J Intell Inf Syst manuscript No.
(will be inserted by the editor)

Link Classification with Probabilistic Graphs

Nicola Di Mauro · Claudio Taranto · Floriana Esposito

Received: date / Accepted: date

Abstract The need to deal with the inherent uncertainty in real-world relational or networked data leads to the proposal of new probabilistic models, such as probabilistic graphs. Every edge in a probabilistic graph is associated with a probability whose value represents the likelihood of its existence, or the strength of the relation between the entities it connects.

The aim of this paper is to propose two machine learning techniques for the link classification problem in relational data exploiting the probabilistic graph representation. Both the proposed methods will exploit a language-constrained reachability method to infer the probability of possible hidden relationships that may exists between two nodes in a probabilistic graph.

Each hidden relationships between two nodes may be viewed as a feature (or a factor), and its corresponding probability as its weight, while an observed relationship is considered as a positive instance for its corresponding link label. Given a training set of observed links, the first learning approach is to use a propositionalization technique adopting a L2-regularized Logistic Regression to learn a model able to predict unobserved link labels. Since in some cases the edges' probability may be not known in advance or they could not be precisely defined for a classification task, the second proposed approach is to exploit the inference method and to use a mean squared technique to learn the edges' probabilities. Both the proposed methods have been evaluated on real world data sets and the corresponding results proved their validity.

Keywords Probabilistic graphs · Logistic Regression · Stochastic Gradient Descent · Statistical Relational Learning · Link Classification

N. Di Mauro
Department of Computer, University of Bari "Aldo Moro", 70125 Bari, Italy
E-mail: nicola.dimauro@uniba.it

C. Taranto
Department of Computer, University of Bari "Aldo Moro", 70125 Bari, Italy
E-mail: claudio.taranto@uniba.it

F. Esposito
Department of Computer, University of Bari "Aldo Moro", 70125 Bari, Italy
E-mail: floriana.esposito@uniba.it

1 Introduction

Many real-world relational/networked domains, such as biological and social networks, need tools able to deal with both the inherent uncertainty of the data and their complex relational structure. Traditional approaches for structured domains do no allow one to deal with uncertainty and hence their probabilistic extensions are necessary for handling and mining such data. This need leads in the last years to the proposal of Statistical Relational Learning (SRL) (Getoor and Taskar, 2007) and Probabilistic Inductive Logic Programming (PILP) (De Raedt et al, 2008) methods that combine the expressive power of relational representations with statistical tools to handle the uncertainty. Another important research topic, strongly connected to SRL, emerged in the last few years is the extension of graph structures with uncertainty (Potamias et al, 2010; Zou et al, 2010a; Pfeiffer III and Neville, 2011), leading to the *probabilistic graph* model.

With probabilistic graphs one can model structured domain, as with the classical graph structure, but with the advantage to also handle uncertain data. Uncertainty is modeled by means of *probabilistic edges* whose value quantifies the likelihood of the edge existence, or the strength of the link between the nodes it connects. Here the edges are not assumed to absolutely exist, but, adopting the *possible world semantics*, it may exist according to its own probability. Since we have to deal with probabilistic edges, then arises the problem of transposing the tasks in classical graph structure to this probabilistic setting.

Computing the connectivity of the network represents one of the main issues in probabilistic graphs. A generalization of the pairwise reachability is the network reliability problem (Colbourn, 1987), where the aim is to determine the probability that all pairs of nodes are reachable from one another. However, unlike in a deterministic graph in which the reachability function returns a binary value representing the existence of a path connecting two nodes, in the case of probabilistic graphs this existence assumes a probabilistic value.

The concept of reachability, along with its specializations, used to compute how two nodes are likely to be connected, represents one of the main tool in probabilistic graphs (entities not directly linked may be related by chains of links). It is quite similar to the concept of *link prediction* (Getoor and Diehl, 2005), whose task may be formalized as follows. Given a networked structure (V, E) made up of a set of data instances V and a set of observed links E among some nodes in V, the task corresponds to predict how likely should exist an unobserved link between two nodes. Extending the concept of link prediction to probabilistic graphs adds an important ingredient that should be adequately exploited. Indeed, the key difference with respect to classical link prediction is that in a probabilistic graph the observed links between two nodes cannot be considered always true, and hence methods exploiting probabilistic links are needed. When the edges are labeled they may describe different kind of links among the nodes and hence *link classification* methods, returning a categorical value and not just a binary value as in link prediction, are needed to identify the correct unobserved link.

In this paper we provide two machine learning methods to predict the most likely link between two nodes in probabilistic graphs. Given a probabilistic graph, in both the approaches we exploited the reachability tool to infer the probability of some possible interconnections that may exists between two nodes, as already reported in (Taranto et al, 2013).

In the first proposed machine learning approach, each of these unobserved connections may be viewed as a feature, or a factor, between the two nodes and the corresponding probability as its weight. Each observed labeled link is considered as a positive instance for the label it represents. The link label may be viewed as the value of the output random variable y_i, while, the features between two nodes, computed with the inference tool, correspond to the components of the corresponding input vector $\mathbf{x}_i$. Hence, given the constructed training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, obtained from a set of n observed links, a L2-regularized Logistic Regression has been adopted to learn a model to be used to predict unobserved link labels. This approach is quite similar to that of *propositionalization* (Kramer et al, 2000) proposed in the fields of PILP and SRL, where the relational data are flattened to a propositional representation using relational features in order to have efficient learning results. Here, the further problem that we have to handle is that the relational representation is associated with uncertainty.

Another problem that we have to consider is that in many real world domains the data are provided with a flat representation and not with a relational one, or that the links among the involved entities are hidden. In particular, we are in the case where we have not an observed graph structure of the domain. Hence, in order to apply the proposed learning approach, based on probabilistic graphs, we used a method that elicits, from a flat data set, the hidden relational structure with its corresponding probabilistic information, by exploiting similarity functions.

In many cases the relational structure of the domain is known, but the edges' probabilities could not be observed or they are not precisely computable by a similarity function. The second proposed machine learning approach, extending that proposed in (Taranto et al, 2013), try to overcome this problem by exploiting the inference method and using a mean squared technique to learn the edges' probabilities. Given the training set $\mathcal{D}$, we proposed a stochastic gradient descent method (Bottou, 1998) to learn the edge probabilities in order to minimize a given objective function.

We used as application domains to validate the proposed techniques that of recommender systems (Desrosiers and Karypis, 2011), where the aim is to predict the unknown rating between an user and an item, and that of classification of web pages, a benchmark data set that has been the subject of prior study in machine learning. Experiments proved that the proposed approaches achieves significant results when compared to state-of-the-art methods for the recommendation task (for the domain of recommender systems) and to state-of-the-art SRL systems (for the case of web pages classification).

The rest of this paper is organized as follows. The next section reports some related works. Section 3 introduces the probabilistic graph framework. Section 4 describes how the link classification problem is solved adopting the two proposed learning methods. Then, Section 5 shows the method we used to elicit the hidden relations and their corresponding probabilities from flat data (Section 5.1), and the corresponding results of the proposed learning approaches on some real world problems. Lastly, Section 6 concludes the paper.

2 Related Works

Given a graph (or a network), the goal we are dealing with is to predict edges (or links) that could be most likely added to the graph in the future, sometimes called *link prediction problem* (Getoor and Diehl, 2005). In particular, we focus on the task of *link classification*, where we try to both predict and classify a set of links (Taskar et al, 2003). Applications where the link prediction can be used are those such as identifying the structure of a criminal network, overcoming the data-sparsity problem in recommender systems using collaborative filtering (Zan et al, 2005), analyzing users navigation history to generate users tools that increase navigational efficiency (Zhu, 2003). The link prediction problem is also related to the problem of inferring missing links from an observed network. One constructs a network of interactions based on observable data and then tries to infer additional links that are most likely to exist (Goldberg and Roth, 2003; Popescul and Ungar, 2003; Taskar et al, 2003).

The methods that solve this problem assign a connection weight to pairs of nodes, based on the input graph, and then produce a ranked list in decreasing order of such connections. This could correspond to compute a measure of proximity or a similarity between nodes, by using the length of their shortest path in the graph. Such a measure follows the notion that collaboration networks are *small worlds*, in which individuals are related through short chains (Newman, 2001b).

Other methods try to compute the similarity between two nodes by looking their corresponding neighborhoods. Given two nodes, if the set of their corresponding neighbors have a large overlapping then the nodes should be very similar. It has been shown in (Newman, 2001a) that there is a correlation between the number of common neighbors between two nodes at a given time, and the probability they will be similar in the future.

Another method uses random walks on the graph (von Luxburg et al, 2011), where starting from a node, the selection of the next node to visit is randomly chosen among its neighbors. Adopting this approach it is possible to compute the hitting time between two nodes x and y as the expected number of steps required for a random walk starting at x to reach y.

All the methods described above consider the space of representation as a graph with the nodes of the network indicating the objects of the world and the edges with a numeric value that indicates their weight.

Over the last few years uncertain graphs have become an important research topic (Potamias et al, 2010; Zou et al, 2010a,b). In these graphs each edge is associated with an edge existence probability that quantifies the likelihood that the edge exists in the graphs. Using this representation it is possible to adopt the *possible world* semantics to model it. One of the main issue in uncertain graphs is how to compute the connectivity of the network. As already said, unlike in a deterministic graph in which the reachability function is a binary function indicating whether or not there is a path that connects two nodes, in the case of the reachability on uncertain graphs the function assumes probabilistic values. In (Potamias et al, 2010), the authors provide a list of alternative shortest-path distance measures for uncertain graphs in order to discover the k closest nodes to a given node. Another work (Jin et al, 2011) tries to deal with the concept of $x - y$ distance-constraint reachability problem. In particular, given two nodes x and y, they try to solve the problem of computing the probability that the distance from x to y is lesser than

or equal to a user-defined threshold. In order to solve this problem, they proposed an exact algorithm and two reachability estimators.

