (MLN) [40] is a set of pairs (F_h, λ_h), where F_h is a formula in first-order logic and λ_h is a real number. Given a finite set of constants C_h, defining the groundings of all the variables appearing in the F_h, a Markov network MLN is defined as:

1. an MLN contains one node for each possible grounding of each predicate, the value of the node is 1 iff the atom is true.
2. an MLN contains one feature for each possible grounding of each formula F_h. The value of this feature is equal to 1 iff the ground formula is true. The weight of the feature is λ_h.

The probability distribution over possible assignments v specified by the ground Markov network MLN is given by

$$P(V = v) = \frac{1}{Z}\exp\left(\sum_h \lambda_h n_h(v)\right)$$

where $n_h(v)$ is the number of true groundings of F_h in v.

Definition 8.1. A Markov Fuzzy Logic Network (MFLN) is a set of pairs (F_h, λ_h), where F_h is a formula in first-order logic and λ_h is a real number. Together with a finite set of constants C_h, defining the groundings of all the variables appearing in the F_h, it defines a Markov network MFLN as follows:

1. an MFLN contains one node for each possible grounding of each predicate, the value of the node is the degree of truth of the atom.
2. an MFLN contains one feature for each possible grounding of each formula F_h. The value of this feature is equal to the degree of truth of the formula computed via a fuzzy FOL generalization of F_h. The weight of the feature is λ_h.

The resulting network will contain one clique for each grounded formula. An MFLN extends an MLN allowing non-binary features and a continuous degree of truth for the predicates. While a MLN counts the number of true groundings of a formula in the world, an MFLN computes the sum of the degrees of satisfaction of the formula computed via a t-norm.

Let's assume that the unknown predicates are approximated with the output of the kernel machines $\boldsymbol{f}$ applied over the groundings. Let $\boldsymbol{x}_{i_h}$ be the set of pattern representations associated to the i-th grounding of F_h and $\boldsymbol{f}(\boldsymbol{x}_{i_h})$ be the set of

values returned by the functions when computed over the $\boldsymbol{x}_{i_h}$. The value of the feature associated to this grounding is the t-norm value $t_{\Phi_h}\big(\boldsymbol{f}(\boldsymbol{x}_{i_h})\big)$.

Therefore,

$$P_\Phi(V = \boldsymbol{f}) = \frac{1}{Z}\exp\left(\sum_h \lambda_h \sum_{i_h} t_{\Phi_h}\big(\boldsymbol{f}(\boldsymbol{x}_{i_h})\big)\right). \tag{9}$$

By comparing Equations (8) and (9), it emerges that SBR can be seen as a MLN where the FOL formulas and node values are replaced by their fuzzy generalization with the node values computed by kernel machines.

8.2. Discussion

A fundamental difference among MLNs and SBR is that MLNs either in their standard or hybrid version [49] cannot incorporate the feature-based pattern representations if not associating a rule and weight to each feature: for example see how text categorization with MLNs is performed by Domingos and Summer [12]. On the other hand, SBR implements a multi-layer architecture where pattern representations can be dealt using kernel machines at the first layer. This has various advantages in terms of training and flexibility. Indeed, MLNs see all the weights (defining the input and higher level inference) at the same level, while SBR can employ more appropriate and efficient learning schema for the input level, or it can even perform different training phases like the simple strategy employed described in Section 6.2. Unlike MLNs, SBR can deal with continuous, high-dimensional and highly correlated feature vectors. Furthermore, SBR can be defined in cases where the input representation is not vectorial or even unknown, as only the kernel values over the inputs are needed in order to perform training and inference.

The connections that have been enlighten in this paragraph between MLNs and SBR suggest that it is possible to reuse the MLN training mechanism to learn the rule weights λ_h in the output layer of SBR using Equation (9). Like in MLNs this would require to compute the partition function, which often leads to inefficiency and approximations. However, the experimental results show that learning the input layer weights in SBRs is often more effective than solving the MLN learning task in many contexts.

9. Experimental results

The proposed framework has been the subject of a large experimental analysis that was carried out by using the *SBR software package*.[1]

9.1. Transductive learning: text categorization on the CORA dataset

The CORA research paper dataset is a relational dataset containing information about papers and associated authors [30]. The CORA dataset collects papers classified into a topic hierarchy of 80 classes with 10 classes at the first level.[2] In addition, authors have been classified into a set of 10 classes depending on their major topic of research interest. A random sample of 2000 papers belonging to at least one of the 10 top level classes was extracted together with the 3928 authors having at least one publication in the selected sample of papers. Each paper was associated with a vectorial representation containing the title represented as bag-of-words with TF-IDF weights. There is no profile of authors, being symbolic entities without feature representation. The learning task consists of predicting the category of the papers and the author research area. This is a multi-label dataset, since authors can be associated to multiple categories. Papers have been split into three sets: published before the year 1995, papers published in year 1995 and all later papers. The resulting sets contain 949 papers and 2272 authors, 316 papers and 605 authors, and 735 papers and 1051 authors, respectively. The three sets form the pools from which the training, validation and test datasets are sampled from, respectively. This simulates a real world scenario, where the training process is performed at a certain time and testing is expected to involve new incoming papers. In particular, five folds have been generated by randomly selecting n% of the papers and n% of authors ($n = 10, 20, 30, 40, 50$) for which supervisions are kept in the first and second sets as training and validation data. Experiments have been carried out in a transductive context, where the test data is available as unsupervised data during training, and by averaging over 5 folds.

