Machine Learning manuscript No.
(will be inserted by the editor)

Applying the Information Bottleneck to Statistical Relational Learning

Fabrizio Riguzzi · Nicola Di Mauro

Received: date / Accepted: date

Abstract In this paper we propose to apply the Information Bottleneck (IB) approach to the sub-class of Statistical Relational Learning (SRL) languages that are reducible to Bayesian networks. When the resulting networks involve hidden variables, learning these languages requires the use of techniques for learning from incomplete data such as the *Expectation Maximization* (EM) algorithm. Recently, the IB approach was shown to be able to avoid some of the local maxima in which EM can get trapped when learning with hidden variables. Here we present the algorithm Relational Information Bottleneck (RIB) that learns the parameters of SRL languages reducible to Bayesian Networks. In particular, we present the specialization of RIB to a language belonging to the family of languages based on the distribution semantics, Logic Programs with Annotated Disjunction (LPADs). This language is prototypical for such a family and its equivalent Bayesian networks contain hidden variables. RIB is evaluated on the IMDB, Cora and artificial datasets and compared with LeProbLog, EM, Alchemy and PRISM. The experimental results show that RIB has good performances especially when some logical atoms are unobserved. Moreover, it is particularly suitable when learning from interpretations that share the same Herbrand base.

1 Introduction

Probabilistic Inductive Logic Programming (De Raedt et al, 2008) and Statistical Relational Learning (SRL) (Getoor and Taskar, 2007) have been recently proposed for overcoming the limitations of traditional and relational Machine Learning by integrating approaches for learning graphical models with Inductive Logic Programming (Muggleton and De Raedt, 1994) techniques. This combination has been highly successful in a

Fabrizio Riguzzi
Dipartimento di Ingegneria, Università di Ferrara, Italy
E-mail: fabrizio.riguzzi@unife.it

Nicola Di Mauro
Dipartimento di Informatica, Università di Bari "Aldo Moro", Italy
E-mail: ndm@di.uniba.it

variety of fields, from social networks analysis to entity resolution, from collective classification to information extraction. With probabilistic logical languages such as Probabilistic Logic Programs (Dantsin, 1991), the Independent Choice Logic (ICL) (Poole, 1997), Bayesian Logic Programs (Kersting and De Raedt, 2001), Stochastic Logic Programs (Muggleton, 2002), CLP($\mathcal{BN}$) (Costa et al, 2003), Markov Logic Networks (MLNs) (Richardson and Domingos, 2006) PRISM (Sato, 1995) or ProbLog (De Raedt et al, 2007) one can represent the different type of objects and the uncertain relations among them that are typical of most application domains.

Some of these languages can be translated into Bayesian networks. When the networks contain hidden variables, learning the parameters of these languages requires the use of techniques for learning from incomplete data such as the *Expectation Maximization* (EM) algorithm (Dempster et al, 1977; Lauritzen, 1995). This algorithm performs a greedy search of the likelihood surface converging to a local stationary point, usually a local maximum. When there are many local maxima, EM can be trapped in a poor solution. The *Information Bottleneck* (IB) framework, originally proposed in (Tishby et al, 1999), was shown to be superior to EM for learning parameters of Bayesian networks with hidden variables (Elidan and Friedman, 2005) because it can avoid some local maxima. Moreover, it can be easily extended for inducing the structure of the network, including the number and cardinality of hidden variables. Given the advantages of IB with respect to EM, it is interesting to investigate its application to statistical relational languages that can be converted to Bayesian networks. In this paper, we discuss how IB can be applied to the problem of learning the parameters of these languages and present the Relational Information Bottleneck (RIB) algorithm. RIB modifies IB by taking into account parameter tying in its M-step.

In order to describe a concrete example of RIB, we specialize it for the case of Logic Programs with Annotated Disjunction (LPADs) (Vennekens et al, 2004), a recent formalism that is prototypical for the class of languages based on the *distribution semantics* (Sato, 1995) such as Probabilistic Logic Programs, ICL, PRISM and ProbLog. In the distribution semantics, a probabilistic program defines a joint distribution over queries and programs and the probability of a query is obtained by marginalization. All these languages have the same expressive power: there are transformations that can convert each one into the others (Vennekens and Verbaeten, 2003; De Raedt et al, 2008). In this paper we will use LPADs because they have the most general syntax and thus allow more modeling freedom.

Acyclic (Apt and Bezem, 1991) LPADs can be translated into Bayesian networks with hidden variables (Vennekens et al, 2004), thus an algorithm that handles incomplete data is necessary in order to perform learning. Previous approaches for learning this language include (Blockeel and Meert, 2007; Meert et al, 2007, 2008) that proposed to use the EM algorithm for inducing parameters and the Structural EM algorithm (Friedman, 1998) for inducing the structure of ground LPADs, and (Riguzzi, 2004, 2007a, 2008a) that adopts constraint optimization techniques to learn a subclass of ground programs.

RIB has been tested on the IMDB and Cora datasets and compared with LeProbLog (Gutmann et al, 2008, 2010), EM and Alchemy (implementing MLNs (Richardson and Domingos, 2006)). The experiments show that RIB is competitive with the other algorithms, with EM performing particularly well. To further investigate the relative strengths and weaknesses of RIB and EM, they are compared on a number of artificial datasets in which the example interpretations share the same Herbrand base. Two of these datasets also have unobserved atoms. On all these datasets RIB achieves

better mean absolute difference of the parameters, log likelihood and area under the precision-recall curve than EM, thus showing the suitability of RIB for learning from interpretations that share the same Herbrand base.

The paper is organized as follows. Section 2 presents notation and some preliminaries on Bayesian networks. In Section 3 we discuss the IB approach. Section 4 describes IB for learning Bayesian networks with hidden variables. Section 5 presents the RIB algorithm. In Section 6 we introduce the distribution semantics and LPADs and in Section 7 we illustrate how RIB has been tailored to LPADs. Section 8 discusses related work and Section 9 the experiments performed. Finally, Section 10 concludes the paper.

2 Preliminaries

In this section we briefly describe the adopted notation and we provide some preliminary notions regarding Bayesian networks.

We will use capital letters, such as X, Y, T, for variable names and lowercase letters x, y, t for specific values taken by the variables. Sets of variables will be usually denoted by boldface capital letters, such as $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{T}$, while assignments of values to those variables will be denoted by boldface lowercase letters, $\mathbf{x}$, $\mathbf{y}$, $\mathbf{t}$. Finally, we will use $P(x|y)$ as a shorthand for $P(X = x|Y = y)$.

Let $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ be a set of random variables, where each variable X_i may assume values from a finite set. Formally, a *Bayesian network* (Pearl, 1988) over $\mathbf{X}$ is a pair $\langle \mathcal{G}, \Theta \rangle$, where $\mathcal{G}$ is a directed acyclic graph whose nodes correspond to the random variables in $\mathbf{X}$, and the edges represent direct dependencies between the variables. The component Θ represents the set of parameters quantifying the network. For each variable X_i, the set of parents of X_i in $\mathcal{G}$ will be denoted by $\mathbf{Pa}_{X_i}$ or simply by $\mathbf{Pa}_i$.

Each node in the graph is annotated with a conditional probability table (CPT) $P(X_i|\mathbf{Pa}_i)$ defined by the parameters $\theta_{x_i|\mathbf{pa}_i} \in \Theta$ for each value x_i of X_i and $\mathbf{pa}_i$ of $\mathbf{Pa}_i$. A Bayesian network defines a unique joint probability distribution over $\mathbf{X}$ given by $P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\mathbf{Pa}_i)$. The graph $\mathcal{G}$ represents independence properties holding in the P distribution. In particular, each X_i is independent of its non-descendants given its parents $\mathbf{Pa}_i$. Moreover, X_i is independent of the rest of the variables given its *Markov blanket*: the set of its parents, its children and the parents of its children.

3 The IB Framework

In this section we briefly review the IB framework on which our approach is based.

IB has its origin in clustering (Tishby et al, 1999). Given the variables X and Y and their joint distribution $Q(X, Y)$, the aim of clustering is to group values of Y such that as much information as possible is preserved about X. For example, if Y are the words appearing in a set of documents and X are the documents' topics, the aim is to cluster words in a way that is most relevant to the documents' topics. The *information* that Y contains about X (and vice versa), or, in other words, the relevance of the variable X with respect to the variable Y, is naturally measured in terms of the

mutual information

$$\mathbf{I}_Q(X;Y) \triangleq \sum_{x,y} Q(x,y) \log \frac{Q(x,y)}{Q(x)Q(y)} = \sum_{x,y} Q(x)Q(y|x) \log \frac{Q(y|x)}{Q(y)}. \quad (1)$$

This quantity, as defined in Equation (1), is symmetric, non-negative, and is equal to zero iff the variables are independent. It measures how many bits are needed on average to convey the information X has about Y (or vice versa).

The aim of clustering in this case is to find (soft) partitions of Y's values that are informative about X. This requires balancing two goals: a) losing irrelevant distinctions made by Y, and at the same time b) maintaining relevant ones. To cluster Y's values, (Tishby et al, 1999) introduces a bottleneck variable T and a function $Q(T|Y)$: the values of T identify the various clusters and $Q(T|Y)$ represents the degree of membership of the values of Y to the clusters. The T variable must compress Y while capturing as much as possible the information about X. In other words, T must be such that $\mathbf{I}_Q(T;Y)$ is minimized while $\mathbf{I}_Q(X;T)$ is maximized. Clustering in this case can be performed by finding the parameters of the Q distribution such that the function

$$\mathcal{L}[Q] = \mathcal{L}[Q(t|y)] = \mathbf{I}_Q(Y;T) - \beta \mathbf{I}_Q(T;X) \quad (2)$$

is minimized, where β determines the trade-off between information compression and preservation (Elidan and Friedman, 2005). At $\beta = 0$ the compression is maximal (everything is assigned to a single cluster), while as $\beta \to \infty$ the quantization becomes arbitrarily detailed.

In (Friedman et al, 2001) the IB approach has been extended to the case of multiple observed variables $\mathbf{X}$ using several bottleneck variables. Let us consider first the case of a single bottleneck variable. The interactions are represented by two Bayesian networks: $\mathcal{G}_{in}$, representing the Q distribution (i.e., the required compression), and $\mathcal{G}_{out}$, representing the P distribution (i.e., the independences that should be obtained between the bottleneck variables and the target variables), see Figure 1. In particular, in $\mathcal{G}_{in}$ $\mathbf{X}$ and T have to be conditionally independent given Y, while, in $\mathcal{G}_{out}$, Y has to be conditionally independent of $\mathbf{X}$ given T. In other words, $\mathcal{G}_{in}$ represents that T is the compressed version of the observed variables, while $\mathcal{G}_{out}$ represents which relations should be maintained or predicted, since it specifies which variables are predicted by T. Any structure for $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ can be chosen provided they encode the above mentioned independences. The networks of Figure 1, for example, satisfy these constraints.

