Appears in *Proceedings of the Fifteenth International Conference on Machine Learning*
pp. 454-462, Madison, WI, July 1998

Theory Refinement for Bayesian Networks with Hidden Variables

Sowmya Ramachandran,
Stottler Henke and Associates, Inc.,
1660, So. Amphlett Blvd. Ste. 350,
San Mateo, CA, 94402
sowmya@shai.com

Raymond J. Mooney
Department of Computer Sciences,
University of Texas at Austin,
Austin, TX, 78712
mooney@cs.utexas.edu

Abstract

While there has been a growing interest in the problem of learning Bayesian networks from data, no technique exists for learning or revising Bayesian networks with hidden variables (i.e. variables not represented in the data), that has been shown to be efficient, effective, and scalable through evaluation on real data. The few techniques that exist for revising such networks perform a blind search through a large space of revisions, and are therefore computationally expensive. This paper presents BANNER, a technique for using data to revise a given Bayesian network with noisy-or and noisy-and nodes, to improve its classification accuracy. The initial network can be derived directly from a logical theory expressed as propositional rules. BANNER can revise networks with hidden variables, and add hidden variables when necessary. Unlike previous approaches, BANNER employs mechanisms similar to logical theory refinement techniques for using the data to focus the search for effective modifications. Experiments on real-world problems in the domain of molecular biology demonstrate that BANNER can effectively revise fairly large networks to significantly improve their accuracies.

1 Introduction

Bayesian networks have become the most popular approach to uncertain reasoning due to their precise probabilistic semantics as well their success in practical applications. In an attempt to automate their construction, induction of Bayes nets has become a topic of increasing interest. A number of learning methods have been developed for the case where all relevant variables are observable (Heckerman, 1995). Parameter learning methods for networks with *hidden variables* (variables not represented in the data) have also been developed (Russell, Binder, Koller, & Kanazawa, 1995; Thiesson, 1995). However, learning both the structure and the parameters of a Bayesian network with hidden variables remains a problem. Many of the existing methods can be adapted to discover hidden variables, but only by conducting extensive search that is impractical for most problems. A recent development is MS-EM (Friedman, 1997), which learns the structure of a network with hidden variables; however, it requires specifying the number of hidden variables and has not been tested on real data.

As demonstrated by *theory refinement* research on rule-bases, using empirical data to revise an initial imperfect knowledge base can significantly improve performance over induction from scratch (Opitz & Shavlik, 1993; Ourston & Mooney, 1994; Towell & Shavlik, 1994; Mahoney & Mooney, 1994; Brunk & Pazzani, 1995). A few techniques have been developed for revising Bayesian networks (Lam & Bacchus, 1994; Buntine, 1991); however, they do not handle hidden variables. Many existing Bayes-net induction methods could be adapted to revision, but only by examining all possible individual modifications. By contrast, rule-revision systems use classification errors on the training data to propose specific modifications rather than blindly examining all possible options. The result is an efficient, directed revision process.

We have developed a technique, BANNER, for refining Bayesian networks with hidden variables that, like rule-refinement algorithms, uses the data to focus the search for effective modifications. BANNER's goal is to improve the accuracy of an initial network for a specific inference task by modifying both its parameters and structure, including adding new hidden variables. Although Bayesian networks can simultaneously support many types of inference, training directly for the desired classification task results in better performance (Friedman & Goldszmidt, 1996). Since general Bayesian networks are impractical for many large problems because the number of parameters grows exponentially in the fan-in of a node, we focus on networks with *noisy-or* and *noisy-and* nodes, specialized models that require only a linear number of parameters (Pearl, 1988; Pradhan, Provan, Middleton, & Henrion, 1994). Since these models are close to logical functions, they also allow a rule-base to be used as an initial theory by mapping the rules to a network in the obvious way. Existing results show that the accuracy of rule bases can be dramatically improved by mapping them to a representation that provides numerical summing of evidence (Towell & Shavlik, 1994; Mahoney & Mooney, 1994). However, the neural networks or certainty-factor rules employed in these results do not provide an interpretable knowledge base with parameters that have a precise semantics. An important goal of theory refinement is to provide interpretable knowledge, and we believe Bayes nets are preferable in this regard.