Knowledge base The available prior knowledge is modeled in terms of FOL predicates. Let $\mathcal{B} := \{\texttt{false}, \texttt{true}\}$ and let us denote the paper and author domains by $\mathcal{P}$ and $\mathcal{A}$, respectively. Notice that while $\mathcal{P} \in \mathbb{R}^d$, $\mathcal{A}$ is simply a set of author identifiers. Let us define the following predicates according to the relational representation in Fig. 3:

[1] The SBR package can be downloaded at https://sites.google.com/site/semanticbasedregularization/home/software.
[2] The data base can be downloaded at http://people.cs.umass.edu/~mccallum/data.html.

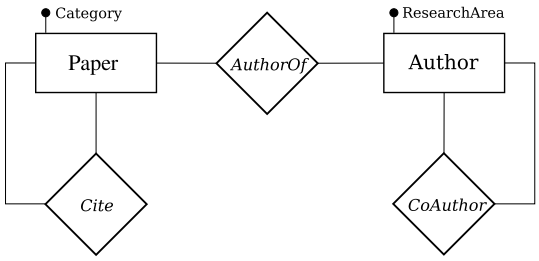

Fig. 3. Relational representation of the CORA Research Paper Dataset.

$C_i(\cdot) : \mathcal{P} \rightarrow \mathcal{B},\ \ C_i(x) = \texttt{true} \Leftrightarrow$ *x is of category* $i = 1, \ldots, 10$

$AC_i(\cdot) : \mathcal{A} \rightarrow \mathcal{B},\ \ AC_i(x) = \texttt{true} \Leftrightarrow$ *x is of category* $i = 1, \ldots, 10$

$CoAuthor(\cdot, \cdot) : \mathcal{A}^2 \rightarrow \mathcal{B},\ CoAuthor(x, y) = \texttt{true} \Leftrightarrow$ *x, y are co-authors*

$AuthorOf(\cdot, \cdot) : \mathcal{A} \times \mathcal{P},\ \ AuthorOf(x, y) = \texttt{true} \Leftrightarrow$ *x is author of y*

$Cite(\cdot, \cdot) : \mathcal{P}^2 \rightarrow \mathcal{B}^2,\ \ Cite(x, y) = \texttt{true} \Leftrightarrow$ *x cites y*

When considering all the categories $i = 1, \ldots, 10$, the knowledge base (KB) that we use to represent the learning task is composed of 52 rules:

$$0 \quad \forall x\, P_i(x) \Rightarrow C_k(x)\ ,\ \ \forall x\, N_i(x) \Rightarrow \neg C_k(x)$$

$$i \quad \forall x \forall y\, Cite(x, y) \Rightarrow (C_i(x) \wedge C_i(y)) \vee (\neg C_i(x) \wedge \neg C_i(y))$$

$$ii \quad \forall x \forall y\, CoAuthor(x, y) \Rightarrow (AC_i(x) \wedge AC_i(y)) \vee (\neg AC_i(x) \wedge \neg AC_i(y))$$

$$iii \quad \forall x \forall y\, AuthorOf(x, y) \Rightarrow (AC_i(x) \wedge C_i(y)) \vee (\neg AC_i(x) \wedge \neg C_i(y))$$

$$iv \quad \forall x \bigvee_{i=1}^{10} \left([\neg]_{i \neq 1} C_1(x) \wedge [\neg]_{i \neq 2} C_2(x) \wedge \cdots \wedge [\neg]_{i \neq 10} C_{10}(x) \right)$$

$$v \quad \forall x\, AC_1(x) \vee AC_2(x) \vee \ldots \vee AC_{10}(x)$$

The 0 formulas express the supervised data. The 10 *i* formulas state that papers tend to cite other papers of the same category, while the *ii* ones state that co-authors belong to the same research area. The *iii* formulas enforce the coherence of the categories given to papers and authors. *iv* is a formula in Disjunctive Normal Form stating the exclusive classification assumption ($[\neg]_{i \neq j}$ adds a negation in front of each predicate excluding the j-th), such that each paper is assigned to one and only one of the 10 classes. Finally, *v* states the close-world assumption forcing each author to be assigned to at least one of the 10 classes.

Results and discussion In a first experiment we validated the *dom* heuristic proposed in Section 6.2. In particular, SBR was tested using the following configurations:

- $(0, i, ii, iii, iv, v)$: introducing all the constraints together at the beginning of training;
- $(0),\ (i, ii, iii, iv, v)$: learning from the supervised data first and, after the training data has been learned, introducing all the other constraints. In particular, the second group of constraints is added when the gradient module becomes small, which was empirically observed to happen after around 30 iterations of gradient decent in this experiment;
- $(0),\ (i, ii, iii),\ (iv),\ (v)$: splitting the training into multiple stages, where the constraints are sequentially introduced according to the *dom* heuristic by looking at the degree of freedom in the assignments of each formula. A new set of constraints is added when the gradient module becomes small. The formulas in the sets (i, ii, iii) have the same degree of freedom (possible different assignments to the query predicates verifying the formula) and are introduced together.

