Learning Parameters for Relational Probabilistic Models with Noisy-Or Combining Rule

Sriraam Natarajan, Prasad Tadepalli*, Gautam Kunapuli, Jude Shavlik
University of Wisconsin, Madison * Oregon State University USA
{natarasr,kunpag,shavlik}@biostat.wisc.edu, *tadepall@cs.orst.edu

Abstract

Languages that combine predicate logic with probabilities are needed to succinctly represent knowledge in many real-world domains. We consider a formalism based on universally quantified conditional influence statements that capture local interactions between object attributes. The effects of different conditional influence statements can be combined using rules such as Noisy-OR. *To combine multiple instantiations of the same rule we need other combining rules at a lower level. In this paper we derive and implement algorithms based on gradient-descent and EM for learning the parameters of these multi-level combining rules. We compare our approaches to learning in Markov Logic Networks and show superior performance in multiple domains.* ***Keywords*-Relational Learning;**

I. Introduction

Complex real world domains, such as citation analysis and social network analysis, motivate the need for rich, knowledge-representationist languages that combine predicate logic with probabilities. There have been several first-order probabilistic languages developed in recent years including probabilistic relational models (PRMs) [1], Markov Logic Networks (MLNs) [2], and many others. The important underlying similarity between all these approaches is that they define a probabilistic model over the attributes and relationships between the objects in a domain. The principle attraction of these relational languages is that they are much more succinct than their propositional counterparts leading to easier specification of their structure by the domain experts, and faster learning of their parameters.

The underlying semantics of almost all the languages based on directed graphical models can be captured by a set of universally quantified influence statements which hold under some conditions. Since, in general, multiple rules (that predict the same target variable) might apply, to keep these models succinct, each such rule is considered independent of others; this is called the assumption of "independence of causal influences" (ICI). In many cases, a single parameterized rule can result in multiple instantiated sets of parents that influence a single ground target variable. Also, the number of these parent variables might change from one instance to the other.

Consider for example, a university domain where we are interested in predicting the satisfaction of a student in a particular year. One might consider two rules to predict the satisfaction. The first rule says that the student's satisfaction is influenced by the grades obtained in the courses for the given year. The second rule says that satisfaction is influenced by the quality of papers published in the year. The students could take multiple courses in a year and/or publish multiple papers in a year. Moreover, the number of courses/papers for every student need not be the same. We model this situation by having a single parameterized conditional distribution for each rule. When the rule is instantiated with a specific instance, e.g., a single course, its influents, e.g., grade, determine the conditional probability of the target attribute, i.e., satisfaction.

A common problem in these models is accounting for the impact of multiple rules and multiple instantiations of the same rule on the target attribute. One possible solution is the use of aggregators [1], which operate on the values of the influents. In this work, we use combining rules that operate on the distributions; this has been proposed earlier and used in [3], [4], [5]. In this approach we assume ICI, where multiple causes on a target variable can be decomposed into several independent causes whose effects are combined to yield a final value. In other words, each parent or set of related parents produces a different value for the child variable, all of which are combined using a deterministic or stochastic function. Depending on how the causes are decomposed and the effects are combined, we can express the conditional distribution of the target variable given all the causes as a function of the conditional distributions of the target variable given each independent cause using a "decomposable combining rule."

We make three contributions in this paper. First, we extend the work above by deriving the EM algorithm for the setting where different influence statements are combined by Noisy-OR at the rule level and by Mean at the instance level. Second, we derive and implement two versions of

the gradient-descent algorithm, one to minimize the mean-squared error (MSE) and the other to maximize the log-likelihood. Third, we perform experiments on two real-world data sets: the UW data set [2] and the `citeseer` data set [6] and a synthetic data set and compare our results against the models of [5] and MLNs [2].

The rest of the paper[1] is organized as follows: in Section II, we present the required background. We then derive the algorithms for learning the parameters in the presence of Noisy-OR combining rule in Section III. We present the experimental results in the Section IV and conclude the paper with some directions for future research.

