Learning Relational Probabilistic Models from Partially Observed Data - Opening the Closed-World Assumption

Tushar Khot[1], Sriraam Natarajan[2], Kristian Kersting[3], and Jude Shavlik[1]

[1] University of Wisconsin-Madison, USA
[2] Wake Forest University School of Medicine, USA
[3] Fraunhofer IAIS, Germany

Abstract. Recent years have seen a surge of interest in learning the structure of Statistical Relational Learning (SRL) models that combine logic with probabilities. Most of these models apply the closed-world assumption i.e., whatever is not observed is false in the world. In this work, we consider the problem of learning the structure of SRL models in the presence of hidden data i.e. we open the closed-world assumption. We develop a functional-gradient boosting algorithm based on EM to learn the structure and parameters of the models simultaneously and apply it to learn different kinds of models – Relational Dependency Networks, Markov Logic Networks and relational policies. Our results in a variety of domains demonstrate that the algorithms can effectively learn with missing data.

Introduction

Traditional machine learning assumes that the world can be described in terms of features, but the world is made up of objects that interrelate. Data about these worlds is inherently noisy and relational. In addition, the data in the real world can have missing values due to several reasons. Statistical Relational Learning (SRL) [1] deals with uncertainty and relations among objects. SRL models seek to avoid explicit state enumeration through a symbolic representation of states. The advantage of these models is that they can succinctly represent probabilistic dependencies among the attributes of different related objects leading to a compact representation of learned models. The compactness and even comprehensibility gained by SRL, however, comes at the expense of a typically much more complex learning task. There have been some advances in this problem, especially in the case of Markov Logic Networks (MLN) [2–5]. More recently, algorithms based on functional-gradient boosting [6] have been developed for learning SRL models such as Relational Dependency Networks (RDN) [7], and MLNs [4]. One of the key advantages of the functional-gradient based algorithms is that the structure and parameters of the models are learned simultaneously and empirical results show that these methods outperform the respective state-of-the-art algorithms.

While these methods exhibit good empirical performance, they apply the closed-world assumption, i.e., whatever is unobserved in the world is considered to be false. Research with missing data in SRL has mainly focused on learning the parameters. In such cases, algorithms based on classical EM [8] have been developed for several SRL

models [9–11]. There has been some work on learning structure of SRL models from hidden data [12, 13]. These approaches, inspired by Friedman's structural EM approach for Bayesian networks [14], compute the sufficient statistics over the hidden states and perform a greedy hill-climbing search over the clauses.

We significantly extend this approach – inspired by the success of structural EM on propositional graphical models [14] and the success of boosting in learning SRL models [4, 7], we propose an EM algorithm for functional-gradient boosting. We derive and present the update equations of the E and M-steps of the algorithm. One of the key features of our algorithm is that we consider the set of distributions in the models to be a product of potentials and this allows us to learn different models such as MLNs [15], RDNs [16] and even relational policies [17]. After deriving the EM algorithm, we then also show how to adopt the standard approach of approximating the full likelihood by the MAP states (i.e., hard EM). We empirically evaluate the proposed algorithm in different datasets and demonstrate the superiority of the proposed approach against different baseline algorithms.

To summarize, this paper makes several key contributions: First, we propose an algorithm that can learn the structure and parameters of SRL models in the presence of hidden data. The algorithm is based on two different successful methods – EM for learning with hidden data and functional-gradient boosting for SRL models – and hence is theoretically interesting. Second, following previous work, we show how to adapt the algorithm for learning RDNs, MLNs, and relational policies. So, our algorithm extends multiple earlier proposed methods to the problem of learning from missing data. Third, as far as we are aware, this is the first work on the adaptation of the successful structural EM algorithm for relational functional-gradient boosting. Finally, we evaluate the algorithm on a variety of different types of tasks to demonstrate the broad applicability of our algorithm.

Background

Statistical Relational Learning Models: RDNs [16] are relational extensions of dependency networks [18], which are directed graphical models that may contain cycles. The joint distribution can be factored as a product of individual conditionals and can be represented by Relational Probability Trees (RPT; [19]) or Relational Regression Trees (RRT; [20]) with a sigmoid applied to the regression output. MLNs [15] are relational undirected models where first-order logic formulas correspond to the cliques of a Markov network and formula weights correspond to the clique potentials. An MLN can be instantiated as a Markov network with a node for each ground predicate (atom) and a clique for each ground formula, where all groundings of the same formula are assigned the same weight.

