Integrating Logic Knowledge into Graph Regularization: an application to image tagging

Claudio Saccà
Information Engineering Department
Via Roma 54
Siena, Italy
claudiosac@gmail.com

Michelangelo Diligenti
Information Engineering Department
Via Roma 54
Siena, Italy
diligmic@dii.unisi.it

Marco Gori
Information Engineering Department
Via Roma 54
Siena, Italy
marco@dii.unisi.it

Marco Maggini
Information Engineering Department
Via Roma 54
Siena, Italy
maggini@dii.unisi.it

ABSTRACT

This paper studies how prior knowledge in form of First Order Logic (FOL) clauses can be converted into a set of continuous constraints. These constraints can be directly integrated into a learning framework allowing to jointly learn from examples and semantic knowledge. In particular, in this paper we show how the constraints can be integrated into a regularization schema working over discrete domains. We consider tasks in which items are connected to each other by given relationships, thus yielding a graph, whose nodes correspond to the available objects. It is required to estimate a set of functions defined on each node of the graph, given a small set of labeled nodes for each function. The FOL constraints enforce dependencies, resulting from the FOL knowledge, among the values that the functions assume over the nodes. The experimental results evaluate the proposed technique on an image tagging task, showing how the proposed approach provides a significantly higher tagging accuracy than simple graph regularization. The experimental results show how the selection of a proper conversion process of the FOL clauses is fundamental in order to achieve good results.

1. INTRODUCTION

Tagging of resources like images, textual documents and videos is an important functionality in social networks and it is emerging as an effective technique to enrich shared resources with semantic meta-information. A high precision tagging allows the deployment of more sophisticated information retrieval mechanisms that the ones currently provided by search engines. In particular, tags associated to images usually summarize their semantic content that would be difficult to interpret automatically with state-of-the-art image understanding techniques. However, a manual collective tagging process has many limitations, as it is not suited for very large or dynamic collections and it does not provide high consistency of the tags across the images, creating many issues for their subsequent use [8]. In fact, in the context of social networks and folksonomies the tag dictionary is freely built by the users without an actual agreement on a common semantic interpretation or knowledge model.

Automatic image tagging can be formulated as a classification problem, where the number of tags is typically high and the classes are not mutually exclusive, thus yielding a multi-label classification task. In this paper we take the approach of using the manually inserted tags as training data for a tagger, that generalizes the supervisions to new images. However, when an automatic image tagger is trained over the tags inserted by the users of a folksonomy, it may inherit the same inconsistencies of the training data.

The approach presented in this paper exploits a multi relational representation of the data as proposed in [10], where each image corresponds to a node in a graph and different semantic relations like content similarity, friendship in the social network and authorship can be used to establish connections between the corresponding nodes. A few nodes are labeled with explicit tags inserted by one or more users. Like other transductive approaches [1, 14, 13], the proposed solution generalizes the tags to the unlabeled data via a diffusion process. However, the proposed solution is also able to take into account the available prior knowledge defined by FOL clauses, that enforce the consistency of the assigned tags without depending on specially trained human taggers. In particular, the FOL clauses are built on variables ranging over the nodes of the graph (the images) and task predicates, each one associated to a tag [3]. The FOL clauses are compiled into a set of equivalent continuous constraints, and the integration between logic and learning is implemented via a novel multi-task learning scheme, which combines the loss on the supervised examples and a penalty term resulting from the conversion of the logic knowledge. This paper proposes

MLG '11 San Diego, CA, USA

and evaluates different approaches for the conversion of the logic knowledge, using t-norms or different functions based on mixture of Gaussians.

Other specific tagging methods have been devised for classifying networked items. In particular, some approaches considering the images in their isolation perform image annotation and tagging using generative or latent models [7, 9]. Other approaches perform tag propagation on the network using random walks or semi-supervised learning [2]. Other more recent methods take advantage of the semantic relations among tags [11, 12] exploit tag co-occurrence in order to improve their predicted ranking.

The experimental results show that the employed conversion schema has a very strong impact on the precision of the tagging and that an appropriate conversion can determine a significant improvement of the tagging accuracy with respect to a standard classifier not incorporating the prior knowledge.

The paper is organized as follows. The next section presents the proposed approach for multi-task learning on a graph of entities when a set of known constraints is to be enforced. Then, section 3 describes how the available FOL knowledge base can be converted into a set of continuous constraints involving the tagging functions. Finally, section 4 reports the evaluation on a dataset collected from Flicker[1], where the tag dictionary is large and not centrally controlled, resulting in noisy and inconsistent training data.

2. MULTI-TASK TRANSDUCTIVE LEARNING WITH CONSTRAINTS

We consider a given set of entities and relationships among them. Hence, the input can be modeled by a graph where each node represents an entity and an edge encodes some semantic relation between the connected nodes. We assume that the pertinence of each node to each tag has to be determined given a partial labeling of all the nodes in the graph. Therefore, it is possible to approach this task using regularization over discrete domains, performing a separate diffusion over the graph for each tag.