Experimental evaluation of Bayes net learning has largely been conducted on artificial data and not adequately compared to other methods on real problems (exceptions include Provan and Singh (1994), Friedman and Goldszmidt (1996)), and we know of no Bayes-net results on revising real knowledge bases to fit actual data. We have evaluated BANNER on several realistic problems used to test other theory refinement systems, obtaining performance competitive with the current best results while maintaining the advantages of a Bayes-net representation. The remainder of the paper presents an overview of BANNER's learning algorithm and the promising results of this evaluation.

2 Refinement Algorithm

As in general in theory refinement, the goal is to minimally modify the initial theory to make it consistent with the available training data. Taking the standard approach, BANNER employs one procedure to revise the parameters of a network and another to revise the structure. First, the parameters are revised to improve classification accuracy. If the resulting network does not adequately fit the training data, the structure of the network is modified and the parameters are retrained. This process repeats until it is determined that additional training results in over-fitting.[1] In this paper, we focus on structure revision. Our current implementation includes two parameter revision algorithms, BANNER-PR (Ramachandran & Mooney, 1996) and C-APN (based on (Russell et al., 1995)), which use different forms of gradient descent. Ramachandran (1998) presents further details.

[1] The parameter revision component uses 10-fold internal cross-validation on the training set to determine when to stop (Mitchell, 1997).

Structure revision exploits the idea that networks with noisy-or/and nodes are similar to logical theories and therefore techniques used to revise rule bases are useful. These methods attribute classification errors on particular examples to specific portions of the theory and directly construct revisions to handle the misclassified cases. Most logical refinement systems use abduction to diagnose faults (Mooney, 1997). Since Bayesian networks place no restrictions on the direction of inference, abduction can be performed using the standard inference algorithms. In addition, *leak nodes* (Pradhan et al., 1994) provide a way to model the incompleteness and incorrectness of a Bayesian network with noisy-or/and nodes. A leak node is a source in the graph added as an extra input to a node in order to represent a possible unknown cause. BANNER diagnoses faults in a network by temporarily instrumenting each node with leak nodes that indicate potential revision points. It then uses training data to select a small set of revision points and construct appropriate refinements.

2.1 Selecting Revision Points

The procedure for instrumenting a network with leak nodes is best illustrated with an example, such as that shown in Figure 1 (A–G are the original nodes). Each noisy-or/and has an added parent called a *node-leak* node. In order to avoid significantly altering the semantics of the net, the prior of the leak node and its link parameter are initially set very low. However, when the algorithm detects misclassifications, it reestimates the prior probabilities by training a copy of the network augmented with leak node-leak nodes using the parameter revision module. All of the original noisy-or (noisy-and) nodes also have their parents routed through an *intervening* noisy-and (noisy-or) node. The intervening nodes themselves have attached leak nodes called *link-leak* nodes. To avoid altering the semantics, the weights on the links are set to simulate logical functions and the prior probability of the link-leak node is set to the weight on the original link. The leak nodes effectively represent possible faults in the theory, with node-leak nodes representing the need for new inputs to a node, and link-leak nodes representing the need for new intervening hidden variables between two nodes.