Functional Gradient Boosting: A standard method of supervised learning is based on gradient-descent where the learning algorithm starts with initial parameters θ_0 and computes the gradient of the likelihood function. Dietterich et al. [21] used a more general approach to train the potential functions based on Friedman's [6] gradient-tree boosting algorithm, where the potential functions are represented by sums of regression trees that are grown stage-wise.

Functional gradient method starts with an initial potential ψ_0 and iteratively adds gradients Δ_i. Hence after m iterations, the potential is given by $\psi_m = \psi_0 + \Delta_1 + ... + \Delta_m$. Here, Δ_m is the functional gradient at episode m. Instead of computing the functional gradients over the potential function, they are instead computed for each training example i, given as $\langle \mathbf{x_i}, y_i \rangle$. This set of local gradients forms a set of training examples for learning the gradient at stage m. Friedman [6] suggested fitting a regression tree to these derived examples i.e., fit a regression tree h_m on the training examples $\{(x_i, y_i), \Delta_m(y_i; x_i)\}$. We replace the propositional regression trees with RRTs.

Functional Gradient Boosting in SRL: Functional gradient boosting has been applied to SRL models such as RDNs [7] and MLNs [4]. Since computing the true likelihood function for MLNs is prohibitive, the pseudo-likelihood (PL) function is popularly used for learning in MLNs. PL in MLNs is defined as the product of the conditional probabilities of the ground variables given their Markov blankets (MB). Note that in the case of RDNs, the joint distribution is approximated by the product of the conditional distributions. Hence the learning problem for both RDN and MLN optimizes the product of conditional distributions. Earlier work [7, 4] represented these conditional distributions as a sigmoid over a function ψ, i.e. $P(x|Pa(x))$ in RDNs and $P(x|MB(x))$ in MLNs were represented as $\frac{e^{\psi(x)}}{[1+e^{\psi(x)}]}$. For both these problems, the functional gradient of the likelihood for each example $\langle y_i, \mathbf{x}_i \rangle$ with respect to $\psi(y_i = 1; \mathbf{x_i})$ was shown to be: $\frac{\partial \log P(y_i; \mathbf{x_i})}{\partial \psi(y_i = 1; \mathbf{x_i})} = I(y_i = 1; \mathbf{x_i}) - P(y_i = 1; \mathbf{x_i})$, where I is the indicator function that is 1, if $y_i = 1$ and 0 otherwise. Since the learning procedure and the gradients for both these models are identical, we show that one can use our Structural EM approach for both RDNs and MLNs by just changing the scoring criterion and probability calculations. In the case of imitation learning, the policy $P(action|state)$ can be represented using a sigmoid function and prior work [17] showed that these policies can be represented as sets of RRTs. The gradients were similar to the one presented above. Hence, as we show, our algorithms can also be employed for imitation learning.

Structural EM

We first define some notations that will be used throughout the paper. Please note that since the algorithm is employed across different formalisms, we use the same notations for the different models. We use capital letters such as X, Y, Z to represent variables (predicates in our formalisms) and small letters such as x, y, z to represent values taken by the variables. We use bold-faced letters to represents sets. Letters such as **X, Y, Z** represent sets of variables and **x, y, z** represent sets of values. We shall use $\mathbf{z}_{-z}$ to denote $\mathbf{z} \setminus z$ and $\mathbf{x}_{-i}$ to represent $\mathbf{x} \setminus x_i$.

The high-level overview of our RFGB-EM (Relational Functional Gradient Boosting - EM) approach is shown in Figure 1. Similar to other EM approaches, we sample the states for the hidden groundings based on our current model in the E-step and use the sampled states to update our model in the M-step. ψ_t represents the model in the t^{th} iteration. The initial model, ψ_0 can be as simple as a uniform probability for all examples or could be a model specified by an expert. We sample certain number of assignments of the hidden groundings (denoted as $|W|$) using the current model ψ_t. Based on these samples, we create regression examples which are then used to learn T

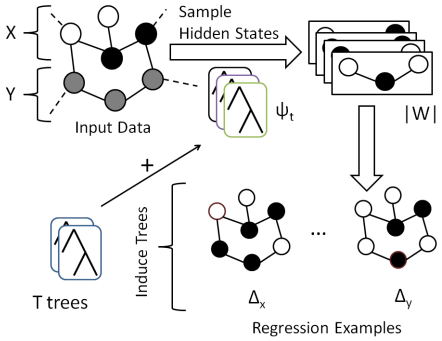

Fig. 1: RFGB-EM in action. Shaded nodes indicate variables with unknown assignments, while the white (or black) nodes are assigned true (or false) values. The input data has observed (indicated by X) and hidden (indicated by Y) groundings. We sample $|W|$ assignments of the hidden groundings using the current model ψ_t. We create regression examples based on these samples, which are used to learn T relational regression trees. The learned trees are added to the current model and the process is repeated.

relational regression trees. The learned regression trees are added to the current model and the process is repeated. To perform the M-step, we update the current model using functional gradients. We now derive the M-step.