Table 1 reports the classification scores for the 3 different learning schemas, obtained on the test set for patterns in the paper and research area domains. Since each pattern in the paper domain belongs to a single class (single-label classification task), standard classification accuracy has been used as metric for this task. The F1 metric has been used for the research area domain, since this is a multi-label multi-class classification task. For all configurations, SBR used the minimum t-norm and the linear kernel with meta-parameters selected to maximize the classification accuracy for the paper category on the validation set of the considered fold. It is clear from the results that the *dom* heuristic effectively breaks the learning complexity, allowing to find better solutions. As shown by the smaller gains from moving from the $(0),\ (i, ii, iii, iv, v)$ to the $(0),\ (i, ii, iii),\ (iv),\ (v)$ learning schema, a significant portion of the gains comes from introducing higher level semantic rules after the predicates have been approximated by fitting the supervised data.

Table 1
Results on the CORA dataset (average over the 5 available folds) using different Semantic Based Regularization models, trained using a variable number of supervised patterns and different heuristics controlling the sequence of incorporation of the FOL rules in the training process.

Papers (accuracy)					
	10%	20%	30%	40%	50%
SBR (0,i,ii,iii,iv,v)	0.479	0.538	0.584	0.616	0.623
SBR (0), (i,ii,iii, iv,v)	0.492	0.554	0.599	0.630	0.647
SBR (0), (i,ii,iii), (iv), (v)	0.511	0.565	0.612	0.638	0.656
Research areas (F1)					
	10%	20%	30%	40%	50%
SBR (0,i,ii,iii,iv,v)	0.464	0.511	0.551	0.577	0.585
SBR (0), (i,ii,iii,iv,v)	0.471	0.520	0.565	0.590	0.606
SBR (0), (i,ii,iii), (iv), (v)	0.482	0.526	0.561	0.595	0.610

A comparison against Markov Logic Networks both in their discrete (MLN) and hybrid versions (HMLN) has been carried out by using the implementation provided by the Alchemy software package[3] using discriminative weight training optimized via rescaled conjugate gradient, which provided the best results on this task. MLNs have been trained using the same knowledge base previously employed by SBR, where rules *iv* and *v* has been added as hard constraints. Furthermore, the following formulas have been added to take care of the bag-of-words representation of the page by linking the words to the document categories in the MLN rule definitions:

$$HasWord(+word, x) \Rightarrow C(x, +class) ,$$

where x is a variable spanning over all the papers and, following the Alchemy syntax, the "+" means that one rule is added for each $(word, class)$ pair. This follows the experimental set up for text categorization suggested in Domingos et al. [12]. In the case of Hybrid Markov Logic Networks, the TF-IDF score is used to associate a numeric feature to each ground clause for the previous rule. In particular, any TF-IDF score below 1 has been gaussian decayed using Alchemy soft equality penalty function.

In order to compare against Probabilistic Soft Logic (PSL), some modifications to the KB have been required, since PSL can process only rules with conjunctive bodies and single-literal head (e.g. any propositional formula obtained after the evaluation of the grounded predicates must be a Horn Clause). Therefore, each formula, creating a manifold structure over the authors or papers, has been split into two corresponding formulas for the PSL evaluation, such that the new formulas can be processed by PSL and the logical AND of them is equivalent to the original formulation. For example the first manifold rule in SBR has been replaced by the pair of formulas:

$$\forall x \forall y \; Cite(x, y) \wedge C_i(x) \Rightarrow C_i(y)$$
$$\forall x \forall y \; Cite(x, y) \wedge \neg C_i(x) \Rightarrow \neg C_i(y)$$

The same procedure has been performed for the formulas expressing the manifold with respect to the *CoAuthor* and *Author* predicates. The formulas implementing the logic OR operation over the classes cannot be implemented in PSL. Unlike SBR, PSL has no direct integration with an SVM processing the input pattern representations. However, like done in the Bröcheler et al. [5] for their Wikipedia Category Prediction experiment, PSL can employ the output of a previously trained feature-based classifier as a prior for its assignments. This was done by reusing the same SVMs previously trained and then adding the following rules to the PSL learning task definition:

$$\forall x \; SVMCategory(x, i) \Rightarrow C_i(x)$$
$$\forall x \; \neg SVMCategory(x, i) \Rightarrow \neg C_i(x)$$

where $SVMCategory(x, i)$ is an evidence predicate holding a true value iff the SVM assigns the tag i to the pattern x (e.g. the SVM trained for the tag i provides an output greater than 0 when processing as input the pattern x). PSL formula weights have been learned over the validation set using the LazyMaxLikelihoodMPE algorithm (Most Probable Explanation Max likelihood), which provided the best results on the task and is also the employed training algorithm in all the examples provided with the PSL software package.

SBR has also been compared against standard, Structured and Transductive SVMs for the paper classification task. Structured SVMs have been used to perform native multi-class classification as explained by Tsochantaridis et al. [47]. Like for SBR, a linear kernel with meta-parameters selected to maximize the accuracy and F1 scores on the validation set have been used for all SVM experiments. The libSVM software package[4] was used as implementation for plain SVMs, while the SVMlight software package[5] was used for Transductive and Structured SVM. Since authors are not associated with a

[3] http://alchemy.cs.washington.edu/.
[4] http://www.csie.ntu.edu.tw/~cjlin/libsvm/.
[5] http://svmlight.joachims.org/.