Once the network is properly instrumented, BANNER performs abduction on each misclassified example to generate a set of repairs that could correct the example. This involves instantiating both the evidence and the target variables in the augmented network to

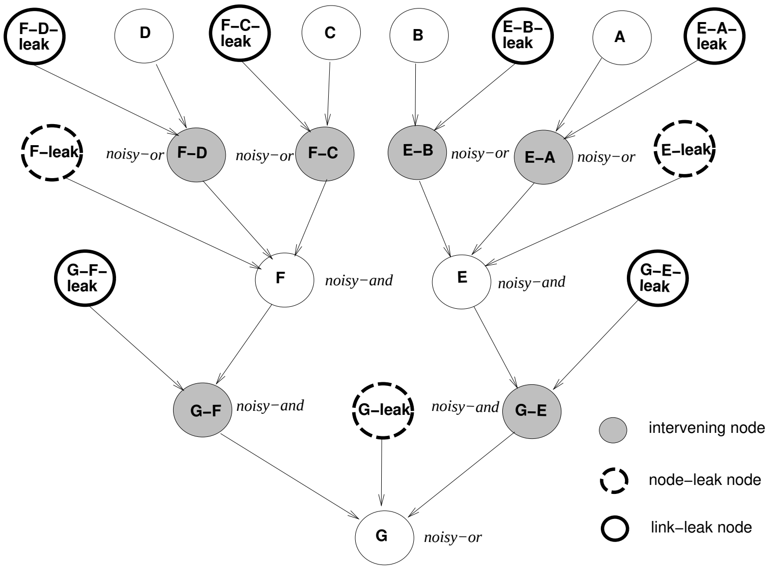

Figure 1: Augmenting a network with leak nodes

their observed values and inferring the beliefs associated with the leak nodes using standard Bayesian inference. For each misclassified example, it collects a set of leak nodes, whose beliefs deviate from their prior probability by more than 10%. Such leak nodes are said to *cover* the example, and indicate potential revision points in the theory. When the belief in the truth of a leak node decreases from its prior, it is called an *inhibitor* for that example; if it increases, it is called an *enabler*. Each leak node covering an example is associated with the degree to which its belief deviated from its prior, indicating the extent to which it is blamed for the misclassification. Once leak nodes are collected for all misclassified examples, Banner uses a greedy set covering algorithm (where the contribution of each leak node is weighted by its degree) to generate a small set of leak nodes that cover all of the misclassified examples. While Banner uses only misclassified examples to generate a set of revision points, it performs abduction on all the examples, generating leak nodes that are enablers or inhibitors for each example. This information is used during the generation of appropriate revisions.

2.2 Revision Operators

For each revision point in the covering set, Banner implements one of the following modifications to help correct the misclassified examples covered by the corresponding leak node: 1) Add a new parent, 2) Add a new hidden node, 3) Delete a link. The first operator is invoked when a revision point is a node-leak node, in which case it adds a new parent to the appropriate node in the original network. In the example, if *G-leak* is a selected revision point, then a new parent is added to G. The heuristic for selecting the new parent is discussed below.

If a revision point is a link-leak node, Banner modifies the corresponding link. One option is to introduce a new hidden variable with an additional parent and the same type as the corresponding intervening node. In the example, if *E-A-leak* is the revision point, a new noisy-or node is added between E and A (see Figure 2). The rationale for such a revision is that the previous step of abduction with the augmented network indicated that such a structure would better explain the misclassified data.

However, in some cases, the problematic link is simply deleted. For example, if *E-A-leak* is an enabler for several examples but never an inhibitor, the link may be deleted to correct the misclassified examples without affecting other examples since the link is effectively an always-true input to a noisy-and which therefore has no effect. A dual argument can be made for noisy-or nodes. A link is also deleted if, when a hidden node is added, the chosen parent has the same effect as link deletion. For example, if the negation of A is chosen as the new parent of E-A, the link between E and A is deleted.

New parents are selected based on the examples for

Given: An initial network, and a set of training data. **Output:** A revised network.