II. Background

Our learning algorithm does not assume any representation for the rules. It merely uses an abstract similar to the one provided in [5]. Each statement has the form:

If $\langle condition \rangle$ *then* $\langle qualitative\ influence \rangle$

where *condition* is a set of literals, each literal being a predicate symbol applied to the appropriate number of variables. The set of literals is treated as a conjunction. Such abstract statements are called as First-Order Conditional Influence (FOCI) statements. A $\langle qualitative\ influence \rangle$ is of the form $X_1, \ldots, X_k$ *Qinf* Y, where the X_i and Y are of the form $V.a$, and V is a variable that occurs in *condition* and a is an object attribute. This expresses a directional dependence of the *resultant* Y on the *influents* X_i. Associated with each statement is a *conditional probability function* that specifies a probability distribution of the resultant conditioned on the influents, e.g. $P(Y|X_1, \ldots, X_k)$ for the above statement. We will use P_i to denote the probability function of the i^{th} FOCI statement. Consider the statement,

```
If {student(S), course(C), takes(T,S,C)}
    then T.grade Qinf S.satisfaction,
```

which indicates that a student's satisfaction depends on the course grade. The conditional probability distribution $P(T.\text{grade} \mid S.\text{satisfaction})$ associated with this statement (partially) captures the quantitative relationships between the attributes.

A. Combining Rules

```
Noisy-Or{
  If \{student(S), course(C), takes(T,S,C)\}
    then T.grade Qinf (Mean) S.satisfaction.
  If \{student(S),paper(P,S)\}
    then P.quality Qinf (Mean) S.satisfaction.
}
```

We explained the first rule earlier. The second states that if the student has authored a paper, then its quality influences the satisfaction of the student. Multiple instantiations of the respective rules (the different course grades or the different paper qualities) are combined using the Mean combining rule and the distributions due to different rules using Noisy-OR combining rule.

[1]We extend a journal version of the paper (to appear) by elaborating and performing experiments in real-world data sets for comparison with the weighted mean combining rule and MLNs.

III. Learning Parameters

Consider, if $\langle condition \rangle$ then $X_i^1, \ldots, X_i^k$ *Qinf* Y. For notational simplicity, we assume that the i^{th} influence statement, 'rule i' for short, has k influents, X_i^1 through X_i^k (which we jointly denote as $\mathbf{X}_i$). When this rule is instantiated or "grounded" on a database, it generates multiple, say m_i, sets of influent instances, which we denote as $\mathbf{X}_i^1 \ldots \mathbf{X}_i^{m_i}$. The role of the combining rule is to express the probability $P_i(Y|\mathbf{X}_i^1 \ldots \mathbf{X}_i^{m_i})$ as a function of the probabilities $P_i(Y|\mathbf{X}_i^j)$, one for each j, where P_i is the CPT associated with rule i. Since these instance tuples are unordered and can be arbitrary in number, combining rule should be symmetric, i.e., its value should not depend on the order of the arguments.

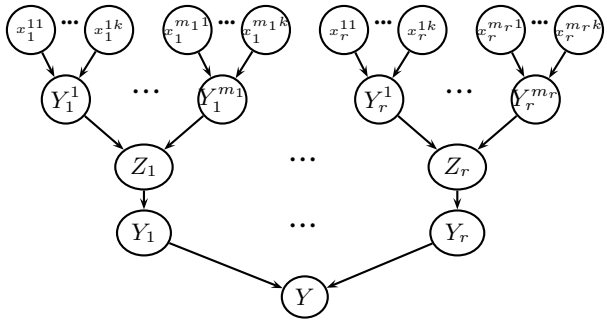

Figure 1. Value based Bayesian Network for the Noisy-OR combining rule. The nodes $Z_1 \ldots Z_r$ are distributed according to the mean of their parent distributions. $P(Y_i = 1|Z_i = 1) = q_i$ and $P(Y_i = 1|Z_i = 0) = 0$. Y is a deterministic OR function of $Y_1, \ldots, Y_r$.

We write $x_i^{j,1}, \ldots, x_i^{j,k} \equiv \mathbf{x}_i^j$ to denote the values of $\mathbf{X}_i^j$ and y to denote the value of Y. We write $\theta_{y|\mathbf{x}_i}$ to denote $P_i(y|\mathbf{x}_i)$. We omit the superscript j because the parameters θ are shared across the different instantiations of the same rule. We use l to index an example, and use y_l and $\mathbf{x}_l$ to denote the Y-value and the X-vector of the l^{th} example.