As regards the need to elicit hidden relations, research in the past decade on SRL has shown the power of the underlying network of relations in relational data. However, many data sets do not contain explicit relations. Trying to elicit and to exploit structured information, instead of applying machine learning methods on flat descriptions has been investigated in recent years, such as in (Witsenburg and Blockeel, 2011; Macskassy, 2011). In (Witsenburg and Blockeel, 2011) the authors proposed a method that exploiting the graph structure, by including link information, yields a more relevant similarity measure to cluster the nodes of the graph. (Macskassy, 2011) investigated the possibility of constructing relations, by using simple similarity-based rules, such that relational inference results in better classification performance than non-relational inference.

3 Probabilistic Graphs

Let $G = (V, E)$, be a graph where V is a collection of nodes and $E \subseteq V \times V$ is the set of edges, or relationships, between the nodes.

Definition 1 (Probabilistic graph) A *probabilistic graph* is a system $G = (V, E, \Sigma, l_V, l_E, s, t, p)$, where (V, E) is an directed graph, V is the set of nodes, E is the set of ordered pairs of nodes where e=(s,t), Σ is a set of labels, $l_V : V \to \Sigma$ is a function assigning labels to nodes, $l_E : E \to \Sigma$ is a function assigning labels to the edges, $s : E \to V$ is a function returning the source node of an edge, $t : E \to V$ is a function returning the target node of an edge, $p : E \to [0, 1]$ is a function assigning *existence probability* values to the edges.

The existence probability $p(a)$ of an edge $a = (u, v) \in E$ is the probability that the edge a, connecting the node u to the node v, can exist in the graph. A particular case of probabilistic graph is the *discrete graph*[1], where binary edges between nodes represent the presence or the absence of a relationship between them, i.e., the existence probability value on all observed edges is 1.

The *possible world semantics*, specifying a probability distribution on discrete graphs and formalized in the *distribution semantics* of Sato (Sato, 1995) for the first order logic, is usually used for probabilistic graphs. We can imagine a probabilistic graph G as a sampler of worlds, where each world is an instance of G. A discrete graph G' is sampled from G according to the probability distribution P, and denoted as $G' \sqsubseteq G$, when each edge $a \in E$ is selected to be an edge of G' with probability $p(a)$. Edges labeled with probabilities are treated as mutually independent random variables indicating whether or not the corresponding edge belongs to a discrete graph.

Assuming independence among edges, the probability distribution over discrete graphs $G' = (V, E') \sqsubseteq G = (V, E)$ is given by

$$P(G'|G) = \prod_{a \in E'} p(a) \prod_{a \in E \setminus E'} (1 - p(a)). \quad (1)$$

[1] Sometimes called *certain graph*.

Definition 2 (Simple path) Given an uncertain graph $G = (V, E)$, a *simple path* of length k from u to v in G, expressed as $\pi \in G$, is a sequence of edges denoted as $\pi = \langle e_1, e_2, \dots e_k \rangle$, where $e_1 = (u, v_1)$, $e_k = (v_{k-1}, v)$, $e_i = (v_{i-1}, v_i)$ for $1 < i < k - 1$, $e_i \in E$ and $u, v, v_i \in V$, or equivalently as $\pi = u \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2 \cdots v_{k-1} \xrightarrow{e_k} v$.

Given an uncertain graph G, and $\pi = \langle e_1, e_2, \dots e_k \rangle$ a path in G from the node u to the node v, $\ell(\pi) = l_E(e_1) l_E(e_2) \cdots l_E(e_k)$ denotes the ordered concatenation of the labels of all the edges belonging to π.

We will adopt a *regular expression* $\mathtt{R}$ to denote what is the exact sequence of the labels that the path must contain.

Definition 3 (Language-constrained simple path) Given a probabilistic graph G and a *regular expression* $\mathtt{R}$, a *language constrained simple path* is a simple path $\pi \in G$ such that $\ell(\pi) \in L(\mathtt{R})$, where $L(\mathtt{R})$ is the language described by $\mathtt{R}$. We also say that π satisfies $\mathtt{R}$ in G.

3.1 Inference

Given a probabilistic graph G, a main task corresponds to compute the probability that there exists a simple path between two nodes u and v, that is, querying for the probability that a randomly sampled discrete graph contains a simple path between u and v. More formally, the *existence probability* $P(\pi|G)$ of a simple path π in a probabilistic graph G corresponds to the marginal $P(\pi, G'|G)$ with respect to π:

$$P(\pi|G) = \sum_{G' \sqsubseteq G} \mathbb{I}\{\pi \in G'\} \cdot P(G'|G), \tag{2}$$

where $\mathbb{I}\{\pi \in G'\} = 1$ if there exits the simple path π in G', and $\mathbb{I}\{\pi \in G'\} = 0$ otherwise[2]. In other words, the existence probability of the simple path π is the probability that the simple path π exists in a randomly sampled discrete graph.

Definition 4 (Language-constrained simple path probability) Given a probabilistic graph G and a *regular expression* $\mathtt{R}$, the probability of a *language-constrained simple path* $\pi \in G$ is

$$P(\pi|G, \mathtt{R}) = \sum_{G' \sqsubseteq G} \mathbb{I}\{\pi \in G'|\mathtt{R}\} \cdot P(G'|G), \tag{3}$$

where $\mathbb{I}\{\pi \in G'|\mathtt{R}\} = 1$ if there exists a simple path π in G' such that $\ell(\pi) \in L(\mathtt{R})$, and $\mathbb{I}\{\pi \in G'|\mathtt{R}\} = 0$ otherwise.

The previous definition give us the possibility to compute the probability of a set of simple path queries, or patterns, fulfilling the structure imposed by a regular expression. In this way we are interested in discrete graphs that contain at least one simple path belonging to the language denoted by the regular expression.

[2] In the rest of the paper, if not otherwise specified, $\mathbb{I}\{C\}$ denotes the indicator function returning 1 if the condition C is true, and 0 otherwise.

Computing the existence probability directly using (2) or (3) is intensive and intractable for large graphs since the number of discrete graphs to be checked is exponential in the number of probabilistic edges. It involves computing the existence of the simple path in every discrete graph and accumulating their probability.

A natural way to overcome the intractability of computing the existence probability of a simple path is to approximate it using a Monte Carlo sampling approach (Jin et al, 2011):

1. we sample n possible discrete graphs, $G_1, G_2, \dots G_n$ from the probabilistic graph G by sampling the edges of each discrete graph uniformly at random according to their existence probabilities; and
2. we check if the simple path exists in each sampled graph G_i.

This process provides the following basic sampling estimator for $P(\pi|G)$:

$$P(\pi|G) \approx \widehat{P(\pi|G)} = \frac{\sum_{i=1}^{n} \mathbb{I}\{\pi \in G_i\}}{n}. \tag{4}$$

Note that it is not necessary to sample all the edges to check whether the graph contains the path. For instance, assuming to use an iterative depth first search (DFS) procedure to check the path existence. When a node is just visited, we will sample all its adjacent edges and pushing them into the stack used by the iterative procedure. We will stop the procedure either when the target node is reached or when the stack is empty (non existence).

Algorithm 1 reports the algorithm to solve the inference step. The function `sampledAsTrue` implements a memoization technique in order to sample the edges. If the function is called for the first time on a given edge e, then it samples the edge and returns true whether the edge has been sampled as true and false otherwise. All successive calls on the same already sampled edge consist in returning the previous sampled value. The algorithm corresponds to a DFS starting from the node u and ending to the node v if possible. If the search ends in v a positive count is accumulated. Then the estimated probability is computed by dividing the accumulated positive counts by the number of samplings n.

Given a probabilistic graph G with d the mean degree of its node, and k the length of a path π, then the complexity of the Algorithm 1 is $O(pd^k)$, where p is the mean probability of the edges.

4 Link Classification

After having defined the probabilistic graph framework, we can adopt language-constrained simple paths in order to extract probabilistic features (i.e., patterns) to describe the correlation between two nodes in the graph.

Given a probabilistic graph G, with V the set of nodes, E the set of edges, Σ the set of edges' labels, and $Y \subseteq \Sigma$, we have a set of edges $D \subseteq E$ such that for each element $e \in D$: $l_E(e) \in Y$. In particular, D represents the set of observed links whose label belongs to the set Y, and we are interested in learning how the l_E function should assign the subset of the label Y to unobserved edges.

Given the set of training links D and the set of labels Y we want to learn a model able to correctly classify unobserved links. A way to solve the classification task can be that of using a language based classification approach. Given an

Algorithm 1 INFER(G, π, R, n)

Require: G: the probabilistic graph; π: the path $u \xrightarrow{e_1} v_1 \xrightarrow{e_2} v_2 \cdots v_{k-1} \xrightarrow{e_k} v$; R: the regular expression; n: the number of samplings;
Ensure: $\widehat{P(\pi|G,\mathtt{R})}$

```
1: c = 0
2: for i = 1 to n do
3:    visited = { u }
4:    S.clear()
5:    sampler.clear()
6:    depth = 1
7:    prevNode[0] = u
8:    proven = false
9:    for all adjacent node a_j of the node u do
10:      S.push((a_j,depth))
11:   while not S.empty() and not proven do
12:      (a, depth) = S.top()
13:      S.pop()
14:      e = (prevNode[depth-1],a)
15:      prevNode[depth] = a
16:      if a ∉ visited and ℓ(e) == e_depth then
17:         sampled = sampledAsTrue(e, sampler)
18:         if depth == k then
19:            c++
20:            proven = true
21:         visited.add(a)
22:         for all adjacent node a'_j of the node a do
23:            S.push((a'_j,depth+1))
24: return c/n
```

unobserved edge $e_i = (u_i, v_i)$, in order to predict its class $\widehat{y_i} \in Y$ (i.e., correctly predicting its label $l_E(e_i)$) we have to solve the following maximization problem:

$$\widehat{y_i} = \arg\max_{q_j} P(q_j(u_i, v_i)|G), \tag{5}$$

where $q_j(u_i, v_i)$ is the unknown link with label $q_j \in Y$ between the nodes u_i and v_i. In particular, the maximization problem corresponds to compute the link prediction for each $q_j \in Y$ and then choosing that label with maximum likelihood. In other ways, this should correspond to learn how the function $l_E : E \to \Sigma$ assigns labels to the edges in D in order to predict the label of unobserved links.