2.1 Regularization in Discrete Domains

We consider a general formulation of transductive learning in discrete domains based on a regularization principle [14, 13]. This class of algorithms exploits a set of examples (labeled nodes) and the connections (edges) among the objects. Entity classification is performed by computing a function that is defined over the graph nodes (each node representing an object to be classified) and that is *smooth* when considering nearby nodes. Generalization of the labels assigned to a small set of supervised nodes to the other nodes is obtained by exploiting the graph topology. Typically, a score equal to $+1$ (-1) is used to indicate that a target property is to be attached (not attached) to a given node. The values in the range $[-1, 1]$ can be used to indicate a degree of confidence, assuming the value 0 as the threshold for the decision. This approach allows us to define a binary classification scheme for each considered property (i.e. we decide to attach a given label to an item without considering the results for the other properties). When facing a multi-label classification problem with mutually exclusive classes, the procedure is applied for each class and the final classification is performed by selecting the class yielding the maximum value (eventually considering rejection if the maximum is below a given confidence threshold).

More formally, let us suppose we are given a graph G and let V_G be the set of nodes in G. Each edge of the graph is assigned a weight representing the strength of the connection between the corresponding nodes. In particular, given two nodes $u, z \in V_G$, $w_{uz} \geq 0$ indicates the weight of the connection between u and z. $w_{uz} = 0$ is equivalent to not having a connection between the nodes. In the considered framework, the connection weights define the strength of the nodes' similarity (i.e. a higher weight value should imply a higher similarity between the two connected nodes).

The graph G can be represented via its adjacency matrix $\boldsymbol{W}$, whose $(u, z) - th$ element is equal to w_{uz}. Let $\boldsymbol{y}$ be a vector of size $|V_G|$ having in the u-th position the available classification score of node u, if it is labeled, and 0 otherwise. Discrete domain regularization estimates the function $f(u), u \in V_G$ by computing a score value for each node in V_G, in order to yield a good fitting of the target vector on the labeled nodes with "smooth" variations over the graph connections. The values $f(u), u \in V_G$ are collected in the vector $\boldsymbol{f}$. In particular, the learning problem determines the vector $\boldsymbol{f}^\star$ minimizing the following cost functional

$$\boldsymbol{C}_G[\boldsymbol{f}] = \frac{1}{2}\left\|\boldsymbol{f} - \boldsymbol{y}\right\|^2 + \lambda_l \boldsymbol{f}^T \boldsymbol{R}_G \boldsymbol{f} \tag{1}$$

where $\boldsymbol{R}_G$ is a regularization matrix defined to penalize non-smooth solutions, $\boldsymbol{y}$ is the vector of target scores, whose non-supervised entries are set to 0, and $0 \leq \lambda_l \leq 1$ is a constant determining the trade off between regularization and the fitting error over the nodes.

The optimal solution minimizing equation (1) can be computed by finding its stationary points, obtained by solving

$$\nabla_{\boldsymbol{f}} \boldsymbol{C}_G[\boldsymbol{f}] = (\boldsymbol{f} - \boldsymbol{y}) + \lambda_l \boldsymbol{R}_G \boldsymbol{f} = 0 \ .$$

If $(\boldsymbol{I} + \lambda_l \boldsymbol{R}_G)$ is invertible, $\boldsymbol{f}^\star$ exists, it is unique and equal to

$$\boldsymbol{f}^\star = (\boldsymbol{I} + \lambda_l \boldsymbol{R}_G)^{-1} \boldsymbol{y} \ . \tag{2}$$

Equation (2) requires the inversion of a square matrix, which has size equal to the number of nodes in the input graph. When the graph is large, direct inversion is not feasible. However, if the largest eigenvalue of $\lambda_l \boldsymbol{R}_G$ lays inside the unit circle and $\boldsymbol{R}_G$ is sparse, the solution can be efficiently found by solving the following iterative equation: $\boldsymbol{f}^{t+1} = \boldsymbol{y} - \lambda_l \boldsymbol{R}_G \boldsymbol{f}^t$. Interestingly, this iterative equation represents a diffusion process of the labels through the graph.

In order to provide a meaningful definition of $\boldsymbol{R}_G$, we start from a regularization functional $\boldsymbol{C}_G^R[\boldsymbol{f}]$ that penalizes the distance between the values computed for pairs of connected nodes, weighted by the strength of the connection,

$$\boldsymbol{C}_G^R[\boldsymbol{f}] = \frac{1}{2} \sum_{u=1}^{|V_G|} \sum_{z=1}^{|V_G|} w_{uz} (f(u) - f(z))^2 \ . \tag{3}$$

This functional favors functions assuming close values on nodes that are strongly connected. Equation (3) can be rearranged as

$$\boldsymbol{C}_G^R[\boldsymbol{f}] = \frac{1}{2} \sum_{u=1}^{|V_G|} \sum_{z=1}^{|V_G|} \frac{(w_{uz} + w_{zu})}{2} (f(u) - f(z))^2 \ . \tag{4}$$

