Structure Learning with Hidden Data in Relational Domains

Tushar Khot, Sriraam Natarajan[*], Kristian Kersting[*+], Jude Shavlik
University of Wisconsin-Madison, USA
[*] Wake Forest University School of Medicine, USA
[+] Fraunhofer IAIS, Germany

Abstract

Recent years have seen a surge of interest in learning the structure of Statistical Relational Learning (SRL) models, which combine logic with probabilities. Most of these models apply the closed-world assumption i.e., whatever is not observed is false in the world. 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 two kinds of models – Relational Dependency Networks (RDNs) and Markov Logic Networks (MLNs). Our results in two testbeds demonstrate that the algorithms can effectively learn with missing data.

1. Introduction

Statistical Relational Learning (SRL) (Getoor & Taskar, 2007) elevates propositional graphical models to a first-order level, achieving a more compact representation of states. 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 (Kok & Domingos, 2009; 2010; Khot et al., 2011). More recently, algorithms based on functional-gradient boosting (Friedman, 2001) have been developed for learning SRL models such as RDNS (Natarajan et al., 2012), and MLNs (Khot et al., 2011).

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 on handling missing data in SRL has mainly focused on learning the parameters (Natarajan et al., 2009; Jaeger, 2007; Kameya & Sato, 2000). Li and Zhou (2007) directly learn the structure of a PRM model from hidden data. Since they are learning a directed model, they have to perform the expensive check of ensuring acyclicity in the ground model. Kersting and Raiko (2005) learn the structure of logical HMMs in the presence of missing data. Their approach computes the sufficient statistics over the hidden states and does a greedy hill-climbing search over the clauses. We significantly extend this approach – inspired by the success of structural EM on propositional graphical models (Friedman, 1998) and the success of boosting in learning SRL models, we propose an EM algorithm for functional-gradient boosting. 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. This allows us to learn different models such as MLNs (Domingos & Lowd, 2009; Khot et al., 2011) and RDNs (Neville & Jensen, 2007; Natarajan et al., 2012). After deriving the EM algorithm, we adopt the standard approach of approximating the full likelihood by the MAP states (i.e., hard EM). We empirically evaluate the proposed algorithm in two datasets.

2. Background

Statistical Relational Learning Models: RDNs (Neville & Jensen, 2007) are relational extensions of dependency networks (Heckerman et al., 2001), 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; Neville et al. 2003) or Relational Regression Trees (RRT; Blockeel & Raedt 1998) with a sigmoid applied to the regression output. MLNs (Domingos & Lowd, 2009) 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. (2004) used a more general approach to train the potential functions based on Friedman's (2001) gradient-tree boosting algorithm, where the potential functions are represented by sums of regression trees that are grown stage-wise.

More formally, functional gradient ascent 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 (2001) suggested fitting a regression tree to these derived examples i.e., fit a regression tree h_m on the training examples $\{(x_i, y_i), \Delta_m(y_i; x_i)\}$. In this work, we replace the propositional regression trees with relational regression trees.

Functional Gradient Boosting in SRL: Functional gradient boosting has been applied to SRL models such as RDNs (Natarajan et al., 2012) and MLNs (Khot et al., 2011). Since structure learning using the true likelihood function for MLNs is computationally prohibitive due to the computation of the normalization constant, the pseudo-likelihood function is popularly used for learning in MLNs. Pseudo-likelihood 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 (Natarajan et al., 2012; Khot et al., 2011) 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. We refer the reader to previous work (Natarajan et al., 2012; Khot et al., 2011) for further details.

3. Structural EM

We first define some notation that will be used throughout the paper. Note that we use the same notation across the formalisms. 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 use script letters such as $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ to represent world states, i.e. $\mathcal{Y}$ would represent a set of **y**.

3.1. RFGB-EM

We present the basic pseudocode for our RFGB-EM (Relational Functional Gradient Boosting - EM) approach in Algorithm 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 a domain expert. We perform T steps of the EM algorithm, where we fix $T = 10$ since results didn't change much after 10 iterations. We now derive the update equations for the M-step for functional-gradient boosting and then present the approximations for the E-step.

