Structure Refinement in First Order Conditional Influence Language

Sriraam Natarajan NATARASR@EECS.OREGONSTATE.EDU
Weng-Keen Wong WONG@EECS.OREGONSTATE.EDU
Prasad Tadepalli TADEPALL@EECS.OREGONSTATE.EDU
School of EECS, Oregon State University, USA

Abstract

In this paper, we present preliminary results from learning the structure of first-order conditional influence statements from data for the purpose of classification. In order to reduce the search space over structures, we formulate and address the structure learning problem as a problem of refining the structure of a first-order probabilistic program using training data. We use variants of the conditional BIC scoring metric to refine the program to best fit the data. We use a previously introduced language called FOCIL which consists of statements that can be instantiated and composed into a propositional Bayesian network. The results on a synthetic dataset and a real-world task show that the algorithm achieves error rates comparable to the gold standard program with a reasonable amount of training data.

1. Introduction

There have been several first-order probabilistic languages developed in recent years with different motivations, syntax, and semantics. These include probabilistic relational models (PRMs) (Getoor et al., 2001), Bayesian logic programs (BLPs) (Kersting & De Raedt, 2000), relational Bayesian networks (RBNs) (Jaeger, 1997), Markov logic networks (MLNs) (Domingos & Richardson, 2004) and many others. One principle attraction of these relational languages is that they are more succinct than their propositional counterparts, and hence their structure can be more easily specified by the domain experts. For example, in First-Order Conditional Influence Language (FOCIL) (Natarajan & Altendorf, 2005), one can specify the influents (parents in the Bayes nets jargon) of an attribute in a rule-like syntax with appropriate conditions. This allows us to naturally translate English sentences such as "the difficulty of the courses one takes and the work one puts into them affects one's grades" into the corresponding first-order conditional influence (FOCI) statements. Given a database of factual relationships between different entities, e.g. students, courses, and teachers, these statements can in turn be "unrolled" into a concrete Bayesian network where the nodes represent the attributes of specific objects which are reasoned about. Since the parameters of this concrete network are shared between all instantiations, the parameterization is much more succinct and learning is more data-efficient than in a propositional setting.

This raises the question and possibility of learning the structure of the FOCI-statements from data in order to predict the class of a target variable. Indeed, there have been some suggested algorithms for structure search in the frameworks of PRMs (Getoor et al., 2001) and MLNs (Kok & Domingos, 2005). However, structure search over the space of relational models remains difficult due to the combinatorially explosive search space of structures that is a result of the expressiveness of the first-order languages. Another challenging aspect of our task is the need to learn a discriminative classifier. We will use techniques from learning the structure of Bayesian network classifiers. However, in a Bayesian network setting, learning a discriminative model requires maximizing a scoring metric derived from the conditional likelihood. Computing the value of this scoring metric is computationally expensive (Friedman et al., 1997) but several approximations have been suggested (Grossman & Domingos, 2004; Guo & Greiner, 2005). The conditional approximation of the BIC scoring metric was proposed in (Guo & Greiner, 2005), while that of the MDL scoring metric was used in (Grossman & Domingos, 2004).

In this paper, we report preliminary results from learn-

Presented at the ICML Workshop on Open Problems in Statistical Relational Learning, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

ing the structure of FOCI-statements from data. Instead of an unconstrained structure search, we set ourselves a less ambitious goal. We formulate the problem of refining FOCI-statements from a partial specification. We call the partial specification of a set of FOCI-statements, a partial FOCI-program. Following the philosophy of dependency networks (Heckerman et al., 2000), we formulate one separate FOCI-program for each query variable of interest and learn them separately.

FOCIL is especially conducive for the domain expert to specify sparse network structures because each rule groups together potentially related influents. Thus, the search is more constrained than say, in relational probability trees, where the learning algorithm has to consider all the potential influents while picking the best influent for the current split (Neville et al., 2003). The structure search can facilitate easy specification by experts by taking over the responsibility to find the exact set of influents starting from a larger superset. In the future we will also allow the expert to specify a set of possible combining rules, e.g., noisy-or, noisy-max, weighted mean, etc. and allow the structure search to pick the one that best fits the data. Our work is also inspired by the work of (Heckerman et al., 1994) where a Bayesian network structure search is conducted starting from an initial structure and by defining a prior which penalizes large deviations from it. In contrast, our work limits the structure search to be consistent with the user-defined partial FOCI-program and allows the specification of a prior over the consistent FOCI-programs.