4.1 Propositionalization of probabilistic graphs

The previous link prediction task can be solved by querying the probability of some language-constrained simple path. In particular, predicting the probability for each label q_j as $P(q_j(u_i, v_i)|G)$ in (5) corresponds to compute the probability $P(\pi|G)$ for a query simple path $\pi \in G$ satisfying the language $L(\mathtt{R}_j)$, i.e., computing $P(\pi|G)$ as in (3):

$$\widehat{y_j} = \arg\max_{q_j} P(q_j(u_i, v_i)|G) \approx \arg\max_{q_j} P(\pi|G, \mathtt{R}_j). \tag{6}$$

This query based approach consider the languages used to compute the (6) as independent form each other without considering any correlation between them.

This is the approach used in (Taranto et al, 2012a,b). A more interesting approach that we want to investigate in this paper is to learn from the probabilistic graph a linear model of classification combining the prediction of each language constrained simple path.

In particular, given an edge e_i and a set of k languages $\mathcal{L} = \{L(\mathtt{R}_1), \ldots, L(\mathtt{R}_k)\}$, we can generate k real valued features x_{ij} where $x_{ij} = P(\pi|G, \mathtt{R}_i)$, $1 \leq j \leq k$. The original training set of observed links D can hence be transformed into the set of instances $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,n}$, where $\mathbf{x}_i$ is a k-component vector of features $x_{ij} \in [0, 1]$, and y_i is the class label of the corresponding example $\mathbf{x}_i$.

Linear classification represents one of the most promising learning technique for problems with a huge number of instances and features aiming at learning a weight vector $\mathbf{w}$ as a model. L2-regularized Logistic Regression belongs to the class of linear classifiers and solves the following unconstrained optimization problem:

$$\min_{\mathbf{w}} f(\mathbf{w}) = \left(\frac{\mathbf{w}^T\mathbf{w}}{2} + C \sum_{i=1}^{n} \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) \right), \tag{7}$$

where $\log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)) = \xi(\mathbf{w}; \mathbf{x}_i, y_i)$ denotes the specific loss function, $\frac{1}{2}\mathbf{w}^T\mathbf{w}$ is the regularized term, and $C > 0$ is a penalty parameter. The decision function corresponds to $\mathrm{sgn}(\mathbf{w}^T \mathbf{x}_i)$. In case of binary classification $y_i \in \{-1, +1\}$, while for multiclass problems the one vs the rest strategy can be used.

Among many methods for training logistic regression models, such as iterative scaling, nonlinear conjugate gradient, quasi Newton, a new efficient and robust truncated Newton, called trust region Newton method, has been proposed in (Lin et al, 2008), whose corresponding algorithm has been implemented in the LIBLINEAR[3] system that we used in this paper.

Having the set D of observed links for which we want to learn how the l_E function labels them, and setting the set $\mathcal{L} = \{L(\mathtt{R}_1), \ldots, L(\mathtt{R}_k)\}$ of languages (done by the expert of the domain), for each observed link $e_i \in D$, whose observed label is y_i, we compute its corresponding feature vector representation $\mathbf{x}_i$ as $x_{ij} = P(\pi|G, \mathtt{R}_i)$, $1 \leq j \leq k$. The pair $(\mathbf{x}_i, y_i)$ represents the propositionalized instance corresponding to e_i of the new training set $\mathcal{D}$ that the LIBLINEAR will use to learn the weight vector $\mathbf{w}$ as a model. Then, for each new unobserved link we firstly compute its feature vector representation as above and then we use the learned model $\mathbf{w}$ with LIBLINEAR in order to predict the corresponding label.

An example of the propositionalization approach is reported in Table 1. We are supposing to have a data set of rating values between a set of users and a set of items. Each rating is a link between a user node and an item node labeled as `low`, `medium`, or `high`. Now supposing to have an edge between two user, named `similar`, denoting whether they have rated the same items in the same way, then a possible simple path that we can use to predict that a user `u` will rate a `low` an item `i` could be $\pi = \mathtt{u} \xrightarrow{\mathtt{similar}} \mathtt{u}_j \xrightarrow{\mathtt{low}} \mathtt{i}$. The probability $P(\pi|G)$ will be the value of the corresponding feature.

[3] `http://www.csie.ntu.edu.tw/~cjlin/liblinear`.

Table 1 Propositionalization of the observed links.

edge	label	x_1	x_2	$\cdots$	x_k	y
e_1	low	0.1	0.4	$\cdots$	0.9	1
e_2	medium	0.2	0.1	$\cdots$	0.3	2
e_2	high	0.3	0.5	$\cdots$	0.1	3
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
e_n	low	0.7	0.6	$\cdots$	0.5	1

4.2 Learning probabilistic graphs parameters

In this section we present a least mean squared method that is able to learn the parameters of a probabilistic graph. In particular, we know the structure of the graph and we want to learn the edges' probabilities minimizing a loss function with respect to a set of queries.

In a multiclass learning setting, supposing to have a set $\mathcal{R}$ of k regular expressions, if the probability $P(\pi|G,\mathtt{R_i})$ is very high, where $\mathtt{R_i} \in \mathcal{R}$, then a query path π may be labeled as belonging to the class c_i. If $P(\pi|G,\mathtt{R_i}) \geq P(\pi|G,\mathtt{R_j})$ for each $j \neq i$ then we assume that the class of π is c_i.

Given a data set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{M}$, where $\mathbf{x}_i$ is the observation and y_i is the *state of nature*, or the corresponding class, the decision rule is

$$\widehat{y} = \arg\max_{y_j} P(y_j|\mathbf{x}_i).$$

In our case the class to associate to the query π is

$$\widehat{c} = \arg\max_{c_i} P(\pi|G,\mathtt{R_i}).$$

Here the goal is to learn the edges' probabilities (i.e., the parameters) of the probabilistic graph in order to correctly predict the class for each query.

For a multiclass problem, we can define the following cost function:

$$J(\mathbf{p}) = \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{C}\left(P(y_i|\mathbf{x}_i) - \mathbb{I}(y_i = j)\right)^2, \tag{8}$$

where M is the number of training examples, C is the number of distinct classes, and $\mathbf{p}$ is the vector of the edges' probabilities. The aim is to choose $\mathbf{p}$ so as to minimize $J(\mathbf{p})$. This is often obtained by using an iterative process applying changes to the parameters $\Delta\mathbf{p}$ at each iteration. Denoting the parameters at the j-th iteration as $\mathbf{p}_j$, the simple update rule becomes:

$$\mathbf{p}_{j+1} = \mathbf{p}_j + \Delta\mathbf{p}_j.$$

Lets consider the gradient descent algorithm, which starts with some initial $\mathbf{p}$ and attempt to optimize the objective function by following the steepest descent direction given by the negative of the gradient of the parameters at the j-th iteration $\frac{\partial}{\partial \mathbf{p}_j} J(\mathbf{p}_j)$ by repeatedly performing the following update:

$$\mathbf{p}_{j+1} = \mathbf{p}_j - \mu\frac{\partial}{\partial \mathbf{p}_j} J(\mathbf{p}_j). \tag{9}$$

Algorithm 2 SGD(X, μ, e)

Require: μ: global learning rate; e: number of epochs; X: the set of training observations;
1: $j = 1$
2: initialize $\mathbf{p}_0$
3: **for** epoch = 1 **to** e **do**
4: $X' = \text{randomShuffle}(X)$;
5: **for** i = 0 **to** $|X|$ **do**
6: **for all** parameters l **do**
7: $p_{j+1,l} = p_{j,l} - \frac{\mu}{\sqrt{\sum_{k=1}^{j} g_{k,l}^2}} g_{j,l}$
8: $j = j + 1$

Here, μ is called the learning rate controlling how large the step in the direction of the negative gradient must be taken.

Since the p_i must be probabilities, we can represent them using a sigmoid function as

$$p_i = (1 + e^{-w_i})^{-1} = \sigma(w_i)$$

and then minimizing the cost function with respect to the weights w_i.

In order to minimize the function in (8), instead of using a standard gradient descent performing the update in Equation 9, we use a stochastic gradient descent (Robbins and Monro, 1951), that approximate the true gradient of $J(\mathbf{p}_j)$ by a gradient at a single observation giving a local estimate of which direction minimizes the cost using the following update:

$$\mathbf{p}_{j+1} = \mathbf{p}_j - \mu \frac{\partial}{\partial \mathbf{p}_j} J_i(\mathbf{p}_j), \tag{10}$$

where

$$J_i(\mathbf{p}) = \frac{1}{2} \sum_{j=1}^{k} \left(P(y_i|\mathbf{x}_i) - \mathbb{I}(y_i = j) \right)^2, \tag{11}$$

is the cost function applied to the single observation i.

We used an adaptive learning rate recently proposed in (Duchi et al, 2010), in an approach called AdaGrad, whose update rule, for each problem dimension l, takes the form

$$\Delta p_{j,l} = -\frac{\mu}{\sqrt{\sum_{k=1}^{j} g_{k,l}^2}} g_{j,l}, \tag{12}$$

where $p_{k,l}$ is the lth component (parameter) of the vector $\mathbf{p}_k$, and $g_{j,l} = \frac{\partial}{\partial p_{j,l}} J_i(\mathbf{p}_j)$ is the gradient of the lth parameter at iteration j. In particular, we have a dynamic learning rate for each parameter proportional to the $l2$ norm of all previous gradients for that parameter.