Table 2
Accuracy and F1 scores (average over the 5 available folds) on the CORA dataset for patterns in the paper and research area domains, respectively. Papers have been classified using plain SVMs (SVM), Structured SVMs (SSVM), Transductive SVMs (TSVM), Semantic Based Regularization (SBR), plain (MLN) and Hybrid (HMLN) Markov Logic Networks and PSL using SVM (PSL-SVM) or Transductive SVM (PSL-TSVM) as base classifier providing the classification priors. The classification has been performed for different numbers of supervised patterns. Research area classification has been carried out using only SRL methods (SBR, and the different MLN and PSL versions), which can process patterns not associated to a vectorial feature representation.

Papers (accuracy)					
	10%	20%	30%	40%	50%
SVM	0.211	0.274	0.317	0.348	0.376
SSVM	0.352	0.385	0.410	0.419	0.430
TSVM	0.357	0.396	0.419	0.436	0.453
MLN	0.272	0.336	0.382	0.407	0.435
HMLN	0.257	0.328	0.375	0.407	0.426
PSL-SVM	0.258	0.367	0.404	0.443	0.460
PSL-TSVM	0.396	0.425	0.442	0.460	0.472
SBR	**0.511**	**0.565**	**0.612**	**0.638**	**0.656**
Research areas (F1)					
	10%	20%	30%	40%	50%
MLN	0.278	0.335	0.372	0.386	0.417
HMLN	0.275	0.322	0.375	0.395	0.404
PSL-SVM	0.232	0.304	0.345	0.385	0.403
PSL-TSVM	0.365	0.394	0.419	0.434	0.447
SBR	**0.482**	**0.526**	**0.561**	**0.595**	**0.610**

feature representation, they can only be classified by a purely relational approach and no comparison against SVM-based classification schemas was possible.

PSL has been tested using either SVM or Transductive SVM as base classifier providing a prior for the classification. These two versions will been indicated as PSL-SVM and PSL-TSVM, respectively.

Table 2 reports the accuracy and F1 scores on the test set for patterns representing papers and authors, respectively. Since some methods do not allow to express exclusive classification via an explicit rule in the KB, the output class for these methods has been selected as the class associated to the highest value among the classification outputs for each pattern.

Metrics in bold represent statistically significant gains (95%) over all the other classifiers. Since this is a transductive learning task, it is no surprise that Transductive SVMs are the best performers among the non-relational classifiers. Both PSL-SVM and PSL-TSVM outperform the corresponding base non-relational classifiers, and PSL-TSVM outperforms PSL-SVM because of the better performing base classifier. The SBR model outperforms all the non-relational classifiers, since it can integrate a much richer prior knowledge. The large improvement of SBR over the tested SRL models is due to two important factors. First, SBR provides a larger flexibility in designing the KB: some rules had to be dropped or modified in the PSL definitions, not having a conjunctive body and single head. Secondly, SBR integrates the processing of the input feature representations and of the higher level logic knowledge. This means that SBR natively back-propagates the output of the inference process performed during learning using the KB to the underlying SVMs, significantly improving their performances when little supervised data is available. PSL uses a frozen SVM as prior for its classifications and has no ability to improve the underlying classifier using the unsupervised data. This is huge advantage for SBR in a transductive context with a large portion of unsupervised data.

9.2. Collective classification in WebKB

This experiment evaluates the proposed collective classification approach on the WebKB benchmark. The WebKB dataset is a relational dataset which consists of labeled web pages from the computer science departments of 4 universities. We used the version of the dataset from Craven and Slattery [7], used also in many other following papers [29,32], which features about 4100 webpages and 10 000 hyperlinks. Each webpage is associated with a vectorial representation of its content represented as bag-of-words, while each link is associated with its anchor text. Each webpage belongs to at least one of the 5 categories: *person, course, department, researchproject, other*. In addition, the anchor's links belong at least one of the 5 classes: *toPerson, toCourse, toDepartment, toResearchProject, toOther*, depending on the category of the pointed webpage. The goal of the benchmark is to predict the categories of the webpages and of the links from the given data.

Each university represents an independent world, therefore a 4-fold crossvalidation is the most natural evaluation procedure to evaluate the performance of classification. 4 folds are generated by keeping the data of one university out as test set, then selecting the first two other remaining universities as training set and the last one as validation data.

Knowledge base Let us assume that $\mathcal{W}$ and $\mathcal{A}$ denote the set of web pages and anchor text identifiers, respectively. $\mathcal{B}$ represents a Boolean value. We consider the following predicates, following the relational representation shown in Fig. 4

$$C_i(\cdot) : \mathcal{W} \rightarrow \mathcal{B}, \quad C_i(x) = \texttt{true} \Leftrightarrow x \text{ is of category } i = 1, \ldots, 5$$

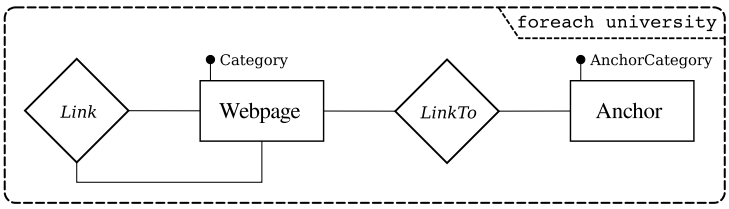

Fig. 4. Relational representation of the WebKB Dataset.