The rest of the paper is organized as follows: In the next section we provide some background on our language. In the third section, we formulate the problem of learning the FOCI statements given a partial FOCI program. We also derive a conditional BIC score for our problem and outline the search. Next, we provide empirical evaluation of the algorithms on two domains: a folder prediction data set and a synthetic data set. We then conclude the paper by outlining some areas for future research.

2. First Order Conditional Influence Language

In this section, we summarize the main features of first-order conditional influence language (FOCIL) which forms the basis of our work (Natarajan & Altendorf, 2005). The core of FOCIL consists of first-order conditional influence (FOCI) statements, which are used to specify probabilistic influences among the attributes of objects in a given domain. The domain expert is assumed to know the structure of the ER diagram while he or she writes down the FOCI statements. Each FOCI statement has the form:

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

where *condition* is a conjunction of literals, each literal being a predicate symbol applied to the appropriate number of variables. The conditions are used to identify the objects that participate in the qualitative influence. A ⟨*qualitative influence*⟩ is of the form $X_1, \ldots, X_k$ *Qinf* Y, where the X_i and Y are of the form $V.a$, where V is a variable in *condition* and a is an object attribute. This statement simply expresses a directional dependence of the *resultant* Y on the *influents* X_i. Associated with each FOCI 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. Consider for example the FOCI statement,

```
If {Person(p)} then p.diettype Qinf
p.fitness.
```

It means that a person's type of diet influences their fitness level. There is an entity *Person* and *diettype* and *fitness* are two attributes of *Person*. As can be seen, the influent and the resultants are of the form *Object.attribute*. Also, the condition is used to determine the type of the variable p, which in this case is a *Person*. The conditional probability distribution $P(p.\text{fitness} \mid p.\text{diettype})$ associated with this statement (partially) captures the quantitative relationships between these attributes. We refer the reader to (Natarajan & Altendorf, 2005) for more details about the language.

Consider an intelligent desktop assistant that must predict the folder of a document to be saved. Assume that there are several tasks that a user can work on, such as proposals, courses, budgets, etc. The following FOCI statement says that a task and the role the document plays in that task influence its folder.

```
If {task(t), document(d), role(d,r,t)} then
     t.id,r.id Qinf d.folder.
```

Typically a document plays several roles in several tasks. For example, it may be the main document of one task but only a reference in some other task. Thus there are multiple task-role pairs $(t_1, r_1), \ldots, (t_m, r_m)$, each yielding a distinct folder distribution $P(d.folder \mid t_i.id, r_i.id)$. We need to combine these distributions into a single distribution for the folder variable.

In FOCIL, a combining rule is applied to combine the distributions due to different influent instances of a single FOCI statement. In addition, combining rules can be employed to combine distributions arising from multiple FOCI statements with the same resultant. The following example captures such a case (see Figure 2 for the unrolled network):

```
WeightedMean{
  If {task(t), doc(d), role(d,r,t)} then
    t.id, r.id Qinf (Mean) d.folder.
  If {doc(s), doc(d), source(s,d)} then
    s.folder Qinf (Mean) d.folder.}
```

Figure 1. Example of specifying combining rules in FOCIL.

The expression in Figure 1 includes two FOCI statements. One statement is the task-role influence statement discussed above. The other says that the folder of the source document of d influences d's folder. The source of a document is a document that was edited to create the current document. There can be multiple sources for a document. There are two distributions $P(d.folder \mid t_i.id, r_i.id)$ and $P(d.folder \mid s_i.folder)$. The distributions corresponding to different instances of the influents in the same statement are combined via the *mean* combining rule (indicated by the keyword "Mean"). The two resulting distributions are then combined with a *weighted mean* combining rule. Equation 5 shows the computation of the probability of a resultant given the set of instantiations of its influents. In our earlier work(Natarajan et al., 2005), we have implemented algorithms based on Gradient-descent and EM to learn the parameters of the conditional probability distributions and also the weights of the weighted mean. In this work, we consider the problem of refining the structure of FOCI statements from data.

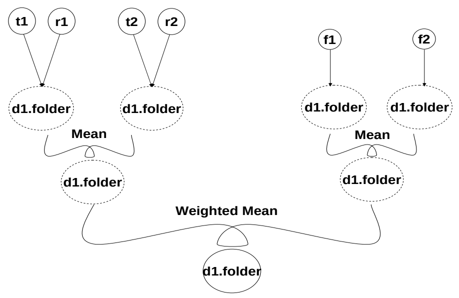

Figure 2. Use of Combining rules to combine the influences of task and role on the one hand and the source folder on the other on the folder of the current document.