The application of IB to $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ attempts to find the values of $Q(T|Y)$ assuming that Q can be approximated by a distribution that factorizes according to $\mathcal{G}_{in}$. (Elidan and Friedman, 2005) showed that clustering can be performed by minimizing the following objective function

$$\mathcal{L}^{(2)}[Q,P] = \mathbf{I}_Q(Y;T) + \gamma \mathbf{D}(Q(Y,T,\mathbf{X}) || P(Y,T,\mathbf{X})), \quad (3)$$

where Q and P are the joint probabilities that are represented by the networks $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$, respectively, and $\mathbf{D}$ is the Kullback-Leibler divergence. The parameter γ balances the above two factors. When γ is zero one is interested in compressing the variable Y, while when γ is high one concentrates on choosing $Q(T|Y)$ that is close to a distribution satisfying the independences encoded by $\mathcal{G}_{out}$.

The case of multiple bottleneck variables can be treated similarly, by considering networks $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ containing a vector $\mathbf{T}$ of variables.

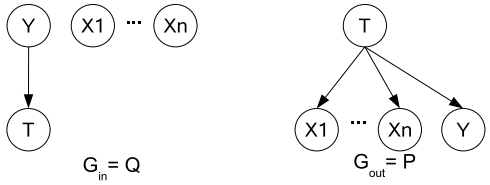

Fig. 1 G_{in} and G_{out} for the multivariate information bottleneck framework.

4 IB for Learning Probabilistic Graphical models

The IB approach has been applied in (Elidan and Friedman, 2005) to the problem of learning Bayesian networks with hidden variables. The aim is to compress information about the training data and to make the hidden variables informative about the observed attributes to ensure they preserve the relevant information. The approach is based on the multivariate extension of IB.

While the aim of IB is to learn a distribution $Q(T|Y)$, in (Elidan and Friedman, 2005) the authors focus on a somewhat different problem. In particular, given some data $\mathcal{D} = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$ over the observed variables $\mathbf{X}$, the aim is to find a generative model P over the variables $\mathbf{X}$ and the hidden variable T that describes $\mathcal{D}$. The variable Y is used in this case to represent the instance identity and takes values from $\{1, \ldots, M\}$. For each instance y, $\mathbf{x}[y]$ are the values that the variables $\mathbf{X}$ take in instance y. The goal is to find the parameters (and possibly the structure) of P such that T explains the observed data.

(Elidan and Friedman, 2005) proved that one can learn the parameters of P by minimizing the IB objective function $\mathcal{L}^{(2)}[Q, P]$ of Equation (3). In this case it takes the form

$$\mathcal{L}_{EM} = \mathbf{I}_Q(T;Y) - \gamma \left(\mathbb{E}_Q[\log P(\mathbf{X}, T)] - \mathbb{E}_Q[\log Q(T)]\right) \tag{4}$$

In the general case, we may have a vector $\mathbf{T}$ of hidden variables. Any distribution for $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ can be chosen, provided that $\mathbf{T}$ is independent of $\mathbf{X}$ given Y in $\mathcal{G}_{in}$ and Y is a leaf in $\mathcal{G}_{out}$ with $\mathbf{T}$ as its only parents. In order to make the treatment feasible, a factorized form for $Q(\mathbf{T}|Y)$ can be used, for example a naive Bayes assumption can be made, in which $Q(\mathbf{T}|Y)$ is factorized as $\prod_i Q(T_i|Y)$. Different factorizations correspond to different choices for $\mathcal{G}_{in}$. In the naïve Bayes case, the objective function takes the following form (Elidan and Friedman, 2005):

$$\mathcal{L}^{+}_{EM} = \sum_i \mathbf{I}_Q(T_i;Y) - \gamma \left(\mathbb{E}_Q[\log P(\mathbf{X}, \mathbf{T})] - \sum_i \mathbb{E}_Q[\log Q(T_i)] \right). \tag{5}$$

4.1 The IB-EM Algorithm

The Information Bottleneck EM algorithm (IB-EM) consists of the repetition of the following two steps:

- **E-step**: maximize $-\mathcal{L}^{+}_{EM}$ by optimizing $Q(\mathbf{T}|Y)$ while holding P fixed;
- **M-step**: maximize $-\mathcal{L}^{+}_{EM}$ by optimizing P while holding Q fixed.

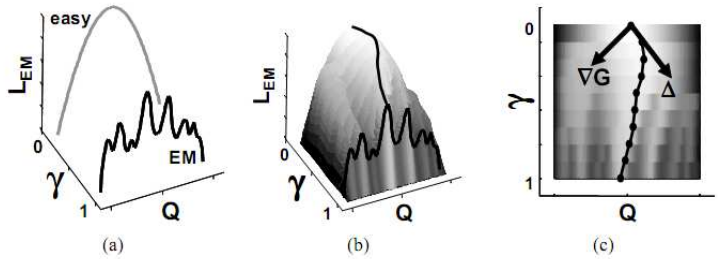

Fig. 2 Illustration of IB-EM (reproduced with permission from (Elidan and Friedman, 2005)).

The M-step is a standard maximum likelihood optimization of Bayesian networks. Indeed, the term involving P is $\mathbb{E}_Q[\log P(\mathbf{X}, \mathbf{T})]$ having the form of a log likelihood function with the empirical distribution Q. The distribution is over all the variables, and hence sufficient statistics can be used. The E-step requires the following result, reported in (Elidan and Friedman, 2005).

Proposition 1 *Let $\mathcal{L}^+_{EM}$ be defined via $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ as in Equation (5). Assuming a naive Bayes approximation for $Q(\mathbf{T}|Y)$, a stationary point of $\mathcal{L}^+_{EM}$ satisfies the following equations for all i, t_i and y:*

$$Q(t_i|y) = \frac{1}{Z(i,y,\gamma)} Q(t_i)^{1-\gamma} e^{\gamma \mathbb{EP}(t_i,y)}, \tag{6}$$

where

$$Z(i,y,\gamma) = \sum_{t'_i} Q(t'_i)^{1-\gamma} e^{\gamma \mathbb{EP}(t'_i,y)} \tag{7}$$

is a normalizing constant, and

$$\mathbb{EP}(t_i, y) \equiv \mathbb{E}_{Q(\mathbf{T}|t_i,y)}[\log P(\mathbf{x}[y], \mathbf{T})]. \tag{8}$$

The parameter γ balances between compression of the data and fitness of the parameters to $\mathcal{G}_{out}$: for $\gamma = 1$, Equation (5) is equivalent to the objective function of the EM algorithm (Elidan and Friedman, 2005). The IB-EM can bypass local maxima of EM by varying γ in Equation (5) using a deterministic annealing strategy (Rose, 2002): γ is initially set to 0, where a single, easy to compute solution exists (see Figure 2 (a)) and then it is gradually incremented towards higher values, tracking the solution through various stages, hopefully bypassing local maxima by staying close to the optimal solution at each value of γ. The aim is to follow a smooth path from the trivial solution at $\gamma = 0$ to a good solution at $\gamma = 1$, see Figure 2 (b).

4.2 E-Step

The characterization of such paths may be done as follows. As reported in Proposition 1, when the gradient of $\mathcal{L}^+_{EM}$ is zero, Equation (6) holds for all t_i and y, and thus we

will consider paths where all of these equations hold. Taking a log of Equation (6), it is possible to define a set of functions G as

$$G_{t_i,y}(Q,\gamma) = -\log Q(t_i|y) + (1-\gamma)\log Q(t_i) + \gamma \mathbb{EP}(t_i,y) - \log Z(i,y,\gamma).$$

$G_{t_i,y}(Q,\gamma) = 0$ when Equation (6) holds for all t_i and y. The goal is hence to follow an equi-potential path where all $G_{t_i,y}(Q,\gamma)$ functions are zero starting from some small value of γ up to the desired solution at $\gamma = 1$. Starting from a point (Q_0, γ_0), where $G_{t_i,y}(Q_0,\gamma_0) = 0$ for all t_i and y, the aim is to move in a direction $\Delta = (dQ, d\gamma)$ s.t. $G_{t_i,y}(Q_0 + dQ, \gamma_0 + d\gamma) = 0$. Hence, one wants to find a direction Δ s.t.

$$\forall t_i, y \nabla_{Q,\gamma} G_{t_i,y}(Q_0,\gamma_0) \cdot \Delta^T = 0, \tag{9}$$

see Figure 2 (c), where $\nabla_{Q,\gamma} G_{t_i,y}(Q_0,\gamma_0)$ is the gradient of $G_{t_i,y}(Q_0,\gamma_0)$ with respect to the parameters $Q(t_i|y)$ and γ that results in a derivative matrix

$$H(Q,\gamma) = \left(\frac{\partial G_{t_j,y_1}(Q,\gamma)}{\partial Q(t_i|y_2)} | \frac{\partial G_{t_j,y}(Q,\gamma)}{\partial \gamma} \right),$$

for $i,j = 1,\ldots,n$, where n is the number of hidden variables and y_1 and y_2 are two values of Y. To find the direction Δ satisfying Equation (9) we have to satisfy the matrix equation

$$H(Q_0|\gamma_0)\Delta^T = 0. \tag{10}$$

The matrix H is of size $\prod_i |T_i| \times |Y| \times (\prod_i |T_i| \times |Y| + 1)$, so in practice we approximate it by a matrix that contains only the diagonal entries $\frac{\partial G_{t_i,y}(Q,\gamma)}{\partial Q(t_i|y)}$ and the last column $\frac{\partial G_{t_i,y}(Q,\gamma)}{\partial \gamma}$. Denoting the first expression with $h_{t_i,y}$ and the second with $h^{\gamma}_{t_i,y}$, we can write Equation (10) as a set of equations of the form

$$d_{t_i,y} h_{t_i,y} + d_\gamma h^{\gamma}_{t_i,y} = 0$$

where $d_{t_i,y}$ and d_γ are the elements of Δ. Since we are only interested in the direction of the step and not on the magnitude at the moment, we can use d_γ as a parameter and obtain

$$d_{t_i,y} = -\frac{h^{\gamma}_{t_i,y}}{h_{t_i,y}} d_\gamma.$$

In order to decide the size of the step, we can use a *continuation method* in the deterministic annealing strategy: we do not fix the size in advance but we let it vary in the process depending on the expected change of $\mathbf{I}_Q(\mathbf{T};Y)$ that represents a measure of the progress. We want to normalize the step so that, when $\mathbf{I}_Q(\mathbf{T};Y)$ is not sensitive to changes in the parameters, we proceed rapidly and, when it is sensitive, we proceed in small steps. Therefore, we rescale Δ in a way that produces a fixed increase ϵ of $\mathbf{I}_Q(\mathbf{T};Y)$, by choosing d_γ so that:

$$\nabla_{Q,\gamma} \mathbf{I}_Q(\mathbf{T};Y) \cdot \Delta^T = \epsilon.$$

Usually, a minimum and maximum values for the step are used in order to ensure that γ is always increased but not too much.