1. Initialize the parameters of the network either randomly or based on some prior knowledge.
2. Repeat steps a-e until there is no improvement in training accuracy over a pre-specified number of consecutive cycles.
   (a) set *train-net* = initial network.
   (b) set *leak-net* = train-net augmented with *node-leak* nodes.
   (c) Train network *train-net* to revise parameters.
   (d) If the previous step indicates overfitting, *or* all examples are correctly classified, return *train-net.*
   (e) else
      i. Train network *leak-net* to estimate prior probabilities of the *node-leak* nodes.
      ii. Set *augmented-net* = *train-net* augmented with *node-leak* and *link-leak* nodes.
      iii. Copy priors of leak nodes from *leak-net* to *augmented-net.*
      iv. For each example,
         A. Instantiate input and target nodes of *augmented-net* with values from the example.
         B. Infer beliefs of all the nodes in *augmented-net.*
         C. Collect all enabled and inhibited *node-leak* and *link-leak* nodes.
      v. Set *revision-points* = small set of *node-leak* and *link-leak* nodes that cover all the misclassified examples (computed using greedy set covering)
      vi. For each revision point in *revision-points*, revise *train-net* at the revision point using one of the revision operators.

Figure 3: Outline of the Refinement Algorithm

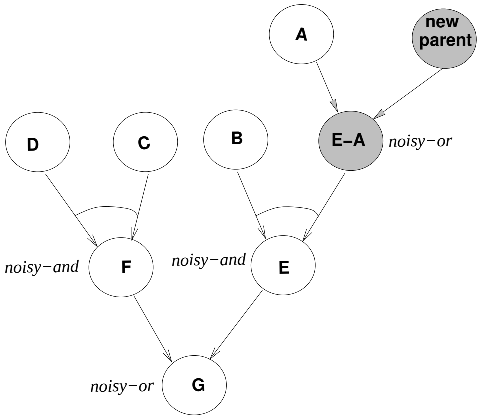

Figure 2: Revision operator: Adding a hidden node

which the chosen leak-node is an enabler or inhibitor. The new parent needs to be true for the examples it must enable and false for the ones it must inhibit. BANNER uses a standard *information gain* metric (Quinlan, 1990) to choose a parent that best discriminates between these two sets of examples. This metric, commonly used in inductive learning algorithms (Mahoney & Mooney, 1994; Quinlan, 1990, 1986), estimates the information gained about a target function value from knowing the value of an attribute. Two versions of this metric that are commonly used. The version used by Quinlan (1990) to learn propositional Horn-clause theories, is designed to pick a feature that best discriminates between sets of examples, with the additional constraint that the feature have specific values (e.g. true or false) for each set of examples. This version is most appropriate for our theory refinement algorithm because we need to select a new parent that discriminates between the examples that need an enabling influence, and the examples that need an inhibitory influence, with the additional constraint that the new parent be true for the former set of examples and false for the latter set of examples.

Suppose that we are given a set of examples, S, of size N, of which N^+ a re *positive* examples of a given class C, and N^- are *negative* examples of C. Also assume that all the features in the examples are boolean-valued. For any given feature F, let N_f be the number of examples for which F is true; of these let, N_f^+ be the number of examples which are positive examples of C, and N_f^- be the number of examples which are negative examples of C. Then, the reduction due to F in the total number of bits required to encode the positive members of C is given by

$$Gain(C, F) = N_f^+ * (I(S) - I(N_f)),$$

where $I(S) = -\log_2\left(\frac{N^+}{N^- + N^+}\right)$ is the number of bits required to encode a positive member of class C, and $I(N_f) = -\log_2\left(\frac{N_f^+}{N_f^- + N_f^+}\right)$ is the number of bits required to encode the positive members of class C, given that F is true. The higher the value of this function, the greater the correlation between the examples for which F is true and the positive examples of C. Note that this computation can be easily generalized to hidden variables and variables with missing values.

Information gain for such nodes can be obtained by weighting the frequency measures N_f^+ and N_f^- by the degree of belief associated with these nodes for each example.

So far, we have described this metric with a view to selecting an enabling parent. The same metric is used to select an inhibitory parent by defining N_f to be the number of examples for which F is false. Every other term in the computation of the metric is defined as before. In general, all nodes in the network and their negations are potential candidates; however, to avoid redundancy and the introduction of loops, the existing parents and descendents of the recipient of the new parent are excluded. Figure 3 shows a summary of the overall algorithm.