3. Partial FOCI-programs and Structure Learning

In this section, we present an algorithm for learning the structure of the FOCI statements, starting from a partial FOCI-program. A partial FOCI-program consists of a set of partial FOCI-statements, each of which specifies some constraints on the target FOCI-statements. In this paper, we only consider refinements obtained by dropping the influents of the rules in the partial program. Other constraints which we have not implemented, might include a min or max constraint on the number of parents, a collection of possible combining rules only one of which is to be chosen, and constraints on the rule conditions. The refinement problem of partial FOCI-programs is taking as input a partial FOCI program and some training data and producing a FOCI-program and a set of parameters that most correctly classify the target variable given the inputs.

We will approach the refinement problem of partial FOCI-programs in a manner similar to learning the structure of Bayesian networks. In Bayesian network structure search, the number of possible structures to be considered is superexponential in the number of variables (Robinson, 1973). (Chickering, 1996) shows that Bayesian network structure learning is NP-hard. Therefore, the majority of structure learning algorithms typically use local search techniques. These local search techniques use operators such as adding, deleting, or reversing edges to search over the space of possible DAGs. The candidate structures are scored using a scoring metric and the search returns with a best-scoring Bayesian network structure.

Since we are dealing with relational models, the complexity of structure learning is even greater than that of learning Bayesian networks. We will employ an exhaustive search and a hill climbing procedure to search over the space of possible FOCI statements. However, we will only consider a highly restricted search space, namely the FOCI-programs obtained by dropping the influents from the FOCI-statements in the input partial program.

3.1. Scoring Metric

We need a scoring metric to guide the search for the best structure. We use a variant of the Bayesian Information Criterion (BIC) (Schwarz, 1978). Suppose we have a set of candidate models M_i, $i = 1, ..., m$ for the given data set and the corresponding model parameters θ_i and we are interested in choosing the model that best fits the data D. Then the posterior probability of the model given the data is

$$P(M \mid D) \propto P(M) \cdot P(D \mid M) \tag{1}$$

Applying a Taylor series expansion to the posterior probability followed by the use of Laplace method for integrals yields the BIC score (Raftery, 1995). The BIC score for a given model is

$$\begin{aligned} BICScore &= -2 * log(P(D \mid \hat{\theta}, M)) + d_m log N \\ &= -2 * loglikelihood + d_m log N \end{aligned} \tag{2}$$

where $log(P(D \mid \hat{\theta}, M))$ is the likelihood of the data given the model M and the maximum likelihood estimate $\hat{\theta}$. The second term can be considered as a penalty term that penalizes complex structures, i.e., structures with nodes having a large number of parents and a large number of parameters. Since we are interested in learning the FOCI programs for a single "target" attribute, our problem belongs to the discriminative setting rather than the generative setting which aims to model the entire set of objects and their attributes. Hence it is more appropriate to optimize the conditional likelihood instead of the joint likelihood over all the variables. Several others have advocated using conditional likelihood as the scoring metric (Guo & Greiner, 2005; Stanford & Raftery, 2002; Bouchard & Celeux, 2006). We define the problem of learning the FOCI program for a target resultant Y as that of finding a set of r rules, each with a set of influents $\mathbf{X}^i$. The conditional BIC (CBIC) scoring metric for our problem is thus defined as

$$\begin{aligned} CBICScore &= -2 * log(P(Y \mid \mathbf{X}^1 ... \mathbf{X}^r, \hat{\theta}, M)) \\ &\quad + d_m log N \\ &= -2 * CLL + d_m log N \end{aligned} \tag{3}$$

where CLL is the conditional log-likelihood of the target variable given the set of influents. We now show how to compute the log-likelihood and the number of free parameters in our setting.