[1]http://www.flicker.com

If we define $\bar{w}_{uz} = \frac{(w_{uz}+w_{zu})}{2}$, equation (4) becomes

$$\begin{aligned}
\boldsymbol{C}_G^R[\boldsymbol{f}] &= \frac{1}{2}\sum_{u=1}^{|V_G|}\sum_{z=1}^{|V_G|}\bar{w}_{uz}(f(u)-f(z))^2 = \\
&= \frac{1}{2}\sum_{u=1}^{|V_G|}\sum_{z=1}^{|V_G|}\bar{w}_{uz}f^2(u) + \frac{1}{2}\sum_{u=1}^{|V_G|}\sum_{z=1}^{|V_G|}\bar{w}_{uz}f^2(z) + \\
&\quad -\sum_{u=1}^{|V_G|}\sum_{z=1}^{|V_G|}\bar{w}_{uz}f(u)f(z)\ .
\end{aligned}$$

The weights $\bar{w}_{uz}$ are symmetric ($\bar{w}_{uz} = \bar{w}_{zu}\ \forall u, z$), therefore it holds that

$$\sum_{u=1}^{|V_G|}\sum_{z=1}^{|V_G|}\bar{w}_{uz}f(u)^2 = \sum_{u=1}^{|V_G|}\sum_{z=1}^{|V_G|}\bar{w}_{uz}f(z)^2\ .$$

Thus, we obtain

$$\boldsymbol{C}_G^R[\boldsymbol{f}] = \sum_{u=1}^{|V_G|}\sum_{z=1}^{|V_G|}\bar{w}_{uz}f^2(u) - \sum_{u=1}^{|V_G|}\sum_{z=1}^{|V_G|}\bar{w}_{uz}f(u)f(z)\,.$$

Let $\bar{\boldsymbol{W}}$ be a symmetric square matrix having $\bar{w}_{uz}$ as the (u, z)-th element and $\boldsymbol{D}$ be a diagonal matrix with its u-th element d_u equal to $\sum_{z=1}^{|V_G|}\bar{w}_{uz}$, then the two terms on the right side of equation (5) can be expressed as $\boldsymbol{f}^T\boldsymbol{D}\boldsymbol{f}$ and $\boldsymbol{f}^T\bar{\boldsymbol{W}}\boldsymbol{f}$, respectively.

Therefore, $\boldsymbol{C}_G^R[\boldsymbol{f}]$ can be compactly rewritten as

$$\boldsymbol{C}_G^R[\boldsymbol{f}] = \boldsymbol{f}^T(\boldsymbol{D}-\bar{\boldsymbol{W}})\boldsymbol{f}$$

which expresses the regularizer in the form required by equation (1). Now, setting $\boldsymbol{R}_G = \boldsymbol{D}-\bar{\boldsymbol{W}}$ into equation 2 allows us to compute the optimal score vector as

$$\boldsymbol{f}^\star = \left(\boldsymbol{I} + \lambda_l\boldsymbol{D} - \lambda_l\bar{\boldsymbol{W}}\right)^{-1}\boldsymbol{y}\ . \qquad (5)$$

Since $\boldsymbol{D}-\bar{\boldsymbol{W}}$ is diagonally dominant, $\boldsymbol{I} + \lambda_l\boldsymbol{D} - \lambda_l\bar{\boldsymbol{W}}$ is also diagonally dominant and, therefore, invertible. Thus, the optimal solution $\boldsymbol{f}^\star$ exists and it is uniquely defined by the graph and the supervised vector of the targets.

2.2 Learning with constraints

We assume to have available a dictionary T of tags (categories), whose size we indicate with $|T|$. We consider a multitask learning problem in which we must decide which tags are to be assigned to each node. Therefore, a set of $|T|$ functions $\boldsymbol{F} = \{\boldsymbol{f}_k, k = 1, \ldots, |T|\}$ must be estimated, where $\boldsymbol{f}_k$ collects the predictions for the assignment of the k-th tag. If the tasks are uncorrelated, the optimization of the objective function with respect to the $|T|$ functions is faced as a set of stand-alone graph regularization problems, as defined by equation (1). However, we consider the case when the task functions $\boldsymbol{f}_k$ are not independent but have to meet a set of H constraints that can be expressed by a set of functionals $\Phi = \{\phi_h(\boldsymbol{F}) = 0,\ h = 1, \ldots, H\}$ which must hold for any valid choice of $\boldsymbol{f}_k : k = 1, \ldots, |T|$. In particular, in the next section we will show how appropriate functionals can be defined to force the function values to meet a set of first-order logic constraints. Without loss of generality we can assume that $\phi_h(\boldsymbol{F}) > 0$ when the constraint is violated.

The learning problem can be formulated as a soft-constrained optimization problem, where the cost function to optimize takes the following form

$$\begin{aligned}
\boldsymbol{C}_G[\boldsymbol{F}, \Phi] &= \sum_{k=1}^{|T|}\boldsymbol{C}_G[\boldsymbol{f}_k] + \qquad (6)\\
&\quad +\lambda_c\sum_{h=1}^{H}\phi_h(\boldsymbol{f}_1, \ldots, \boldsymbol{f}_{|T|}) = \\
&= \sum_{k=1}^{|T|}\Big(\frac{\left\|\boldsymbol{f}_k - \boldsymbol{y}_k\right\|^2}{2} + \lambda_l\boldsymbol{f}_k{}^T\boldsymbol{R}_G\boldsymbol{f}_k\Big) + \\
&\quad +\lambda_c\sum_{h=1}^{H}\phi_h(\boldsymbol{F})\ ,
\end{aligned}$$

where $\boldsymbol{y}_k$ is the target vector for the k-th function.