$LC_i(\cdot) : \mathcal{A} \rightarrow \mathcal{B},\ \ LC_i(x) = \texttt{true} \Leftrightarrow$ *x is of category* $i = 1, \ldots, 5$

$Link(\cdot,\cdot) : \mathcal{W}^2 \rightarrow \mathcal{B}\ \ Link(x, y) = \texttt{true} \Leftrightarrow$ *x is linked to y*

$LinkTo(\cdot,\cdot) : \mathcal{A} \times \mathcal{W} \rightarrow \mathcal{B}\ \ Link(x, y) = \texttt{true} \Leftrightarrow$ *x is linked to y*

The following KB was used for the task:

$$0 \quad \forall x\, P_i(x) \Rightarrow C_k(x)\ ,\ \ \forall x\, N_i(x) \Rightarrow \neg C_k(x)$$

$$i \quad \forall x \forall y\, Link(x, y) \Rightarrow (C_i(x) \wedge C_j(y)) \vee (\neg C_i(x) \wedge \neg C_j(y))$$

$$ii \quad \forall x \forall y\, LinkTo(x, y) \Rightarrow (LC_i(x) \wedge C_i(y)) \vee (\neg LC_i(x) \wedge \neg C_i(y))$$

$$iii \quad \forall x\, C_1(x) \vee C_2(x) \vee C_3(x) \vee C_4(x) \vee C_5(x)$$

$$iv \quad \forall x\, LC_1(x) \vee LC_2(x) \vee LC_3(x) \vee LC_4(x) \vee LC_5(x)$$

The 0 formulas express the fitting of the supervised data. The *i* formulas state that linked pages tend to be of the same class (manifold regularization), while the *ii* ones dictate classification consistency among the predicted class for pages and anchors. Formulas *iii* and *iv* impose the close-world assumption, forcing each web page and anchor to belong to at least one of the 5 classes in the corresponding domain. The overall knowledge base consists of 42 rules.

The following formulas representing how words are correlated to webpage and link categories have also been added to the MLN rule definitions:

$$HasWord(+word, page) \Rightarrow Category(page, +class)\ ,$$

$$HasWord(+word, anchor) \Rightarrow AnchorCategory(anchor, +anchorclass)\ ,$$

where the "+" is used to define one rule for each (*word*, *class*) and (*word*, *anchorclass*) pairs, respectively. The rules *iii*, *iv* have been added as hard constraints in the MLN experiments as they should be always verified.

For the PSL experimental comparison, the same procedure described in the previous experiment has been used to express the manifolds built by the *Link* and *LinkTo* predicates. The input pattern representations have been embedded into the PSL classification by adding a set of rules expressing the consistency between PSL and SVM class assignments, where one SVM has been previously trained for each class. PSL formula weights have been learned over the validation set using the LazyMaxLikelihoodMPE algorithm.

Results and discussion Since the training set for each single fold is completely supervised, one SBR classifier per fold has been trained using the 0 rules converted via the minimum t-norm (e.g. a plain kernel machine with a linear kernel). Classification performance on the validation set was used to select the optimal regularization parameter for each fold. The output of the learned functions has been used to initialize the collective classification step for the validation and test sets. In particular, collective classification was performed separately on the validation set of each fold using different λ_c parameters, then selecting the $\bar{\lambda_c}$ value providing the best results for each fold. Finally, collective classification was performed on the test set of each fold, using $\lambda_c = \bar{\lambda_c}$, as selected at the previous step. The results obtained by Semantic Based Regularization using the minimum t-norm and collective classification (SBR-CC) have been compared against a plain SVM and Markov Logic Network (MLN). We used the Alchemy software implementation of MLNs, using discriminative weight training optimized via rescaled conjugate gradient. The libSVM software package was used to implement plain SVMs using a linear kernel. The SVM *C* parameter (trade off between model complexity and fitting of the supervised data) was selected to maximize the classification metrics on the validation set.

Table 3 shows the F1 and AUC scores obtained on the test set for the webpage and anchor link categories as an average over the 4 folds. SVM classification metrics are very good for webpages, where the rich feature-based representations allow to well discriminate the patterns. SVM performances are much worse for anchors, where the anchor text represented in the feature vector is small, noisy and often not very representative. MLNs perform slightly worse than SVMs on the webpage classification task, because MLNs do not get full advantage of the information available in the feature vectors. On the other hand, MLNs can get advantage of the available KB to improve the classification of the anchors with respect to SVMs. PSL

Table 3
F1 and AUC scores on the test set for the webpage and anchor link categories of the WebKB Dataset using a plain SVM (SVM), PSL, Markov Logic Networks (MLN) and Semantic Based Regularization with collective classification (SBR-CC) as an average over the 4 WebKB folds.

	F1		AUC	
	Webpage	Anchors	Webpage	Anchors
SVM	0.768	0.319	0.861	0.634
MLN	0.730	0.564	0.670	0.763
PSL	0.777	0.510	0.763	0.753
SBR-CC	0.810	0.700	0.895	0.775

performs quite well on the webpage classification task thanks to the good performance of its base classifier. However, the less rich additional KB that can be integrated in PSL limits its performance on the anchor patterns. SBR-CC is the best performer on both domains (even if with a small margin over the best competitor for each separate domains): it can fully get advantage of the feature-based representations, plus it can perform inference over the anchor domain using the entire KB.