First, let us look at the number of free parameters in our setting. Consider the network shown in Figure 3. It should be noted that since we are using combining rules to combine the distributions, the nodes $1, 2, ..., m_1$ on the left side of the Figure all share the CPTs, i.e., they all have the same distribution $P(Y \mid X^j_{i,1}, ... X^j_{i,k})$. Hence the number of free parameters in the left part of the network is the number of free parameters in this CPT. If Y takes l values, the number of free parameters is $N_{\mathbf{X}^1} \times (l-1)$, where $N_{\mathbf{X}^1}$ refers to the number of parent configurations of the first rule. The same argument is true for the second rule. Hence the total number of free parameters

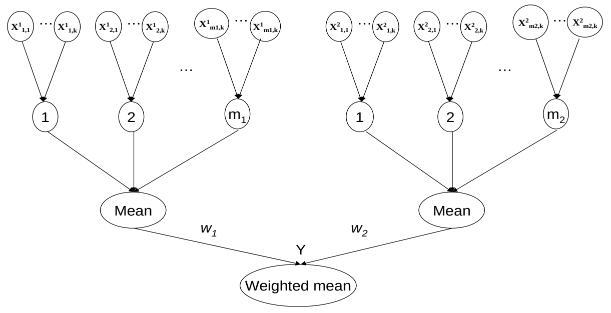

Figure 3. An example of "unrolled" network

is the sum of the free parameters in the CPTs of all the rules which is given by

$$d_m = (l-1) \times \sum_i N_{\mathbf{X}^i} \tag{4}$$

It has been noted that the above CBIC score underfits data by penalizing large networks too heavily (Guo & Greiner, 2005). We also empirically observed that the penalty is too large when the domain size l of the resultant Y is high. To compensate for this, we scaled the penalty by dividing it by $l-1$, i.e, we use $\frac{d_m}{l-1} \log N$ as the penalty.

The conditional likelihood of the target resultant given the set of influents is given by,

$$P(Y \mid X^1_{1,1} ... X^r_{m_r,k}) = \frac{\sum_{i=1}^{r} w_i \frac{1}{m_i} \sum_{j=1}^{m_i} P_i(y \mid \mathbf{X}^i_j)}{\sum_{i=1}^{r} w_i} \tag{5}$$

We then sum over the log-values of the conditional likelihoods for all the data cases and use it in the CBIC scoring metric.

$$\begin{aligned} CBICScore = -2 * \sum_d [log(P(Y \mid X^1_{1,1} ... X^r_{m_r,k}))] \\ + \sum_i N_{\mathbf{X}^i} log N \end{aligned} \tag{6}$$

Note that $r = 2$ for the network in Figure 3.

The CPTs and the weights of the mean combining rule are computed using the EM algorithm presented in our earlier paper(Natarajan et al., 2005). In the *expectation* step, we compute the responsibility of each instantiation of each rule. The responsibilities reflect the relative density of the training points under each rule (Hastie et al., 2001). We consider the weight of the current rule and the number of instantiations while computing the responsibility of an instantiation of the current rule. In the *maximization* step, we use these responsibilities to update the CPTs. Note that EM finds the parameters that maximize joint loglikelihood estimate rather than the conditional loglikelihood, which is the more correct thing to maximize. However, in previous work it has been found to give good results with less computational effort (Grossman & Domingos, 2004). One possible future direction is to use the gradient descent to optimize the conditional loglikelihood as in (Natarajan et al., 2005). We use the responsibilities of the instantiations of an example to compute the weights if at least two rules are instantiated in the example. If an example matches less than two rules, the weights do not affect the distribution. Consider the update of $P(y_1 \mid \mathbf{x}^i)$. This is the fraction of the sum of all the responsibilities when $Y = y_1$ over all Y given $\mathbf{x}^i$. Likewise, the weight of the current rule is the fraction of the sum of the responsibilities of all instantiations of the rule over the number of examples with two or more rules instantiated.

3.2. Structure Search

In general, a structure learning algorithm has to specify the search operators in the structure space and the search strategy. In our case, the only operators in the search space are to add or drop an influent (or a rule) from the given set of influents. Figures 4 and 5 provide an example of the prior set of FOCI statements that are specified by the domain expert and the set of FOCI statements that are learned by the learning algorithm.

```
WeightedMean{
  If {task(t), doc(d), role(d,r,t)} then
    t.id, r.id, t.creationDate, t.LastAccessed
    Qinf (Mean) d.folder.
  If {doc(s), doc(d), source(s,d)} then
    s.folder, s.creationDate
    Qinf (Mean) d.folder.}
```

Figure 4. Example of a prior set of FOCI statements

```
WeightedMean{
  If {task(t), doc(d), role(d,r,t)} then
    t.id, r.id Qinf (Mean) d.folder.
  If {doc(s), doc(d), source(s,d)} then
    s.folder Qinf (Mean) d.folder.}
```