Derivation for M-step For ease of explanation, let $\mathbf{X}$ be all the observed predicates and $\mathbf{Y}$ be all the hidden predicates (their corresponding groundings are $\mathbf{x}$ and $\mathbf{y}$). Since $\mathbf{Y}$ is unobserved, it can have multiple possible assignments denoted by $\mathcal{Y}$ and $\mathbf{y} \in \mathcal{Y}$ represents one such hidden state assignment.

Traditionally, EM approaches for parameter learning find θ that maximize the $\mathcal{Q}(\theta|\theta_t)$ function. The $\mathcal{Q}(\theta|\theta_t)$ function is defined as the expected log-likelihood of missing and observed data (based on θ) where the expectation is measured based on the current distribution (θ_t) for the missing data i.e.

$$\mathcal{Q}(\theta|\theta_t) = \sum_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y}|\mathbf{x}; \theta_t) \log P(\mathbf{x}, \mathbf{y}|\theta)$$

In our work, we are not just interested in estimating the parameters. We are also learning the structure of the models (RDNs or MLNs as appropriate). Moreover, our functional-gradient boosting approach for learning the structure is non-parametric i.e., we do not have fixed set of parameters θ but instead use a regression function ψ. This is a subtle yet important distinction to the EM learning methods for SRL [9, 10, 22] that estimate the parameters given a fixed structure. The regression function in our models are captured using relational regression trees. The trees define the structure of the potential function and the leaves of the trees represent the parameters of these potentials. Hence we rewrite the $\mathcal{Q}$ function as $\mathcal{Q}(\psi \mid \psi_t)$.

Following prior work [17, 7, 4], we seek to maximize the pseudo-loglikelihood because computing the log-likelihood for relational data is prohibitively expensive. Hence the $\mathcal{Q}$ function now becomes

$$\mathcal{Q}(\psi \mid \psi_t) = \sum_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y}|\mathbf{x}; \psi_t) \sum_{z \in \mathbf{x}, \mathbf{y}} \log P(z|\mathbf{z}_{-z}; \psi)$$

Commonly EM methods find parameters that maximize the $\mathcal{Q}$ function and set them as the parameters for the next iteration, i.e. $\theta_{t+1} = \arg\max_\theta \mathcal{Q}(\theta|\theta_t)$. In our case, $\mathcal{Q}$

does not have a closed form solution for ψ. Following the procedures used in generalized EM algorithms [8] rather than finding the maximum at every step, we perform gradient descent (via functional gradient boosting) over the $\mathcal{Q}$ function, i.e., $\psi_{t+1} = \psi_t + \Delta_\psi \mathcal{Q}(\psi|\psi_t)$. Due to the computational cost of performing gradient descent till convergence, we perform S gradient steps in the M-step. In our experiments, S was set to 2. This allowed us to amortize the cost of sampling the world states and run enough EM iterations in reasonable time without making the model too large. It can be shown that our approach guarantees a monotonic increase in the pseudo-loglikelihood of the observed data, if the new ψ_{t+1} improves over $\mathcal{Q}(\psi_t|\psi_t)$.

We present our proposed approach for updating the model in Algorithm 1. We iterate through all the query and hidden predicates and learn one tree for each predicate. We compute the gradients for the groundings of predicate p given by E_p, using the world states W and current model ψ. We then learn a relational regression tree using this dataset and add it to our current model. The $learnTree$ function uses different scoring functions depending on the model (MLNs vs RDNs) as we show later. The set E_p may not contain all the groundings of the predicate p, since we downsample the negative examples during every iteration by randomly selecting the negatives so that there are twice as many negative as positive examples. Relational datasets generally have many more negative examples than positives and it has been shown that ensemble methods perform better if the majority class is downsampled [23].