Algorithm 1 RFGB-EM(P, H, D)

```
1: {P is the set of target and hidden predicates}
2: {H is the set of hidden groundings}
3: {D is the set of observed groundings}
4: Set initial model, $\psi_0$
5: t:= 0
6: for $t \leq T$ do
7:     W := sampleWorlds(H, D, $\psi_t$) {E-step}
8:     $\psi_{t+1}$ := updateModel(W, P, D, $\psi_t$) {M-Step}
9: end for
```

3.1.1. Derivation for M-step

For ease of explanation, let **X** be all the observed predicates and **Y** be all the hidden predicates (their corresponding groundings are **x** and **y**). Following prior work (Natarajan et al., 2012; Khot et al., 2011), we seek to maximize the log likelihood of the observed groundings. We average the likelihood function over all possible values of the hidden groundings to compute the marginal probabilities of the observed groundings shown below,

$$\begin{aligned}\ell(\psi) &\equiv \log P(\mathbf{X}=\mathbf{x}|\psi) = \log \sum_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{x};\mathbf{y}|\psi) \\ &= log \sum_{\mathbf{y}\in\mathcal{Y}} \left\{ P(\mathbf{y}|\mathbf{x};\psi) \frac{P(\mathbf{x};\mathbf{y}|\psi)}{P(\mathbf{y}|\mathbf{x};\psi)} \right\}\end{aligned}$$

As explained in the background section, ψ is the regression function that is used to calculate $P(X = x|Pa(x))$. As mentioned earlier, the regression function is represented as a sum of relational regression trees. The trees define the structure of the potential function and the leaves of the trees represent the parameters of this potentials. After t iterations of the EM steps, ψ_t represents the current model and we must update this model by finding a ψ that maximizes the log-likelihood. Similar to previous EM approaches, we can maximize the lower bound on $\ell(\psi)$ given by

$$\ell(\psi) \geq \mathcal{Q}(\psi;\psi_t) - \mathcal{Q}(\psi_t;\psi_t) + \ell(\psi_t)$$

where the only term that depends on ψ is given by

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

For computational purposes following prior work on MLNs and Markov nets, we use the pseudo-loglikelihood instead of the loglikelihood in $\mathcal{Q}(\psi;\psi_t)$. We define Z as the union of all the predicates i.e. $Z = X \cup Y$. We shall use $\mathbf{z}_{-z}$ to denote $\mathbf{z} \setminus z$ and $\mathcal{Y}_{-i}$ to represent the world states for the set of groundings $\mathbf{y}_{-y_i}$ (i.e. $\mathbf{y} \setminus y_i$). Hence we can now rewrite $\mathcal{Q}(\psi;\psi_t)$:

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

Since we do not have a closed form solution for ψ, we use steepest descent with functional gradients. Also, following the procedures used in generalized EM algorithms (Dempster et al., 1977) rather than finding the maximum at every step, we take S gradient steps in the M-step. Finding a ψ that improves over $\mathcal{Q}(\psi;\psi_t)$ might suffice since it will ensure that the log-likelihood would increase. In prior work, each gradient step was approximated by a single tree. Hence in this work, we learn S trees for every M-step.

We present the algorithm for updating the model in Algorithm 2. We do not maximize the $\mathcal{Q}$ function but take S gradient steps in each iteration. 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. We would learn $S \times T$ trees in the end, which would be 20 trees in our case. 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 depends on the model class that we are learning. 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).

Algorithm 2 updateModel(W, P, D, ψ)

```
1: $S := 2$ {Number of trees learned in M-step}
2: for $i \leq S$ do
3:   {Iterate over target and hidden predicates, P}
4:   for $p \in P$ do
5:     {$E_p :=$ Downsampled groundings of $p$}
6:     $D_p := buildDataset(E_p, W, D, \psi)$
7:     $T_p := learnTree(D_p, D)$
8:     $\psi = \psi + T_p$
9:   end for
10: end for
11: return $\psi$
```