Figure 5. An example of the set of rules that could be learned by the algorithm

Given the possible influents, we use a greedy search with random restarts to search over the space of FOCI statements and select the set of FOCI statements that has the minimum CBIC score. We start with an empty FOCI program and then search through the space of structures by adding influents and the corresponding rules. We continue the search through the space until we reach a local maxima. Once we reach a local maxima, we restart the search from a random configuration in the structure space. As an example, consider the partial program presented in Figure 4. There are 4 influents in the first rule and 2 influents in the second. Hence the total number of possible structures is $2^4 2^2$ corresponding to whether each influent is present or not. Note that when all the 4 influents of the first rule are absent, the second rule alone fires and vice-versa. The exhaustive search would start from an empty program and searches through all the 2^6 structures. In the greedy hillclimbing with random restarts, we start with the empty network and search through the space of structures until we reach a local maxima. Once a local maxima is reached, we randomly start with some structure and continue the search. The CBIC score that we derived is used to compare different structures. It must be noted that to compute the CBIC score, in the inner loop we use EM to compute the parameters of the CPTs and the weights that best fit the data given the current model. In this work, we assume that the appropriate combining rules are given.

4. Experiments and Results

In this section, we describe results on the two data sets that we employed to test the learning algorithms. The first is based on the folder prediction task, where we applied two rules to predict the folder of a document from other information. The second data set is a synthetic one where there are two rules as well.

4.1. Folder Prediction

As part of the Task Tracer project (Dragunov et al., 2005), we collected data for 500 documents and 6 tasks. The documents were stored in 11 different folders. Each document was manually assigned to a role with respect to each task with which it was associated. A document was assigned the *main* role if it was modified as part of the task. Otherwise, the document was assigned the *reference* role, since it was opened but not edited. A document is a *source* document if it was opened, edited, and then saved to create a new document or if large parts of it were copied and pasted into the new document. Since the documents could play several roles in several tasks, the number of $\langle t, r \rangle$ pairs vary[1]. Hence, the data set consisted of the following, the document's folder the set of source documents' folder, and the set of tasks and the roles the document played in those tasks.

Rank	Exhaustive -R	HC+RR - R	Exhaustive - I	HC+RR - I
1	349	354	312	311
2	107	98	128	130
3	22	26	26	26
4	15	12	20	23
5	6	4	3	4
6	0	0	1	1
7	1	4	2	0
8	0	2	1	2
9	0	0	0	0
10	0	0	0	0
11	0	0	2	3
Score	0.8299	0.8325	.7926	0.7841

Figure 6. Results of the Exhaustive and the hill-climbing algorithms on the folder prediction task. The results are averaged over 20 runs. *Exhaustive-R* and *HC+RR-R* refers to the algorithms when provided with only the relevant influents and *Exhaustive-I* and *HC+RR-I* refer to the algorithms with irrelevant attributes.

We performed two kinds of experiments: One to test if the learning algorithm when provided with just the relevant influents was able to retain all the relevant influents and the second to test if provided with irrelevant attributes, the algorithm learns to ignore these irrelevant attributes. We used the data set described above for the first experiment, and in the second we used the FOCI program in Figure 5 where we randomized the values of *creationDate* and *lastAccessed* attributes. We used two forms of the structure learning algorithm. One that searches exhaustively through the space of structures and the other that uses hill climbing with random restarts for search. The results were averaged over 20 runs of the algorithms.

We employed 10-fold cross-validation to evaluate the results. Within each fold, the learned network was applied to rank the folders of the current document and the position of the correct folder in this ranking was computed (counting from 1). The results are shown in Figure 6, where the counts report the total number of times (out of 500) that the correct folder was ranked first, second, and so on. The final row of the table reports the mean reciprocal rank of the correct folder (the average of the reciprocals of the ranks). The mean reciprocal rank would be 1 if the correct folder is ranked at the top in all tests. We compare the exhaustive search with the hill climbing with random restarts. The results are averaged over 20 runs. In the first experiment where the algorithms are provided with only the relevant attributes, it can be observed from the second and third columns of Figure 6 that the greedy hill-climbing with random restarts achieves a comparable mean reciprocal rank to that of the exhaustive search.The exhaustive search retrieved the "true" network in all the runs in the first experiment. On the other hand, hill climbing with random restarts retrieved the true network in 17 of the 20 runs. In the other 3 runs, the hill climbing algorithm learned a network with only the second rule.