Algorithm 1 updateModel(W, ψ)

```
S := 2 {Number of trees learned in M-step}
for i ≤ S do
    {Iterate over target and hidden predicates, P}
    for p ∈ P do
        {E_p := Downsampled groundings of p}
        D_p := buildDataset(E_p, W, ψ)
        T_p := learnTree(D_p)
        ψ = ψ + T_p
    end for
end for
return ψ
```

To apply functional-gradient boosting, we need to compute the gradients for each example (i.e. hidden and observed groundings of the target and hidden predicates) which will be used to learn the next regression tree (T_p). In previous work, the gradients were computed w.r.t. $\psi(x)$, where x is a ground query literal. The value returned by the ψ function also depends on the other ground literals, since their values will influence the path taken in the regression tree. One can include them as arguments to the function definition i.e. $\psi(x; \mathbf{x})$. But $\mathbf{x}$ is observed and constant across all examples and so the function is simplified to $\psi(x)$. In our definition, the assignment to the hidden variables $\mathbf{y}$ is not observed and each assignment may return a different value for a given exam-

ple. Hence, we include the assignment to the hidden variables in our function definition ($\psi(x; \mathbf{y})$) and compute the gradients for an example and hidden state assignment.

Gradients for hidden groundings We now focus on obtaining the gradients of $\mathcal{Q}$ w.r.t the hidden groundings by taking partial derivatives of $\mathcal{Q}$ w.r.t $\psi(y_i; \mathbf{y}_{-i})$ where y_i is a hidden grounding. The value of $\psi(y_i; \mathbf{y}_{-i})$ is only used to calculate $P(y_i|\mathbf{x}, \mathbf{y}_{-i}; \psi)$ for two world states: where y_i is true and where y_i is false. So the gradient w.r.t. $\psi(y_i; \mathbf{y}_{-i})$ can be calculated as
$P(y_i = 1, \mathbf{y}_{-i}|\mathbf{x}; \psi_t) \frac{\partial \log P(y_i=1|\mathbf{x}, \mathbf{y}_{-i}; \psi)}{\partial \psi(y_i; \mathbf{y}_{-i})} + P(y_i = 0, \mathbf{y}_{-i}|\mathbf{x}; \psi_t) \frac{\partial \log P(y_i=0|\mathbf{x}, \mathbf{y}_{-i}; \psi)}{\partial \psi(y_i; \mathbf{y}_{-i})}$

As shown in previous work [21], the gradients would correspond to the difference between the true value of y_i and the current predicted probability of y_i (i.e. $I(y_i = y) - P(y_i = y)$). Since we have two terms involving $P(y_i)$, one for each value of y_i, we get two different gradients.

$$\begin{aligned} & P(y_i = 1, \mathbf{y}_{-i}|\mathbf{x}; \psi_t)(1 - P(y_i = 1|\mathbf{x}, \mathbf{y}_{-i}; \psi)) \\ & \quad + P(y_i = 0, \mathbf{y}_{-i}|\mathbf{x}; \psi_t)(0 - P(y_i = 1|\mathbf{x}, \mathbf{y}_{-i}; \psi)) \\ & = P(y_i = 1, \mathbf{y}_{-i}|\mathbf{x}; \psi_t) - P(\mathbf{y}_{-i}|\mathbf{x}; \psi_t) P(y_i = 1|\mathbf{x}, \mathbf{y}_{-i}; \psi)) \end{aligned} \quad (1)$$

With the PLL assumption, the gradients can be written as $\prod_{j \neq i} P(y_j|\mathbf{x}, \mathbf{y}_{-j}; \psi_t)[P(y_i = 1|\mathbf{x}, \mathbf{y}_{-i}; \psi_t) - P(y_i = 1|\mathbf{x}, \mathbf{y}_{-i}; \psi)]$. Intuitively, the gradients correspond to the difference between the probability predictions weighted by the probability of the hidden state assignment.

Gradients for observed groundings To compute the gradients for the observed groundings, we take partial derivatives of $\mathcal{Q}$ with respect to $\psi(x_i; \mathbf{y})$ where x_i is observed in the data. Similar to the gradients for hidden groundings, we use $\mathbf{y}$ as an argument in the ψ function and only consider the world states that matches with the given argument. The gradient is

$$P(\mathbf{y}|\mathbf{x}; \psi_t) \frac{\partial \log P(x_i|\mathbf{x}_{-i}, \mathbf{y}; \psi)}{\partial \psi(x_i; \mathbf{y})} = P(\mathbf{y}|\mathbf{x}; \psi_t)[I(x_i) - P(x_i = 1|\mathbf{z}_{-x_i}; \psi)] \quad (2)$$

Similar to the hidden groundings, the gradients correspond to the difference between the predictions weighted by the probability of the hidden state assignment.

