# Applying the Information Bottleneck Approach to SRL: Learning LPAD Parameters

Fabrizio Riguzzi[1] and Nicola Di Mauro[2]

[1] Dipartimento di Ingegneria, Università di Ferrara - Italy,
fabrizio.riguzzi@unife.it
[2] Dipartimento di Informatica, Università di Bari "Aldo Moro"- Italy,
ndm@di.uniba.it

**Abstract.** In this paper, we propose to apply the Information Bottleneck (IB) approach to a sub-class of Statistical Relational Learning (SRL) languages. Learning parameters in SRL dealing with domains that involve hidden variables requires the use of techniques for learning from incomplete data such as the *expectation maximization* (EM) algorithm. Recently, IB was shown to overcome well known problems of the EM algorithm. Here we show that learning in SRL languages reducible to Bayesian Networks can be obtained by applying the IB approach. In particular, our focus is on the problem of learning the parameters of Logic Programs with Annotated Disjunction (LPADs). We adopt a reductionist approach in which an acyclic LPAD is translated into a Bayesian network. The reduction process introduces in the network some hidden variables thus naturally requiring the use of the IB approach. The paper illustrates the algorithm Relational Information Bottleneck (RIB) that learns LPAD parameters and shows some promising experimental results.

## 1 Introduction

Probabilistic Inductive Logic Programming [6] and Statistical Relational Learning (SRL) [3] 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 techniques. This combination has been proved highly successful in a variety of fields, from social networks analysis to entity resolution, from collective classification to information extraction.

Learning parameters in SRL dealing with domains that involve hidden variables requires the use of techniques for learning from incomplete data such as the *expectation maximization* (EM) algorithm. The Information Bottleneck (IB) framework [2] was shown to be superior to EM for learning parameters of Bayesian networks with hidden variables. 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. In this paper, we

discuss how IB can be applied to the problem of learning the parameters of Logic Programs with Annotated Disjunction (LPADs) [9].

An LPAD can be translated into a Bayesian network with hidden variables [9], thus an algorithm that handles incomplete data is necessary in order to perform learning. Previous approaches for learning this language include: [4], that proposed to use the EM algorithm for inducing the parameters and the Structural EM algorithm for inducing the structure of ground LPADs, and [7], that adopts constraint optimization techniques to learn a subclass of programs.

In this paper, we specialized the IB approach for the case of LPADs obtaining the Relational Information Bottleneck (RIB) algorithm. RIB is tested on artificial datasets and on the IMDB dataset by comparing it with the EM algorithm. The results on the artificial datasets show that RIB is more accurate in finding the values of parameters, while the results on IMDB show that RIB achieves higher area under the precision-recall curve in less time.

## 2 Information Bottleneck Framework

The Information Bottleneck [2] had its origin in clustering [8]. Given two variables $X$ and $Y$ and their joint distribution $Q(X, Y)$, the aim of clustering is to group values of $Y$ so 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, we want to cluster words in a way that is most relevant to the documents' topics. The *information* that $Y$ contains about $X$ (and vice versa) is naturally measured in terms of the *mutual information* $\mathbf{I}_Q(X;Y) \triangleq \sum_{x,y} Q(x,y) \log \frac{Q(x,y)}{Q(x)Q(y)}$. To cluster $Y$'s values, [8] introduces a bottleneck variable $T$ whose values identify the various clusters and the function $Q(T|Y)$ represents the degree of membership of the values of $Y$ to the clusters. $T$ must compress $Y$ while capturing as much as possible the information about $X$. Clustering in this case can be performed by finding the parameters of the $Q$ distribution such that the function

$$\mathcal{L}[Q] = \mathbf{I}_Q(Y;T) - \beta \mathbf{I}_Q(T;X) \tag{1}$$

is minimized, where $\beta$ determines the trade-off between information compression and preservation.