Since we are minimizing the cost function with respect to the weights w_i, we need to compute the gradients $\frac{\partial}{\partial w_{j,l}} J_i(\mathbf{w}_j)$. The modification to the cost function is simple and it gives us the following gradient:

$$\frac{\partial}{\partial w_j} J_i(\mathbf{w}) =$$
$$\frac{\partial}{\partial w_j} \frac{1}{2} \sum_{k=1}^{C} \left(P(y_k|x_i) - \mathbb{I}(y_i = c_k)\right)^2 =$$
$$\sum_{k=1}^{C} \left(\left(P(y_k|x_i) - \mathbb{I}(y_i = c_k)\right) \frac{\partial}{\partial w_j} \left(P(y_k|x_i) - \mathbb{I}(y_k = c_k)\right) \right) =$$
$$\sum_{k=1}^{C} \left(\left(P(y_k|x_i) - \mathbb{I}(y_i = c_k)\right) \frac{\partial}{\partial w_j} \left(\sum_{G' \sqsubseteq G} \prod_{e_h \in E'} p_{e_h} \prod_{e_h \in E \setminus E'} (1 - p_{e_h}) \right) \right) =$$
$$\sum_{k=1}^{C} \left(\left(P(y_k|x_i) - \mathbb{I}(y_i = c_k)\right) \frac{\partial}{\partial w_j} \left(\sum_{G' \sqsubseteq G} \prod_{e_h \in E'} \sigma(w_{e_h}) \prod_{e_h \in E \setminus E'} (1 - \sigma(w_{e_h})) \right) \right)$$

Being $\frac{\partial}{\partial x}\sigma(x) = \sigma(x)(1 - \sigma(x))$, then:

$$\frac{\partial}{\partial w_j} J_i(\mathbf{w}) =$$
$$\sum_{k=1}^{C} (P(y_k|x_i) - \mathbb{I}(y_i = c_k))\sigma(w_j)(1 - \sigma(w_j))$$
$$\left(\sum_{\substack{G' \sqsubseteq G \\ e_j \in G'}} \prod_{\substack{e_h \in G' \\ e_h \neq e_j}} \sigma(w_h) \prod_{e_h \notin G'} (1 - \sigma(w_h)) - \sum_{\substack{G' \sqsubseteq G \\ e_j \notin G'}} \prod_{e_h \in G'} \sigma(w_h) \prod_{\substack{e_h \notin G' \\ e_j \neq e_h}} (1 - \sigma(w_h)) \right) \tag{13}$$

Algorithm 2 reports the corresponding algorithm for the parameter learning. The update process can be done for a fixed number of epochs, where an epoch corresponds to update the parameters after having considered all the training observations. Before to start the update process we have to inizialize the parameters (row 2), and for each epoch the training examples are randomly shuffled.

If we set:

$$\begin{aligned} A_k &= (P(y_k|x_i) - \mathbb{I}(y_i = c_k))\sigma(w_j)(1 - \sigma(w_j)) \\ B_k &= \sum_{\substack{G' \sqsubseteq G \\ e_j \in G'}} \prod_{\substack{e_h \in G' \\ e_h \neq e_j}} \sigma(w_h) \prod_{e_h \notin G'} (1 - \sigma(w_h)) \\ C_k &= \sum_{\substack{G' \sqsubseteq G \\ e_j \notin G'}} \prod_{e_h \in G'} \sigma(w_h) \prod_{\substack{e_h \notin G' \\ e_j \neq e_h}} (1 - \sigma(w_h)) \end{aligned} \tag{14}$$

then

$$\frac{\partial}{\partial w_j} J_i(\mathbf{p}) = \sum_{k=1}^{M} A_k B_k C_k$$

The component B_k regards all the worlds (i.e., a discrete graph) proving the query and containing the edge e_j whose probability is p_j. On the contrary, the component C_k regards all the worlds proving the query but that are not required to contain the edge e_j. Both the quantities can be estimated using a Monte Carlo approach. In particular, if we sample m worlds, then

$$B_k = \frac{\#(e_j = true)}{m * p_j},$$

$$C_k = \frac{\#(e_j = false)}{m * (1 - p_j)}.$$

5 Experimental evaluation

We used as application domains to validate the proposed techniques that of recommender systems (Desrosiers and Karypis, 2011), where the aim is to predict the unknown rating between a user and an item, and that of classification of web pages, a benchmark data set that has been the subject of prior study in machine learning.

In order to validate the proposed approach in recommender systems the first data set we used is the MovieLens data set[4], made available by the GroupLens research group at University of Minnesota for the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems. We used the MovieLens 100K version consisting of 100000 ratings (ranging from 1 to 5) regarding 943 users and 1682 movies, divided into five folds. Each user has rated at least 20 movies and there are simple demographic info for the users (such as age, gender, occupation, and zip code). In this paper we used the ratings only without considering the demographic information.

The Hetrec2011-lastfm (Cantador et al, 2011) data set, related to recommender systems domain (music recommendation), is the second data set we used to validate the proposed method. This data set contains social networking, tagging, and music artist listening information from a set of 2K users from Last.fm online music system[5]. In this data set we have 1892 users, 17632 artists, 12717 bi-directional user friend relations and 92834 user-artist relations. We have discretized the user-listened artist relations into three (equal bins) classes (play1, play2 and play3, where play1 $<$ play2 $<$ play3) indicating the frequency with which a user has listened to a specific artist. Hetrec2011-lastfm data set has been divided into 4 fold made up of $\sim$70000 training ratings and $\sim$20000 testing ratings.

For these two data sets the goal is to predict the user's interest with respect to an unknown object. In Movielens data set, we want to predict the user's interest with respect to a new film, while in the Hetrec2011-lastfm data set the goal is to predict the frequency with which a user may listen to a new artist.

For both the data set it is necessary to elicit the uncertain relationships among the given evidence. The next section proposes a method to solve this task.

[4] `http://ir.ii.uam.es/hetrec2011/datasets.html`

[5] `http://www.lastfm.com`

5.1 Probabilistic graph creation in recommender system domain

When we work with a set of data, in which the probabilistic relationships between data are hidden, a common approach to elicit these connections is based on using similarity measures. To model the data with a graph we can adopt different similarity measures for each type of node involved in the relationships.

In a recommender system domain we have two types of entities: the users and the items, and the only observed relationship corresponds to the ratings that a user has assigned to a set of items. The goal is to predict the rating a user could assign to an object that he never rated in the past. In the collaborative filtering approach there are two methods to predict unknown rating exploiting users or items similarity. User-oriented methods estimate unknown ratings based on previous ratings of similar users, while in item-oriented approaches ratings are estimated using previous ratings given by the same user on similar items.

Let U be a set of n users and I a set of m items. A rating r_{ui} indicates the preference degree the user u expressed for the item i, where high values mean stronger preference. Let S_u be the set of items rated from user u. A user-based approach predicts an unobserved rating $\widehat{r_{ui}}$ as follows:

$$\widehat{r_{ui}} = \overline{r_u} + \frac{\sum_{v \in U | i \in S_u} \sigma_u(u,v) \cdot (r_{vi} - \overline{r_v})}{\sum_{v \in U | i \in S_u} |\sigma_u(u,v)|}, \tag{15}$$

where $\overline{r_u}$ represents the mean rating of user u, and $\sigma_u(u,v)$ stands for the similarity between users u and v, computed, for instance, using the Pearson correlation:

$$\sigma_u(u,v) = \frac{\sum_{a \in S_u \cap S_v} (r_{ua} - \overline{r_u}) \cdot (r_{va} - \overline{r_v})}{\sqrt{\sum_{a \in S_u \cap S_v} (r_{ua} - \overline{r_u})^2 \sum_{a \in S_u \cap S_v} (r_{va} - \overline{r_v})^2}}.$$

On the other side, item-based approaches predict the rating of a given item using the following formula:

$$\widehat{r_{ui}} = \frac{\sum_{j \in S_u | j \neq i} \sigma_i(i,j) \cdot r_{uj}}{\sum_{j \in S_u | j \neq i} |\sigma_i(i,j)|}, \tag{16}$$

where $\sigma_i(i,j)$ is the similarity between the item i and j.

These neighborhood approaches see each user connected to other users or consider each item related to other items as in a network structure. In particular they rely on the direct connections among the entities involved in the domain. However, as recently proved, techniques able to consider complex relationships among the entities, leveraging the information already present in the network, involves an improvement in the processes of querying and mining (Witsenburg and Blockeel, 2011; Taranto et al, 2011).

Given the set of observed ratings $\mathcal{K} = \{(u, i, r_{ui}) | r_{ui} \text{ is known}\}$, we add a node with label `user` for each user in $\mathcal{K}$, and a node with label `item` for each item in $\mathcal{K}$. The next step is to add the edges among the nodes. Each edge is characterized by a label and a probability value, which should indicate the degree of similarity between the two nodes.

Two kind of connections between nodes are added. For each user u, we added an edge, labeled as `simU`, between u and the k most similar users to u. The similarity

between two users u and v is computed adopting a weighted Pearson correlation between the items rated by both u and v.

In particular, the probability of the edge `simU` connecting two users u and v is computed as:

$$P(\texttt{simU}(u,v)) = \sigma_u(u,v) \cdot w_u(u,v),$$

where $\sigma_u(u,v)$ is the Pearson correlation between the vectors of ratings corresponding to the set of items rated by both user u and user v, and $w_u(u,v) = |S_u \cap S_v|/|S_u \cup S_v|$.

For each item i, we added an edge, with label `simI`, between i and the most k similar items to i. In particular, the probability of the edge `simI` connecting the item i to the item j has been computed as:

$$P(\texttt{simI}(i,j)) = \sigma_i(i,j) \cdot w_i(i,j),$$

where $\sigma_i(i,j)$ is the Pearson correlation between the vectors corresponding to the histogram of the set of ratings for the item i and the item j, and $w_i(i,j) = |\overline{S}_i \cap \overline{S}_j|/|\overline{S}_i \cup \overline{S}_j|$, where $\overline{S}_i$ is the set of users rating the item i.

Finally, edges with probability equal to 1, and with label $\texttt{r}_k$ between the user u and the item i, denoting the user u has rated the item i with a score equal to k, are added for each element (u, i, r_k) belonging to $\mathcal{K}$.