9.3. Arnetminer

The Arnetminer Movie benchmark[6] is a relational dataset containing information about movies and associated directors, writers and actors. The Arnetminer dataset assigns a set of tags to each movie. We selected all the movies with at least one tag containing one of the 12 most common keywords: *horror*, *drama*, *comedy*, *television*, *teen*, *musical*, *adventure*, *western*, *mystery*, *thriller* and *biographical*. Each of these tags was assumed to correspond to an underlying genre that we want to predict. The goal of this experiment is to predict the movie genres by looking at the movie title, represented as bag-of-words. Please note that this is a multi-label dataset, since movies can be associated to multiple genres. The movies have been split into three sets: one set composed by the movies released up to the year 1979, another by the movies from 1980 to 1997 and, finally, and all later movies have been added to the last set. The resulting datasets contain 7567, 4234 and 5653 movies, respectively.

The first, second and third sets form the pools from which the training, validation and test data are selected for the single experiments, respectively. This is to simulate a real world scenario, where training is performed at some point in time over some available previous data and then the trained model is used to perform predictions over newly received data. A variable percentage of supervised labels from the movies in the first and second sets has been kept for training to evaluate how the performance is affected by the amount of labeled data. The remaining unlabeled patterns are still provided in the training set as unsupervised data. 5 different folds have been generated by performing different samples for each percentage of labels that are kept available in the training and validation sets.

Together with the rules expressing the fitting of the supervised data, the following rules have been added as prior knowledge for each tag C_i of the dataset:

$$\begin{aligned}
\forall x \forall y\ SameDirector(x, y) &\Rightarrow \big(C_i(x) \wedge C_i(y)\big) \vee \big(\neg C_i(x) \wedge \neg C_i(y)\big) \\
\forall x \forall y\ SameProducer(x, y) &\Rightarrow \big(C_i(x) \wedge C_i(y)\big) \vee \big(\neg C_i(x) \wedge \neg C_i(y)\big) \\
\forall x \forall y\ SameWriter(x, y) &\Rightarrow \big(C_i(x) \wedge C_i(y)\big) \vee \big(\neg C_i(x) \wedge \neg C_i(y)\big)
\end{aligned}$$

to express the fact that movies sharing the same director, producer or writer tend to belong to the same genres. The following two rules for each tag C_i express the Transductive SVM assumption:

$$\forall x\ C_i(x) \vee \neg C_i(x)$$

$$\exists_n x\ C_i(x) \wedge \exists_m x\ \neg C_i(x) :\ n + m = N$$

where N is the overall number of available movies in the considered dataset and n, m are estimated looking at the distribution of $C_i, \neg C_i$ in the training data of each fold, respectively.

After training the SBR models using the minimum t-norm and the linear kernel for different λ_c meta-parameters, the best model has been selected on the validation set. Finally, Semantic Based Regularization collective classification (SBR-CC) has been performed over the test set using the selected model. The SBR-CC results have been compared against Semantic Based Regularization (SBR), where no collective classification is performed on the test set, but the trained kernel machines are directly used to perform the predictions. Furthermore, SBR-CC was compared against standard SVM (using only supervised labels), Transductive SVM (TSVM), Laplacian SVM (LSVM) using the same director, producer or writer rules to build the manifold of data and Probabilistic Soft Logic using SVM (PSL-SVM) and TSVM classifiers as priors (PSL-TSVM). One separate binary classifier for each class has been built for SVM, LSVM, TSVM. In particular, the same software simulator used for

[6] It can be downloaded at http://arnetminer.org/lab-datasets/soinf.

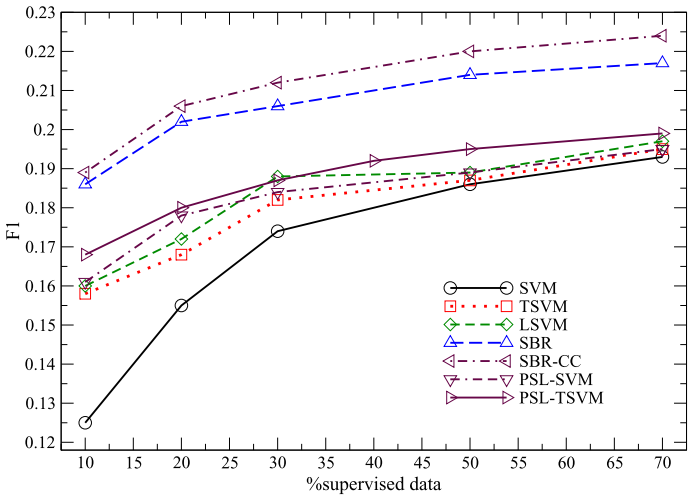

Fig. 5. F1 scores as average over 5 folds on the Arnetminer dataset obtained using SVM, TSVM, LSVM, PSL using SVM (PSL-SVM) or Transductive SVM (PSL-TSVM) as base classifier providing the classification priors, SBR and SBR with collective classification (SBR-CC) varying the number of supervised patterns.

the experiments in Melacci and Belkin [31] was used as implementation of Laplacian SVMs. The same software simulators used in the previous experiments have been employed as implementations of SVMs and TSVMs. All SVM-based classifiers employed a linear kernel and the models have been trained using different meta-parameters. The best model has been fold-by-fold selected by cross validation on the validation set.