Intuitively, Noisy-OR models the case where there are multiple causes, any one of which can cause the target effect. However, this effect is disabled with some probability $(1 - q_i)$ independently of each other (see Figure 1). Thus, it is modeled by the following function, where all variables are binary. The variables Z_i are the parent variables and Y is the child variable:

$$P(Y = 1 \mid Z_1, \ldots, Z_n) = 1 - \prod (1 - q_i)^{Z_i}$$

Here q_i is defined as $P(Y = 1|Z_i = 1, \not\forall j \neq i, Z_j = 0)$. If all Z_i's are 0, Y is zero. Otherwise, Y is 0 if all its 1-inputs are disabled, and 1 otherwise. Noisy-OR is equivalent to a network where the inputs $Z_1, \ldots, Z_r$ are transformed to $Y_1, \ldots, Y_r$, such that $P(Y_i = 1 \mid Z_i = 1) = q_i$, $P(Y_i = 1 \mid$

$Z_i = 0) = 0$, and Y is a deterministic OR function of Y_i's. The distribution of the value at each Z_i is the mean of the distributions of the values of its parent nodes, $Y_i^1, \ldots, Y_i^k$.

A. Gradient for Mean Squared Error

We now derive the gradient equations for the mean-squared error function for the prediction of the target variable, when multiple FOCI-statements are combined by the Noisy-OR combining rule. Let the l^{th} training example e_l be denoted by $(\langle x_{l,1}^{1,1}, \ldots, x_{m_{l,r_l},k}^{l,r_l} \rangle, y_l)$. Recall that $x_{l,i}^{j,p}$ is the p^{th} input value of the j^{th} instance of the i^{th} rule of e_l. Here we exploit the Independence of Causal Influence (ICI) of Noisy-OR. Since the output variable Y is 0 only when all its immediate inputs $Y_1, \ldots, Y_r$ are 0's, and their influences on Y are independent of each other, the predicted probability of class y on e_l can be decomposed as follows:

$$
\begin{aligned}
&P(y=1|e_l) \\
&= 1 - \prod_i^{r_l} \frac{1}{m_{l,i}} \sum_j^{m_{l,i}} \Big[P_i(y=0|\mathbf{x}_{l,i}^j) + (1-q_i) P_i(y=1|\mathbf{x}_{l,i}^j) \Big] \\
&= 1 - \prod_i^{r_l} \frac{1}{m_{l,i}} \sum_j^{m_{l,i}} \Big[1 - q_i + q_i P_i(y=0|\mathbf{x}_{l,i}^j) \Big]
\end{aligned}
$$

r_l is the number of rules the example satisfies, i is an index of the applicable rule, and $m_{l,i}$ is the number of instances of rule i on the l^{th} example. The squared error is given by

$$
\begin{aligned}
E &= \tfrac{1}{2} \sum_{l=1}^{n} \sum_y \left(I(y_l, y) - P(y|e_l) \right)^2 \\
&= \tfrac{1}{2} \sum_{l=1}^{n} \big[\left(I(y_l, y=0) - P(y=0|e_l) \right)^2 \\
&\qquad + \left(I(y_l, y=1) - P(y=1|e_l) \right)^2 \big] \\
&= \tfrac{1}{2} \sum_{l=1}^{n} \big[\left(I(y_l, y=0) - P(y=0|e_l) \right)^2 \\
&\qquad + \left(I(y_l, y=1) - (1 - P(y=0|e_l)) \right)^2 \big]
\end{aligned}
\tag{1}
$$