3 Experimental Evaluation

We conducted experiments on realistic problems and data to demonstrate that BANNER is effective at revising networks to improve their classification accuracy. We also compared its performance to naive Bayes which learns a simple Bayes net that includes all features and assumes conditional independence,[2] with KBANN (Towell & Shavlik, 1994) a neural-network refinement method, RAPTURE (Mahoney & Mooney, 1994) a certainty-factor refinement method, and with two standard inductive algorithms: C4.5 (Quinlan, 1993) for decision trees and BACKPROP (McClelland & Rumelhart, 1988) for neural networks. In order to study the contribution of BANNER's components, we also performed *ablation* studies, where we disabled parts of the algorithm and compared performance to the full system. BANNER-IND, is an inductive version which does not utilize an initial theory but starts with a default network with input and output variables but no links, and BANNER-PR (parameter revision), which uses an initial theory but does not perform structure revision. Finally, we specifically evaluated structure revision by attempting to fix an artificially corrupted initial theory.

We present results on two molecular biology problems employed in previous refinement experiments: recognizing promoters and splice-junctions in DNA strands (Towell & Shavlik, 1994). These problems include imperfect, expert-provided theories represented as propositional rules. These theories contain fan-ins of up to 17 inputs, which would require more than 130,000 parameters for general nodes, demonstrating the importance of using noisy-or/ands. Here we present the splice-junction results and results on a corrupted version of the promoter theory. BANNER also performs well on revising the original promoter theory, but since its structure is already adequate, this problem does not test structure revision. The system also performed well on revising a knowledge base on C++ programming to model students for an intelligent tutoring system (Baffes & Mooney, 1996). Ramachandran (1998) presents complete results.

In order to compare to previous results, we generated learning curves in which the data was randomly split into independent training and test sets, systems were trained on the training data, and then tested on classifying the test examples. Results were averaged over 20 random training/test splits. This was done for training sets with increasing number of examples. A two-tailed paired t-test is used to evaluate the statistical significance of differences in performance given a specific number of training examples.

3.1 DNA Splice-Junction

This problem addresses the task of detecting *splice-junctions*, the boundaries between the utilized and unutilized sequences in DNA. The data set consists of 3190 examples consisting of strings of 60 nucleotides with the values A, C, G, or T, and assigned to three different categories. The initial theory consists of 47 propositional rules.

Figures 4 shows the primary results and Figure 5 shows the ablation results. The experiment provides evidence that BANNER is successful at improving the accuracy of the initial theory significantly with just a small number of examples. The accuracy of the initial theory has risen from 55%, before revision, to 73.6% when trained on just 20 examples, and to about 91.2% when trained on 400 examples. The performance of the three refinement algorithms RAPTURE, BANNER, and KBANN are similar, although RAPTURE performs slightly better. The differences between RAPTURE and BANNER are small but statistically significant for all points on the learning curve at the 0.01 level. The inductive algorithms all perform significantly worse for smaller training sets, although NAIVE BAYES catches up with RAPTURE at 200 examples. The differences between the BANNER and NAIVE BAYES are significant at at least the 0.01 level for 20, 50, and 100 examples, where the former performs considerably better, at the 0.001 level for 400 examples where it performs slightly worse.

[2] Our version includes smoothing with Laplace estimates which significantly improves performance (Kohavi, Becker, & Sommerfield, 1997)