Regression tree learning The input examples to our regression tree learner are of the form $< (z; \mathbf{y}), \Delta >$. For every ground literal $z \in \mathbf{x} \cup \mathbf{y}$, we calculate the gradients for an assignment to the hidden variables. Algorithm 2 describes the $buildDataset$ function used to generate these examples. For every ground literal e and every world state w (i.e., $\mathbf{y}$), we compute the gradient of the example ($gradient(e, w)$). For examples that are observed, we use equation 2 to compute $gradient(e, w)$ and for examples that are hidden, we use equation 1. Similar to previous work [4], we use only a subset of the examples for learning the regression function. Apart from subsampling the ground literals, we also pick $|W|$ hidden state assignments from $\mathcal{Y}$. Since our gradients are weighted

Algorithm 2 buildDataset(E_p, W, ψ)

```
D_p := ∅
for e ∈ E_p do
    for w ∈ W do
        Δ_e := gradient(e, w)
        D_p := D_p ∪ < (e; w), Δ_e >
    end for
end for
return D_p
```

by the probability of the hidden state assignment $\mathbf{y}$, an unlikely assignment will result in small gradients and thereby have little influence on the learned tree. Hence, we use Gibbs sampling to sample the most likely hidden state assignments. Also we approximate the joint probability of an hidden state assignment with the pseudo-likelihood i.e., $P(\mathbf{y}|\mathbf{x}; \psi_t) = \prod_i P(y_i|\mathbf{x}, \mathbf{y}_{-i}; \psi_t)$.

Adapting RFGB-EM for different models

Natarajan et al. [7] describe learning RDN structure using functional-gradient boosting where all the trees are learned for a target predicate before the next predicate. Since the gradients for each predicate are independent of the model for other predicates, one can learn all the trees independently. We, on the other hand, update the hidden world states after every two iterations (note $S = 2$) and hence for every predicate we learn two trees at a time. We then resample the hidden states and use the sampled states for the next two iterations. We use RRTs with the weighted variance scoring function for fitting the gradients for each example. The $learnTree$ function in Algorithm 1 can use any off-the-shelf RRT learner.

For MLNs, we learned RRTs for the gradients presented earlier with the modified scoring function as described by Khot et al. [4]. MLNs are approximated by a product of conditional distributions in their approach where the set of trees for each predicate correspond to the conditional distributions. To compute the marginal probability of any example, trees for all the predicates would be used. Hence while learning, a single tree is learned for each predicate and the gradients are computed based on the trees learned till the current iteration. In our EM approach, we resample the hidden states after two such iterations over the target and hidden predicates. Khot et al. also present an approach to learn MLN clauses to fit the gradients. While we only present the results for learning trees, it is trivial to extend this work to learn clauses.

For imitation learning, we are learning the distribution over the actions for every state using the training trajectories provided by an expert. The set of predicates, P contains all the action predicates and the hidden predicates. We can then learn RRTs to predict each action while updating the hidden values. Natarajan et al. [17] learned all the trees for each action independently whereas we learn two trees for every predicate before resampling the hidden ones.

Experiments

We now present the results of our approaches on four different problems. We use *SEM* to represent the structural EM approach which uses Gibbs sampling for generating the samples. We present results for SEM with a suffix to indicate the number of hidden state samples used i.e., $|W|$ mentioned in the previous section (e.g. *S-10* uses ten samples while *S-1* uses the single MAP estimate). *S-10* corresponds to the soft-EM approach whereas *S-1* corresponds to the hard-EM approach. We also present the results of using RFGB without using EM while setting all hidden groundings to false i.e. using the closed world assumption (*CWA*). This is essentially the prior work on RDNs [7], MLNs [4] and imitation learning [17]. Each of these methods were run for 10 gradient iterations. We observed that this number was enough for convergence in all our domains. In the case where we used MLNs, we used the default settings in Alchemy (http://alchemy.cs.washington.edu). We compare the methods using two different measures: conditional log likelihood (CLL) and area under the PR curve (AUC-PR). We use **bold-face** to indicate results that are statistically significantly better (at p-value=0.05) than all the other methods. In these experiments we attempt to empirically investigate the following questions:
Q1: Can opening CWA for relational structure learning improve the performance?
Q2: Can soft-EM outperform hard-EM in relational domains?