4.3 M-Step

In the M-step, the parameters of P can be obtained from counts of the following form:

$$\mathcal{N}(v, \mathbf{pa}_v) = \sum_y Q(y) Q((v\mathbf{pa}_v \cap \mathbf{T})|y) 1\{(v\mathbf{pa}_v \cap \mathbf{X})[y] = (v\mathbf{pa}_v \cap \mathbf{X})\} + \alpha(v, \mathbf{pa}_v)$$
$$\mathcal{N}(\mathbf{pa}_v) = \sum_v \mathcal{N}(v, \mathbf{pa}_v)$$

where v is a variable from $\mathbf{X} \cup \mathbf{T}$, $\mathbf{pa}_v$ are its parents in P, α are the hyper-parameters of the Dirichlet prior distribution, $1\{\}$ is the indicator function, the notation $\mathbf{V}[y]$ indicates the values of the variables of the set $\mathbf{V}$ in instance y and with $v\mathbf{V}$ we denote $\{v\} \cup \mathbf{V}$. The parameters of P can then be expressed as

$$\theta_{v|\mathbf{pa}_v} = \frac{\mathcal{N}(v, \mathbf{pa}_v)}{\mathcal{N}(\mathbf{pa}_v)}$$

for every variable V either observed or hidden.

Having expressed the parameters, we are thus able to compute the derivatives of $\log P(\mathbf{x}[y], \mathbf{t})$, and thus also of $\mathbb{EP}(i, y)$, that take the form

$$\frac{\partial \mathbb{EP}(i, y)}{\partial Q(t_i|y)} = Q(y) \mathbb{E}_{Q(\mathbf{T}|t_i, y)} \mathcal{D}(y, t_i, mb_i),$$

where mb_i are the values of the variables in the Markov blanket of t_i and $\mathcal{D}(y, t_i, mb_i)$ is a formula whose expression for a single hidden variable can be found in (Elidan and Friedman, 2005).

5 Relational Information Bottleneck

The IB approach can be applied to any statistical relational language that can be converted to Bayesian networks. In particular, it can be applied to those that follow a Knowledge Based Model Construction (KBMC) (Breese et al, 1994) approach: expressions in the language are a compact way of representing a number of Bayesian network portions. The network portions, when combined, produce a graphical model of the domain. Examples of languages that follow the KBMC approach are (Bacchus, 1993), Bayesian Logic Programs (BLP) (Kersting and De Raedt, 2001), CLP($\mathcal{BN}$) (Costa et al, 2003), or Relational Bayesian Networks (Jaeger, 1997).

For these languages, a formula of the language is a template for a set of families of variables: it compactly encodes the dependencies of a set of variables in the ground Bayesian network from their parents or further ancestors. Typically, a formula r encodes the fact that a template variable S depends on a set of template parents $\mathbf{Pa}_S$ with parameters $\theta_{s|\mathbf{pa}_s}$. A function i takes as input a template variable and returns the set of its instantiations, i.e. $i(S)$ contains the child variables in the families encoded by r. The parents of each variable of $i(S)$ are specified by $i(\mathbf{Pa}_S)$ and r usually specifies also the CPT, which is shared by all the families.

We further distinguish between Bayesian network-based SRL languages that naturally contain hidden variables such as those based on the distribution semantics (LPADs (Vennekens et al, 2004), ICL (Poole, 1997), ProbLog (De Raedt et al, 2007), PRISM (Sato, 1995)) and BLP with combining rules, from those that do not, such as (Bacchus, 1993), BLP without combining rules and CLP(BN). The hidden variables in the first

class of languages are used to model the result of probabilistic choices either for the head of rules (LPADs, BLP with combining rules) or for probabilistic facts (PRISM, ICL, ProbLog). The second class of languages instead models Bayesian network more directly, without introducing extra variables. Both classes of languages may also have some logical atom variables hidden in the data.

IB can be used for learning when both types of hidden variables are present, obtaining the Relational Information Bottleneck (RIB) algorithm. Let us call $\mathbf{CH}$ the set of choice variables and $\mathbf{T}$ the set of unseen atom variables. Moreover, let us consider a naive Bayes factorization for the Q distribution:

$$Q(\mathbf{CH}, \mathbf{T}|Y) = \prod_i Q(CH_i|Y) \prod_i Q(T_i|Y). \tag{11}$$

The function $\mathcal{L}_{EM}$ from Equation (4) becomes:

$$\mathcal{L}^+_{EM} = \mathbf{I}_Q(\mathbf{CH}, \mathbf{T}; Y) - \gamma(\mathbb{E}_Q[\log P(\mathbf{X}, \mathbf{CH}, \mathbf{T})] - \mathbb{E}_Q[\log Q(\mathbf{CH}, \mathbf{T})]) \tag{12}$$

where

$$\begin{aligned}\mathbf{I}_Q(\mathbf{CH}, \mathbf{T}; Y) = &-\mathbb{E}_Q[\log Q(\mathbf{CH}, \mathbf{T})] + \mathbb{E}_Q[\log Q(\mathbf{CH}, \mathbf{T}|Y)] = \\ &-\mathbb{E}_Q[\log Q(\mathbf{CH}, \mathbf{T})] + \sum_i \mathbb{E}_Q[\log Q(CH_i|Y)] + \sum_i \mathbb{E}_Q[\log Q(T_i|Y)]\end{aligned}$$

and

$$\mathbb{E}_Q[\log Q(\mathbf{CH}, \mathbf{T})] \approx \sum_i \mathbb{E}_Q[\log Q(CH_i)] + \sum_i \mathbb{E}_Q[\log Q(T_i)].$$

following the approximation used in (Elidan and Friedman, 2005). Thus the objective function takes the following form

$$\begin{aligned}\mathcal{L}^+_{EM} = &\mathbb{E}_Q[\log Q(\mathbf{CH}, \mathbf{T}|Y)] - \gamma\mathbb{E}_Q[\log P(\mathbf{X}, \mathbf{CH}, \mathbf{T})] + (\gamma - 1)\mathbb{E}_Q[\log Q(\mathbf{CH}, \mathbf{T})]) = \\ &\sum_i \mathbb{E}_Q[\log Q(CH_i|Y)] + \sum_i \mathbb{E}_Q[\log Q(T_i|Y)] - \gamma\mathbb{E}_Q[\log P(\mathbf{X}, \mathbf{CH}, \mathbf{T})] + \\ &(\gamma - 1) \sum_i \mathbb{E}_Q[\log Q(ch_i)] + (\gamma - 1) \sum_i \mathbb{E}_Q[\log Q(t_i)]\end{aligned}$$

As for the classical IB approach, the aim is to bypass local maxima, following a smooth path from the trivial solution at $\gamma = 0$ to a good solution $\gamma = 1$.

In the case of Bayesian network-based languages, the G functions are

$$\begin{aligned}G_{ch_k,y}(Q, \gamma) = &-\log Q(ch_k|y) + (1 - \gamma) \log Q(ch_k) \\ &+\gamma\mathbb{EP}(ch_k, y) - \log Z_{\mathbf{CH}}(k, y, \gamma)\end{aligned} \tag{13}$$

$$\begin{aligned}G_{t_k,y}(Q, \gamma) = &-\log Q(t_k|y) + (1 - \gamma) \log Q(t_k) + \\ &\gamma\mathbb{EP}(t_k, y) - \log Z_{\mathbf{T}}(k, y, \gamma)\end{aligned} \tag{14}$$

where $\mathbb{EP}(ch_k, y)$, $\mathbb{EP}(t_k, y)$, $Z_{\mathbf{CH}}(k, y, \gamma)$ and $Z_{\mathbf{T}}(k, y, \gamma)$ have expressions analogous to Equations (8) and (7). We want to compute the derivatives of $G_{ch_i,y}(Q, \gamma)$ and $G_{t_i,y}(Q, \gamma)$ for all ch_i, t_i and y with respect to the parameters and γ and then use the orthogonal direction as the update step.

The P distribution that is needed in order to compute the derivatives is determined by the ground Bayesian network but we must ensure that the parameters of families

that are instantiations of the same rule r are "tied", i.e. they are the same in the different families. To do so, in the maximization phase, the parameters have to be computed by taking into account the counts for all the families that are instantiation of the same rule

$$\theta_{s|\mathbf{pa}_s} = \frac{\mathcal{N}(s, \mathbf{pa}_s)}{\mathcal{N}(\mathbf{pa}_s)} = \frac{\sum_{v \in i(s)} \mathcal{N}(v, \mathbf{pa}_v)}{\sum_{v \in i(s)} \mathcal{N}(\mathbf{pa}_v)}.$$

In general, each propositional variable V will be observed or hidden separately from each other. In practice, most often all the instances of the same template variable will share the same condition. The values of these parameters are then used in the formulas of the derivatives.

6 Distribution Semantics

The distribution semantics (Sato, 1995) is shared my many languages, including ICL, PRISM, LPADs and ProbLog. A program in one of these languages defines a probability distribution over normal logic programs called *worlds*. This distribution is then extended to queries and the probability of a query is obtained by marginalizing the joint distribution of the query and the programs.

If the program does not contain function symbols, the set of worlds W is finite, otherwise it is infinite. The semantics has been defined for both cases, we review here the case of no function symbols for simplicity. Let us call $P(W)$ the distribution over worlds. The probability of a query Q given a world w is $P(Q|w) = 1$ if $w \models Q$ and 0 otherwise, where $\models$ is truth in the well-founded model (Van Gelder et al, 1991). Thus the probability of a query Q is given by

$$P(Q) = \sum_{w \in W} P(Q, w) = \sum_{w \in W} P(Q|w)P(w) = \sum_{w \in W : w \models Q} P(w) \tag{15}$$

The languages following the distribution semantics differ in the way they define the probability distribution over worlds. In ICL and PRISM the facts of a program may be probabilistic statements that specify sets of atoms together with their probability of being selected. A world is then obtained from the union of the normal rules together with one fact selected from every grounding of each probabilistic statement. The probability of a world is obtained by multiplying the probabilities of selecting the facts from probabilistic statements because these are assumed to be independent from each other. In ProbLog, the probabilistic facts are required to have only two alternatives: an atom and a dummy atom that does not appear in the body of any clause. (De Raedt et al, 2008) showed that ICL and PRISM can be converted to ProbLog: a probabilistic statement with n alternatives is encoded with a set of probabilistic ProbLog facts with the appropriate probabilities. The opposite transformation is straightforward since a ProbLog program is a valid ICL/PRISM program.

In Logic Program with Annotated Disjunctions (Vennekens et al, 2004) the alternatives are encoded in the head of clauses in the form of a disjunction in which each atom is annotated with a probability. Each groundind of an annotated disjunctive clause represents a probabilistic choice between a number of ground normal clauses. By choosing a head atom for each grounding of each clause of an LPAD we get a world. The probability of the world is given by the product of the annotations of the atoms

selected. An ICL, PRISM or ProbLog program is a valid LPAD, since the clauses can have an empty body. (Vennekens and Verbaeten, 2003) showed that LPADs can be translated to ICL, so all these languages have the same expressive power.