Here y is a class label, and y_l is the true label of l^{th} example. $I(y_l, y)$ is an indicator variable that is 1 if $y_l = y$ and 0 otherwise. Taking the derivative of negative squared error with respect to $P(y|\mathbf{x}_i) = \theta_{y|\mathbf{x}_i}$, we get

$$
\frac{-\partial E}{\partial \theta_{y|\mathbf{x}_i}} = \sum_{l=1}^{n} \big[1 + I(y_l, y=0) - I(y_l, y=1) - 2P(y=0|e_l) \big] \cdot \delta(e_l) \tag{2}
$$

$$
\delta(e_l) = q_i \frac{\#(\mathbf{x}_i|e_l)}{m_{l,i}} \prod_{i' \neq i} \frac{1}{m_{l,i'}} \sum_j \big[1 - q_i + q_i P_{i'}(y=0 \mid \mathbf{x}_{l,i'}^j) \big] \tag{3}
$$

B. Gradient for Log-Likelihood

We now give the derivation of the gradient for log-likelihood with Noisy-OR as the combining rule. The log-likelihood LL is given by,

$$
LL = \sum_l \log P(y_l|e_l) \tag{4}
$$

$$
\begin{aligned}
&P(y=1|e_l) \\
&= 1 - \prod_i^{r_l} \frac{1}{m_{l,i}} \sum_j^{m_{l,i}} P_i(y=0|\mathbf{x}_{l,i}^j) + (1-q_i) P_i(y=1|\mathbf{x}_{l,i}^j) \\
&= 1 - \prod_i^{r_l} \frac{1}{m_{l,i}} \sum_j^{m_{l,i}} 1 - q_i + q_i P_i(y=0|\mathbf{x}_{l,i}^j).
\end{aligned}
\tag{5}
$$

Taking the derivative of the likelihood with respect to $P(y|\mathbf{x}_i) = \theta_{y|\mathbf{x}_i}$ we get,

$$
\begin{aligned}
\frac{\partial L}{\partial \theta_{y|\mathbf{x}_i}} &= \sum_l \frac{1}{P(y_l|e_l)} \frac{\partial P(y_l|e_l)}{\partial \theta_{y|\mathbf{x}_i}} \\
&= \sum_l \left[\frac{1}{P(y_l|e_l)} (-1)^y \delta(e_l) \right]
\end{aligned}
\tag{6}
$$

where $\delta(e_l)$ is as defined in (3). The gradient in the above equation will have different signs corresponding to whether the final value of y is a 0 or an 1.

C. Expectation Maximization for Noisy-OR

In their work, Natarajan et al. [5], developed an EM-algorithm that uses the notion of responsibilities to compute the parameters of the CPTs and the weights of the weighted mean combining rule. Since Noisy-OR is asymmetric with respect to its values, and all its inputs are responsible for the output, we directly estimate the expectations of values of random variables of interest without using auxiliary random variables such as responsibilities. Previous work on implementing EM for Noisy-OR models includes [7] for propositional networks and [8] for PRMs. Since our formulation involves multiple levels of combining rules, our approach is different from the prior work and is derived from first principles using the value-based network of Figure 1.

We now describe the EM procedure to learn the CPTs for Y_i^j's. In the E-step of EM, we seek to estimate the value of Y_i^j for each i, j pair and each value of the target variable Y. First, let us consider the easy case of $Y = 0$. Since Y is a result of OR'ing its parents, each of Y's parents including Y_i must be 0.

$$
\begin{aligned}
&P(Y_i^j = 1 \mid Y = 0, \mathbf{x}) = P(Y_i^j = 1 \mid Y_i = 0, \mathbf{x}) \\
&= \alpha P(Y_i^j = 1 \mid \mathbf{x}) P(Y_i = 0 \mid Y_i^j = 1, \mathbf{x}) \\
&= \alpha P(Y_i^j = 1 \mid \mathbf{x}) \cdot \\
&\quad \Big(P(Y_i = 0 \mid Z_i = 0) P(Z_i = 0 \mid Y_i^j = 1, \mathbf{x}) \\
&\quad + P(Y_i = 0 \mid Z_i = 1) P(Z_i = 1 \mid Y_i^j = 1, \mathbf{x}) \Big) \\
&= \alpha \cdot P(Y_i^j = 1 \mid \mathbf{x}) \cdot \frac{1}{m_i} \cdot \\
&\left(1 - q_i + \sum_{1 \leq j' \neq j}^{m_i} P(Y_i^{j'} = 0 \mid \mathbf{x}) + (1 - q_i) P(Y_i^{j'} = 1 \mid \mathbf{x}) \right)
\end{aligned}
\tag{7}
$$