5.2 Propositionalization validation

After having constructed the probabilistic graph, the next step to validate the propositionalization approach corresponds to the features construction that will serve as input to the LIBLINEAR classification model.

5.2.1 Feature construction

Adopting a recommender system data set we can assume that the values of r_{ui} are discrete and belonging to a set R. Given the recommender probabilistic graph G, the query path based classification approach try to solve the problem

$$\widehat{r_{ui}} = \arg\max_{\texttt{r}_j} P(\texttt{r}_j(u,i)|G),$$

where $\texttt{r}_j(u,i)$ is the unknown link with label $\texttt{r}_j$ between the user u and the item i.

This link prediction task is based on querying the probability of some language constrained simple path. For instance, a user-based collaborative filtering approach may be obtained by querying the probability of the edges, starting from a user node and ending to an item node, denoted by the regular expression $\texttt{R}_i = \{\texttt{simU}^1\texttt{r}_i^1\}$, corresponding to use the simple path $\pi = u \xrightarrow{\texttt{simU}} u_i \xrightarrow{\texttt{r}_i} i$.

In particular, predicting the probability of the rating j as $P(\texttt{r}_j(u,i))$ corresponds to compute the probability $P(\pi|G)$ for a query path in $L(\texttt{R}_j)$, i.e.,

$$\widehat{r_{ui}} = \arg\max_{\texttt{r}_j} P(\texttt{r}_j(u,i)|G) \approx \arg\max_{\texttt{r}_j} P(\pi|G, \texttt{R}_j).$$

In the same way, item-based approach could be obtained by computing the probability of the paths constrained by the language $L(\texttt{R}_i)$, where $\texttt{R}_i = \{\texttt{r}_i^1\texttt{simI}^1\}$.

The power of the proposed framework is evident when the labels of the edges are heterogeneous. In such a situation our approach gives us the possibility to construct more complex queries such as that constrained by the regular expression $\mathtt{R}_i = \{\mathtt{r}_i\mathtt{simI}^n : 1 \leq n \leq 2\}$, that gives us the possibility to explore the graph by considering not only direct connections. Hybrid queries, such as those constrained by the regular expression $\mathtt{R}_i = \{\mathtt{r}_i\mathtt{simI}^n : 1 \leq n \leq 2\} \cup \{\mathtt{simU}^m\mathtt{r}_i^1 : 1 \leq m \leq 2\}$, give us the possibility to combine the user information with item information.

In order to use the feature based classification approach proposed in this paper we can define a set of regular expression $\mathcal{R}$ and then computing for each language $L(\mathtt{R}_i)$, with $\mathtt{R}_i \in \mathcal{R}$, the probability $P(\pi|G, \mathtt{R}_i)$ between two nodes in the graph. In particular in recommender system case, the set of observed ratings $\mathcal{K} = \{(u, i, r_{ui})|r_{ui} \text{ is known}\}$ is mapped to the training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,n}$, where x_{ij} is the probability $P(\pi|G, \mathtt{R}_j)$ between the nodes u and i, and y_i is equal to r_{ui}. The proposed link classification method has been implemented in the `Eagle` system[6] that provides a set of tools to deal with probabilistic graphs.

5.2.2 Results

For each data set, given the training/testing set, the validation procedure followed the steps:

1. creating the probabilistic graph from the training ratings data set as reported in Section 5.1;
2. defining a set $\mathcal{R}$ of regular expressions to be used to construct a specific set of features as described in Section 5.2.1;
3. learning the L2-regularized Logistic Regression model; and,
4. testing the links reported in the testing data set $\mathcal{T}$ by computing, for each pair $(u, i) \in \mathcal{T}$ the predicted value adopting the learned classification model and comparing the result with the true prediction reported in $\mathcal{T}$.

In order to learn the classification model as reported in Section 4.1, we used the L2-regularized Logistic Regression implementation included in the LIBLINEAR system (Lin et al, 2008). Given a set $\mathcal{T}$ of testing instances, the accuracy of the proposed framework has been evaluated according to the *Mean Absolute Error* (MAE), which measures the deviation of the predicted ratings from their true values as specified by the users. For each pair of a true user-specified rating r_{ui} and a predicted rating $\widehat{r_{ui}}$, the absolute difference $|r_{ui} - \widehat{r_{ui}}|$ is calculated between them as the error for that pair. Then MAE is computed as the average error over all pairs in $\mathcal{T}$:

$$MAE = \sum_{i}^{L} \frac{|r_{ui} - \widehat{r_{ui}}|}{|\mathcal{T}|}.$$

Sometimes, the accuracy is also computed using the *macroaveraging mean absolute error* (Baccianella et al, 2009), for the recommender case,

$$MAE^M(\widehat{r_{ui}}, \mathcal{T}) = \frac{1}{k} \sum_{j=1}^{k} \frac{1}{|T_j|} \sum_{x_i \in T_j} |\widehat{r_{ui}} - r_{ui}|,$$

[6] `http://www.di.uniba.it/~claudiotaranto/eagle.html`

Table 2 Regular expressions for the language constrained simple paths used for the MovieLens data set.

$\texttt{ml}_1 = \{\texttt{simU}^1\texttt{r}_k^1\}$
$\texttt{ml}_2 = \{\texttt{r}_k^1\texttt{simF}^1\}$
$\texttt{ml}_3 = \{\texttt{r}_k^1\texttt{simF}^n : 1 \leq n \leq 2\}$
$\texttt{ml}_4 = \{\texttt{simU}^n\texttt{r}_k^1 : 1 \leq n \leq 2\}$
$\texttt{ml}_5 = \{\texttt{simU}^n\texttt{r}_k^1 : 1 \leq n \leq 2\} \cup \{\texttt{r}_k^1\texttt{simF}^n : 1 \leq n \leq 2\}$
$\texttt{ml}_6 = \{\texttt{r}_k^1\texttt{simF}^n : 1 \leq n \leq 3\}$
$\texttt{ml}_7 = \{\texttt{simU}^n\texttt{r}_k^1 : 1 \leq n \leq 3\}$
$\texttt{ml}_8 = \{\texttt{simU}^n\texttt{r}_k^1 : 1 \leq n \leq 3\} \cup \{\texttt{r}_k^1\texttt{simF}^n : 1 \leq n \leq 3\}$
$\texttt{ml}_9 = \{\texttt{simU}^n\texttt{r}_k^1 : 1 \leq n \leq 4\} \cup \{\texttt{r}_k^1\texttt{simF}^n : 1 \leq n \leq 4\}$

where $T_j \subset \mathcal{T}$ denotes the set of test rating whose true class is j.

For the MovieLens graph construction, edges are added using the procedure presented in Section 5.1, where we set the parameter $n = 30$, indicating that an user (resp., a film) is connected, respectively, to 30 most similar users (resp., films). The value of each feature have been obtained with the Monte Carlo inference procedure by sampling M discrete graphs. In order to construct the set of features, we proposed to query the paths constrained by the set of regular expressions reported in Table 2.

The first language-constrained simple paths $L(\texttt{ml}_1)$ corresponds to adopt a user-based approach, while the second language $L(\texttt{ml}_2)$ gives us the possibility to simulate an item-based approach. Then, we propose to extend these two basic languages, $L(\texttt{ml}_1)$ and $L(\texttt{ml}_2)$, in order to construct features that consider a neighborhood with many nested levels. Finally, we constructed hybrid features by combining both the user-based and item-based methods and the large neighborhood explored with paths whose length is greater than one ($\texttt{ml}_5$, $\texttt{ml}_8$ and $\texttt{ml}_9$).

We defined two sets of features: the set F_1 based on simple languages and corresponding to the probabilities of the paths constrained by the set of regular expressions $\{\texttt{ml}_1, \texttt{ml}_2, \texttt{ml}_3, \texttt{ml}_4, \texttt{ml}_5\}$; and the set F_2 consisting of querying more complex paths constrained by the set of regular expressions $\{\texttt{ml}_3, \texttt{ml}_4, \texttt{ml}_5, \texttt{ml}_6, \texttt{ml}_7, \texttt{ml}_8, \texttt{ml}_9\}$.

Table 3 shows the results on the MovieLens data set obtained adopting the proposed approach implemented in the `Eagle` system. The last row reports the mean value of the MAE^M averaged on the five folds obtained with the proposed method. The results improve when we use the set F_2 of features. The last two columns report the results of two baseline methods. The second last column reports the results obtained with a system that predicts a rating adopting a uniform distribution, while the last column reports the results of a system that uses a categorical distribution that predicts the value k of a rating with probability $p_k = |D_k|/N$, where D_k is the number of ratings belonging to the data set having value k, and N is the total number of ratings.

In Table 4 we can see the errors committed for each rating. The rows for the methods `U` and `C` report the mean of the MAE^M value for each fold using a system adopting a uniform or a categorical distribution. The data set is not balanced and the method adhere more to the categorical distribution proving that they are able to recognize the unbalanced distribution of the data set.

We compared the prediction quality achieved by our model to that of previous work that used the MovieLens 100k data set, whose results are reported in Table 5.

Table 3 MAE^M values obtained with `Eagle` on the MovieLens data set.

Fold	`Eagle@`F_1	`Eagle@`F_2	`U`	`C`
1	0.8372	0.8044		
2	0.8323	0.8055		
3	0.8429	0.8256		
4	0.8494	0.8231		
5	0.8507	0.8270		
Mean	0.842±0.007	**0.817**±0.011	1.6	1.51

Table 4 MAE^M values for each class obtained with `Eagle` on the MovieLens data set.

Method	**r1**	**r2**	**r3**	**r4**	**r5**
`U`	2.0	1.4	1.2	1.4	2.0
`C`	2.53	1.65	1.00	0.89	1.47
`Eagle@`F_1	1.14	0.80	0.65	0.65	0.93
`Eagle@`F_2	**1.03**	**0.73**	**0.66**	**0.66**	**0.96**

Table 5 Comparison on the MovieLens 100k data set of the prediction quality of various CF models and our model.