Formally, an LPAD L consists of a finite set of formulas of the form

$$\mathtt{H_1} : \theta_1 \vee \mathtt{H_2} : \theta_2 \vee \ldots \vee \mathtt{H_n} : \theta_n \leftarrow \mathtt{B_1}, \mathtt{B_2}, \ldots \mathtt{B_m},$$

called *annotated disjunctive clauses*. In such a clause the $\mathtt{H_i}$ are logical atoms, the $\mathtt{B_i}$ are logical literals and the θ_i are real numbers in the interval $[0, 1]$ such that $\sum_{i=1}^{n} \theta_i \leq 1$. The head of the clause implicitly contains an extra dummy atom `none` whose annotation is $1 - \sum_{i=1}^{n} \theta_i$. For a rule r of the form reported above, we define $H(r, i)$ as $\mathtt{H_i}$ and $\theta(r, i)$ as θ_i. Let $H_B(L)$ be the Herbrand base of L.

A world is identified by means of a selection function. Let L be an LPAD: a *selection* (Vennekens et al, 2004) σ is a function which selects one pair $(\mathtt{H}\delta : \theta)$ from each grounding $r\delta$ of each rule r of L where δ is a substitution grounding r: $\sigma(r\delta) \in head(r)\delta \cup \{\mathtt{none} : 1 - \sum_{i=1}^{n} \theta_i\}$. For each ground rule $r\delta$, we denote the selected atom $\mathtt{H}$ by $\sigma_{atom}(r\delta)$ and the selected probability θ by $\sigma_{prob}(r\delta)$. Let σ be a selection and $g(L)$ be the grounding of L: the *world* L_σ *chosen by* σ is obtained by keeping only the atom selected for c in the head of each rule $c \in g(L)$, i.e., $L_\sigma = \{\text{“}\sigma_{atom}(c) \leftarrow body(c)\text{”} | c \in g(L)\}$.

The *probability of a world* L_σ is the product of the probabilities of the individual choices made by the corresponding selection, i.e. $P(L_\sigma) = \prod_{c \in g(L)} \sigma_{prob}(c)$. We assume that each world has a total model according to the well-founded semantics. The probability of a query Q is then given by Equation (15) where w is replaced by L_σ.

Example 1 Let us see an example of an LPAD.

```
earthquake(X, strong) : 0.3 ∨ earthquake(X, moderate) : 0.5 ←
    fault_rupture(X).
earthquake(X, strong) : 0.2 ∨ earthquake(X, moderate) : 0.6 ←
    volcanic_eruption(X).
fault_rupture(stromboli).
volcanic_eruption(stromboli).
volcanic_eruption(eyjafjallajkull).
```

This program models the occurrence of earthquakes depending on its possible causes. In particular, if an earthquake at a site `X` is caused only by the rupture of a geological fault, we have a strong earthquake with probability 0.3, a moderate earthquake with probability 0.5 and no earthquake with probability $1 - 0.3 - 0.5 = 0.2$. In other words, if only one cause happens, the probability of the effect is given by the parameter in the head. If more than one cause happens, the probabilities of the effect are combined with the noisy-or relation. For example, the probability of $earthquake(stromboli, strong)$ is given by $1 - (1 - 0.3) \cdot (1 - 0.2) = 0.44$. □

To compute the probability of a query given a set of atoms, one needs to use an inference algorithm such as (De Raedt et al, 2007; Riguzzi, 2007b, 2008b, 2010; Meert et al, 2010; Riguzzi and Swift, 2010).

Let us now define the acyclic property for LPADs, extending the definition of (Apt and Bezem, 1991) for normal logic programs. An LPAD is *acyclic* if an integer level can be assigned to each ground atom so that the level of each atom in the head of each ground rule is the same and is higher than the level of each atom in the body.

An acyclic LPAD L can be translated into a Bayesian network $\beta(L)$ (Vennekens et al, 2004). $\beta(L)$ is built by associating each atom $\mathtt{A}$ in $H_B(L)$ with a binary variable A with values *true* (1) and *false* (0). Moreover, for each rule c_i of the following form

$\mathtt{H_1} : \theta_1 \vee \ldots \vee \mathtt{H_n} : \theta_n \leftarrow \mathtt{B_1}, \ldots \mathtt{B_m}, \neg \mathtt{C_1}, \ldots, \neg \mathtt{C_l}$

in $g(L)$ we add to $\beta(L)$ a new variable CH_i (for "choice for rule c_i") that has $B_1, \ldots, B_m$, $C_1, \ldots, C_l$ as parents and has the values h_1, ..., h_n and *none*, corresponding respectively to atoms $\mathtt{H_1}$, ..., $\mathtt{H_n}$ and $\mathtt{none}$. The CPT of CH_i is

	...	$B_1 = 1, \ldots, B_m = 1, C_1 = 0, \ldots, C_l = 0$	...
$CH_i = h_1$	0.0	θ_1	0.0
...			
$CH_n = h_n$	0.0	θ_n	0.0
$CH_i = none$	1.0	$1 - \sum_{i=1}^{n} \theta_i$	1.0

Moreover, each variable A corresponding to the atom $\mathtt{A} \in H_B(L)$ has as parents all the variables CH_i of rules c_i that have $\mathtt{A}$ in the head. The CPT for A is the following:

	at least one parent equal to A	remaining columns
$A = 1$	1.0	0.0
$A = 0$	0.0	1.0

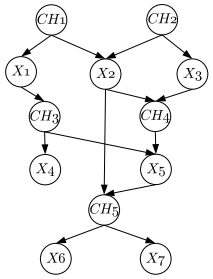

Fig. 3 Bayesian network

Note that in order to convert an LPAD containing variables into a Bayesian network, its grounding must be generated.

Example 2 Consider the following LPAD L:

$r_1 = \mathtt{x1} : 0.4 \vee \mathtt{x2} : 0.3.$

$r_2 = \mathtt{x2} : 0.1 \vee \mathtt{x3} : 0.2.$

$r_3 = \mathtt{x4} : 0.6 \vee \mathtt{x5} : 0.4 \leftarrow \mathtt{x1}.$

$r_4 = \mathtt{x5} : 0.4 \leftarrow \mathtt{x2}, \mathtt{x3}.$

$r_5 = \mathtt{x6} : 0.3 \vee \mathtt{x7} : 0.2 \leftarrow \mathtt{x2}, \mathtt{x5}.$

Its corresponding network is shown in Figure 3. □

7 RIB for LPADs

In order to apply RIB to LPADs, the network $\mathcal{G}_{out}$ is the result of the translation of the LPAD for which we want to learn the parameters plus the addition of the Y variable.

We consider the case in which the data available for training is a set of interpretations, i.e., a set of subsets of $H_B(L)$. Thus the data contain the truth values of atoms

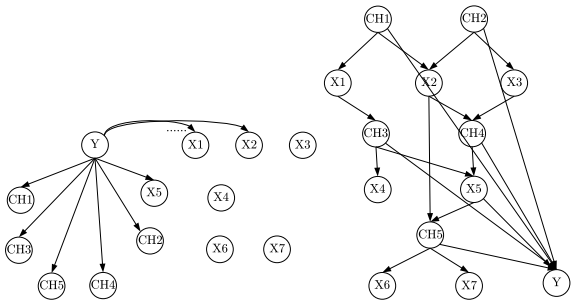

Fig. 4 $G_{in} = Q$ (left) and $G_{out} = P$ (right)

from $H_B(L)$ but not of the choice variables. Moreover, some of the atoms may be unobserved as well. Thus, the set of hidden variables contains the vector of the choice variables **CH** plus the unobserved atoms, **T**. With **X** we indicate the set of atom variables that are observed in the data. For the $\mathcal{G}_{in}$ network, we consider a naïve Bayes factorization as in Equation (11). In $\mathcal{G}_{out}$, the choice and the unobserved variables are the only parents of Y. In fact, given the choice and the unobserved variables, the **X** variables are uniquely determined, and so is the instance identity (assuming there are no duplicate examples, but this can be modeled by assigning them a different prior probability $Q(Y)$).

Consider the LPAD L reported in Example 2. Moreover, suppose that x_5 is unseen in the data. The networks $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ for this LPAD are shown in Figure 4. According to IB, the chosen Q distribution must be such that unobserved variables are independent of observed ones given Y. This requirement is satisfied by $\mathcal{G}_{in}$ in Figure 4. Regarding P, **CH** and **T** must be the only parents of Y, which is true in $\mathcal{G}_{out}$.

Once the equivalent Bayesian networks $\mathcal{G}_{in}$ and $\mathcal{G}_{out}$ are created, we need to compute the update direction Δ. The G functions are given by Equations (13) and (14). We need to compute the derivatives of $G_{ch_i,y}(Q,\gamma)$ and $G_{t_i,y}(Q,\gamma)$ for all ch_i, t_i and y. In the following, we will present the main results and point to (Riguzzi and Di Mauro, 2010) for the proofs.

Let us first express the parameters of P. Note that, if we want to be able to translate the learned network back to an LPAD, some of the parameters are fixed in advance: those belonging to the CPTs for atoms and those in the rows corresponding to a false body in the CPTs for choice variables. So we must minimize the objective function by varying only a subset of the parameters. Moreover, some parameters are "tied": all the choice variables that refer to ground rules obtained from the same non-ground rule share the same parameters.

Let us indicate with $\theta_{x_j|\mathbf{pa}_{X_j}}$ the parameters of the CPT for atom $\mathtt{X_j}$. Thus $\theta_{X_j=1|\mathbf{pa}_{X_j}} = 1$ if the atom $\mathtt{X_j}$ is among the values $\mathbf{pa}_{X_j}$ of its parents, and 0 otherwise. For a non-ground rule r, let $\theta_{HD_r=hd_r|body_r}$ or simply $\theta_{hd_r|body_r}$ be the probability that the head hd_r is selected given that the body has truth value $body_r$. Thus $\theta_{hd_r|false} = 1$ if $hd_r = none$.

Moreover, let $i(r)$ be the set of indexes k instances c_k of r. Given the body $\mathbf{pa}_{CH_k}$ of the instantiated rule c_k, let $bt(\mathbf{pa}_{CH_k})$ be 1 if the observed variables in $\mathbf{pa}_{CH_k}$ do not make the body false, and 0 otherwise. Let $tb(\mathbf{pa}_{CH_k})$ be a set of values for the unobserved variables that are parents of CH_k and that do not make the body false.

The maximum likelihood parameters of the distribution of HD_r in the case of a true body are

$$\theta_{hd_r|true} = \frac{\mathcal{N}(r, hd_r) + \alpha(r, hd_r, true)}{\mathcal{N}(r) + \alpha(r, true)}$$
$$\mathcal{N}(r, hd_r) = \sum_{k \in i(r)} \sum_{y} Q(y) Q(CH_k = hd_r|y) bt(\mathbf{pa}_{ch_k}[y]) \prod_{t_j \in tb(\mathbf{pa}_{ch_k})} Q(t_j|y)$$
$$\mathcal{N}(r) = \sum_{hd_r} \mathcal{N}(r, hd_r)$$

where $\alpha()$ are the hyper-parameters of the Dirichlet prior distribution, and $\mathcal{N}$ is used to denote the total counts used for estimation.

The expressions of the derivatives of the G functions with respect to $Q(ch_i|y)$, $Q(t_i|y)$ and γ, together with the derivatives of $\mathbf{I}_Q(\mathbf{CH}, \mathbf{T}; Y)$, necessary to compute the update step, are reported in Appendix A. Note that none of these expressions require inference in the underlying Bayesian network.