Gradients for hidden groundings

We now focus on obtaining the gradients of $\mathcal{Q}$ w.r.t the hidden groundings before considering the observed predicates. Taking partial derivatives of $\mathcal{Q}$ with respect to $\psi(y_i)$ where y_i is a hidden grounding, produces:

$$\frac{\partial}{\partial\psi(y_i)} \sum_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathbf{x};\psi_t) \sum_{z\in\mathbf{x}\cup\mathbf{y}} logP(z|\mathbf{z}_{-z};\psi) \quad (1)$$

$$= P(y_i = 1|\mathbf{x};\psi_t) - \sum_{\mathbf{y_{-i}}\in\mathcal{Y}_{-i}} P(y_i = 1|\mathbf{z}_{-y_i};\psi))P(\mathbf{y_{-i}}|\mathbf{x};\psi_t) \quad (2)$$

Intuitively, the gradients correspond to the difference between the probability of the hidden grounding being true based on our previous model and expected probability based on the current model averaged over all the hidden groundings states.

Gradients for observed groundings

To compute the gradients for the observed groundings, we take partial derivatives of $\mathcal{Q}$ with respect to $\psi(x_i)$

where x_i is observed in the data.

$$\frac{\partial}{\partial\psi(x_i)} \sum_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathbf{x};\psi_t) \sum_{z\in\mathbf{x}\cup\mathbf{y}} \log P(z|\mathbf{z}_{-z};\psi) = I(x_i) - \sum_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathbf{x};\psi_t) P(x_i = 1|\mathbf{z}_{-x_i};\psi) \quad (3)$$

Intuitively, this corresponds to the difference between the true value of the observed groundings and the expected probability averaged over all the hidden groundings states. We can also rewrite the gradients as a weighted sum of gradients corresponding to each world state.

Algorithm 3 describes the *buildDataset* function used to compute the gradients for the examples in E_p. For every example e and every world state w along with the observed groundings D, we compute the gradient of the example. The gradient formula in equations 2 and 3 can be understood as a weighted mean of gradients computed for each world state. Hence we can compute the gradient of an example by summing the weighted gradients as shown in line 5.

Algorithm 3 buildDataset(E_p, W, D, ψ)

```
1: $D_p := \emptyset$
2: for $e \in E_p$ do
3:     $\Delta_e := 0$
4:     for $w \in W$ do
5:         $\Delta_e := \Delta_e + gradient(e, w \cup D) * prob(w)$
6:     end for
7:     $D_p := D_p \cup < e, \Delta_e >$
8: end for
9: return $D_p$
```

3.1.2. Approximations for the E-step

We approximated the loglikelihood with the pseudo-loglikelihood as shown in the derivation. This approximation is necessary to make the approach computationally feasible and has been used in many structure and weight-learning approaches for SRL (Kok & Domingos, 2009; 2010).

Computing probabilities for all possible world states would be exponential in number of hidden groundings. This would also result in computing the gradients for all examples in each one of these world states. Hence we use Gibbs sampling to generate $|W|$ samples from the distribution $P(\mathbf{y}|\mathbf{x};\psi_t)$ to approximate all the world states, $\mathcal{Y}$ and compute the expected counts over only the sampled world states, W. The gradient for the observed groundings now simplifies to $I(x_i) - \sum_{\mathbf{y}\in W} P(\mathbf{y}|\mathbf{x};\psi_t)P(x_i = 1|\mathbf{z}_{-x_i};\psi)$. Similar gradients can be obtained for the hidden groundings too. This is analogous to the Monte Carlo Expectation Maximization (MCEM) approach used for high-dimensional data (Wei & Tanner, 1990). We shall refer to this approximation of the RFGB-EM approach as SEM from now on.

Computing gradients for each example still involves computing the probability of an example for each sampled world state. We can further approximate the expected probability of an example by using the most likely world state/most probable explanation (MPE) instead of using all the world states. The MPE state corresponds to $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x};\psi_t)$. The gradient for the observed groundings now simplifies to $I(x_i)$ - $P(x_i = 1|\mathbf{x}_{-i};\hat{\mathbf{y}};\psi)$. This is similar to the hard-EM approach and is much faster than SEM. We shall refer to this approach as H-SEM. Using MPE to approximate the expected counts has been used in HMMs (Collins, 2002) and MLNs (Singla & Domingos, 2005).