The objective function of equation (6) can be optimized using gradient descent by computing the derivative with respect to the value assigned by each function to each node. If we indicate the value of the k-th function in the i-th node by $f_k(i)$, the derivative of the cost function with respect to $f_k(i)$ is

$$\begin{aligned}
\frac{\partial \boldsymbol{C}_G[\boldsymbol{F}, \Phi]}{\partial f_k(i)} &= f_k(i) - y_k(i) + 2\lambda_l\big(\sum_{j=1}^{|V_G|}\bar{w}_{ij}\big)f_k(i) - \\
&- \lambda_l\sum_{j=1}^{|V_G|}\bar{w}_{ij}f_k(j) + \lambda_c\sum_{h=1}^{H}\frac{\partial\phi_h(\boldsymbol{F})}{\partial f_k(i)}
\end{aligned}$$

where $y_k(i)$ indicates the i-th value of the k-th target vector.

It is now possible to perform gradient descent optimization by updating $f_k(i)$ as $f_k(i)^{t+1} = f_k(i)^t - \mu\frac{\partial \boldsymbol{C}_G[\boldsymbol{F},\Phi]}{\partial f_k(i)}$, where $\mu > 0$ is the learning rate parameter.

3. TRANSLATION OF FOL CLAUSES INTO REAL-VALUED CONSTRAINTS

The learning from constraints approach described in the previous section requires to devise a conversion process to translate a set of logic formula into real-valued functions. We focus our attention on knowledge-based descriptions given by first-order logic (FOL–KB). While the framework can be easily extended to arbitrary FOL predicates, in this paper we will consider only unary predicates to keep the notation simple. In the following, we indicate by $\mathcal{V} = \{v_1, \ldots, v_N\}$ the set of the variables used in the KB, with $v_s \in V_G$ denoting a node in the graph. A predicate in the KB takes a node (or more nodes for n-ary predicates) in the graph and associates it with a true/false value depending whether a given property applies to the node. Therefore, given the set of predicates used in the KB $\mathcal{P} = \{p_k | p_k : (v \in V_G) \rightarrow \{true, false\},\ k = 1, \ldots, |T|\}$, the clauses will be built from the set of atoms $p(v) : p \in \mathcal{P}, v \in \mathcal{V}$.

Any FOL clause has an equivalent version in *Prenex Normal form* (PNF), that has all the quantifiers ($\forall, \exists$) and their associated quantified variables at the beginning of the clause. Standard methods exist to convert a generic FOL clause into its corresponding PNF and the conversion can be easily automated. Therefore, without loss of generality, we restrict our attention to FOL clauses in the PNF form. We assume that the task functions $\boldsymbol{f}_k, k = 1, \ldots, |T|$ are exploited to implement the predicates in $\mathcal{P}$. In this framework, the predicates yield a continuous real value that can be interpreted as a *truth degree*. The FOL–KB will contain a set of clauses corresponding to expressions with no free variables (i.e. all the variables appearing in the expression are quantified) that are assumed to be *true* in the considered domain. These clauses can be converted into a set of constraints that can be enforced during the learning process. As detailed in the following sections, the conversion process of a clause into a constraint functional consists of the following steps: *Conversion of the Propositional Expression* and *Quantifier conversion*.

3.1 Conversion of the Propositional Expression

Different methods can be used for the conversion of the propositional expression into a continuous function. In particular, two families of conversion approaches have been proposed in the literature: t-norms and mixtures of Gaussians.

T-norms.

In the context of fuzzy logic, t-norms [6] are commonly used as a generalization of logic clauses to continuous variables.

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

T-norms have been employed in previous work in the literature to integrate prior knowledge into kernel machines [4].

Mixture of Gaussians.

A different approach based on mixtures of Gaussians has been proposed in [5] in the context of symbolic learning using neural networks. Unlike t-norms, this approach generalizes the logic clause without making any independence assumption among the variables. In particular, let us consider a propositional logic clause involving n logic variables. The logic clause is equivalent to its truth table containing 2^n rows, each one corresponding to a configuration of the variables. The continuous function approximating the clause is based on a set of Gaussian functions, each one centered on a configuration corresponding to the true value in the truth