8 Related Works

Learning statistical relational models containing hidden variables has been tackled using either a variant of the EM algorithm or by gradient descent methods. With the EM algorithm, the objective function $\mathcal{L}^{(2)}[Q, P]$ with $\gamma = 1$ (Equation (3) is optimized, while with gradient descent methods the objective function is either the likelihood or the mean squared error of the probability of a set of queries. In gradient descent methods, the partial derivatives of the objective function with respect to the parameters are computed and the parameters are updated accordingly.

Since the languages PRISM, ICL, LPADs and ProbLog are equally expressive and one can be translated into another, a system for learning one of these languages can be almost directly be used for the others. The PRISM system (Sato, 1995; Sato and Kameya, 2001) includes one of the first learning algorithms based on EM. The system however makes strong assumptions on the allowed program for inference and learning to work correctly: the probability of a conjunction (A, B) is computed as the product of the probabilities of A and B (independence assumption) and the probability of a disjunction $(A; B)$ is computed as the sum of the probabilities of A and B (exclusiveness assumption). The latter condition in particular requires the body of ground clauses with the same atom in the head to have mutually exclusive bodies. These assumption significantly simplify the inference and learning problems. Differently from PRISM, RIB does not make any assumption on the form of the programs.

These assumptions can be related to the framework of RIB by observing that, when all the logical atoms are observed, the values of the logical atoms given the instance identity are completely determined and so is the truth of clauses' bodies. If the clauses have mutually exclusive bodies then, for each atom, there exists at most one clause with the body true and the atom in the head, so the Q distribution will assign mass 1 to the value of the choice variable associated to the atom. So the E step in RIB in this case would be strongly simplified and the naive Bayes assumption would be true. If some atom variables are unobserved then the instance identity does not determine anymore the truth of clauses' bodies and so the distribution of a choice variable depends on

that of its ancestor choice variables, even if the clauses are disjoint. In these cases, the naive Bayes assumption is only an approximation. The experiment comparing RIB with PRISM in the next section contains unseen atom variables so RIB can not exploit the PRISM modeling assumptions. If all the atom variables are observed and the clauses do not have mutually exclusive bodies, then two choice variables will be dependent given the instance identity if the corresponding ground rules share a ground atom in the head. In this case the naïve Bayes assumption is an approximation as well.

Another work that exploits the EM algorithm is (Koller and Pfeffer, 1997). The authors use EM to learn the structure of *first-order probabilistic logic* (FOPL) rules (first-order rules with associated probabilistic uncertainty parameters). The probabilistic uncertainty parameters of the rules are adaptively learned using an extension of standard EM that deals with an ensemble of networks of varying structure, and in which the same parameters can appear several times. In this approach, a set of Horn rules, with an associated uncertainty parameter, describes the ways in which first-order atoms influence each other. Since more than one set of conditions can cause an atom to be true, the authors use combining rules to indicate how the different possible cause interact. The EM algorithm they propose is able to learn in presence of these combining rules and missing data. In the distribution semantics multiple causes for the truth of an atom are combined with a noisy-or rule, so the system of (Koller and Pfeffer, 1997) can be used for these languages and RIB can be used for FOPL when the combining rule is noisy-or.

The work of Koller and Pfeffer (1997) has been generalized and extended in (Natarajan et al, 2005) to multi-level combining rules, where the first level combines the influences due to different ground instances of the same statement, and the second level combines the influences due to different statements. The authors propose a language consisting of quantified conditional influence statements. This language captures most relational probabilistic models based on directed graphs. Examples of combining rules are mean, weighted-mean or noisy-or. The algorithms for parameter learning in the presence of such combining rules are based on gradient descent and EM. Again, in RIB we consider only the case where the combining rule for both levels is nosiy-or.

(Jaeger, 2007) considered a weighted combination or a nested combination of the combining rules and used a gradient descent algorithm for optimizing the objective function. The proposed technique focuses on the formalism of Relational Bayesian Networks (RBNs) but can also be applied to Probabilistic Relational Models (PRMs) or to Bayesian Logic Programs (BLPs). The approach compiles the RBN model into a computation graph for the likelihood function and uses this graph to perform the necessary computations for a gradient descent likelihood optimization procedure. Adopting the likelihood graph greatly reduces the number of computations needed for the gradient computation.

LeProbLog (Gutmann et al, 2008, 2010) is another method that tackles the problem of learning the parameters of models with hidden variables by using gradient descent. It starts from a set of queries annotated with a probability and from a ProbLog program. It tries to find the values of the parameters of the program that minimize the mean squared error of the probability of the queries. LeProbLog applies a gradient descent directly to the Binary Decision Diagrams that represent the queries.

As regards previous approaches specific to LPADs, the works (Blockeel and Meert, 2007; Meert et al, 2007, 2008) apply the EM algorithm directly to them. (Blockeel and Meert, 2007) first syntactically transforms an LPAD into another LPAD that has a direct correspondence with a Bayesian network. The resulting network is different

from the one of Section 6. Blockeel and Meert then use EM for learning the parameters of the network and to learn the structure by performing greedy search. (Meert et al, 2007, 2008) expand on the application of EM to LPADs and present the first results on an artificial dataset, on which the authors obtained superior performance with respect to a Bayesian network without hidden variables trained with Structural EM.

RIB is most similar to the EM approaches, as its final objective function is the same as EM and it uses an iterative optimization approach as EM. Gradient descent methods instead differ in the optimization method and, in some cases, also on the objective function (e.g. LeProbLog). In the following section we compare the performances RIB with the systems based on EM by using an implementation of it based on `cplint` (Riguzzi, 2007b) and with the gradient descent systems by using LeProbLog.

Another system for LPADs is ALLPAD (Riguzzi, 2008a, 2007a) that learns a restricted set of ground LPADs considering a constraint satisfaction approach. The programs are such that each couple of clauses that share an atom in the head have mutually exclusive bodies. ALLPAD learns this class of LPADs by finding all the clauses satisfying certain properties, by estimating the parameters for each of them and then by solving a mixed integer programming problem for identifying the subset of clauses to be included in a solution.

9 Experiments

We tested RIB on two real world datasets, IMDB[1] (Mihalkova and Mooney, 2007) and Cora[2] (Singla and Domingos, 2005), and on some synthetic datasets.

On IMDB and Cora we compare the performances of RIB with that of EM, LeProbLog[3] and Alchemy[4]. We implemented both RIB and EM in Yap Prolog[5]. IMDB regards movies, actors, directors and movie genres. It is divided into five mega-examples, each containing all the information regarding four movies. It contains 10 predicates and 316 constants divided into 4 types. The number of possible ground atoms is 32,615, of which 1,540 are true.

We used a methodology similar to the one in (Mihalkova and Mooney, 2007): we train on four mega-examples and test on the remaining one. Then we draw a Precision-Recall curve and we compute the Area Under the Curve (AUCPR) using the method reported in (Davis and Goadrich, 2006).

We choose the following LPAD that predicts the value of the target predicate `sameperson/2`.

```
sameperson(X,Y):t :- movie(M,X), movie(M,Y).
sameperson(X,Y):t :- actor(X), actor(Y),
    workedunder(X,Z), workedunder(Y,Z).
sameperson(X,Y):t :- gender(X,Z),gender(Y,Z).
sameperson(X,Y):t :- director(X), director(Y), genre(X,Z), genre(Y,Z).
```

[1] `http://alchemy.cs.washington.edu/data/imdb`

[2] `http://alchemy.cs.washington.edu/data/cora`

[3] `http://dtai.cs.kuleuven.be/problog/`, we used the version of LeProbLog included in the git version of Yap downloaded as of the 1st of September 2010.

[4] `http://alchemy.cs.washington.edu/`, we used the CVS version of Alchemy downloaded as of the 16th of May 2010.

[5] `http://www.dcc.fc.up.pt/~vsc/Yap/`

This theory has a direct translation to ProbLog:

```
sameperson(X,Y) :- movie(M,X),movie(M,Y),f1(X,Y,M).
sameperson(X,Y) :- actor(X), actor(Y),
  workedunder(X,Z),workedunder(Y,Z), f2(X,Y,Z).
sameperson(X,Y) :- gender(X,Z), gender(Y,Z), f3(X,Y,Z).
sameperson(X,Y) :- director(X), director(Y),
  genre(X,Z), genre(Y,Z), f4(X,Y,Z).
```

where `t` stands for a "tunable" parameter and `f1/3`, `f2/3`, `f3/3` and `f4/3` are non-ground probabilistic facts whose parameters are learned with LeProbLog.

Each mega-example has a different Herbrand base: it contains different constants and thus many random variables would appear only in a single example. This may cause a problem to RIB that exploits the dependency of random variables from individual examples. Thus each fold has been divided into smaller examples on the basis of the target predicate: each of the smaller examples is an interpretation that refers to an instance `sameperson(s,p)` and contains the facts for all the other predicates where the constants `s` and `p` appear. We have one positive example for each fact that is true in the data, while we sampled from the complete set of false facts three times the number of true instances in order to generate negative examples. In the small interpretations, the constants from the mega-examples are replaced by dummy constants that are the same for all the small examples.

In order to build the $\mathcal{G}_{out}$ network, a derivation for each atom for `sameperson/2` is built in each interpretation. The clauses and the atoms used in the derivation act as a guide for generating $\mathcal{G}_{out}$: only the network portion relevant to the query is built.

The annotated queries that LeProbLog takes as input are obtained by annotating with 1.0 each positive example for `sameperson/2` and with 0.0 each negative examples for `sameperson/2` obtained by random sampling.

To compare our results with Alchemy, we translated the above theory into an MLN following an approach similar to the one used in (Gutmann et al, 2010) to convert a MLN into ProbLog. Each LPAD clause `h : t :- b1,...,bm` was translated into the MLN clause `h v !b1 v ... v !bm`. This theory is not semantically equivalent to the LPAD/ProbLog one but each exploits the features of the languages.

Then RIB, EM, LeProbLog and Alchemy were used to learn the parameters of the two theories. For RIB we used a minimum of 0.005 and a maximum of 0.1 for the γ update step and 0.01 for ϵ, the fraction of $\mathbf{I}_Q(\mathbf{T};Y)$ for step rescaling. We run LeProbLog for a maximum of 100 iterations or until the difference in Mean Squared Error (MSE) between two iterations gets smaller than 10^{-5}. We used the preconditioned rescaled conjugate gradient discriminative algorithm (Lowd and Domingos, 2007) for Alchemy and we specified `sameperson/2` as the only non-evidence predicates.

Table 1 shows the area under the precision-recall curve averaged over the five folds together with the standard deviation for RIB, LeProbLog, EM and Alchemy. The p column shows the p-value of a two-tailed paired t-test of the significance of the difference in AUCPR between RIB and LeProbLog/EM/Alchemy. Table 2 shows the learning time in hours.