This approach has been applied in [2] to the problem of learning Bayesian networks with hidden variables, where the hidden variables are treated as the bottleneck variable. In other words, we are given some data $\mathcal{D} = \{\mathbf{x}[1], \ldots, \mathbf{x}[M]\}$ over the observed variables $\mathbf{X}$ and we want to find a generative model $P$ over the observed variables $\mathbf{X}$ and the hidden variable $T$ that describes $D$. The variable $Y$ is used in this case to represents the instance identity and it takes values from $\{1, \ldots, M\}$. We want to find the parameters (and possibly the structure) of $P$ so that $T$ explains the observed data. $T$ should compress the training data while at the same time preserving information about the observed attributes. We can model the problem with two Bayesian networks: $\mathcal{G}_{in}$, representing the

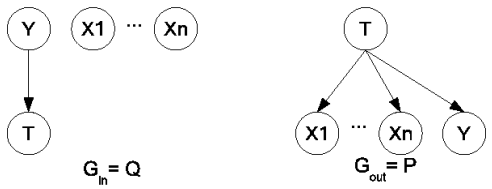


**Fig. 1.** $G_{in}$ and $G_{out}$ for the multivariate information bottleneck framework.

$Q$ distribution (i.e., the required compression), and $\mathcal{G}_{out}$, representing the $P$ distribution, the one we are trying to learn, see Figure 1. 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 $\mathbf{X}$ in $\mathcal{G}_{in}$ given $Y$, 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 this case, the objective function takes the following form

$$\mathcal{L}^+_{EM} = \sum_i \mathbf{I}_Q(T_i;Y) - \gamma(\mathbf{E}_Q[\log P(\mathbf{X},T)] - \sum_i \mathbf{E}_Q[\log Q(T_i)]). \tag{2}$$

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

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

The parameter $\gamma$ balances between compression of the data and the fit of parameters to $\mathcal{G}_{out}$: for $\gamma = 1$, (2) is equivalent to the objective function of the EM algorithm. The IB-EM can bypass local maxima of EM by varying $\gamma$ in (2) using a deterministic annealing strategy: $\gamma$ is initially set to 0, where a single, easy to compute solution exists, 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 $\gamma$. The aim is to follow a smooth path from the trivial solution at $\gamma = 0$ to a good solution at $\gamma = 1$. When the gradient of $\mathcal{L}^+_{EM}$ is zero, the functions $G_{t_i,y}(Q,\gamma)$ (see [2]) for all $t_i$ and $y$ take value 0. Hence the goal is to follow an equi-potential path where all $G_{t_i,y}(Q,\gamma)$ functions are zero starting from some small value of $\gamma$ up to the desired solution at $\gamma = 1$. Starting from a point $(Q_0,\gamma_0)$, where $G_{t_i,y}(Q_0,\gamma_0) = 0$ for all $t_i$ and $y$, we want to move in a direction $\Delta = (dQ, d\gamma)$ s.t. $G_{t_i,y}(Q_0 + dQ, \gamma_0 + d\gamma) = 0$. Hence, we want to find a direction $\Delta$ s.t.

$$\forall t_i, y \nabla_{Q,\gamma} G_{t_i,y}(Q_0,\gamma_0) \cdot \Delta^T = 0, \tag{3}$$

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 $\gamma$. In order to compute the derivatives of $G_{t_i,y}(Q,\gamma)$, we

must express $\log P(\mathbf{x}[y], \mathbf{t})$ as a function of the parameters $Q(t_i|y)$. 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, $\alpha$ 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 in 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.

## 3 Information Bottleneck for LPADs

In this section we briefly present the basics of LPADs and their conversion to Bayesian networks, followed by a description of RIB.

### 3.1 Logic Programs with Annotated Disjunctions

A Logic Program with Annotated Disjunctions $L$ consists of a finite set of formulas of the form

$$(H_1 : \theta_1) \vee (H_2 : \theta_2) \vee \ldots \vee (H_n : \theta_n) \leftarrow B_1, B_2, \ldots B_m,$$

called *annotated disjunctive clauses*. In such a clause the $H_i$ are logical atoms, the $B_i$ are logical literals and the $\theta_i$ 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 atom *null* whose annotation is $1 - \sum_{i=1}^n \theta_i$. For a clause $C$ of the form above, we define $H(C, i)$ as $H_i$ and $\theta(C, i)$ as $\theta_i$.

The semantics of an LPAD was given in [9] for finite ground programs. A non-ground program can be assigned a semantics only if its grounding is finite, i.e., if it does not contain function symbols. The semantics is given in this case in terms of its grounding.

Each ground annotated disjunctive clause represents a probabilistic choice between a number of ground non-disjunctive clauses. By choosing a head atom for each ground clause of an LPAD we get a normal logic program called an *instance* of the LPAD. A probability distribution is defined over the space of instances by assuming independence between the choices made for each clause. The probability of a ground query $\phi$ is given by the sum of the probabilities of the instances where the query is true.