The last line follows from expanding the Mean combining rule, noting that $P(Y_i = 0 \mid Z_i = 0) = 1$, $P(Y_i = 0 \mid Z_i =$

$1) = (1 - q_i)$, and $Y_i^j = 1$, and simplifying. Similarly, for $Y_i^j = 0$, we have:

$$P(Y_i^j = 0 \mid Y = 0, \mathbf{x}) = \alpha \cdot P(Y_i^j = 0 \mid \mathbf{x}) \cdot \frac{1}{m_i} \cdot \left(1 + \sum_{1 \le j' \ne j}^{m_i} P(Y_i^{j'} = 0 \mid \mathbf{x}) + (1 - q_i) P(Y_i^{j'} = 1 \mid \mathbf{x})\right) \quad (8)$$

We can determine α by normalizing Equations (7) and (8). Now we proceed to $Y = 1$, which is only slightly more complicated and obtain

$$\begin{aligned} & P(Y_i^j = 1 \mid \mathbf{x}, Y = 1) = \alpha P(Y_i^j = 1, Y = 1 \mid \mathbf{x}) \\ & = \alpha P(Y_i^j = 1 \mid \mathbf{x}) P(Y = 1 \mid Y_i^j = 1, \mathbf{x}) \\ & = \alpha P(Y_i^j = 1 \mid \mathbf{x}) \left(1 - P(Y = 0 \mid Y_i^j = 1, \mathbf{x})\right) \\ & = \alpha P(Y_i^j = 1 \mid \mathbf{x}) \left(1 - \prod_{1 \le i'}^{r} P(Y_{i'} = 0 \mid Y_i^j = 1, \mathbf{x})\right) \\ & = \alpha P(Y_i^j = 1 \mid \mathbf{x}) \left[1 - \frac{1}{m_i} \cdot \left((1 - q_i) + \sum_{1 \le j' \ne j}^{m_i} P(Y_i^{j'} = 0 \mid \mathbf{x}) + (1 - q_i) P(Y_i^{j'} = 1 \mid \mathbf{x})\right) \cdot \prod_{1 \le i' \ne i}^{r} P(Y_{i'} = 0 \mid \mathbf{x})\right] \end{aligned} \quad (9)$$

The last but one step follows from the fact that the only way $Y = 0$ is if all its parents are 0s, and that they are all independent of each other given $\langle \mathbf{x}, Y_i^j \rangle$. The last step then expands $P(Y_i = 0 \mid Y_i^j = 1, \mathbf{x})$ by marginalizing over Z_i, expanding the Mean combining rule, and using the fact that $Y_{i'}$ is independent of Y_i^j for all $i' \ne i$. Similarly,

$$P(Y_i^j = 0 \mid \mathbf{x}, Y = 1) = \alpha \cdot P(Y_i^j = 0 \mid \mathbf{x}) \cdot \left[1 - \frac{1}{m_i} \cdot \left(1 + \sum_{1 \le j' \ne j}^{m_i} P(Y_i^{j'} = 0 \mid \mathbf{x}) + (1 - q_i) P(Y_i^{j'} = 1 \mid \mathbf{x})\right) \cdot \prod_{1 \le i' \ne i}^{r} P(y_{i'} = 0 \mid \mathbf{x})\right] \quad (10)$$

We determine α by normalizing, as usual. Given the distributions of all $P(Y_i^j \mid \mathbf{x}, Y)$, for all i, j over all examples, the M-step is straightforward. We treat these probabilities as fractional counts and estimate the expected number of groundings j of rule i in which $\mathbf{X}_i^j = \mathbf{v}$ over all examples and out of these the number in which $Y_i^j = 1$. In other words, we estimate $n(Y_i^j = 1, \mathbf{x}_i^j = \mathbf{v}) = \sum_l \sum_{j:\mathbf{x}_{l,i}^j = \mathbf{v}} P(Y_i^j = 1 \mid \mathbf{x}_l, y_l)$. Here, the first index l is over the examples, and the second index j is over different instances of rule i whose influents are grounded to the values $\mathbf{v}$. $\mathbf{x}_l$ and y_l respectively denote the input vector and the output label of the l^{th} example. Similarly, $n(\mathbf{x}_i^j = \mathbf{v}) = \sum_l \sum_{j:\mathbf{x}_{l,i}^j = \mathbf{v}} 1$. We can estimate $P(Y_i^j = 1 \mid \mathbf{x}_j^i = \mathbf{v})$ as $\frac{n(Y_i^j = 1, \mathbf{x}_j^i = \mathbf{v})}{n(\mathbf{x}_j^i = \mathbf{v})}$ for any instance j.