CF Model	`MAE`
SVD PCA (Vozalis et al, 2010)	0.793
H-NLPCA (Vozalis et al, 2010)	0.784
Eagle	0.763
UI-RBM (Georgiev and Nakov, 2013)	0.690
Latent CF (Langseth and Nielsen, 2012)	0.685

Table 6 Regular expressions for the language constrained simple paths used for the hetrec2011-lastfm data set.

$\texttt{lfm}_1 = \{\texttt{simUser}^1\texttt{r}_k^1\}$
$\texttt{lfm}_2 = \{\texttt{r}_k^1\texttt{simArtist}^1\}$
$\texttt{lfm}_3 = \{\texttt{simUser}^n\texttt{r}_k^1 : 1 \leq n \leq 2\}$
$\texttt{lfm}_4 = \{\texttt{r}_k^1\texttt{simArtist}^n : 1 \leq n \leq 2\}$
$\texttt{lfm}_5 = \{\texttt{simUser}^1\texttt{r}_k^1\texttt{simArtist}^1\}$
$\texttt{lfm}_6 = \{\texttt{friend}^1\texttt{r}_k^1\}$
$\texttt{lfm}_7 = \{\texttt{simUser}^1\texttt{friend}^1\texttt{r}_k^1\}$
$\texttt{lfm}_8 = \{\texttt{friend}^1\texttt{r}_k^1\texttt{simArtist}^1\}$

We can see that our approach performs better than two strong models from the literature: the SVD- and the PCA-based approaches of (Vozalis et al, 2010). Our final result gets close to the state-of-the-art results of a probabilistic collaborative filtering model that explicitly represents all items and users simultaneously in the model (Georgiev and Nakov, 2013), and of a framework for collaborative filtering based on Restricted Boltzmann Machines (Langseth and Nielsen, 2012).

For the Hetrec2011-lastfm graph construction, edges are added using the procedure presented in Section 5.1, where we set the parameter $n = 1500$, indicating that an user or an artist is connected, respectively, to 1500 most similar users, resp. artists. Hetrec2011-lastfm data set is composed by two types of edges: similarity edges (`simUser` and `simArtist`) and social relationship edges (`friend`). Here, we want to evaluate whether adopting the social connections improves the classification performances (He and Chu, 2010).

Table 7 MAE^M values for each class obtained with `Eagle` on the Hetrec2011-lastfm data set.

Fold	Method	play1	play2	play3	All
1	`Eagle`@M_1	0.6315	0.5686	0.4216	0.5405
	`Eagle`@M_2	0.6047	0.2524	0.5946	0.4839
2	`Eagle`@M_1	0.6090	0.5975	0.4460	0.5508
	`Eagle`@M_2	0.5794	0.2326	0.6268	0.4796
3	`Eagle`@M_1	0.6194	0.5875	0.4542	0.5537
	`Eagle`@M_2	0.6062	0.1963	0.6796	0.4940
4	`Eagle`@M_1	0.6295	0.6077	0.4181	0.5517
	`Eagle`@M_2	0.5976	0.2432	0.5840	0.4749
Average	`Eagle`@M_1	0.6223	0.5903	0.4349	0.5492
	`Eagle`@M_2	0.5969	0.2311	0.6212	**0.4831**

We defined two sets of features using the set of regular expression reported in Table 6: the set M_1 corresponding to the probabilities of the paths constrained by the set of regular expressions $\{\texttt{lfm}_1, \texttt{lfm}_2, \texttt{lfm}_3, \texttt{lfm}_4, \texttt{lfm}_5\}$ do not considering the social relationships among the elements in the network; and the set M_2 corresponding to the probabilities of the paths constrained by the set of regular expressions $\{\texttt{lfm}_1, \texttt{lfm}_2, \texttt{lfm}_3, \texttt{lfm}_4, \texttt{lfm}_5, \texttt{lfm}_6, \texttt{lfm}_7, \texttt{lfm}_8\}$ where the social connections are taken into account.

Table 7 shows the Hetrec2011-lastfm results for each class comparing `Eagle`@M_1 and `Eagle`@M_2. We can see that `Eagle`@M_1 that adopt social relationship edges achieves better results than `Eagle`@M_2 that does not use these connections.

5.3 Parameter learning evaluation

In this section we report the results of learning the edges' probability on two different application domains: that of recommender systems, already used in the propositionalization method, and that of classification of web pages.

5.3.1 Recommender systems domain

Here we want to investigate whether it is possible to learn the edges' probability without using any similarity function and obtaining results at least comparable as those reported in the previous section.

In particular, given the regular expressions $\texttt{R}_j$, we firstly used the graph with the probabilities computed with the similarity function and the inference to solve the maximization

$$\widehat{r_{ui}} = \arg\max_{\texttt{r}_j} P(\texttt{r}_j(u,i)|G) \approx \arg\max_{\texttt{r}_j} P(\pi|G, \texttt{R}_j).$$

Then, we tried to solve the same maximization but after having learned the probabilities with respect to the regular expressions $\texttt{R}_j$.

For the Hetrec2011-lastfm data set we defined a set of regular expressions as reported in Table 8. Each language has been used individually for each experiment. Before to start the learning process all the edges have been initialized with values uniformly chosen from the interval $[0.01 - 0.1]$. The learning rate has been set to 0.1.

Table 8 Regular expressions for the language constrained simple paths used to learn the graph parameters in the case of Hetrec2011-lastfm data set.

L1 = $\{\texttt{simU}^1\texttt{r}_k^1\}$
L2 = $\{\texttt{r}_k^1\texttt{simArtist}^1\}$
L3 = $\{\texttt{simU}^n\texttt{r}_k^1 : 1 \leq n \leq 2\}$
L4 = $\{\texttt{r}_k^1\texttt{simArtist}^n : 1 \leq n \leq 2\}$

Table 9 The difference of the MAE value for each adopted regular expression on the Hetrec2011-lastfm data set with and without learning the graph parameters.

Language	w learning	w/o learning	p-value
L1	0.756±0.005	0.825±0.004	0.0000006
L2	0.570±0.003	0.575±0.004	0.0924
L3	0.687±0.078	0.783±0.136	0.2666
L4	0.513±0.006	0.641±0.004	0.00000003

Table 10 The difference of the MAE^M value for each adopted regular expression on the Hetrec2011-lastfm data set with and without learning the graph parameters.

Language	w learning	w/o learning	p-value
L1	0.761±0.005	0.830±0.003	0.00000037
L2	0.571±0.003	0.576±0.004	0.0924
L3	0.691±0.080	0.787±0.138	0.274
L4	0.513±0.006	0.645±0.004	0.00000003

Table 11 Language constrained simple paths used to learn the graph parameters in the case of MovieLens data set.

L1 = $\{\texttt{simU}^1\texttt{r}_k^1\}$
L2 = $\{\texttt{r}_k^1\texttt{simF}^1\}$
L3 = $\{\texttt{simU}^n\texttt{r}_k^1 : 1 \leq n \leq 2\}$
L4 = $\{\texttt{r}_k^1\texttt{simF}^n : 1 \leq n \leq 2\}$

Tables 9 and 10 reports, respectively, the difference of the MAE and the MAE^M, value for each adopted regular expression when learning the parameters (w learning) and when we use the similarity function to set the parameters (w/o learning). The algorithm has been run for 1 epoch for each of the four fold. The third row reports the results when we do inference by using the probability assigned by using the similarity function. As we can see, we always improve the results obtained without learning the parameters.

An insight of the behavior on the optimization phase is reported in Figure 1 where we plotted the curves of the progressive training squared error. In particular, for each observation, before to update the parameters, we computed the squared error. At a given iteration, the accumulated squared error is averaged by the number of observed observations. As we can see, the average error go down near to a minimum proving the validity of the method.

The second experiment has been done on the MovieLens data set. We defined a set of regular expressions as reported in Table 11. Each regular expression has been used individually for each experiment. Before to start the learning process all the edges have been initialized with values uniformly chosen from the interval $[0.49 - 0.51]$. The learning rate has been set to 0.1.

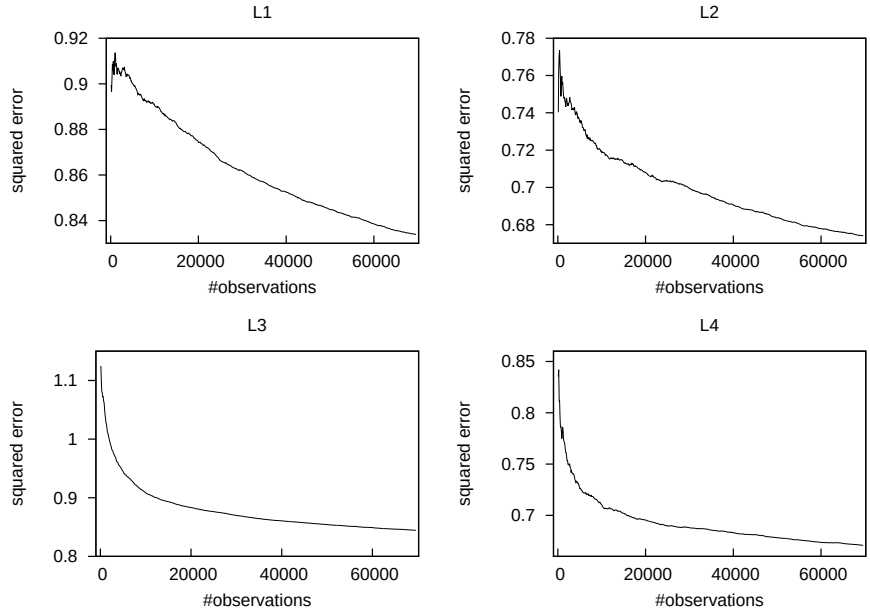

Fig. 1 Progressive training squared error curves, as a function of the number of training observations, for the four query languages of Table 8 applied to the Hetrec2011-lastfm data set.