[1]On average, each document participated in 2 $\langle t, r \rangle$ pairs, although a few documents participated in 5 to 6 $\langle t, r \rangle$ pairs.

In the second experiment the algorithm ignored the irrelevant attributes in the first rule and retrieved the true influents. In the second rule, however, it included the randomized *creationDate* as the influent, instead of the *folder* of the source. This is because the penalty for including the source *folder* is higher than the penalty for including the *creationDate* as the CPT is much larger in the former case. Hence the CBIC score of the *learned* FOCI program was lower than the program that had the best mean reciprocal rank. Despite not using the best influent in the second rule, the mean reciprocal rank of the learned network was comparable to the mean reciprocal rank of the best network, as can be seen in the 4^{th} column of Figure 6. This is because only a few documents had other documents as sources and hence the rule did not fire in many of the training examples. The hill-climbing search with irrelevant attributes (column 5 in the Figure 6) had a reasonably good mean reciprocal rank on all the runs except one and hence has a comparable score to that of the exhaustive search.

4.2. Synthetic dataset

To estimate the accuracy of the learned probabilistic model, we used the a synthetic data set for our experiments that was also used in (Natarajan et al., 2005). 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 two influents in each rule have a range of 10 and 3 values respectively. The target attribute can take 3 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 probabilistic predictions on examples are not too uncertain. The rule weights are fixed to be 0.1 and 0.9 to make them far from the default, 0.5. 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 at the rule level.

Again, we use this dataset to perform two kinds of experiments: the first one to determine whether the learning algorithm when given the true set of influents learns to retrieve the true structure, and the second one to determine whether the learning algorithm can ignore irrelevant attributes. We present the results of both the experiments here.

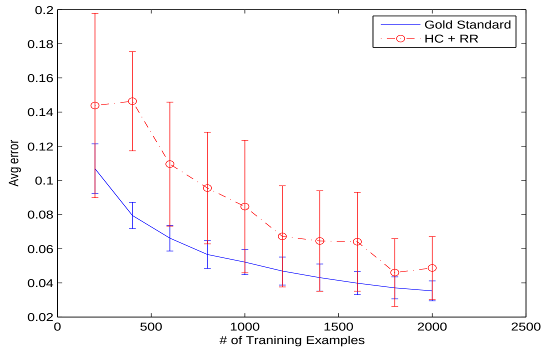

Figure 7. Learning curves in the synthetic domain with only the relevant influents on 30 data sets

We trained the learning algorithms on 30 sets of 2000 training examples and tested them on a set of 1000 test examples. The hill climbing with random restarts learned the gold-standard structure for 18 of the 30 data sets[2]. The performance of the hill-climbing depends on the number of restarts that are allowed. In our experiments, we allowed three restarts. The average absolute difference between corresponding entries in the true distribution and the predicted distribution was averaged over all the test examples. The results are presented in Figure 7. The Figure has two curves: one for the gold standard network and the other for the hill climbing with random restarts. It can be observed that though the hill climbing search did not find true structure in 12 of the 30 datasets, there is no statistically significant difference in the average errors when compared to the true network. Also, with smaller number of examples, the hill climbing is not able to obtain the true network while with larger samples it retrieved the true network more often.

In the second experiment, we added 2 irrelevant influents of domain size 10 to both the rules such that their values are chosen at random. The results are presented in Figure 8. The results were averaged over 14 data sets and the learning curves are presented. As can be seen from the Figure, there is no significant difference between the error rates of the true network and the network learned by the hill-climbing algorithm for the larger samples.

[2]We also ran an exhaustive algorithm to determine if the CBIC score was useful to retrieve the correct structure and the exhaustive algorithm learned the true network on all the datasets.

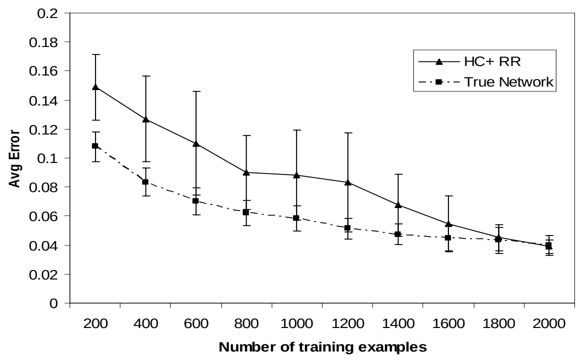

Figure 8. Learning curves in the synthetic domain with irrelevant influents on 8 datasets