IV. Experimental Results

In this section, we present the results on two real-world data sets and one synthetic data set.

A. Synthetic Data set for Noisy-OR

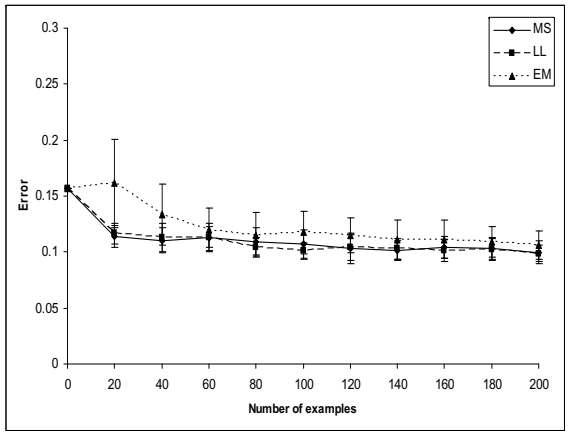

Figure 2. Learning curves. EM: Expectation Maximization; MS: Gradient descent for mean squared error; LL: Gradient descent for log-likelihood

To estimate the accuracy of the learned model using Noisy-OR, we constructed a synthetic data set. The data are generated using a synthetic target as defined by two FOCI statements, each of which has two influents and the same target attribute. The different instances of the same rule are combined using Mean and the different rules are combined using the Noisy-OR combining rule. The two influents in each rule have a range of 10 and 2 values respectively. The target attribute can take 2 values. The probability values in the distribution of the synthetic target are randomly generated to be either between 0.9 and 1.0 or between 0.0 and 0.1. This is to make sure that the probabilities are hard to predict and not too close to the default probability of $\frac{1}{2}$ each. Each example matches a rule with probability 0.5, and when it does match, it generates a number of instances randomly chosen between 3 and 10. This makes it imperative that the learning algorithm does a good job of inferring the hidden distributions both at the instance level and the rule level.

The goal is to evaluate the different versions of the Noisy-OR learning algorithms on this data set to determine the accuracy of the learned distributions. We trained the learning algorithms on 15 sets of 2000 training examples and tested them on a set of 1000 test examples. The average absolute difference between corresponding entries in the true distribution of the test examples and the predicted distribution was averaged over all the test examples. Since the gradient-descent methods optimize the MSE and the log-likelihood while the EM optimizes the log-likelihood, there is a need for comparing the different algorithms using the same performance metric. We choose the average absolute error for this purpose.

The results are presented in Figure 2. The x-axis has the number of examples and y-axis has the average absolute error for the examples. As can be seen, all the algorithms eventually converge to almost the same error rate (and there is no statistically significant difference). Initially, EM seems to perform worse, but, with more training data, achieves comparable performance to the gradient-descent methods. We flattened the data set by using the counts of the instances of the parents as features and used `Weka`[2] to run Naive Bayes on this modified data set. The Naive Bayes algorithm performed poorly and had a high error rate of close to 0.42, even with about 2000 training examples. Since the performance is poor, we omit the propositional classifier from the learning curves.

B. Real-World Data Sets

	Algorithm	UW	Citeseer
`Weighted Mean`	EM	0.7293	0.701
	GDMS	0.756	0.705
	GDLL	0.7406	0.664
`Noisy OR`	EM	0.7907	0.686
	GDMS	0.7962	0.678
	GDLL	0.7669	0.6544
`Alchemy`	MLN-2	0.5	0.34
	MLN-N	0.52	0.40

Table I. Results. GD-MS: Gradient descent for Mean Square error; GD-LL: Gradient descent for log-likelihood

We compared our relational algorithms against the ones developed by Natarajan et al. [5] and Markov Logic Networks [2] on two real-world data sets - UW data set and the `citeseer` data set. Since these are real-world data sets, it is not clear which combining rule can best fit the data without an empirical comparison.