Table 12 The difference of the MAE value for each adopted language on the MovieLens data set with and without learning the graph parameters.

Language	w learning	w/o learning	p-value
L1	0.9287±0.0154	0.9455±0.0139	0.1078
L2	0.8493±0.0137	0.8425±0.0156	0.4848
L3	0.8532±0.0052	0.8592±0.0089	0.2293
L4	0.7810±0.0114	0.7863±0.0034	0.3483

Figure 2 shows the progressive testing squared error curves for each language used to learn the parameters. As we can see, the algorithm does not make overfitting to the training data since the error go down on the test observation too.

Figure 3 and 4 reports, respectively, the progressive testing MAE and MAE^M, curves. The horizontal line represent the error obtained with the probability set by using the similarity function without learning the parameters. While, the other curve, with its corresponding error bars, in the same graph corresponds to the error obtained for each epoch after having learned the parameters. We obtained good results when using the languages L1 and L2 and comparable results when using the remaining two languages.

Table 12 (resp., Table 13) reports the mean and the standard deviation after the last epoch (the tenth) of the MAE (resp. the MAE^M) values averaged over the five folds. As we can see from the p-values, the difference is statistically significant for the languages L1 and L3.

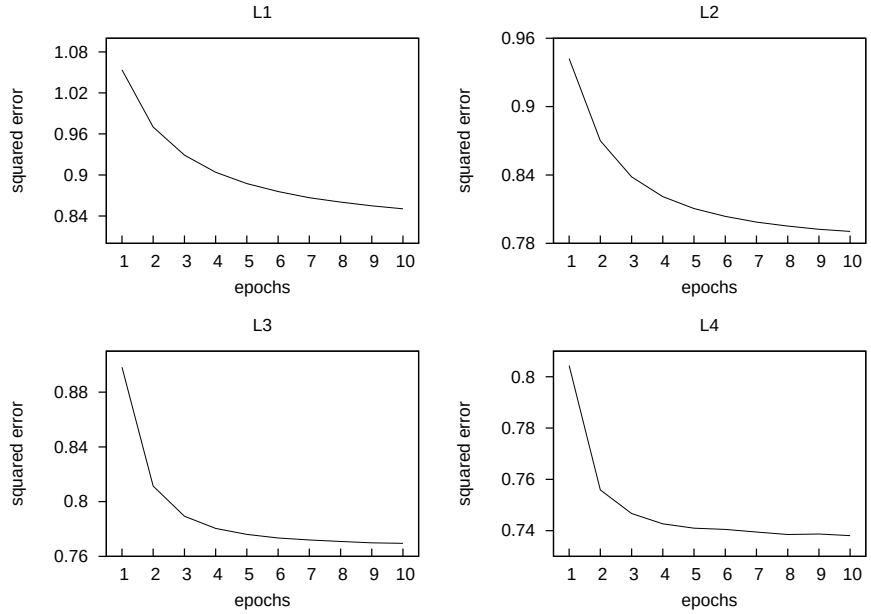

Fig. 2 Progressive testing squared error curves, as a function of the number of training epochs, for the four query languages of Table 11 applied to the MovieLens data set.

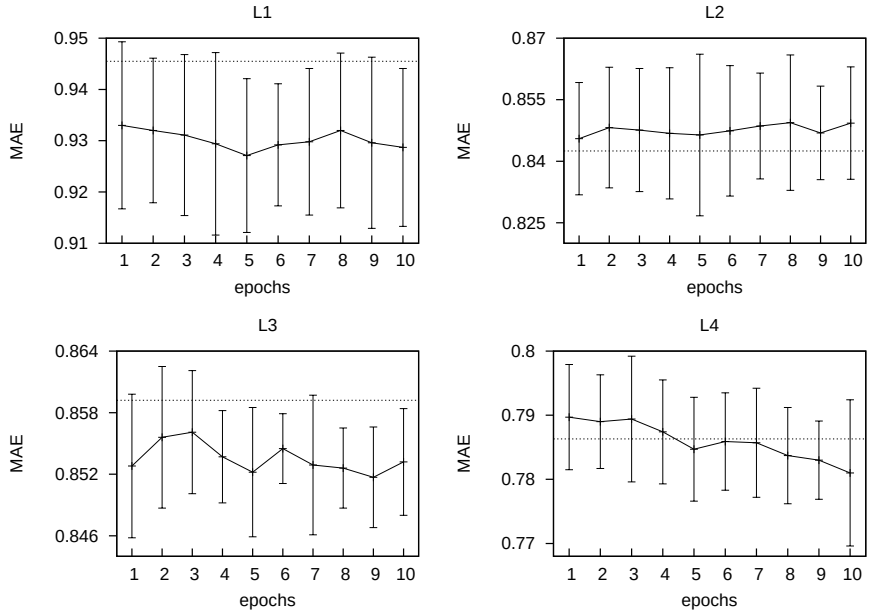

Fig. 3 Progressive testing MAE value curves, as a function of the number of training epochs, for the four query languages of Table 11 applied to the MovieLens data set.

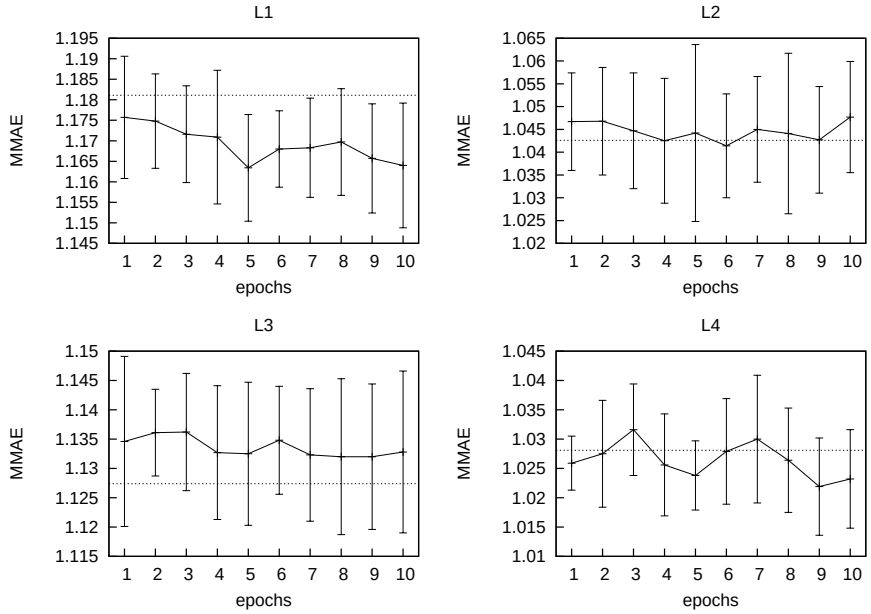

Fig. 4 Progressive testing MAE^M value curves, as a function of the number of training epochs, for the four query languages of Table 11 applied to the MovieLens data set.

Table 13 The difference of the MAE^M value for each adopted language on the MovieLens data set with and without learning the graph parameters.

Language	w learning	w/o learning	p-value
L1	1.1640±0.0152	1.1811±0.0143	0.1043
L2	1.0477±0.0122	1.0426±0.0199	0.6382
L3	1.1328±0.0138	1.1274±0.0074	0.4628
L4	1.0232±0.0084	1.0281±0.0074	0.3564

5.3.2 Web pages classification domain

In order to demonstrate the generality of our proposed approach we compared it to other existing state-of-the-art SRL approaches. In this experiment, we used the WebKB data set (Craven and Slattery, 2001), a classical example of collective classification task. The data set is divided into four folds, each describing the link structure of web pages from the computer science departments of four universities (i.e., Cornell, Texas, Washington and Wisconsin). Here the aim is to predict the class of a page depending on the classes of the pages that link to it and depending on the words being used in the text.

Following (Domingos and Lowd, 2009) and (Gutmann et al, 2011), we used the relational version of the data set from (Craven and Slattery, 2001), consisting of 4165 web pages and 10935 web links. Each web page is marked with one of the following six classes: `course`, `faculty`, `other`, `student`, `staff`, or `researchproject`. As done in (Gutmann et al, 2011), the class `person` has not been considered.

The graph constructed starting from the WebKB data set contains nodes denoting specific pages ($\texttt{page}_i$), words ($\texttt{word}_i$) and classes ($\texttt{class}_i$, there are six class

nodes), the links connecting the words that a page contains ($\texttt{page}_i \xrightarrow{\texttt{hasword}} \texttt{word}_j$), the connections among the pages ($\texttt{page}_i \xrightarrow{\texttt{linkto}} \texttt{page}_j$), and the links from a page to its corresponding class ($\texttt{page}_i \xrightarrow{\texttt{isclass}} \texttt{class}_j$). Then, there are two probabilistic edges, whose probabilities have to be learned, that are the following: `pwordclass`, for each pair of word and class, denoting the probability that a page of a specific class contain a given word ($\texttt{word}_i \xrightarrow{\texttt{pwordclass}} \texttt{class}_j$), and `pclass`, for each pair of classes, ($\texttt{class}_i \xrightarrow{\texttt{pclass}} \texttt{class}_j$), denoting the probability that a page of a class i link to a page of a class j.

In order to predict the class of a given page, the following regular expression has been used: $\texttt{R} = \{\ \texttt{linkto}^1\texttt{isclass}^1\texttt{pclass}^1\} \cup \{\ \texttt{hasword}^1\texttt{pwordclass}^1\}$. Given a web page `p`, this regular expression corresponds to use the simple paths $\pi_1 = \texttt{p} \xrightarrow{\texttt{linkto}} \texttt{p'} \xrightarrow{\texttt{isclass}} \texttt{c}_k \xrightarrow{\texttt{pclass}} \texttt{c}_i$ or $\pi_2 = \texttt{p} \xrightarrow{\texttt{hasword}} \texttt{w} \xrightarrow{\texttt{pwordclass}} \texttt{c}_i$ to predict the class $\texttt{c}_i$ of the page `p`. As already said, the class $\texttt{c}_i$ of the page `p` is predicted solving $\arg\max_{c_i} P(\pi_1, \pi_2 | G, \texttt{R})$.