An LPAD is acyclic if to the ground atoms can be assigned an integer level so that the level of each atom in the head of each ground rule is the same and it is higher than the level of each atom in the body. An acyclic LPAD $L$ can be translated into a Bayesian network $\beta(L)$ [9] that has a Boolean random variable

for each ground atom plus a random variable $choice_{C\Theta}$ for each grounding $C\Theta$ of each clause $C$ of $L$. $choice_{C\Theta}$ assumes value $H(C,i)\Theta$ with probability $\theta(C,i)$ if the configuration of its parents makes the body true, while it assumes value *null* with probability 1 if the configuration makes the body false. Ground atom $A$ has as parents all the $choice_{C\Theta}$ variables for which $A$ appears in the head of $C\Theta$. $A$ assumes value true with probability 1 if one of the parent choice variables assumes value $A$, otherwise it assumes value false with probability 1. Note that in order to convert an LPAD containing variables into a Bayesian network, its grounding must be generated.

### 3.2 Relational Information Bottleneck

In order to apply IB 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. The set of hidden variables contains the vector of the choice variables **CH** plus those atoms that are unobserved in the data, let us call them **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 naive Bayes factorization, so $Q(\mathbf{CH}, \mathbf{T}|Y) = \prod_i Q(CH_i|Y) \prod_j Q(T_j|Y)$.

*Example 1.* Consider the following LPAD $L$:

$$r_1 : x_1 \vee x_2. \qquad r_2 : x_2 \vee x_3.$$
$$r_3 : x_4 \vee x_5 \leftarrow x_1. \qquad r_4 : x_5 \leftarrow x_2, x_3.$$
$$r_5 : x_6 \vee x_7 \leftarrow x_2, x_5.$$

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 2. 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 2. As regards $P$, **CH** and **T** must be the only parents of $Y$. This requirement is also satisfied by $\mathcal{G}_{out}$: in fact, the observed variables are completely determined by knowing **CH** and **T** and so it the instance identity.

Let us now sketch how to compute the direction $\Delta$. We want to compute the derivatives of $G_{ch_i,y}(Q,\gamma)$ and $G_{t_j,y}(Q,\gamma)$ for all $ch_i$, $t_j$ and $y$ with respect to the parameters and $\gamma$, and then use the orthogonal direction as the update step. In the following, we will sketch the derivations.

Let us first express the parameters of $P$. Note that, if we want to be able to translate the network back to an LPAD, some of them are fixed in advance: those belonging to the CPTs for atoms and those in the rows corresponding to the body false for 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 $X_j$. Thus $\theta_{X_j=1|\mathbf{pa}_{X_j}} = 1$ if the atom $X_j$ is among the values $\mathbf{pa}_{X_j}$ of its parents, and 0

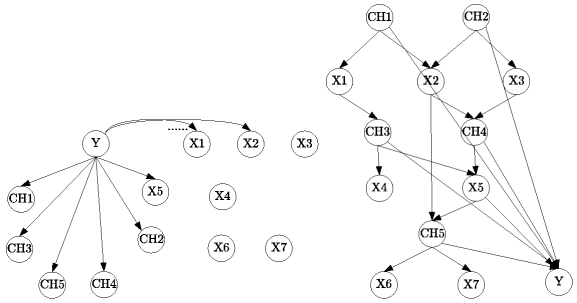


**Fig. 2.** $G_{in} = Q$ (left) and $G_{out} = P$ (right)

otherwise. For a non ground rule $r$, let $\theta_{hd_r,body_r}$ 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 $hd_r = null$.

Moreover, let $i(r)$ be the set of instances of $r$. Given the body $pa_{CH_s}$ of instantiated rule $s$, let $bt(pa_{CH_s})$ be 1 if the observed variables in $pa_{CH_s}$ do not make the body false and 0 otherwise. Let $tb(pa_{CH_s})$ be a set of values for the unobserved variables that are parents of $CH_s$ and that do not make the body false.

The maximum likelihood parameters of the distribution of $HD_r$ with body true are

$$\theta_{hd_r|true} = \frac{\mathcal{N}(r, hd_r) + \alpha(r, hd_r, true)}{\mathcal{N}(r) + \alpha(r, true)}$$

$$\mathcal{N}(r, hd_r) = \sum_{s \in i(r)} \sum_{y} Q(y) Q(CH_s = hd_r|y) bt(pa_{ch_s}[y]) \prod_{t_j \in tb(pa_{ch_s})} Q(t_j|y)$$

$$\mathcal{N}(r) = \sum_{hd_r} \mathcal{N}(r, hd_r)$$

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

## 4 Evaluation

We tested RIB on some synthetic datasets and on the IMDB dataset [5][3].