1) UW Dataset: One test-bed was the UW-CSE domain, where the goal is to predict the `advisedBy` relationship between a professor and a student. This database consists of 278 faculty members and students. We used 2 rules to predict `advisedBy`. The rules are as follows:

```
CR{
  If {student(S), professor(P), course(C)}
   then taughtBy(P,C,Q), ta(S,C,Q)
    Qinf (Mean) advisedBy(S,P).
  If {student(S), professor(P)}
   then publication(P,W), publication(S,W)
    Qinf (Mean) advisedBy(S,P).
}
```

The first rule states that being a TA for a course that the professor offers influences the `advisedBy` relationship. The second rule states that being a co-author for a paper with a professor influences the `advisedBy` relationship with the professor. `CR` in the above rule denotes the combining rule and is either Weighted-Mean or Noisy-OR in our experiments. We predict the likelihood of the target predicate using 5-fold cross validation. We learn the parameters using 4 folds and predict the probability of the target predicate in the other fold and average the likelihood of the target predicate across all the test folds. Note that there are 4 independent parameters for each rule.

The results are presented in Table I. The results of the algorithms that use Noisy-OR are marginally better than the algorithms that use weighted mean. This is due to the fact that when one of the two rules is fired, Noisy-OR would yield a higher probability of the student being advised by the professor. Recall that Noisy-OR predicts that the possibility of a student being advised by a professor is false *if and only if* both the rules return false with high probability.

We used Alchemy system[3] to compare against MLNs. Markov Logic Networks (MLNs in short)[2], consider first-order clauses and softens them by learning their weights. The weight of a clause can be understood as the difference in log probability between a world that satisfies the clause and one that doesnt. While using MLNs, we use the following clauses: `student(S)` $\wedge$ `professor(P)` $\wedge$ `course(C)` $\wedge$ `taughtBy(P,C,Q)` $\wedge$ `ta(S,C,Q)` `:-` `advisedBy(S,P).`
`student(S)` $\wedge$ `professor(P)` $\wedge$ `publication(P,W)` $\wedge$ `publication(S,W)` `:-` `advisedBy(S,P).`

where `:-` denotes implication. The last two rows of the table present our results with MLNs. *MLN-2* is the result using the above 2 rules, and *MLN-N* is the result of using all possible combinations of the truth value of the predicates (8 weights overall). As can be seen, the performance of MLNs is not comparable to that of the directed models. We included 7 more rules provided in the UW dataset that used the *advisedBy* predicate and the performance improved (the average likelihood was close to 0.57). We also evaluated using most of the clauses (about 55 of them) in the UW-dataset where we excluded the clauses that used existential quantifiers and the ones that used the predicates such as *samePerson*, *sameCourse*, etc. for efficiency. This resulted in a much better performance where the average likelihood was 0.67. This suggests that the performance of MLNs is highly sensitive to the number and form of the rules.

2) Citeseer Data Set: The other testbed we used is the `citeseer` domain, where the goal is to predict if 2 citations refer to the same work. The data consisted of 4300 pairs of similar publications. We selected 500 of them at random. Each pair was assigned as positive with a probability of 0.7. This is to say that we made 30% of them negatives (in order to introduce noise). We then used two rules - first rule states that if the papers have similar title and same venue, they are likely to be similar. The second rule is a transitive rule on the target. The goal of this experiment was to verify if the algorithms can retrieve the true likelihood of the data.

[2] `http://www.cs.waikato.ac.nz/ml/weka/`

[3] `www.alchemy.cs.washington.edu/`

As can be seen from the last column of the Table I, the different learning algorithms are capable of learning the true likelihood of the data which is close to 0.70. In this domain, the weighted mean has a marginally better performance (although a very negligible one) compared to that of the Noisy-OR models. In this domain, both the rules are very informative and hence weighted mean has a good performance. There was no statistical significance in the likelihood between the two combining rules. This strengthens our hypothesis that we ultimately need a set of algorithms that can search over the space of combining rules to find the best one that fits the data. We ran Alchemy in this data set with the above 2 clauses. As with the previous domains, MLNs does not seem to capture the true model. These results inspire a new research direction of understanding the equivalence between MLNs and directed models that capture conditional distributions and use combining rules.