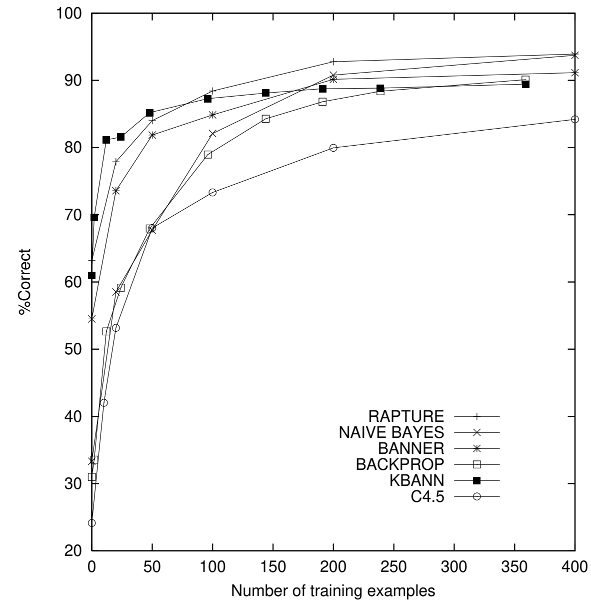

Figure 4: Splice-Junction: Performance of Various Systems

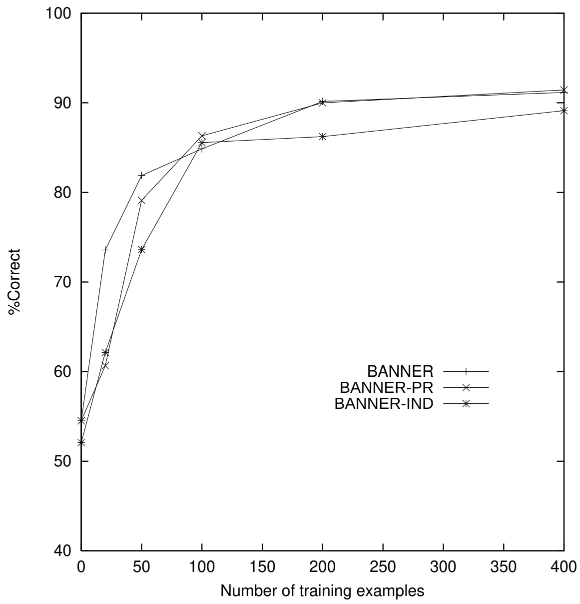

Figure 5: Splice-Junction: Banner Ablations

Figure 5 demonstrates that structure revision contributes significantly to BANNER's performance on smaller training sets. Structure revision has contributed to an improvement in accuracy of about 13% over BANNER-PR for 20 examples (significant at 0.001 level), and an improvement of about 2.8% for 50 examples (significant at the 0.05 level). The revisions that contributed the most to this improvement were deletions of the links between nodes IE and PR, and nodes EI and $P5G$. The differences between BANNER and BANNER-PR are not statistically significant at the rest of the points on the learning curve. As expected, starting out with an initial theory gives BANNER a significant edge over BANNER-IND. The difference in performance between these systems is statistically significant for all points on the learning curves, except at 100 example, at levels of at least 0.02.

3.2 Evaluation of Structure Revision on DNA Promoter

In order to more directly study structure revision, an existing theory with adequate structure was corrupted and BANNER's ability to recover the lost structure was examined. The DNA promoter recognition problem involves identifying DNA sequences that indicate the start of a new gene. Figure 6 shows a portion of the Bayesian network derived from the initial theory for this problem, The data set contains 468 examples, consisting of strings of 57 nucleotides classified as promoters or non-promoters. Although in refinement experiments theories are sometimes corrupted randomly (Pazzani & Brunk, 1993), we found that the redundancy in this theory makes it very robust to small corruptions. Therefore, we generated a corrupt theory by deleting a portion of the theory we knew to be critical, namely the intermediate concept $minus\_35$ (deleted portion shown in bold in Figure 6).