The synthetic datasets have been obtained by writing some LPADs and by generating training sets from them. Four different LPADs have been considered: one of them is the shop example from [4] and contains 4 rules, while the others have been obtained by progressively extending that example to 8, 10 and 12 rules respectively. For each of these LPADs, all the possible interpretations with

[3] Available at `http://alchemy.cs.washington.edu/data/imdb`.

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.

Then the parameters of the programs are learned by RIB and EM, both implemented in Yap Prolog. The parameters obtained are compared with the true ones by computing a mean squared error (MSE). The result show that RIB achieves a much lower MSE in each experiment while taking less time: the average MSE over the four experiments was 0.00218 for RIB and 0.03929 for EM, the total time for RIB was 15.563 seconds against a total time of 56.026 seconds for EM.

We also run experiments with the smallest and the largest theory in which the training set was composed respectively by 1,000 and 10,000 examples obtained from the theories by random sampling. These results confirmed the previous ones: the average MSE was 0.00376 for RIB and 0.04398 for EM and the total time was 538.597 seconds for RIB and 865.709 seconds for EM.

IMDB is a real world database that we used to compare the performances of RIB with that of EM in terms of area under the precision-recall curve. 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 only 1,540 are true.

We used a methodology similar to the one in [5]: we trained on four mega-examples and tested on the remaining one. We choose an LPAD that predicts the value of the target predicate $sameperson(A, B)$.

Each mega-example 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 example 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 instance that is true in the data, while we sampled from the complete set of false instances three times the number of true instances in order to generate negative examples. In the interpretation, the constants $s$ and $p$ are replaced respectively with two dummy constants that are the same for all the examples, so that the overall number of different constants representing persons is reduced to two while the constants for the other types are kept intact.

Our implementation of the EM algorithm was able to complete the first fold, while reporting a memory resource limit error on the remaining folds. The learned parameters were then used to generate the precision-recall curves by drawing the points corresponding to the testing set and by interpolating among them, using the technique described in [1]. On the first fold, RIB achieved an area under the precision-recall curve of 0.2695, while EM achieved an area of 0.2399.

Learning took 599.4 seconds for RIB and 1091.1 seconds for EM. Overall, the average area for RIB was 0.3316 obtained in 470.4 seconds on average.

The experiments discussed in [2] that showed IB-EM superior to EM are thus confirmed by the results.

## 5 Conclusions

We have presented the RIB algorithm that applies the Information Bottleneck to the problem of learning the parameters of an LPAD. Experimental evaluation proves the validity of RIB on artificial and real-life datasets. Supplementary material can be found at `http://sites.google.com/a/unife.it/rib`.

In the future, we plan to extend RIB for learning the structure of LPADs by using the techniques presented in [2]. Furthermore we plan to extend the work to deal with networks with cycles.

## References

1. Davis, J., Goadrich, M.: The relationship between precision-recall and roc curves. In: Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006). pp. 233–240. ACM (2006)
2. Elidan, G., Friedman, N.: Learning hidden variable networks: The information bottleneck approach. Journal of Machine Learning Research 6, 81–127 (2005)
3. Getoor, L., Taskar, B. (eds.): Introduction to Statistical Relational Learning. MIT Press (2007)
4. Meert, W., Struyf, J., Blockeel, H.: Learning ground cp-logic theories by leveraging bayesian network learning techniques. Fundamenta Informaticae 89(1), 131–160 (2008)
5. Mihalkova, L., Mooney, R.J.: Bottom-up learning of markov logic network structure. In: Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007). pp. 625–632. ACM (2007)
6. Raedt, L.D., Frasconi, P., Kersting, K., Muggleton, S. (eds.): Probabilistic Inductive Logic Programming - Theory and Applications, Lecture Notes in Computer Science, vol. 4911. Springer (2008)
7. Riguzzi, F.: ALLPAD: approximate learning of logic programs with annotated disjunctions. Machine Learning 70(2-3), 207–223 (2008)
8. Tishby, N., Pereira, F., Bialek, W.: The information bottleneck method. In: Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing. pp. 368–377 (1999)
9. Vennekens, J., Verbaeten, S., Bruynooghe, M.: Logic programs with annotated disjunctions. In: The 20th International Conference on Logic Programming (ICLP 2004). LNCS, vol. 3131, pp. 195–209. Springer (2004)