Disjunctive dataset

We generated a simple synthetic dataset to compare *SEM* against *CWA* using RDNs as the base model. We used three predicates *q(X,Y), r(X,Y)* and *s(X)*. The range of X was $1, ..., 100$, and varied Y to have two different values $|Y| = 3$ and $|Y| = 10$ as shown in Table 1. We treated the predicate r as hidden and the goal was to predict s. To generate the training data, we used a distribution $P(r|q)$. We then combine `r(X,Y)` for different values of Y using an OR condition to generate `s(X)`. Hence `s(X)` given `r(X,Y)` is a deterministic rule where `s(X)` is true if for some Y, `r(X,Y)` is true. We generated 10 synthetic datasets with randomly sampled hidden data, trained one model on each dataset and evaluated each model on the other nine datasets. We average the results from all these runs.

We used this synthetic dataset as it allows us to evaluate approaches against varying importance of accurately predicting the missing data. For $|Y| = 10$, it is very likely that at least one of the observed values of `r(x,Y)` is true. Hence even if the missing data values are not accurately predicted, the model will learn the OR rule for `s(x)`. On the other hand for $|Y| = 3$, this scenario is less likely, thereby requiring a more accurate representation of the missing data to learn the OR rule.

The results on this domain are presented in Table 1. We only present the CLL values since the AUC-PR values are nearly equal for all the approaches. The EM approaches outperform CWA in all scenarios thereby affirmatively answering *Q1* for this domain. *SEM-10* outperforms both *SEM-1* and *CWA* methods on this dataset for $|Y| = 3$, whereas *SEM-1* outperforms the others for $|Y| = 10$. As expected, *SEM-10* provides a more accurate representation of the missing data which is needed when $|Y| = 3$. For

$|Y| = 10$, the simpler *SEM-1* approach is sufficient to capture the underlying distribution. This experiment clearly demonstrates the difference between soft (*SEM-10*) and hard (*SEM-1*) versions of EM.

Hidden %	20%		40%	
Algorithm	$\|Y\| = 3$	$\|Y\| = 10$	$\|Y\| = 3$	$\|Y\| = 10$
SEM-10	**-0.049**	-0.107	**-0.109**	-0.181
SEM-1	-0.071	**-0.087**	-0.114	**-0.128**
CWA	-0.093	-0.163	-0.170	-0.223

Table 1: CLL values on the Disjunctive dataset.

Hidden %	20%		40%	
Algorithm	CLL	AUC-PR	CLL	AUC-PR
SEM-10	-0.168	0.334	-0.170	0.376
SEM-1	**-0.150**	0.346	**-0.151**	0.367
CWA	-0.187	0.329	-0.192	0.344

Table 2: Results on the UW data set.

UW-CSE: RDN Structure Learning

The UW dataset [24] is one of the most popular datasets for learning SRL models. The goal here is to predict the `advisedBy` relationship between a student and a professor. The data set consists of details of professors, students and courses from five different sub-areas of computer science. Predicates include `professor`, `student`, `publication`, `advisedBy`, `hasPosition`, `tempAdvisedby`, `inPhase`, `courseLevel`, `taughtBy` etc. We randomly hid groundings of the `tempAdvisedby`, `inPhase` and `hasPosition` predicates during training. We performed five-fold cross-validation and present the results in Table 2. We also varied the amount of hidden data in our experiments ("Hidden %" in the table indicates the percentage of the groundings being hidden).

In general, the EM methods perform statistically significantly (with p-value < 0.05) better than the closed world assumption. Hence, we can answer *Q1* affirmatively in this real world domain too. The difference between the two EM methods is not statistically significant for the different levels of hidden data for AUC-PR (although there is a difference in CLL). It appears that in this domain, using a single sample for the hidden state has the same performance as that of using 10 samples. This is in line with most EM algorithms where using a single state (MAP) approximation generally suffices.

IMDB: RDN Structure learning

The IMDB dataset [25] contains five predicates: `actor`, `director`, `genre`, `gender` and `workedUnder` (Following [2], we omitted the four equality predicates). We predicted the `gender` predicate given all the other predicates. We randomly hid the groundings of `actor` and `workedUnder` predicates during learning and inference. We performed five-fold cross-validation and averaged the results across all the folds.

We present the CLL values for hiding 10% and 20% of the groundings of the two hidden predicates in Table 3. Similar to the disjunctive dataset, there is no statistically significant difference between the three methods in the AUC-PR values and hence are not reported here. In general, the EM methods perform statistically significantly (with p-value < 0.05) better than the closed world assumption. Hence we can again affirmatively answer *Q1* in this domain. Between the two EM methods, using one sample is sufficient to capture the underlying distribution and hence the simpler SEM-1 has a higher CLL value than SEM-10.

Hidden %	10%	20%
SEM-10	-0.501	-0.551
SEM-1	**-0.423**	**-0.467**
CWA	-0.586	-0.80

Table 3: CLL values for IMDB.