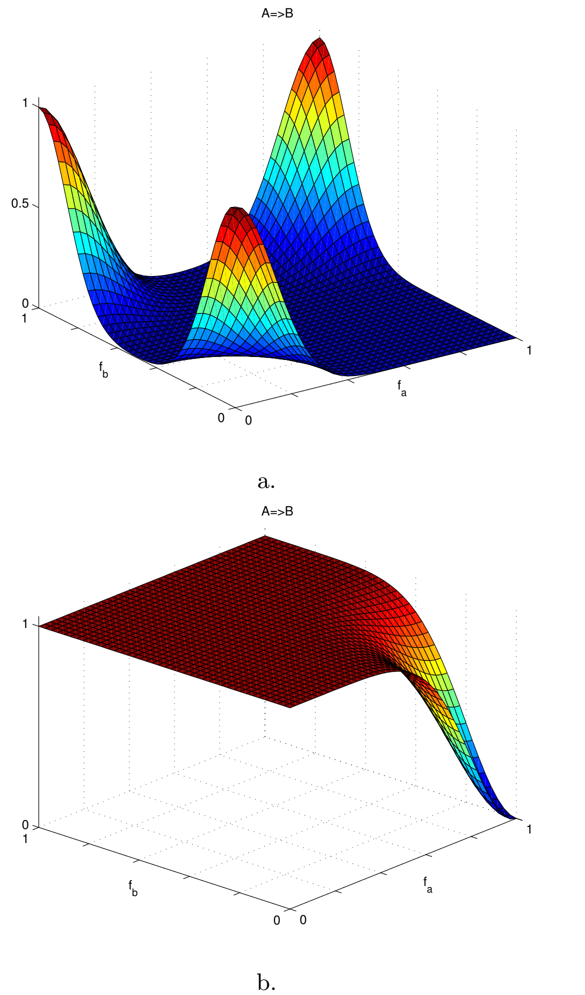

Figure 1: Function resulting from the conversion of $a \Rightarrow b$ using PGAUSS (a.) and NGAUSS (b.).

table. The mixture function sums all the Gaussians:

$$t(x_1, \ldots, x_n) = \\ = \sum_{[c_1,\ldots,c_n] \in \mathcal{T}} \exp\left(-\frac{||[x_1, \ldots, x_n]' - [c_1, \ldots, c_n]'||^2}{2\sigma^2}\right) . \quad (7)$$

where $x_1, \ldots, x_n$ is the set of variables in the propositional clause and $\mathcal{T}$ is the set of all possible configurations of the input variables which correspond to the true value in the table. We indicate as $PGAUSS$ this conversion procedure.

For example, let us consider the clause $x \vee y$, which is verified by the three configurations $[true, true]$, $[true, false]$, and $[false, true]$. The clause is converted as

$$\begin{aligned} t(x, y) \quad = & \quad \exp\left(-\frac{||[x,y]' - [1,1]'||^2}{2\sigma^2}\right) + \\ + & \quad \exp\left(-\frac{||[x,y]' - [1,0]'||^2}{2\sigma^2}\right) + \\ + & \quad \exp\left(-\frac{||[x,y]' - [0,1]'||^2}{2\sigma^2}\right) . \end{aligned}$$

If $[x, y]'$ assumes values corresponding to a configuration verifying the clause, $t(x, y) \ge 1$ holds. Otherwise, the value of $t(x, y)$ will decrease depending on the distance from the closest configuration verifying the clause. The variance σ^2 is a parameter that can be used to determine how quickly $t(x, y)$ decreases when moving away from a configuration verifying the constraint. Please note that each configuration verifying the constraint is always a global maximum of t when using a small enough σ value. See [5] for a complete discussion on how to select a suitable σ value.

A variant of this approach (MGAUSS) is to consider only the Gaussian centered in the closest true configuration

$$
\begin{aligned}
&t(x_1, \ldots, x_n) = \\
&= \max_{[c_1,\ldots,c_n]\in\mathcal{T}} \exp\left(-\frac{||[x_1,\ldots,x_n]'-[c_1,\ldots,c_n]'||^2}{2\sigma^2}\right) .
\end{aligned} \tag{8}
$$

In this case the the clause $x \vee y$ is converted as

$$
\begin{aligned}
t(x, y) \quad = \quad \max\Big[\quad & \exp\left(\frac{-||[x,y]'-[1,1]'||^2}{2\sigma^2}\right) , \\
& \exp\left(\frac{-||[x,y]'-[1,0]'||^2}{2\sigma^2}\right) , \\
& \exp\left(\frac{-||[x,y]'-[0,1]'||^2}{2\sigma^2}\right) \Big] .
\end{aligned}
$$

An alternative approach, that we indicate as NGAUSS, is to represent the false values of the truth table of the clause. In this case, one negative Gaussian is centered on each configuration of variables yielding a false value of the considered clause. A bias value equal to 1 is introduced to obtain a default true value when distant from a false configuration

$$
\begin{aligned}
&t(x_1, \ldots, x_n) = \\
&= 1 - \sum_{[c_1,\ldots,c_n]\in\mathcal{F}} \exp\left(-\frac{||[x_1,\ldots,x_n]'-[c_1,\ldots,c_n]'||^2}{2\sigma^2}\right) ,
\end{aligned} \tag{9}
$$

where $\mathcal{F}$ is the set of input configurations corresponding to a *false* value in the truth table.

Using this conversion procedure, the clause $x \vee y$ would be converted as

$$
t(x, y) = 1 - \exp\left(-\frac{||[x,y]'||^2}{2\sigma^2}\right) .
$$

The conversion schema modelling the true or false configurations resemble the duality in the representation of any propositional formula in Disjunctive or Conjunctive Normal Form. Figure 1 shows the functions obtained by converting the clause $a \Rightarrow b$ using PGAUSS and NGAUSS. Any formula can be converted using both forms. However, depending on the formula, one configuration can be more compact. Compact mixtures should be generally preferred as, when integrated into a cost function to be optimized as usually done in machine learning applications, they introduce less local minima. The experimental results, reported in the last section of this paper, confirm this claim.