Adapting RFGB-EM for RDN and MLN
In the prior work for learning RDN structure (Natarajan et al., 2012), all the trees are learned for a given target predicate before going on to the next predicate. In our EM approach, we update the hidden world states after every two iterations and hence for every predicate we would learn two trees at a time using the last sampled world state. When applying our approach to learn MLNs, the current set of trees for all the predicates are used to compute the marginal probability (and thereby the gradients) of each example. A single tree is learned for each predicate and the gradients are computed based on all the trees learned before the current iteration. We resample the hidden states after every two iterations of learning trees for the target and hidden predicates. RFGB-EM can be extended to relational imitation learning where functional gradient boosting has been successfully applied (Natarajan et al., 2011). For imitation learning, we get results (not reported here) similar to the ones presented for RDNs and MLNs in the next section.

4. Experiments

In this section, we present the results of our approaches on two different problems. We use *SEM* to represent the structural EM approach, which uses Gibbs sampling for generating the samples. We use *H-SEM* to represent the EM approach that uses the single MPE state instead of sampling multiple hidden states, since it is similar to the hard EM approach. We also present the results of using RFGB without using EM but instead setting all hidden groundings to false, i.e. use the closed world assumption. We denote this as *CWA*.

Please note that we use the general term *CWA* to denote our prior work on RDNs (Natarajan et al., 2012) and MLNs (Khot et al., 2011). They correspond to the appropriate prior work based on the experiment. 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.

4.1. Disjunctive dataset

We generated a simple synthetic dataset to compare *SEM* against *H-SEM* using RDNs as the base model. We generated groundings of `q(X,Y)` using a prior distribution over `q` where $X = \{1 \ldots 100\}$. We ran two different experiments with different number of possible values (i.e., range) of Y, $|Y| = 3$ and $|Y| = 10$ as shown in Table 1. Similarly, we also vary the amount of hidden data (Hidden % indicates the percentage of the groundings being hidden). We generate `r(X,Y)` given `q(X,Y)` using a conditional probability 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.

The results on this dataset are presented in Table 1. We only present the CLL values since the AUC-PR values are same for all the approaches. *SEM* outperforms both *H-SEM* and *CWA* methods on this dataset. *H-SEM* assigns the most likely value to each hidden grounding which happens to be false in this dataset. Hence, there is no difference between *H-SEM* and *CWA* in this dataset. On the other hand, *SEM* samples multiple world states where some of the groundings might be true. Hence, there is some probabilistic mass on `r(X,Y)` $\rightarrow$ `s(X)` being true unlike *H-SEM* that assumes all hidden `r(X,Y)` to be false.

Hidden %	20%		40%	
Algorithm	$\|Y\| = 3$	$\|Y\| = 10$	$\|Y\| = 3$	$\|Y\| = 10$
SEM	**-0.050**	**-0.065**	**-0.055**	**-0.078**
H-SEM	-0.093	-0.108	-0.170	-0.175
CWA	-0.093	-0.108	-0.170	-0.175

Table 1. CLL values on the Disjunctive dataset.

4.2. Cancer dataset

The cancer MLN has been popularly referred and used in SRL literature. We created a friend network 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 a lot of friends who smoke. Similarly, a person is likely to have cancer if he smokes or 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 such as `smokes(y)` $\wedge$`friend(x, y)` $\rightarrow$ `smokes(x)`. The target predicate is `cancer` while `smokes` has some missing groundings.

For this synthetic dataset, we trained one model on each of the 10 generated datasets with randomly sampled hidden data and evaluate each model on the other nine datasets. We average the results from all these runs. As with the previous case, we present the results for different amounts of hidden data (20% and 40%). It can be clearly seen from Table 2 that the *SEM* algorithm dominates the *H-SEM* and the *CWA* algorithm of Khot et al. (2011). Comparing *CWA* and *H-SEM* it is clear that the latter dominates in all the measures (at p-value=0.05). Since Alchemy does not have a mechanism to handle missing data for structure learning, we ran weight learning on hand-written rules with their version of EM. Alchemy weight learning assigns the MPE state to the missing values and hence is similar to our *H-SEM* approach. As seen in Table 2, *Alch*, which is only doing weight learning, performs better than *CWA* and *H-SEM* structure learning approaches, but is worse than SEM.