Hidden %	20%		40%	
Algorithm	CLL	AUC-PR	CLL	AUC-PR
SEM-10	**-1.445**	**0.482**	**-1.315**	**0.510**
SEM-1	-1.648	**0.483**	-1.586	0.500
CWA	-1.629	0.478	-1.693	0.488

Table 4: Results on Cancer dataset.

Cancer dataset: MLN Structure Learning

The cancer MLN is a popular synthetic data set [26, 15] used in SRL. We created a friend network which is represented using a symmetric predicate, `friends(X,Y)`. Each person has three attributes: `stress(X)`, `cancer(X)` and `smokes(X)`. A person is more likely to smoke if he has stress or has a lot of friends who smoke. Similarly, a person is likely to have cancer if he smokes or he has a lot of friends who smoke. The more smoker friends a person has, the more likely he is to get cancer. Such rules can be captured by MLNs since the probabilities are proportional to the number of groundings of a clause (e.g. $smokes(y) \wedge friend(x, y) \rightarrow smokes(x)$). The target predicate is `cancer` while `smokes` has some missing groundings.

We trained the model on 10 generated datasets with randomly sampled hidden data and evaluated each model on the other nine datasets. We average the results from all these runs. We present the results for different amounts of hidden data (20% and 40%). As seen in Table 4, *SEM-10* mostly outperforms the other approaches both in terms of CLL and AUC-PR. For 20% missing data, there is no statistically significant difference between the two EM approaches but both methods outperform CWA. Unlike the previous domains, SEM-10 is at least as good as or better than SEM-1 in this domain. Hence for this domain, we can affirmatively answer both *Q1* and *Q2*. Since Alchemy does not have a mechanism to handle missing data for structure learning, we ran weight learning (generative with 10000 iterations and 1e-5 threshold) on hand-written rules and simply learned the weights for this MLN using the Alchemy package. The AUC PR values were around 0.6. This shows that simply learning the parameters is reasonably comparable to our models that learn both the structure and parameters with hidden data.

Wumpus world: Learning relational policies

As a third application of RFGB-EM, we performed imitation learning in a relational domain for a partially observed MDP (POMDP). We created a simple version of the Wumpus task where the location of wumpus is partially observed. We used a 5×5 grid with the wumpus placed at a random location in every training trajectory. The wumpus is always surrounded by stench on all four sides. Figure 2 shows one instantiation of the initial grid locations. The agent can perform 8 possible actions: 4 move actions in each direction and 4 shoot actions in each direction. The agent's task is to move to a cell such that he can fire an arrow to kill the wumpus. The dataset contains predicates for each cell such as `cellAt`, `cellRight` and `cellAbove` and obstacle locations such as `wumpus` and `stench`. The wumpus is not observed in all the trajectories although the stench is always observed. Trajectories were created by human users whose

policy generally is to move towards the wumpus' row or column and shoot accordingly.

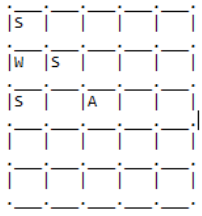

Fig. 2: Wumpus world. W indicates the wumpus location, S indicates the stench location and A is the agent.

Hidden %	20%		40%	
Algorithm	CLL	AUC-PR	CLL	AUC-PR
SEM-10	**-0.245**	**0.857**	**-0.261**	**0.853**
SEM-1	-0.278	0.845	-0.283	0.839
CWA	-0.282	0.826	-0.270	0.826

Table 5: Results for Wumpus dataset.

The EM approaches (using the trajectories where wumpus is observed) learn that wumpus is surrounded by stench and fill the missing values in other trajectories. The *CWA* approach [17] on the other hand assumes that the wumpus is not present and relies on the stench to guess the action to be performed. The results are presented in Table 5. From the results, it can be easily observed that the EM methods are superior to that of the prior work on imitation learning. Moreover, *SEM-10* which uses multiple samples outperforms the single-sample *SEM-1* approach. This domain clearly shows that the previous method of boosting in imitation learning is not sufficient in problems with partial observability and it is imperative to employ methods that do not assume closed-world. Similar to the Cancer domain, we can affirmatively answer *Q1* and *Q2* in this domain too.

In conclusion, our experiments have shown that the opening the closed-world assumption definitely results in an improvement in the performance. Between the two EM approaches, we have shown empirically that for certain domains (e.g. UW, IMDB) a single sample (hard-EM) might be sufficient, whereas in certain domains (e.g. Cancer, Wumpus) multiple samples (soft-EM) are needed to capture the true distribution. Thus, both questions *Q1* and *Q2* can generally be answered affirmatively.