Similarly to the previous experiments, each formula, creating a manifold structure over the movies, has been split into two corresponding formulas for the PSL evaluation. For example the first manifold rule in SBR has been replaced by the pair of formulas in the PSL rule definitions:

$$\forall x \forall y \; SameDirector(x, y) \wedge C_i(x) \quad \Rightarrow \quad C_i(y)$$
$$\forall x \forall y \; SameDirector(x, y) \wedge \neg C_i(x) \quad \Rightarrow \quad \neg C_i(y)$$

The same procedure has been performed for the formulas expressing the manifold with respect to the *SameProducer* and *SameWriter* predicates. The formulas implementing the Transductive SVM assumption cannot be expressed in PSL and have been not used for this experimental comparison. Like for the previous experiments, the rules expressing the consistency between PSL class assignments and the output of the SVMs trained for each class have been added to the PSL rule definitions. PSL formula weights have been learned over the validation set using the LazyMaxLikelihoodMPE algorithm, which provided the best results on this task. Finally, the PSL collective classification step using the rule weights learned in training has been performed over the test set. The output of the collective classification step has been used to determine PSL classification performances.

Fig. 5 reports the classification results as an average over the 5 folds. SVMs are, as expected, the worst performers on this task, as they are the only tested model that cannot benefit from the unsupervised data seen in training. PSL, TSVM and LSVM perform similarly, but they are outperformed by SBR. PSL-TSVM takes advantage of the better performing base classifier and outperforms PSL-SVM. Both PSL versions perform worse than SBR in this task, because PSL cannot get advantage of the unsupervised data to improve the underlying classifiers that it uses as classification priors. Finally, SBR with collective classification (SBR-CC) slightly improves over SBR in all tested configurations by enforcing the rules also over the test set.

10. Conclusions

In this paper we have proposed a semantic-based regularization approach for learning and inference that generalizes both different kernel machine models, processing real-valued features, and statistical relational learning approaches working on symbolic identifiers and domain knowledge expressed in term of FOL formulas. The resulting inference mechanism also involves a novel collective classification schema that exploits real-valued features. As shown in the experimental results, this is useful whenever the feature representation is relatively poor, so as the involvement of the constraints at the time of test significantly improves the performance. This paper significantly extends the theoretical results previously published by presenting intriguing connections with probabilistic models and proposing novel heuristics allowing to tackle more complex learning tasks.

Acknowledgements

This research was partially supported by the research grant 2009LNP494 from the PRIN2009 program of the Italian MURST.