Figure 7 shows BANNER-PR and BANNER's performance with this damaged theory compared to BAN-

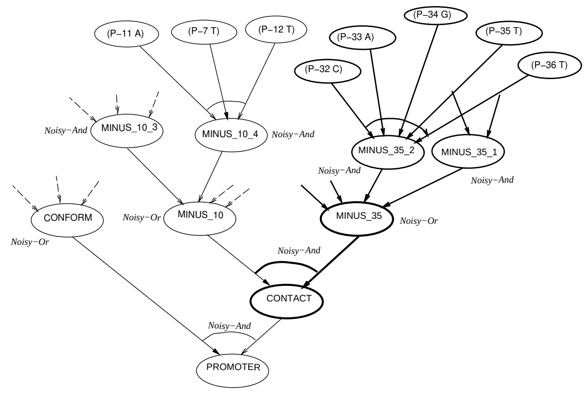

Figure 6: DNA Promoter Recognition - Initial Bayesian Network

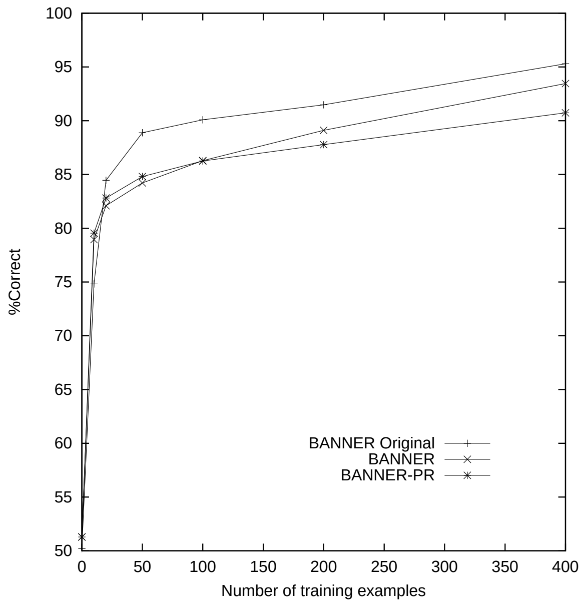

Figure 7: Effect of structure revision on corrupted promoter theory

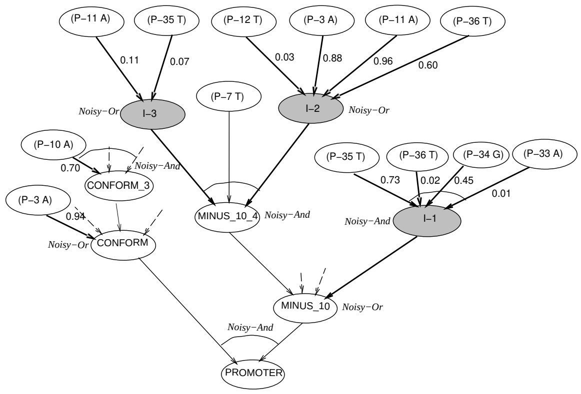

Figure 8: Example of a revised promoter network

NER's performance with the original theory. The graph shows that removing *minus*_35 degrades the theory to the extent that, for most points in the learning curve, parameter revision alone cannot recover the accuracy attained with the original theory. The results shows that, for larger training sets, structure revision is effective at recovering a fair bit of the accuracy lost due to the corruption, although the difference between Banner-Pr and Banner is only significant (at the 0.05 level) at 400 examples.

The fact that Banner and Banner-Pr result in comparable accuracies for smaller training sets can be explained by the fact that none of the trials with 10 and 20 training examples, and less than half the trials with 50 examples required structure revision. Notice that the corrupted theory results in better networks than the original when trained on 10 examples. With 20 and 50 examples, the corrupted theory is still usually able to fit the training examples without structure revision, but results in poorer generalization. This leads to the hypothesis that, for smaller training sets, there are several theories that are as good as the original theory in fitting the training set, but are worse in terms of generalization, which would partially explain the observation that structure revision leads to improved training accuracies without any improvements in generalization, when trained on 50 and 100 examples.