In order to investigate the performance of the algorithms when some atoms are unseen, we also trained the following theory on the IMDB dataset:

```
sameperson_pos(X,Y):t :- movie(M,X), movie(M,Y).
sameperson_pos(X,Y):t :- actor(X), actor(Y),
```

```
   workedunder(X,Z), workedunder(Y,Z).
sameperson_pos(X,Y):t :- director(X),director(Y),genre(X,Z),genre(Y,Z).
sameperson_pos(X,Y):t :- movie(M,X), movie(M,Y).
sameperson_neg(X,Y):t :- movie(M,X), movie(M,Y).
sameperson_neg(X,Y):t :- actor(X), actor(Y),
   workedunder(X,Z), workedunder(Y,Z).
sameperson_neg(X,Y):t :- director(X),director(Y),genre(X,Z),genre(Y,Z).
sameperson_neg(X,Y):t :- movie(M,X), movie(M,Y).
sameperson(X,Y):t :- \+ sameperson_pos(X,Y), sameperson_neg(X,Y).
sameperson(X,Y):t :- \+ sameperson_pos(X,Y), \+ sameperson_neg(X,Y).
sameperson(X,Y):t :- sameperson_pos(X,Y), sameperson_neg(X,Y).
sameperson(X,Y):t :- sameperson_pos(X,Y), \+ sameperson_neg(X,Y).
```

The `sameperson_pos/2` and `sameperson_neg/2` predicates are unseen in the data. In this experiment Alchemy was run with the `-withEM` option that turns on EM learning. The other parameters for Alchemy and for the other algorithms are set as for IMDB. The results of this experiments are shown in Table 1 in the IMDBu row.

The Cora database contains 1295 different citations to 132 different computer science research papers. For each citation, we have information about the title, the authors and the venue. For each title, author and venue, we know which words appear in them. The task is to deduplicate citations, i.e., to predict the predicate `samebib(cit1,cit2)`. The database contains also facts for the predicates `sameauthor(aut1,aut2)`, `sametitle(tit1,tit2)` and `samevenue(ven1,ven2)`, together with facts for `haswordauthor(aut,wor)`, `haswordtitle(tit,wor)` and `haswordvenue(ven,wor)`.

We took the MLN proposed in (Singla and Domingos, 2006)[6] and we removed the transitive closure rules because they would introduce cycles in the LPAD, obtaining the MLN

```
!SameBib(b1,b2)
!SameAuthor(a1,a2)
!SameTitle(t1,t2)
!SameVenue(v1,v2)
Author(bc1,a1) ^ Author(bc2,a2) ^ SameAuthor(a1,a2) => SameBib(bc1,bc2)
Title(bc1,t1) ^ Title(bc2,t2) ^ SameTitle(t1,t2) => SameBib(bc1,bc2)
Venue(bc1,v1) ^ Venue(bc2,v2) ^ SameVenue(v1,v2) => SameBib(bc1,bc2)
HasWordAuthor(a1, +w) ^ HasWordAuthor(a2, +w) => SameAuthor(a1, a2)
!HasWordAuthor(a1, +w) ^ HasWordAuthor(a2, +w) => SameAuthor(a1, a2)
HasWordAuthor(a1, +w) ^ !HasWordAuthor(a2, +w) => SameAuthor(a1, a2)
HasWordTitle(a1, +w) ^ HasWordTitle(a2, +w) => SameTitle(a1, a2)
!HasWordTitle(a1, +w) ^ HasWordTitle(a2, +w) => SameTitle(a1, a2)
HasWordTitle(a1, +w) ^ !HasWordTitle(a2, +w) => SameTitle(a1, a2)
HasWordVenue(a1, +w) ^ HasWordVenue(a2, +w) => SameVenue(a1, a2)
!HasWordVenue(a1, +w) ^ HasWordVenue(a2, +w) => SameVenue(a1, a2)
HasWordVenue(a1, +w) ^ !HasWordVenue(a2, +w) => SameVenue(a1, a2)
```

For RIB, we used the following LPAD

```
samebib(B,C):t :- author(B,D), author(C,E), sameauthor(D,E).
samebib(B,C):t :- title(B,D), title(C,E), sametitle(D,E).
```

[6] Available at `http://alchemy.cs.washington.edu/mlns/er/`.

Table 1 Results of the experiments on the IMDB and Cora datasets. IMDBu refers to the IMDB dataset with the theory containing unseen predicates. AUCPR is the area under the precision-recall curve averaged over the five folds together with the standard deviation and p is the p-value of a paired two-tailed t-test (significant differences at the 5% level in bold). R is RIB, L is LeProbLog, E is EM and A is Alchemy.

	AUCPR				p		
	R	L	E	A	R-L	R-E	R-A
IMDB	0.199±0.036	0.096±0.039	0.202±0.042	0.107±0.022	**0.009**	0.545	**0.011**
IMDBu	0.166±0.027	0.134±0.041	0.120±0.018	0.020±0.005	0.321	**0.012**	**0.000**
Cora	0.939±0.021	0.905±0.074	0.995±0.004	0.469±0.203	0.535	0.073	**0.016**

Table 2 Execution time in hours of the experiments on the IMDB and Cora datasets. R is RIB, L is LeProbLog, E is EM and A is Alchemy.

Dataset	Time (h)			
	R	L	E	A
IMDB	0.02	0.35	0.01	1.54
IMDBu	0.01	0.23	0.01	1.54
Cora	2.49	13.25	1.11	1.30

```
samebib(B,C):t :- venue(B,D), venue(C,E), samevenue(D,E).
samevenue(B,C):t :- haswordvenue(B,word_06), haswordvenue(C,word_06).
...
sametitle(B,C):t :- haswordtitle(B,word_10), haswordtitle(C,word_10).
....
sameauthor(B,C):t :- haswordauthor(B,word_a), haswordauthor(C,word_a).
.....
```

where the dots stand for the rules for all the possible words. Overall, the LPAD contains 559 rules.

The Cora dataset contains five mega-example that have been converted as for IMDB by having a separate interpretation for each `samebib/2`, `samevenue/2`, `sametitle/2` and `sameauthor/2` fact. Moreover, we separately learned the definitions with RIB for the four predicates and we combined the resulting theories at the end. The $\mathcal{G}_{out}$ network is built in a way similar to IMDB. When building the network for `samebib/2`, we consider the atoms for `samevenue/2`, `sametitle/2` and `sameauthor/2` as given. The parameters of RIB were 0.01 for ϵ, 0.08 for the minimum step and 0.1 for the maximum step.

We separately learned the various predicates with LeProbLog as well because learning the whole theory at once would give a lack of memory error on our machines. LeProbLog was run for a maximum of 100 iterations or until the difference in MSE gets smaller than 10^{-5}.

For Alchemy, we used the preconditioned rescaled conjugate gradient discriminative training algorithm specifying `samebib/2`, `samevenue/2`, `sametitle/2` and `sameauthor/2` as the only non-evidence predicates and using the defaults for the other parameters.

Table 1 shows the AUCPR obtained by training on four mega-examples and testing on the remaining one while Table 2 shows the running times of the four systems.

From Table 1 we can observe that RIB achieves a higher AUCPR than both LeProbLog and Alchemy in all three experiments. The difference between RIB and Alchemy is significant at the 5% level in all cases, while the difference between RIB and LeProbLog is statistically significant at the 5% level only on IMDB, even if it is nearly significant on IMBDu. The AUCPR of RIB is lower than that of EM in IMDB and Cora, even if the difference is not statistically significant at the 5% level in all datasets.

On IMDBu, instead, RIB achieves a significantly higher AUCPR than EM. The average training time of RIB is lower than that of LeProbLog in all datasets, lower than that of Alchemy and nearly equal to the one of EM on IMDB and IMBDu, while it is higher than that of Alchemy and EM on Cora. We also tried Alchemy on Cora with a maximum number of iterations of 20000 and a time limit of 20 hours but, despite a longer training time (8.91 hours on average), we got a worse AUCPR (0.2904). When unseen atoms are present, all algorithm do slightly worse than when all atoms are observed, except LeProbLog that surprisingly improves its AUCPR. The improvement of RIB with respect to EM can be explained by the general improvement of IB over EM shown in (Elidan and Friedman, 2005) when hidden variables are present: EM can get trapped in local maxima when to few information is present. Learning in the presence of unseen atoms seems to be still a difficult problem for Alchemy, even with the `-withEM` option. The approach of LeProbLog, based on gradient descent rather than EM, seem to exploit better the degrees of freedom introduced by unseen atoms, but it still does not overcome RIB.

We also consider cases that should be more favorable to RIB, i.e., experiments in which the example interpretations share the same Herbrand base and thus examples do not have to be split up.

We obtain synthetic datasets by writing some LPADs and by generating training sets from them. Four different LPADs have been considered: one of them is the shop example taken from (Meert et al, 2008) containing 4 rules (shop4), while the others have been obtained by progressively extending that example to 8 (shop8), 10 (shop10) and 12 (shop12) rules respectively. For each of these LPADs, all the possible interpretations with a non-zero probability have been generated and inserted into the training set. The probability of each interpretation was taken into account during learning by setting $Q(y)$ to that value, which is equivalent to having different cardinalities for the different interpretations.

We also run experiments with the shop4 and shop12 theories in which the training set was composed respectively by 1,000 and 10,000 examples (shop4s and shop12s) obtained from the theories by random sampling.

Moreover, we consider two further datasets (shop12u1 and shop12u2) in which we deleted respectively one and two ground atoms from the input interpretations, in order to test the performances of RIB and EM when learning from data that contain unseen atoms.

We compare RIB with EM, the best performing algorithm according to the IMDB and Cora experiments. For RIB we used a value for ϵ of 0.01, a minimum step of 0.02 and a maximum step of 0.05.

The results are evaluated according to various metrics. The first is the Mean Absolute Difference (MAD) between the learned parameters and the true ones computed as $MAD = n^{-1} \sum_{i=1}^{n} |\theta_i - \theta_i^{true}|$ as in (Gutmann et al, 2008). The other measures are computed using a testing set composed by 10,000 randomly sampled interpretations. The second measure is the log likelihood assigned by the learned theory to the test interpretations, where the probability of an interpretation is computed by asking a query that is the conjunction of all the true atoms in the interpretations together with the conjunction of the negation of all false atoms in the interpretation. The third measure is obtained by considering individual atoms rather than whole interpretations: the probability of each atom in the Herbrand base is computed for each interpretation given its parent atoms (the atoms it directly depends on). From the probability of

Table 3 Results of the experiments on the shop dataset. MAD is the mean absolute difference of the parameters together with the standard deviation and p is the p-value of a paired two-tailed t-test (significant differences at the 5% level in bold).

Dataset	MAD		
	RIB	EM	p
shop4	5.9e-5±3.5e-5	0.1500±0.1954	0.1189
shop4s	0.0444±0.0210	0.1564±0.1968	0.1817
shop8	0.0193±0.0273	0.1143±0.1388	**0.0293**
shop10	0.0220±0.0320	0.1371±0.1604	**0.0156**
shop12	0.0216±0.0297	0.0890±0.1307	0.0538
shop12s	0.0343±0.0380	0.1142±0.1282	**0.0148**
shop12u1	0.0472±0.0647	0.0904±0.1286	0.2504
shop12u2	0.0603±0.0793	0.1431±0.1360	**0.0310**

Table 4 Results of the experiments on the shop dataset. LL is the log likelihood of the test interpretations, AUCPR is the area under the precision-recall curve averaged over all the atoms in the Herbrand base of the theory together with the standard deviation. The last column gives the learning time in seconds.