5. Conclusion and Future work

In this paper, we considered the problem of learning the structure of the FOCI statements given a partial FOCIL program. The assumption here is that the data is very expensive in many domains and hence cannot be used for searching through the entire space of structures. Also, since learning the entire network is hard, we propose to learn a set of FOCI statements for every query variable. The motivation behind this is the work of dependency networks which was later extended to the relational setting (Neville & Jensen, 2004). We derived the CBIC scoring metric for our setting and also outlined a greedy search through the space of structures. We also empirically showed that our learning algorithm with the CBIC scoring metric was able to retrieve the "true" structure in two domains: a real-world dataset and a synthetic dataset. In the folder data set, the CBIC score of the true network was higher than the one that was learned. We are currently looking at other scoring metrics that could be better suited for the relational setting e.g., Bias-Variance metric that was used in (Guo & Greiner, 2005).

Another possible direction is to investigate the use of a scoring metric like the BDeu scoring metric (Heckerman et al., 1994) which naturally provides a way to specify priors over structures that are consistent with the partial FOCI program. Also it would be useful to have the learning program choose a most accurate combining rule from a given set of candidate rules. An important direction to pursue next is the use of this structure refinement methodology in larger real world domains.

References

Bouchard, G., & Celeux, G. (2006). Selection of generative models in classification. *IEEE Trans. Pattern Anal. Mach. Intell.*, *28*.

Chickering, D. M. (1996). Learning equivalence classes of Bayesian network structures. *UAI* (pp. 150–157).

Domingos, P., & Richardson, M. (2004). Markov logic: A unifying framework for statistical relational learning. *Proceedings of the SRL Workshop in ICML*.

Dragunov, A. N., Dietterich, T. G., Johnsrude, K., McLaughlin, M., Li, L., & Herlocker, J. L. (2005). Tasktracer: A desktop environment to support multi-tasking knowledge workers. *Proceedings of IUI*.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning*, *29*, 131–163.

Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. (2001). Learning probabilistic relational models. *Invited contribution to the book Relational Data Mining, S. Dzeroski and N. Lavrac, Eds.*

Grossman, D., & Domingos, P. (2004). Learning bayesian network classifiers by maximizing conditional likelihood. *Proceedings of ICML '04*.

Guo, Y., & Greiner, R. (2005). Discriminative model selection for belief net structures. *AAAI* (pp. 770–776).

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The Elements of Statistical learning*. Springer.

Heckerman, D., Chickering, D. M., Meek, C., Rounthwaite, R., & Kadie, C. M. (2000). Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, *1*, 49–75.

Heckerman, D., Geiger, D., & Chickering, D. M. (1994). Learning bayesian networks: The combination of knowledge and statistical data. *KDD Workshop* (pp. 85–96).

Jaeger, M. (1997). Relational Bayesian networks. *Proceedings of UAI-97*.

Kersting, K., & De Raedt, L. (2000). Bayesian logic programs. *Proceedings of the Work-in-Progress Track at ILP*.

Kok, S., & Domingos, P. (2005). Learning the structure of markov logic networks. *Proceedings of ICML* (pp. 441–448). New York, NY, USA: ACM Press.

Natarajan, S., & Altendorf, E. E. (2005). *First order conditional influence language* (Technical Report CS05-30-01). Oregon State University School of Electrical Engineering and Computer Science.

Natarajan, S., Tadepalli, P., Altendorf, E., Dietterich, T. G., Fern, A., & Restificar, A. (2005). Learning first-order probabilistic models with combining rules. *Proceedings of ICML-05*.

Neville, J., & Jensen, D. (2004). Dependency networks for relational data. *Proceedings of ICDM'04* (pp. 170–177).

Neville, J., Jensen, D., Friedland, L., & Hay, M. (2003). Learning relational probability trees. *Proceedings of KDD '03* (pp. 625–630).

Raftery, A. (1995). Bayesian model selection in social research. *Sociological Methodology*.

Robinson, R. (1973). Counting labeled acyclic digraphs. *New Directions in the Theory of Graphs*, 239–273.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, *6*, 461–464.

Stanford, D. C., & Raftery, A. E. (2002). Approximate bayes factors for image segmentation: The pseudolikelihood information criterion (plic). *IEEE Trans. Pattern Anal. Mach. Intell.*, *24*.