3.2 Quantifier conversion

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

When processing the universal quantifier, the expression must hold for any node of the graph, and it can be naturally converted measuring the degree of non-satisfaction of the expression over the domain V_G. We indicate with $\boldsymbol{v}_E$ the vector of variables contained in the expression E. The satisfaction measure corresponds to the overall distance of the penalty associated to E, i.e. $\varphi_E(\boldsymbol{v}_E, \boldsymbol{F})$ from the constant function equal to 0 over the domain V_G. Using the infinity norm on discrete domains, this measure is

$$
\forall v_q \in V_G \quad E(\boldsymbol{v}_E, \mathcal{P}) \rightarrow \max_{v_q\in V_G} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{F})| , \tag{10}
$$

where the resulting expression depends on all the variables in $\boldsymbol{v}_E$ except v_q, and $\varphi_E(\boldsymbol{v}_E, \boldsymbol{F}) = max(1 - t_E(A(\boldsymbol{v}_E, \boldsymbol{F})), 0)$, where A maps the values assumed by the predicate functions $\boldsymbol{F}$ to the grounded variables $\boldsymbol{v}_E$ in order to obtain the correct argument list to the function $t_E()$, that implements the propositional expression E. Hence, the result of the conversion applied to the expression $E_q(\boldsymbol{v}_{E_q}, \mathcal{P}) = (\forall v_q \in V_G \quad E(\boldsymbol{v}_E, \mathcal{P}))$ is a functional $\varphi_{E_q}(\boldsymbol{v}_{E_q}, \boldsymbol{F})$, assuming values in $[0, 1]$ and depending on the set of variables $\boldsymbol{v}_{E_q} = [v_{s(1,E_q)}, \ldots, v_{s(n_{E_q},E_q)}]$, such that $n_{E_q} = n_E - 1$ and $v_{s(j,E_q)} \in \{v_r \in \mathcal{V} | \exists i \;\; v_r = v_{s(i,E)}, \; v_r \neq v_q\}$. The variables in $\boldsymbol{v}_{E_q}$ need to be quantified or assigned a specific value in order to obtain a constraint functional depending only on the functions $\boldsymbol{F}$.

The existential quantifier can be realized by enforcing the De Morgan law

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

to hold also in the continuous mapped domain. Using the conversion of the universal quantifier of equation (10), we obtain the following conversion for the existential quantifier

$$
\exists v_q \in V_G \quad E(\boldsymbol{v}_E, \mathcal{P}) \quad \rightarrow \quad \min_{v_q\in V_G} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{F})| .
$$

It is also possible to select a different norm on the discrete domain to convert the universal quantifier. For example, when using the $\|\cdot\|_1$ norm for discrete domains, yields the conversion rule

$$
\forall v_q \in V_G \quad E(\boldsymbol{v}_E, \mathcal{P}) \rightarrow \frac{1}{|V_G|} \sum_{v_q\in V_G} |\varphi_E(\boldsymbol{v}_E, \boldsymbol{F})| .
$$

As an example of the conversion procedure, let $a(\cdot), b(\cdot)$ be two predicates, implemented by the function vectors $\boldsymbol{f}_a, \boldsymbol{f}_b$. The clause $\forall v_1 \in V_G \;\; \forall v_2 \in V_G \quad (a(v_1) \wedge \neg b(v_2)) \vee (\neg a(v_1) \wedge b(v_2))$ is converted starting with the conversion of the quantifier free expression

$$
E_0([v_1, v_2], \{a(\cdot), b(\cdot)\}) = (a(v_1) \wedge \neg b(v_2)) \vee (\neg a(v_1) \wedge b(v_2)) ,
$$

verified if $a(v_1) = true$, $b(v_2) = false$ or $a(v_1) = false$, $b(v_2) = true$, as:

$$
\begin{aligned}
t_{E_0}(f_a(v_1), f_b(v_2)) \quad &= \quad \exp\left(-\frac{f_a(v_1)^2 + (f_b(v_2) - 1)^2}{2\sigma^2}\right) + \\
&+ \quad \exp\left(-\frac{(f_a(v_1) - 1)^2 + f_b(v_2)^2}{2\sigma^2}\right) .
\end{aligned}
$$

Then, $E_0([v_1, v_2], \{a(\cdot), b(\cdot)\})$ is converted into the distance measure and the two universal quantifiers are converted using the infinity norm over the set of nodes, yielding the constraint

$$
\begin{aligned}
\phi(\boldsymbol{F}) \quad &= \quad \varphi_{E_0}([], [\boldsymbol{f}_a, \boldsymbol{f}_b]) = \\
&= \quad \max_{v_1\in V_G} \max_{v_2\in V_G} [\max(1 - t_{E_0}(f_a(v_1), f_b(v_2)), 0)] = \\
&= \quad 0.
\end{aligned}
$$

The partial derivatives of the resulting continuous form of the constraints can be easily computed, even if they are not defined on a set of measure zero due to the presence of the maximum operator. However, these discontinuities in the derivatives do not represent an issue, given that an appropriate gradient descent-based optimization algorithm is exploited.