Dataset	LL		AUCPR		Time (s)	
	RIB	EM	RIB	EM	RIB	EM
shop4	-14861	-18493	0.5707±0.1106	0.5667±0.1134	0.09	0.22
shop4s	-53264	-55825	0.5707±0.1106	0.5666±0.1134	9.62	11.38
shop8	-30888	-36582	0.4423±0.1500	0.4413±0.1491	2.54	9.47
shop10	-32825	-40104	0.4701±0.1617	0.4539±0.1595	3.32	13.89
shop12	-35348	-41330	0.5028±0.1663	0.5016±0.1662	6.39	32.45
shop12s	-81116	-84521	0.5030±0.1663	0.5018±0.1662	528.98	854.33
shop12u1	-35153	-39850	0.5029±0.1663	0.5016±0.1662	15.30	18.16
shop12u2	-35544	-40667	0.5028±0.1663	0.4977±0.1665	16.90	17.67

each atom in each interpretation, we draw a precision-recall curve and we compute the AUCPR.

Table 3 shows the values of MAD together the standard deviation and the p-value of a paired two-tailed t-test (significant differences in bold) while Table 4 lists the log likelihood and the AUCPR together with the running time in seconds. The result show that RIB achieves a much lower MAD with differences that are statistically significant the 5% level in four cases out of eight. Moreover, RIB also obtains a much higher log likelihood in each experiment while taking less time.

The AUCPR of RIB is also higher than that of EM in all cases but the differences are not statistically significant at the 5% level. It must be noted, however, that in these experiments all the ground atoms have a probability of being true significantly higher than zero so this domain is more useful for testing descriptive learning rather than discriminative leaning. LL and MAD are better measures than AUCPR in this case, since the aim is not to perform classification.

These experiments show that RIB performs particularly well when learning from interpretations that share the same Herbrand base, even when there are unseen atoms. In these cases RIB can better exploit the regularities shared by different examples.

Finally, we compared RIB with PRISM. Since none of the above datasets respect the modeling assumptions of PRISM, we considered a different artificial domain. We selected the PRISM program encoding an Hidden Markov Model in (Sato et al, 2005), we translated it into an LPAD and we generated 500 sequences with three states and two output symbols. We gave as input to RIB and PRISM only the atoms encoding the output sequences and we tried to learn the four parameters of the models. This domain has unseen atom variables since the atoms describing partial state and output

sequences are unseen, RIB achieved a MAD of 0.1547±0.0864 while PRISM obtained 0.0420±0.0587 but the difference is not statistically significant (p-value of 0.0752). Thus, even if RIB can not exploit the modeling assumption, it does not perform significantly worse.

10 Conclusions

We have presented the RIB algorithm that applies the Information Bottleneck to the problem of learning the parameters of LPADs, a language that is prototypical of the distribution semantics family. RIB has been evaluated on real-life and artificial datasets. Supplementary material can be found at `http://sites.google.com/a/unife.it/rib`.

RIB has good performances in particular when learning from datasets containing unseen logical atoms and when the training interpretations share the same Herbrand base.

In the future, we will investigate the development of a lifted version of RIB. This should be possible by clustering group of variables with the same Q distribution. This would start by assigning the same value of $Q(t|y)$ to group of variables, for example to all the choice variables that represent instantiations of the same rule. Then the algorithm would keep the variable as a group until some of them are influenced differently by the input data.

Moreover, we will investigate the possibility of having atoms that are unobserved only in a subset of the examples. Finally, we plan to extend RIB for learning the structure of languages reducible to Bayesian networks by using the techniques presented in (Elidan and Friedman, 2005) and for dealing with networks with cycles.

References

Apt KR, Bezem M (1991) Acyclic programs. New Generation Computing 9(3/4):335–364

Bacchus F (1993) Using first-order probability logic for the construction of bayesian networks. In: Heckerman D, Mamdani EH (eds) Proceedings of the 9th Annual Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, pp 219–226

Blockeel H, Meert W (2007) Towards learning non-recursive LPADs by transforming them into Bayesian networks. In: Blockeel H, Ramon J, Shavlik JW, Tadepalli P (eds) Proceedings of the 17th International Conference on Inductive Logic Programming, Springer, LNCS, vol 4894, pp 94–108

Breese JS, Goldman RP, Wellman MP (1994) Introduction to the special section on knowledge-based construction of probabilistic and decision models. IEEE Transactions On Systems, Man and Cybernetics 24(11):1577–1579

Costa VS, Page D, Qazi M, Cussens J (2003) CLP(BN): Constraint logic programming for probabilistic knowledge. In: Meek C, Kjærulff U (eds) Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, Morgan Kaufmann, pp 517–524

Dantsin E (1991) Probabilistic logic programs and their semantics. In: Voronkov A (ed) Proceedings of the Russian Conference on Logic Programming, Springer, LNCS, vol 592, pp 152–164

Davis J, Goadrich M (2006) The relationship between Precision-Recall and ROC curves. In: Cohen WW, Moore A (eds) Proceedings of the 23rd International Confer-

ence on Machine Learning, ACM, ACM International Conference Proceeding Series, vol 148, pp 233–240

De Raedt L, Kimmig A, Toivonen H (2007) ProbLog: A probabilistic prolog and its application in link discovery. In: Veloso MM (ed) Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp 2462–2467

De Raedt L, Demoen B, Fierens D, Gutmann B, Janssens G, Kimmig A, Landwehr N, Mantadelis T, Meert W, Rocha R, Santos Costa V, Thon I, Vennekens J (2008) Towards digesting the alphabet-soup of statistical relational learning. In: Workshop on Probabilistic Programming (in NIPS)

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

Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society Series B (Methodological) 39(1):1–38

Elidan G, Friedman N (2005) Learning hidden variable networks: The information bottleneck approach. Journal of Machine Learning Research 6:81–127

Friedman N (1998) The Bayesian structural EM algorithm. In: Cooper GF, Moral S (eds) Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, pp 129–138

Friedman N, Mosenzon O, Slonim N, Tishby N (2001) Multivariate information bottleneck. In: Breese JS, Koller D (eds) Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, Morgan Kaufmann, pp 152–161

Getoor L, Taskar B (eds) (2007) Introduction to Statistical Relational Learning. MIT Press

Gutmann B, Kimmig A, Kersting K, De Raedt L (2008) Parameter learning in probabilistic databases: A least squares approach. In: Daelemans W, Goethals B, Morik K (eds) Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, LNCS, vol 5211, pp 473–488

Gutmann B, Kimmig A, Kersting K, De Raedt L (2010) Parameter estimation in ProbLog from annotated queries. Tech. Rep. CW 583, Department of Computer Science, Katholieke Universiteit Leuven, Belgium

Jaeger M (1997) Relational Bayesian networks. In: Geiger D, Shenoy PP (eds) Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, pp 266–273

Jaeger M (2007) Parameter learning for relational bayesian networks. In: Ghahramani Z (ed) Proceedings of the 24th International Conference on Machine Learning, ACM, ACM International Conference Proceeding Series, vol 227, pp 369–376

Kersting K, De Raedt L (2001) Towards combining inductive logic programming with Bayesian networks. In: Rouveirol C, Sebag M (eds) Proceedings of the 11th International Conference on Inductive Logic Programming, Springer, LNCS, vol 2157, pp 118–131

Koller D, Pfeffer A (1997) Learning probabilities for noisy first-order rules. In: Proceedings of the 15th International Joint Conference on Artifical Intelligence, Morgan Kaufmann, pp 1316–1321

Lauritzen SL (1995) The EM algorithm for graphical association models with missing data. Computational Statistics & Data Analysis 19(2):191–201

Lowd D, Domingos P (2007) Efficient weight learning for Markov logic networks. In: Kok JN, Koronacki J, de Mántaras RL, Matwin S, Mladenic D, Skowron A (eds) Proceedings of the 18th European Conference on Machine Learning, Springer, LNCS,

vol 4702, pp 200–211
Meert W, Struyf J, Blockeel H (2007) Learning ground CP-logic theories by means of Bayesian network techniques. In: Proceedings of the 6th International Workshop on Multi-Relational Data Mining
Meert W, Struyf J, Blockeel H (2008) Learning ground CP-Logic theories by leveraging Bayesian network learning techniques. Fundamenta Informaticae 89(1):131–160
Meert W, Struyf J, Blockeel H (2010) CP-Logic theory inference with contextual variable elimination and comparison to bdd based inference methods. In: De Raedt L (ed) Proceedings of the 19th International Conference on Inductive Logic Programming, (Revised Papers), Springer, LNCS, vol 5989, pp 96–109
Mihalkova L, Mooney RJ (2007) Bottom-up learning of Markov logic network structure. In: Ghahramani Z (ed) Proceedings of the 24th International Conference on Machine Learning, ACM, ACM International Conference Proceeding Series, vol 227, pp 625–632
Muggleton S (2002) Learning structure and parameters of stochastic logic programs. In: Matwin S, Sammut C (eds) Proceedings of the 12th International Conference on Inductive Logic Programming (Revised Papers), Springer, LNCS, vol 2583, pp 198–206
Muggleton S, De Raedt L (1994) Inductive logic programming: Theory and methods. Journal of Logic Programming 19/20:629–679
Natarajan S, Tadepalli P, Altendorf E, Dietterich TG, Fern A, Restificar A (2005) Learning first-order probabilistic models with combining rules. In: Raedt LD, Wrobel S (eds) Proceedings of the 22nd International Conference on Machine Learning, ACM, ACM International Conference Proceeding Series, vol 119, pp 609–616
Pearl J (1988) Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo, CA: Morgan Kaufman Publishers
Poole D (1997) The Independent Choice Logic for modelling multiple agents under uncertainty. Artifcial Intelligenge 94(1-2):7–56
Richardson M, Domingos P (2006) Markov logic networks. Machine Learning 62(1-2):107–136
Riguzzi F (2004) Learning logic programs with annotated disjunctions. In: Camacho R, King RD, Srinivasan A (eds) Proceedings of the 14th International Conference on Inductive Logic Programming, Springer, LNCS, vol 3194, pp 270–287
Riguzzi F (2007a) ALLPAD: Approximate learning of logic programs with annotated disjunctions. In: Muggleton S, Otero RP, Tamaddoni-Nezhad A (eds) Proceedings of the 16th International Conference on Inductive Logic Programming, Springer, LNCS, vol 4455, pp 43–45
Riguzzi F (2007b) A top-down interpreter for LPAD and CP-Logic. In: Basili R, Pazienza MT (eds) Proceedings of the 10th Congress of the Italian Association for Artificial Intelligence, Springer, LNCS, vol 4733, pp 109–120
Riguzzi F (2008a) ALLPAD: approximate learning of logic programs with annotated disjunctions. Machine Learning 70(2-3):207–223
Riguzzi F (2008b) Inference with logic programs with annotated disjunctions under the well-founded semantics. In: Garcia de la Banda M, Pontelli E (eds) Proceedings of the 24th International Conference on Logic Programming, Springer, LNCS, vol 5366, pp 667–771
Riguzzi F (2010) SLGAD resolution for inference on Logic Programs with Annotated Disjunctions. Fundamenta Informaticae 102(3-4):429–466