We speculate a few reasons - First, it appears that MLNs do not perform very well in the presence of a small number of rules and that their performance improve in the presence of a large number of weakly predictive rules. Secondly, we observe that Alchemy drives the weights of most clauses towards zero[2]. This is a very good solution when the clauses are being automatically learned from data (a.k.a. structure learning). But when provided with minimal number of rules from a domain expert, the weights should not be driven to zero. Another reason is that while our model can learn a set of locally predictive rules, MLNs are not currently capable of learning local models. Finally, the problem of representing arbitrary distributions using a minimal MLN remains an open problem.

V. Discussion

Rather than thinking of combining rules as an alternative to aggregators, one could think of decomposable combining rules as a way of adding special structure to the aggregation problem. This allows us to develop methods that exploit the structure. While our gradient-descent algorithms are derived directly from the combining rules perspective, our approach to the EM algorithm can be understood as aggregation-based, and is explained in terms of the value-based network, where the aggregator function is deterministic, e.g., OR, or stochastic, e.g., random. Thus, decomposable combining rules allow us to view the target distribution, both as a function of the input conditional distributions, and also as a consequence of decomposable causal structure in the corresponding value network. It is our hope that a wide variety of practical real world problems can be captured by a small number of simple decomposable combining rules, thus making arbitrarily complex aggregators unnecessary.

Prior work exists on developing an EM-based learning algorithm for Noisy-OR in the propositional case, notably [7]. Diez and Galan provided a factorization for the more generalized Noisy-MAX function and developed learning algorithms based on the factorization [9]. Learning the parameters of the first-order clauses has also been explored by previous researchers: Koller and Pfeffer (1997) [8] investigated the use of EM algorithm to learn the parameters of the first-order clauses in the presence of combining rules. More recently, Jaeger [4] considered a weighted combination or a nested combination of the combining rules and used a gradient-ascent algorithm to optimize the objective function. This technique has been applied to his formalism of Relational Bayesian Networks(RBNs).

This work can be naturally extended to more general classes of combining rules and aggregators including tree-structured CPTs and noisy versions of other symmetric functions. The relationship between the aggregators and combining rules must be better understood and formalized. Efficient inference algorithms must be developed that take advantage of the decomposability of the combining rules as well as the flexibility of the first-order notation. Finally, it is important to develop more compelling applications in knowledge-rich and structured domains that can benefit from the richness of the first-order probabilistic languages. Extending the SRL languages to dynamic domains with actions and utility makes them much more appropriate for compelling real-world applications.

References

[1] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer, "Learning probabilistic relational models," *Relational Data Mining*, 2001.

[2] P. Domingos and D. Lowd, *Markov Logic: An Interface Layer for AI*. Morgan and Claypool, 2008.

[3] K. Kersting and L. De Raedt, "Bayesian logic programs," in *Proc ILP*, 2000.

[4] M. Jaeger, "Parameter learning for relational bayesian networks," in *Proceedings of ICML*, 2007.

[5] S. Natarajan, P. Tadepalli, E. Altendorf, T. G. Dietterich, A. Fern, and A. Restificar, "Learning first-order probabilistic models with combining rules," in *Proceedings of ICML*, 2005.

[6] S. Lawrence, C. Giles, and K. Bollacker, "Autonomous citation matching," in *ICAA*, 1999.

[7] J. Vomlel, "Noisy-or classifier: Research articles," *Int. J. Intell. Syst.*, vol. 21, no. 3, pp. 381–398, 2006.

[8] D. Koller and A. Pfeffer, "Learning probabilities for noisy first-order rules," in *IJCAI*, 1997, pp. 1316–1323.

[9] F. J. Díez and S. F. Galán, "Efficient computation for the noisy MAX," *International Journal of Approximate Reasoning*, vol. 18, pp. 165–177, 2003.