We performed a 4-fold cross validation, that is, we trained the model on three universities and then tested it on the fourth one. We repeated this for all four universities and averaged the results. We measured the area under the receiver operating characteristic (AUC-ROC) and the area under the precision-recall curve (AUC-PR).

We compared our approach with LeProbLog (Gutmann et al, 2008), LFI-ProbLog (Gutmann et al, 2011), and Alchemy (Domingos and Lowd, 2009). LeProbLog is a regression-based parameter learning algorithm for ProbLog programs (i.e., a probabilistic Prolog), LFI-ProbLog is a parameter estimation algorithm for ProbLog programs from partial interpretations via a Soft-EM algorithm, while Alchemy is an implementation of Markov Logic Networks. On this data set LeProbLog achieves an an AUC-ROC of 0.738 ± 0.014 and an AUC-PR of 0.419 ± 0.014, LFI-ProbLog reaches an AUC-ROC of 0.886 ± 0.01 and an AUC-PR of 0.654 ± 0.03, and Alchemy obtains an AUC-ROC of 0.923 ± 0.016 and an AUC-PR of 0.788 ± 0.036 (Gutmann et al, 2011). Before to start the learning process with Eagle, all the edges have been initialized with values uniformly chosen from the interval $[0.01 - 0.1]$, and the learning rate has been set to 0.2. After 24 epochs, Eagle obtains an AUC-ROC of 0.705 ± 0.134 and an AUC-PR of 0.853 ± 0.094. Hence, it performs better than the other evaluated systems on this data set in terms of AUC-PR that gives a more informative picture of an algorithm performance when compared to the AUC-ROC (Davis and Goadrich, 2006).

6 Conclusions

In this paper we adopt the probabilistic graphs framework to deal with uncertain relational domains exploiting both edges probabilistic values and edges labels denoting the type of relationships among nodes. We proposed a language-constrained reachability method to infer the probability of possible hidden link that may exists between two nodes. Exploiting this inference method two machine learning approaches to link classification in a framework based on probabilistic graphs have been proposed. The first learning approach is to use a propositionalization technique adopting a L2-regularized Logistic Regression to learn a model able to predict unobserved link labels. Since in some cases the edges' probability may be

not known or not precisely defined for a classification task, the second approach is to exploit the inference method and to use a mean squared technique to learn the edges' probabilities. Both the proposed methods are been evaluated on real world data sets and the corresponding results proved their validity.

Acknowledgements This work fulfills the research objectives of the PON02_00563_3489339 project "PUGLIA@SERVICE - L'Ingegneria dei Servizi Internet-Based per lo sviluppo strutturale di un territorio intelligente" funded by the Italian Ministry of University and Research (MIUR).

References

Baccianella S, Esuli A, Sebastiani F (2009) Evaluation measures for ordinal regression. In: Proceedings of the 9th International Conference on Intelligent Systems Design and Applications, IEEE, pp 283–287

Bottou L (1998) Online algorithms and stochastic approximations. In: Saad D (ed) Online Learning and Neural Networks, Cambridge University Press

Cantador I, Brusilovsky P, Kuflik T (eds) (2011) 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011), ACM

Colbourn CJ (1987) The Combinatorics of Network Reliability. Oxford University Press

Craven M, Slattery S (2001) Relational learning with statistical predicate invention: Better models for hypertext. Machine Learning 43(1-2):97–119

Davis J, Goadrich M (2006) The relationship between precision-recall and roc curve. In: Proceedings of the 23rd International Conference on Machine Learning, pp 233–240

De Raedt L, Frasconi P, Kersting K, Muggleton S (eds) (2008) Probabilistic Inductive Logic Programming - Theory and Applications, LNCS, vol 4911, Springer

Desrosiers C, Karypis G (2011) A comprehensive survey of neighborhood-based recommendation methods. In: Ricci F, Rokach L, Shapira B, Kantor PB (eds) Recommender Systems Handbook, Springer, pp 107–144

Domingos P, Lowd D (2009) Markov Logic: An Interface Layer for Artificial Intelligence, 1st edn. Morgan and Claypool Publishers

Duchi JC, Hazan E, Singer Y (2010) Adaptive subgradient methods for online learning and stochastic optimization. In: Kalai AT, Mohri M (eds) The 23rd Conference on Learning Theory, Omnipress, pp 257–269

Georgiev K, Nakov P (2013) A non-iid framework for collaborative filtering with restricted boltzmann machines. In: Dasgupta S, McAllester D (eds) Proceedings of the 30th International Conference on Machine Learning, JMLR Workshop and Conference Proceedings, vol 28, pp 1148–1156

Getoor L, Diehl CP (2005) Link mining: a survey. SIGKDD Explorations 7(2):3–12

Getoor L, Taskar B (2007) Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning). The MIT Press

Goldberg DS, Roth FP (2003) Assessing experimentally derived interactions in a small world. Proceedings of the National Academy of Sciences 100(8):4372–4376

Gutmann B, Kimmig A, Kersting K, Raedt L (2008) Parameter learning in probabilistic databases: A least squares approach. In: Proceedings of the 2008 Euro-

pean Conference on Machine Learning and Knowledge Discovery in Databases - Part I, Springer-Verlag, pp 473–488

Gutmann B, Thon I, De Raedt L (2011) Learning the parameters of probabilistic logic programs from interpretations. In: Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Part I, Springer-Verlag, pp 581–596

He J, Chu WW (2010) A social network-based recommender system (snrs). In: Memon N, Xu JJ, Hicks DL, Chen H (eds) Data Mining for Social Network Data, Annals of Information Systems, vol 12, Springer, pp 47–74

Jin R, Liu L, Ding B, Wang H (2011) Distance-constraint reachability computation in uncertain graphs. Proceedings of the VLDB Endownment 4:551–562

Kramer S, Lavrač N, Flach P (2000) Relational data mining. Springer-Verlag, chap Propositionalization approaches to relational data mining, pp 262–286

Langseth H, Nielsen TD (2012) A latent model for collaborative filtering. International Journal of Approximate Reasoning 53(4):447 – 466

Lin CJ, Weng RC, Keerthi SS (2008) Trust region newton method for logistic regression. Journal of Machine Learning Research 9:627–650

Macskassy SA (2011) Relational classifiers in a non-relational world: Using homophily to create relations. In: Chen X, Dillon TS, Ishbuchi H, Pei J, Wang H, Wani MA (eds) 10th International Conference on Machine Learning and Applications and Workshops, IEEE, pp 406–411

Newman MEJ (2001a) Clustering and preferential attachment in growing networks. Phys Rev E 64

Newman MEJ (2001b) The structure of scientific collaboration networks. Proceedings of the National Academy of Sciences of the United States of America 98(2):404–409

Pfeiffer III JJ, Neville J (2011) Methods to determine node centrality and clustering in graphs with uncertain structure. In: Proceedings of the Fifth International Conference on Weblogs and Social Media, The AAAI Press

Popescul A, Ungar LH (2003) Statistical relational learning for link prediction. In: IJCAI03 Workshop on Learning Statistical Models from Relational Data

Potamias M, Bonchi F, Gionis A, Kollios G (2010) k-nearest neighbors in uncertain graphs. Proceedings of the VLDB Endowment 3:997–1008

Robbins H, Monro S (1951) A stochastic approximation method. Annals of Mathematical Statistics 22(3):400–407

Sato T (1995) A statistical learning method for logic programs with distribution semantics. In: In Proceedings of the 12th International Conference on Logic Programming, MIT Press, pp 715–729

Taranto C, Di Mauro N, Esposito F (2011) Probabilistic inference over image networks. Italian Research Conference on Digital Libraries 2011 CCIS 249:1–13

Taranto C, Di Mauro N, Esposito F (2012a) Uncertain graphs meet collaborative filtering. In: 3rd Italian Information Retrieval Workshop

Taranto C, Di Mauro N, Esposito F (2012b) Uncertain (multi)graphs for personalization services in digital libraries. In: Agosti M, Esposito F, Ferilli S, Ferro N (eds) 8th Italian Research Conference on Digital Libraries, Springer-Verlag, CCIS, vol 354, pp 141–152

Taranto C, Di Mauro N, Esposito F (2013) Learning in probabilistic graphs exploiting language-constrained patterns. In: Appice A, Ceci M, Loglisci C, Manco G, Masciari E, Ras ZW (eds) New Frontiers in Mining Complex Patterns, LNCS,

vol 7765, Springer Berlin Heidelberg, pp 155–169

Taskar B, Wong MF, Abbeel P, Koller D (2003) Link prediction in relational data. In: Thrun S, Saul LK, Schölkopf B (eds) Advances in Neural Information Processing Systems 16

von Luxburg U, Radl A, Hein M (2011) Hitting and commute times in large graphs are often misleading. CORR

Vozalis MG, Markos A, Margaritis KG (2010) Collaborative filtering through svd-based and hierarchical nonlinear pca. In: Proceedings of the 20th international conference on Artificial neural networks: Part I, Springer-Verlag, pp 395–400

Witsenburg T, Blockeel H (2011) Improving the accuracy of similarity measures by using link information. In: Kryszkiewicz M, Rybinski H, Skowron A, Ras ZW (eds) Proceedings of the 19th International conference on Foundations of Intelligent Systems, Springer, LNCS, vol 6804, pp 501–512

Zan H, Xin L, Hsinchun C (2005) Link prediction approach to collaborative filtering. In: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries, ACM Press, pp 141–142

Zhu J (2003) Mining web site link structures for adaptive web site navigation and search. PhD thesis

Zou Z, Gao H, Li J (2010a) Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 633–642

Zou Z, Li J, Gao H, Zhang S (2010b) Finding top-k maximal cliques in an uncertain graph. International Conference on Data Engineering pp 649–652