Hidden %	20%		40%	
Algorithm	CLL	AUC-PR	CLL	AUC-PR
SEM	**-0.709**	**0.650**	**-0.693**	**0.673**
H-SEM	-1.086	0.577	-1.266	0.544
CWA	-1.232	0.535	-1.710	0.521
Alch	-0.967	0.591	-0.730	0.656

Table 2. Results for Cancer dataset.

5. Conclusions

We addressed the challenging problem of learning SRL models in the presence of hidden data. For this goal, 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 employed the proposed approach for two different types of relational learning problems - RDNs (Disjunctive dataset) and MLNs (Cancer dataset). 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 some examples, e.g. the groundings of a predicate whose truth value has changed may be absent from the trees. We could also calculate the marginal probabilities of each hidden grounding and use them as probabilistic facts to compute the gradients. There has been work on improving the MCEM approach (Jank, 2005) by using samples from previous iterations, changing the number of samples in each iteration and deciding the number of iterations that could be applied to this work. Extending our approach to a purely directed model is an interesting avenue for future research.

Acknowledgments

Tushar Khot, Sriraam Natarajan and Jude Shavlik gratefully acknowledge the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government. Kristian Kersting was supported by the Fraunhofer ATTRACT fellowship STREAM and by the European Commission under contract number FP7-248258-First-MM.

References

Blockeel, H. and Raedt, L. De. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101: 285–297, 1998.

Collins, M. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, 2002.

Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B.39:pp. 1–38, 1977.

Dietterich, T.G., Ashenfelter, A., and Bulatov, Y. Training conditional random fields via gradient tree boosting. In *ICML*, 2004.

Domingos, P. and Lowd, D. *Markov Logic: An Interface Layer for AI*. Morgan & Claypool, 2009.

Friedman, J. H. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, pp. 1189–1232, 2001.

Friedman, N. The Bayesian structural EM algorithm. In *UAI*, 1998.

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

Heckerman, D., Chickering, D., Meek, C., Rounthwaite, R., and Kadie, C. Dependency networks for inference, collaborative filtering, and data visualization. *JMLR*, 1: 49–75, 2001.

Jaeger, M. Parameter learning for relational Bayesian networks. In *ICML*, 2007.

Jank, W. Stochastic variants of the EM algorithm: Monte Carlo, quasi-Monte Carlo and more. *ASA*, 2005.

Kameya, Y. and Sato, T. Efficient EM learning with tabulation for parameterized logic programs. In *Computational Logic*, 2000.

Kersting, K. and Raiko, T. 'Say EM' for selecting probabilistic models for logical sequences. In *UAI*, 2005.

Khot, T., Natarajan, S., Kersting, K., and Shavlik, J. Learning Markov logic networks via functional gradient boosting. In *ICDM*, 2011.

Kok, S. and Domingos, P. Learning Markov logic network structure via hypergraph lifting. In *ICML*, 2009.

Kok, S. and Domingos, P. Learning Markov logic networks using structural motifs. In *ICML*, 2010.

Li, X. and Zhou, Z. Structure learning of probabilistic relational models from incomplete relational data. In *ECML*, 2007.

Natarajan, S., Tadepalli, P., Dietterich, T. G, and Fern, A. Learning first-order probabilistic models with combining rules. *Annals of Mathematics and AI*, 2009.

Natarajan, S., Joshi, S., Tadepalli, P., Kristian, K., and Shavlik, J. Imitation learning in relational domains: A functional-gradient boosting approach. In *IJCAI*, 2011.

Natarajan, S., Khot, T., Kersting, K., Guttmann, B., and Shavlik, J. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning*, 2012.

Neville, J. and Jensen, D. Relational dependency networks. In Getoor, L. and Taskar, B. (eds.), *Introduction to Statistical Relational Learning*, pp. 653–692. MIT Press, 2007.

Neville, J., Jensen, D., Friedland, L., and Hay, M. Learning relational probability trees. In *KDD*, 2003.

Singla, P. and Domingos, P. Discriminative training of Markov logic networks. In *AAAI*, 2005.

Wei, G. C. G. and Tanner, M. A. A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. *Journal of the American Statistical Association*, 85(411), 1990.