Riguzzi F, Di Mauro N (2010) Application of the information bottleneck to LPAD learning. Tech. Rep. CS-2010-01, Dipartimento di Ingegneria, Università di Ferrara, Italy, URL http://www.ing.unife.it/docenti/FabrizioRiguzzi/Papers/RigdiM%10-TR-CS-2010-01.pdf

Riguzzi F, Swift T (2010) Tabling and Answer Subsumption for Reasoning on Logic Programs with Annotated Disjunctions. In: Hermenegildo MV, Schaub T (eds) Technical Communications of the 26th International Conference on Logic Programming, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, LIPIcs, vol 7, pp 162–171

Rose K (2002) Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. Proceedings of the IEEE 86(11):2210–2239

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

Sato T, Kameya Y (2001) Parameter learning of logic programs for symbolic-statistical modeling. Journal of Artificial Intelligence Research 15:391–454

Sato T, Kameya Y, Zhou NF (2005) Generative modeling with failure in prism. In: Kaelbling LP, Saffiotti A (eds) Proceedings of the 19th International Joint Conference on Artificial Intelligence, Professional Book Center, pp 847–852

Singla P, Domingos P (2005) Discriminative training of Markov logic networks. In: Veloso MM, Kambhampati S (eds) Proceedings of the 20th National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, AAAI Press / The MIT Press, pp 868–873

Singla P, Domingos P (2006) Entity resolution with Markov logic. In: Proceedings of the 6th IEEE International Conference on Data Mining, IEEE Computer Society, pp 572–582

Tishby N, Pereira F, Bialek W (1999) The information bottleneck method. In: 37th Annual Allerton Conference on Communication, Control and Computing, pp 368–377

Van Gelder A, Ross KA, Schlipf JS (1991) The well-founded semantics for general logic programs. Journal of the ACM 38(3):620–650

Vennekens J, Verbaeten S (2003) Logic programs with annotated disjunctions. Tech. Rep. CW386, Department of Computer Science, Katholieke Universiteit Leuven, Belgium

Vennekens J, Verbaeten S, Bruynooghe M (2004) Logic programs with annotated disjunctions. In: Demoen B, Lifschitz V (eds) Proceedings of the 20th International Conference on Logic Programming, Springer, LNCS, vol 3131, pp 195–209

A Derivatives of the G Functions for LPADs

Theorem 1 *The derivative of the G functions with respect to $Q(ch_i|y)$, $Q(t_i|y)$ and γ are given by*

$$\frac{\partial G_{ch_i,y}(Q,\gamma)}{\partial Q(ch_i|y)} = -\frac{1}{Q(ch_i|y)} + Q(y)(1-Q(ch_i|y)) \cdot \left(\frac{1-\gamma}{Q(ch_i)} + \gamma \mathbb{E}_{Q(\mathbf{CH},\mathbf{T}|ch_i,y)}[\mathcal{D}(y, ch_{t(i,y,\mathbf{x}[y],\mathbf{t})}, ch_i, \mathbf{pa}_{ch_{t(i,y,\mathbf{x}[y],\mathbf{t})}})]\right)$$

$$\frac{\partial G_{t_i,y}(Q,\gamma)}{\partial Q(t_i|y)} = -\frac{1}{Q(t_i|y)} + Q(y)(1-Q(t_i|y)) \cdot \left(\frac{1-\gamma}{Q(t_i)} + \gamma \mathbb{E}_{Q(\mathbf{CH},\mathbf{T}|t_i,y)}[\mathcal{F}(y, ech(t_i), et(t_i))]\right)$$

$$\frac{\partial G_{ch_i,y}(Q,\gamma)}{\partial \gamma} = -\log Q(ch_i) + \mathbb{EP}'(ch_i,y) - \mathbb{E}_{Q(ch_i'|y)}[\mathbb{EP}'(ch_i',y)] - \log Q(ch_i')]$$

where

$$\mathcal{D}(y_0, ch_{t(i,y_0,\mathbf{x}[y_0],\mathbf{t})}, ch_{i0}, \mathbf{pa}_{ch_{t(i,y_0,\mathbf{x}[y_0],\mathbf{t})}}) = \frac{1}{\mathcal{N}(r(i))} \sum_{k \in t(i,y_0,\mathbf{x}[y_0],\mathbf{t})} \frac{\prod_{t_j \in tb(\mathbf{pa}_{ch_{i0}})} Q(t_j|y_0) 1\{ch_{i0} = ch_k\}}{\theta_{HD_{r(i)}=ch_k|true}}$$

$$\mathcal{F}(y_0, ech(t_{i0}), et(t_{i0})) = \sum_{k \in b(t_{i0},y_0,\mathbf{x}[y_0],\mathbf{t})} \sum_{s \in i(r(k))} bt(\mathbf{pa}_{ch_s}[y_0]) \cdot \frac{\prod_{t_j' \in tb(\mathbf{pa}_{ch_s}) \setminus t_{i0}} Q(t_j'|y_0) 1\{t_{i0} \in tb(\mathbf{pa}_{ch_s})\}}{\mathcal{N}(r(k))} \left(\frac{Q(Ch_s' = ch_k|y_0)}{\theta_{HD_{r(k)}=ch_k|true}} + 1\right)$$

$$\mathbb{EP}'(ch_i,y) = \prod_{t_j \in body(i)} Q(t_j|y) \log \theta_{HD_{r(k)}=ch_i|\mathbf{pa}_{ch_k}}[y] 1\{body(i)[y] = true\} + \delta(\sum_{t_j \in \mathbf{pa}^{\mathbf{T}}_{ch_i}} \prod_{t_j \in \mathbf{pa}^{\mathbf{T}}_{ch_i}} Q(t_j|y) 1\{body(i)[y] = false, ch_i \neq none\} + \sum_{j \in p(i), x_j[y]=true, ch_i \neq x_j[y]} \prod_{ch_s \in \mathbf{pa}_{x_j}, s \neq i} Q(ch_s \neq x_j[y]||y) + \sum_{j \in p(i), t_j=true, ch_i \neq t_j} \prod_{ch_s \in \mathbf{pa}_{t_j}, s \neq i} Q(ch_s \neq t_j|y) Q(T_j = true|y) + 1\{ch_i = x_j[y], val(ch_i)[y] = false\} + Q(T_j = false|y) 1\{ch_i = t_j, t_j = false\})$$

$$\mathbb{EP}'(t_i,y) = \sum_{k \in bb(t_i,y)} \sum_{ch_k \neq none} Q(ch_k|y) \prod_{t_j \in body(k), t_j \neq t_i} Q(t_j|y) \log \theta_{HD_{r(k)}=ch_k|true} + \delta\Bigg(\sum_{k \in bb(t_i,y)} Q(Ch_k = none|y) \prod_{t_j \in body(k), t_j \neq t_i} Q(t_j|y) + \sum_{k \in bb(t_i,y)} Q(Ch_k \neq none|y) \Bigg(1 - \prod_{t_j \in body(k), t_j \neq t_i} Q(t_j|y)\Bigg) + \sum_{k, body(k)[y]=true, \overline{t_i} \in body(k)} Q(Ch_k \neq none|y) + \sum_{k, body(k)[y]=false, t_i \in body(k)} Q(Ch_k \neq none|y) + \prod_{Ch_s \in \mathbf{pa}_{t_i}} Q(Ch_s \neq t_i|y) 1\{t_i = true\} +$$

$$\left(1 - \prod_{Ch_s \in \mathbf{pa}_{t_i}} Q(Ch_s \neq t_i|y)\right) 1\{t_i = false\}\Bigg)$$

In these expressions $r(i)$ is the non ground rule of which c_i is an instance; $t(i, y, \mathbf{x}[y], \mathbf{t})$ is the set of indexes k of choice variables ch_k that are instances of rule $r(i)$ and such that the instantiated rule c_k has the body true with respect to $(\mathbf{x}[y], \mathbf{t})$; $ch_{t(i,y),\mathbf{x}[y],\mathbf{t}}$ is the set of choice variables ch_k with $k \in t(i, y, \mathbf{x}[y], \mathbf{t})$; $val(ch_i)$ indicates the atom variable associated to the value of ch_i; $p(i)$ is the set of the indexes of the atom variables appearing in the head of rule i; $\mathbf{pa}^T_{ch_i} = \mathbf{pa}_{ch_i} \cap T$; $body(k)$ is the body of the rule for ch_k; $body(k)[y]$ is the portion of body restricted to $\mathbf{X}$ variables taking the values $\mathbf{x}[y]$; $bb(t_i, y) = \{k|body(k)[y] = true, t_i \in body(k)\}$; $b(t_i, y, \mathbf{x}[y], \mathbf{t})$ is the set of indexes of instantiations of rules for which t_i appears in the body with a matching truth value and such that the body is true with respect to $(\mathbf{x}[y], \mathbf{t})$; $ech(t_i)$ is the set of ch_s where $s \in b(t_i, y_0, \mathbf{x}[y], \mathbf{t})$; $et(t_i)$ is the set of t_i such that t_i appears in the body of a rule c_s with $s \in b(t_i, y_0, \mathbf{x}[y], \mathbf{t})$ with a matching truth value. and δ is used to approximate $\log 0$ *(e.g. $\delta = -10$).*

All these expressions do not require inference over the underlying Bayesian network.

Proof See Theorems 1 and 2 of (Riguzzi and Di Mauro, 2010).

In order to compute the step size we need also to compute expressions for $\mathbf{I}_Q(\mathbf{CH}, \mathbf{T}; Y)$ and $\nabla_{Q,\gamma}\mathbf{I}_Q(\mathbf{CH}, \mathbf{T}; Y))$.

Theorem 2 *$\mathbf{I}_Q(\mathbf{CH}, \mathbf{T}; Y)$ can be expressed as*

$$\begin{aligned}
&\mathbf{I}_Q(\mathbf{CH}, \mathbf{T}; Y) = \\
&\sum_i \sum_{ch_i} \sum_y Q(y)Q(ch_i|y)(\log Q(ch_i|y) - \log Q(ch_i)) + \\
&\sum_i \sum_{t_i} \sum_y Q(y)Q(t_i|y)(\log Q(t_i|y) - \log Q(t_i))
\end{aligned}$$

Proof See Theorems 3 of (Riguzzi and Di Mauro, 2010).

Theorem 3 *The derivatives of $\mathbf{I}_Q(\mathbf{CH}, \mathbf{T}; Y)$ with respect to $Q(ch_i|y)$ and $Q(t_i|y)$ are*

$$\frac{\partial \mathbf{I}_Q(\mathbf{CH}, \mathbf{T}; Y)}{\partial Q(ch_i|y)} = Q(y)(\log Q(ch_i|y) - \log Q(ch_i))$$

$$\frac{\partial \mathbf{I}_Q(\mathbf{CH}, \mathbf{T}; Y)}{\partial Q(t_i|y)} = Q(y)(\log Q(t_i|y) - \log Q(t_i))$$

Proof See Theorems 4 of (Riguzzi and Di Mauro, 2010).