$\forall x$ sea(x) $\Rightarrow$ water(x)
$\forall x$ lake(x) $\Rightarrow$ water(x)
$\forall x$ flower(x) $\Rightarrow$ nature(x)
$\forall x$ mountains(x) $\wedge$ winter(x) $\Rightarrow$ snow(x)
$\forall x$ bird(x) $\wedge$ tree(x) $\Rightarrow$ wildlife(x)
$\forall x$ beach(x) $\Rightarrow$ sea(x) $\vee$ ocean(x)
$\forall x$ sky(x) $\wedge$ newyorkcity(x) $\Rightarrow$ skyline(x)
$\forall x$ eruption(x) $\wedge$ lava(x) $\Rightarrow$ volcano(x)
$\forall x$ sand(x) $\Rightarrow$ beach(x) $\vee$ desert(x)
$\forall x$ lake(x) $\Rightarrow \neg$ sea(x) $\wedge \neg$ ocean(x)
$\forall x$ night(x) $\Rightarrow \neg$ sun(x)

Table 1: Sample of the rules used in the tagging experiments.

4. EXPERIMENTAL RESULTS

The datasets used in the experiments have been constructed by downloading a set of images from the Flicker hosting and online community website. Each image is associated to a textual description and title assigned by the author. The images have also been tagged by the users of the Flicker social network using a set of tags. Typically, most images are tagged by the users with 1 to 6 tags. The dataset has been built by randomly selecting $10,000$ images, each one assigned to at least one of the 80 most frequent tags in the dataset.

The graph of images is built by establishing connections among pairs of images. In particular, let u and z be two nodes of the graph (corresponding to two images in the dataset), the value w_{uz} is attached to the edge (u, z) as $w_{uz} = cos\_sim(title_u, title_z)$, where cos_sim indicates the cosine similarity function and $title_u$ and $title_z$ are the TF-IDF bag-of-words representations of the title of u and z, respectively. To reduce the number of edges, only edges with a weight above 0.1 are kept. The resulting graph contains approximatively $311,000$ edges.

Other similarity metrics could have been exploited like the similarity of the textual descriptions of the images, authorship, friendship between the corresponding authors, etc. We decided to not include any relation based on the visual similarity of the images, as state-of-the-art methods still fail to correlate well with the semantic similarity of the image contents. However, if available, any visual similarly metric could be integrated as the other relations. Once the graph is built, the user-provided tags have been removed from 75% of the nodes. Furthermore, a knowledge base containing a set of rules has been compiled to express the semantic relationships between the tags. In particular, 45 rules have been defined. Table 1 shows a small sample of the rules inserted in the knowledge base.

Each function associated to a tag is obtained by a regularization-based transduction on the graph. The conversion of logic rules into a continuous form has been performed using the different approaches described in section 3. We indicate with MNORM and PNORM the min-max and product t-norms, respectively. PGAUSS, MGAUSS, NGAUSS indicate the mixture-of-gaussians conversion of the logic clause using the definitions in equations (7), (8) and (9), respectively. In the experiments, the σ value for the mixture of Gaussians is a critical parameter. A tuning on this parameter has been performed and an optimal value of 0.05, 0.02, 0.1 has been found for MGAUSS, PGAUSS and NGAUSS respectively.

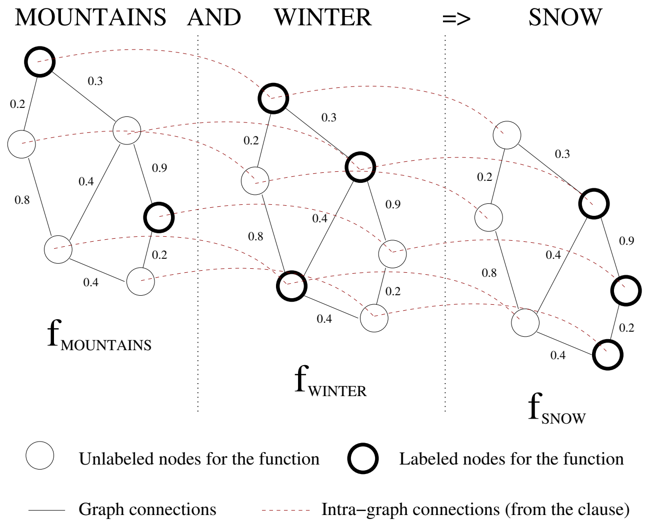

Figure 2: One separate graph regularization run is performed for each tag. The effect of the clause [$\forall x$ mountains(x) $\wedge$ winter(x) $\Rightarrow$ snow(x)] is to introduce inter-graph connections, resulting from the correlation among the function values as imposed by the rule.

Figure 2 shows the effect of a rule on the tagging task. Each rule adds new intra-graph connections, resulting from the correlation among the function values as imposed by the logic clause. The same experiments have been performed using the graph regularization algorithm without enforcing the FOL knowledge (no red connections in figure 2).

Since the user's provided tags are correct when considered one by one, but intrinsically inconsistent when considered globally, the accuracy measured exploiting a test set built using these tags would provide quite unreliable results. Therefore, the evaluation has been performed by using a set of human raters. For each image, the top 5 tags assigned to the image by the graph regularization schema using and not using the semantic constraints have been shown to three independent human raters. The raters were asked to decide whether one tag ranking is better, worse or equally relevant than the other. Please note that the position of the rankings obtained with or without the constraints have been randomized for each trial to avoid any bias in the evaluators.