References

[1] F. Baader, The Description Logic Handbook: Theory, Implementation, and Applications, Cambridge University Press, 2003.
[2] J.F. Bard, Practical Bilevel Optimization: Algorithms and Applications, Nonconvex Optimization and Its Applications, vol. 30, Springer, 1998.
[3] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) 2434.
[4] Y. Bengio, Curriculum learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML0, 2009, pp. 41–48.
[5] M. Broecheler, L. Mihalkova, L. Getoor, Probabilistic similarity logic, in: Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, UAI, 2010, pp. 73–82.
[6] A. Caponnetto, C.A. Micchelli, M. Pontil, Y. Ying, Universal multi-task kernels, J. Mach. Learn. Res. 9 (2008) 1615–1646.
[7] M. Craven, S. Slattery, Relational learning with statistical predicate invention: better models for hypertext, Mach. Learn. (2001) 97–119.
[8] C. Cumby, D. Roth, Learning with feature description logics, in: Proceedings of the 12th International Conference on Inductive Logic Programming, Springer, 2003, pp. 32–47.
[9] C. Cumby, D. Roth, On kernel methods for relational learning, in: Proceedings of the Twentieth International Conference on Machine Learning, ICML, 2003, pp. 107–114.
[10] M. Diligenti, M. Gori, M. Maggini, L. Rigutini, Bridging logic and kernel machines, Mach. Learn. 86 (2012) 57–88.
[11] P. Domingos, M. Richardson, Markov logic: a unifying framework for statistical relational learning, in: ICML-2004 Workshop on Statistical Relational Learning, 2004, pp. 49–54.
[12] P. Domingos, M. Sumner, The alchemy tutorial, http://alchemy.cs.washington.edu/tutorial/tutorial.pdf, 2010.
[13] N. Friedman, L. Getoor, D. Koller, A. Pfeffer, Learning probabilistic relational models, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, 1999, pp. 1300–1309.
[14] G.M. Fung, O.L. Mangasarian, J.W. Shavlik, Knowledge-based support vector machine classifiers, in: Advances in Neural Information Processing Systems, 2002, pp. 521–528.
[15] G.M. Fung, O.L. Mangasarian, J.W. Shavlik, Knowledge-based nonlinear kernel classifiers, in: Learning Theory and Kernel Machines, Springer, 2003, pp. 102–113.
[16] S.W. Golomb, L.D. Baumert, Backtrack programming, J. ACM 12 (1965) 516–524.
[17] M. Gupta, J. Qi, Theory of t-norms and fuzzy inference methods, Fuzzy Sets Syst. 40 (1991) 431–450.
[18] P. Hajek, The Metamathematics of Fuzzy Logic, Kluwer, 1998.
[19] R.M. Haralick, G.L. Elliott, Increasing tree search efficiency for constraint satisfaction problems, Artif. Intell. 14 (1980) 263–313.
[20] D. Haussler, Convolution kernels on discrete structures, Technical report, Department of Computer Science, University of California at Santa Cruz, 1999.
[21] P. Hitzler, S. Holldobler, A.K. Sedab, Logic programs and connectionist networks, J. Appl. Log. 2 (2004) 245–272.
[22] T.N. Huynh, R.J. Mooney, Discriminative structure and parameter learning for Markov logic networks, in: Proceedings of the 25th International Conference on Machine Learning, ICML, ACM, 2008, pp. 416–423.
[23] S. Kok, P. Domingos, Learning the structure of Markov logic networks, in: Proceedings of the 22nd International Conference on Machine Learning, ICML, ACM, 2005, pp. 441–448.
[24] N. Landwehr, A. Passerini, L. De Raedt, P. Frasconi, kfoil: learning simple relational kernels, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2006, pp. 389–394.
[25] N. Landwehr, A. Passerini, L. Raedt, P. Frasconi, Fast learning of relational kernels, Mach. Learn. (2010).
[26] F. Laurer, G. Bloch, Incorporating prior knowledge in support vector machines for classification: a review, Neurocomputing 71 (2009) 1578–1594.
[27] Q.V. Le, A.J. Smola, T. Gärtner, Simpler knowledge-based support vector machines, in: Proceedings of the 23rd International Conference on Machine Learning, ICML, ACM, 2006, pp. 521–528.
[28] M. Lippi, P. Frasconi, Prediction of protein β-residue contacts by Markov logic networks with grounding-specific weights, Bioinformatics 25 (2009) 2326–2333.
[29] D. Lowd, P. Domingos, Efficient weight learning for Markov logic networks, in: Proceedings of the Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases, 2007, pp. 200–211.
[30] A. McCallum, K. Nigam, J. Rennie, K. Seymore, Automating the construction of Internet portals with machine learning, Inf. Retr. 3 (2000) 127–163.
[31] S. Melacci, M. Belkin, Laplacian support vector machines trained in the primal, J. Mach. Learn. Res. 12 (2011) 1149–1184.
[32] L. Mihalkova, R.J. Mooney, Bottom-up learning of Markov logic network structure, in: Proceedings of the 24th International Conference on Machine Learning, ACM, New York, NY, USA, 2007, pp. 625–632.
[33] S. Muggleton, H. Lodhi, A. Amini, M.J. Sternberg, Support vector inductive logic programming, in: Discovery Science, Springer, 2005, pp. 163–175.
[34] J. Neville, D. Jensen, Relational dependency networks, J. Mach. Learn. Res. 8 (2007) 653–692.
[35] F. Niu, C. Ré, A. Doan, J. Shavlik, Tuffy: scaling up statistical inference in Markov logic networks using an RDBMS, in: Proceedings of the VLDB, 2011, pp. 373–384.
[36] V. Novák, First-order fuzzy logic, Stud. Log. 46 (1987) 87–109.
[37] J. Piaget, La psychologie de l'intelligence, Armand Colin, Paris, 1961.
[38] T. Poggio, F. Girosi, A theory of networks for approximation and learning, Technical report, MIT, 1989.
[39] L.D. Raedt, P. Frasconi, K.S.M. Kersting (Eds.), Probabilistic Inductive Logic Programming, Lecture Notes in Artificial Intelligence, vol. 4911, Springer, 2008.
[40] M. Richardson, P. Domingos, Markov logic networks, Mach. Learn. 62 (2006) 107–136.
[41] F. Rossi, P. Van Beek, T. Walsh, Handbook of Constraint Programming, Elsevier, 2006.
[42] B. Scholkopf, A.J. Smola, Learning with Kernels, MIT Press, Cambridge, MA, USA, 2001.
[43] J.W. Shavlik, S. Natarajan, Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, 2009, pp. 1951–1956.
[44] P. Singla, P. Domingos, Memory-efficient inference in relational domains, in: Proceedings of the 21st AAAI Conference on Artificial Intelligence, AAAI Press, 2006, pp. 488–493.
[45] J.T. Starczewski, Advanced Concepts in Fuzzy Logic and Systems with Membership Uncertainty, Springer, 2012.
[46] S.D. Tran, L.S. Davis, Event modeling and recognition using Markov logic networks, in: Proceedings of the European Conference of Computer Vision, ECCV, Springer, 2008, pp. 610–623.

[47] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, Large margin methods for structured and interdependent output variables, J. Mach. Learn. Res. (2005) 1453–1484.
[48] V. Vapnik, The Nature of Statistical Learning Theory, 2nd edn., Springer Verlag, 2000.
[49] J. Wang, P. Domingos, Hybrid Markov logic networks, in: Proceedings of the 23-rd AAAI Conference on Artificial Intelligence, 2008, pp. 1106–1111.
[50] P.M. Williams, Bayesian regularization and pruning using a Laplace prior, Neural Comput. 7 (1995) 117–143.
[51] L.A. Zadeh, Fuzzy sets, Inf. Control 8 (1965) 338–353.
[52] D. Zhou, B. Schölkopf, Regularization on discrete spaces, Pattern Recognit. (2005) 361–368.