Conclusions

We addressed the challenging problem of learning SRL models in the presence of hidden data. We developed an EM-based algorithm for functional-gradient boosting. We derived the gradients for the M-step by maximizing the lower bound of the gradient and showed how to approximate the E-step. We evaluated the algorithm on three different types of relational learning problems - RDNs, MLNs and imitation learning. Our results indicate that the proposed algorithms outperform the respective algorithms that make closed-world assumptions.

Our current approach computes the probability of an example for each world state of the hidden groundings. But based on the relational tree structure, we can avoid recomputing the probabilities for examples, if a different world state will have no impact on the probability of the example. Adapting the different EM heuristics such as random restarts is another interesting direction. We could also calculate the marginal probabilities of each hidden grounding and use them as probabilistic facts to learn the trees. Our approach can handle bidirected and undirected models but extending it to an acyclic directed model is an interesting avenue for future research.

References

1. Getoor, L., Taskar, B., eds.: Introduction to Statistical Relational Learning. MIT Press (2007)
2. Kok, S., Domingos, P.: Learning Markov logic network structure via hypergraph lifting. In: ICML. (2009)
3. Kok, S., Domingos, P.: Learning Markov logic networks using structural motifs. In: ICML. (2010)
4. Khot, T., Natarajan, S., Kersting, K., Shavlik, J.: Learning Markov logic networks via functional gradient boosting. In: ICDM. (2011)
5. Khosravi, H., Schulte, O., Hu, J., Gao, T.: Learning compact Markov logic networks with decision trees. Machine Learning **89**(3) (2012) 257–277
6. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics (2001) 1189–1232
7. Natarajan, S., Khot, T., Kersting, K., Guttmann, B., Shavlik, J.: Gradient-based boosting for statistical relational learning: The relational dependency network case. Machine Learning (2012)
8. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society **B.39** (1977) pp. 1–38
9. Natarajan, S., Tadepalli, P., Dietterich, T.G., Fern, A.: Learning first-order probabilistic models with combining rules. Annals of Mathematics and AI (2009)
10. Jaeger, M.: Parameter learning for Relational Bayesian networks. In: ICML. (2007)
11. Kameya, Y., Sato, T.: Efficient EM learning with tabulation for parameterized logic programs. In: Computational Logic. (2000)
12. Li, X., Zhou, Z.: Structure learning of probabilistic relational models from incomplete relational data. In: ECML. (2007)
13. Kersting, K., Raiko, T.: 'say em for selecting probabilistic models for logical sequences. In: UAI. (2005)
14. Friedman, N.: The Bayesian structural EM algorithm. In: UAI. (1998)
15. Domingos, P., Lowd, D.: Markov Logic: An Interface Layer for AI. Morgan & Claypool, San Rafael, CA (2009)
16. Neville, J., Jensen, D.: Relational dependency networks. In Getoor, L., Taskar, B., eds.: Introduction to Statistical Relational Learning. (2007) 653–692
17. Natarajan, S., Joshi, S., Tadepalli, P., Kristian, K., Shavlik, J.: Imitation learning in relational domains: A functional-gradient boosting approach. In: IJCAI. (2011)
18. Heckerman, D., Chickering, D., Meek, C., Rounthwaite, R., Kadie, C.: Dependency networks for inference, collaborative filtering, and data visualization. Journal of Machine Learning Research **1** (2001) 49–75
19. Neville, J., Jensen, D., Friedland, L., Hay, M.: Learning Relational Probability trees. In: KDD. (2003)
20. Blockeel, H., Raedt, L.D.: Top-down induction of first-order logical decision trees. Artificial Intelligence **101** (1998) 285–297
21. Dietterich, T., Ashenfelter, A., Bulatov, Y.: Training conditional random fields via gradient tree boosting. In: ICML. (2004)
22. Xiang, R., Neville, J.: Pseudolikelihood EM for within-network relational learning. In: ICDM. (2008)
23. Chan, P., Stolfo, S.J.: Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection. In: KDD. (1998)
24. Singla, P., Domingos, P.: Entity resolution with Markov logic. In: ICDM. (2006) 572–582
25. Mihalkova, L., Mooney, R.: Bottom-up learning of Markov logic network structure. In: ICML. (2007) 625–632
26. Kersting, K., Ahmadi, B., Natarajan, S.: Counting Belief Propagation. In: UAI. (2009)