The results are summarized in table 2, where SGR indicates the graph regularization when using logic rules for the specified conversion schema, and GR when no logic rules are used. The first column in the table reports the relative accuracy of the methods that are compared, as the percentage of images for which a rater preferred the SGR generated list over the GR one, given that she/he expressed a preference. The second column, instead, reports the percentage of images for which the raters expressed a preference for one of the two rankings. These latter values measure the impact of the introduction of the logic rules.

PNORM outperforms MNORM by a significant margin both in terms of accuracy and impact. MNORM does not provide a significantly improvement versus not using the logic rules (the raters expressed a preference for the baseline experiment in half of the cases). MGAUSS works similarly

NORM TYPE	Accuracy SGR better than GR	Impact SGR $\neq$ GR
MNORM	51.0%	2.0%
PNORM	63.1%	8.1%
MGAUSS	62.3%	8.3%
PGAUSS	70.3%	6.4%
NGAUSS	72.7%	6.6%

Table 2: Percentages of the cases in dataset of 10000 images in which the raters preferred the Semantic Graph Regularization (SGR) generated tag list with respect to the Graph Regularization (GR) computed list, given the different conversion methods for the rules. The last column reports the percentage of cases in which the users expressed a preference for one of the two lists.

to PNORM and it provides the highest impact among all the conversion schema.

PGAUSS and NGAUSS outperform all the other approaches by a significant margin in terms of tagging accuracy, while providing only a slightly smaller impact than PNORM and MGAUSS. We think that mixture-of-gaussian approaches provide a higher accuracy as they do not hypothesize an independence assumption among the input variables.

NGAUSS performs very well on this task. We think this is related to the specific clauses employed in the experiment. These clauses are implications in the form $A_1 \wedge \ldots \wedge A_n \Rightarrow B$ which are always verified except for a single configuration of the input variables. In this cases, NGAUSS converts the clause using a single negative Gaussian, whereas MGAUSS and PGAUSS employ a larger mixture. The more compact representation has the advantage of limiting the number of local minima, thus yielding an easier optimization problem. In a general setting, it is possible to select the most compact representation for each clause, therefore using PGAUSS or NGAUSS for expressions that are false or true for most configurations, respectively.

5. CONCLUSIONS

This paper presents a novel approach to integrate logic prior knowledge in form of FOL clauses into graph regularization. This methodology is a first attempt to bridge pure transductive machine learning approaches to knowledge based annotators based on logic formalisms. In particular, the presented approach directly injects logic knowledge compiled as continuous constraints into a classical graph regularization schema. Different conversion techniques for the FOL clauses have been proposed and tested. The experimental results have been carried out on an image tagging dataset and show that, using a careful selection of the logic knowledge conversion schema, the presented approach outperforms an image annotator based on graph regularization by a significant margin in terms of tagging accuracy.

6. REFERENCES

[1] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2434, 2006.

[2] L. Cao, J. Luo, and T. Huang. Annotating photo collections by label propagation according to multiple similarity cues. In *Proceeding of the 16th ACM international conference on Multimedia*, pages 121–130. ACM, 2008.

[3] M. Diligenti, M. Gori, M. Maggini, and L. Rigutini. Multitask kernel-based learning with first-order logic constraints. In *The 20th International Conference on Inductive Logic Programming*, 2010.

[4] M. Diligenti, M. Gori, M. Maggini, and L. Rigutini. Multitask Kernel-based Learning with Logic Constraints. In *Proceedings of the 19th European Conference on Artificial Intelligence*, pages 433–438. IOS Press, 2010.

[5] P. Frasconi, M. Gori, M. Maggini, and G. Soda. Representation of finite state automata in recurrent radial basis function networks. *Machine Learning*, 23(1):5–32, 1996.

[6] E. Klement, R. Mesiar, and E. Pap. *Triangular Norms*. Kluwer Academic Publisher, 2000.

[7] J. Li and J. Wang. Real-time computerized annotation of pictures. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 911–920. ACM, 2006.

[8] K. Matusiak. Towards user-centered indexing in digital image collections. *OCLC Systems & Services*, 22(4):283–298, 2006.

[9] F. Monay and D. Gatica-Perez. On image auto-annotation with latent space models. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 275–278. ACM, 2003.

[10] S. Peters, L. Denoyer, and P. Gallinari. Iterative Annotation of Multi-relational Social Networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, pages 96–103. IEEE, 2010.

[11] B. Sigurbjörnsson and R. Van Zwol. Flickr tag recommendation based on collective knowledge. In *Proceeding of the 17th international conference on World Wide Web*, pages 327–336. ACM, 2008.

[12] L. Wu, L. Yang, N. Yu, and X. Hua. Learning to tag. In *Proceedings of the 18th international conference on World wide web*, pages 361–370. ACM, 2009.

[13] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Scholkopf. Learning with Local and Global Consistency. *Advances in Neural Information Processing Systems*, 16:321–328, 2004.

[14] D. Zhou and B. Scholkopf. A regularization framework for learning from graph data. In *ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*, pages 132–137, 2004.