Figure 8 illustrates a revised network. The nodes and links added by Banner are indicated by shaded ellipses and bolder arrows and the numbers beside the links represent parameter values. Note that some nodes have been replicated in the figure for clarity only. Banner added several features to the network: *P-35=T, P-36=T, P-34=G, P-33=A* and *P-3=A* and added new links from features already present in the network: *P-11=A*, and *P-10=A*. In addition, it has added three hidden variables, *I-1* through *I-3*. A comparison with the original theory indicates that the added unit *I-1* roughly corresponds to the deleted *minus*_35 concept. However, in the original theory, *minus*_35 combines conjunctively with *minus*_10, whereas, here it combines disjunctively. That could explain why Banner also added some of these features to the sub-network above *minus*_10_4. However, realize that the initial theory is not known to have the correct structure, it is simply one proposed in the biological literature that is also consistent with the available data. Also, note that the modifications to the network are not confined to any particular level (as they are in Mahoney and Mooney (1994)).

In summary, our experiments demonstrate that Banner is effective in revising an Bayesian networks with hidden variables to significantly improve their accuracy. They also demonstrate that the structure revision algorithm contributes significantly to the overall algorithm and makes semantically interpretable revisions. The effectiveness of the structure revision algorithm is also illustrated by the fact that Banner-Ind learns highly accurate classifiers. Experiments have also been performed that show that Banner-Ind learns more accurate classifiers that Naive Bayes on the problem of classifying chess end-games (Quinlan, 1983). Ramachandran (1998) provides details on these results.

4 Related Work

While recent techniques have begun to address the problem of learning the structure of a Bayesian network from incomplete data (Ramoni & Sebastiani, 1997; Friedman, 1997), only a few address the problem of learning or revising networks with hidden variables. MS-EM (Friedman, 1997) extends EM to learn the structure as well as the parameters of a network from incomplete data. While it works when the initial theory contains hidden variables, it cannot construct new hidden variables. Kwoh and Gillies (1996) present a procedure for adding hidden variables by first learning a Bayesian network from data without hidden variables, and then using statistical analysis to find correlations between variables with the same cause and clustering such variables with a new hidden node. These techniques have been demonstrated on learning small networks, but have not been evaluated on larger, real-world problems. Moreover, it has no mechanism for selecting a candidate set of nodes that need to be revised, instead relying on blind search through the space of all possible revisions.

5 Future Research

Experiments on other realistic problems, particularly ones in which the initial theory is specified as a Bayesian network (rather than translated from rules), is one area for future research. The current results for Banner involve problems of causal inference, tests on tasks involving abductive inference are also needed. More detailed comparisons of different Bayes-net induction and revision algorithms and competing methods on realistic problems measuring both training time and predictive accuracy are clearly needed. The current literature on Bayes-net learning is particularly lacking in this regard relative to other areas of machine learning (Friedman, Goldszmidt, Heckerman, & Russell, 1997).

Extending Banner's general approach to handle nodes other than noisy-or/and ones is an important area for future study. Another is theory refinement for unsupervised learning where there is not a specific targeted inference task. The algorithm can also be extended to use Bayesian metrics to select new nodes to be added to the parent set of a node. A number of interesting ideas for learning and revising Bayes nets have been proposed, but integrating them into an efficient and effective system with clearly demonstrated advantages over other machine-learning methods on realistic problems is still a challenge.

6 Conclusion

We have introduced a novel technique for revising Bayesian networks that can handle existing hidden variables as well as create new ones. We have demonstrated, through experiments on realistic problems, that this approach can efficiently revise large networks and produce highly accurate classifiers. The results are also competitive with those of the best theory refinement systems while maintaining the precise probabilistic semantics of Bayesian networks that we believe make the resulting theories significantly more comprehensible. Whereas existing techniques for revising Bayesian networks must search through the space of all possible revisions, we have presented novel mechanisms for using the information in the data to guide the search for useful revisions, thus focusing the search and making it tractable for larger, more realistic problems.

7 Acknowledgements

We are grateful to Bobby Blumofe, Lorenzo Alvisi, Mike Dahlin, Calvin Lin and other folks at the UT LESS lab for letting us use their computing facilities for this research. The multiprocessor computing facilities in this lab were made available through a generous equipment donation from Sun Microsystems. This research was partially supported by the National Science Foundation through grants IRI-9310819 and IRI-9704943.

References

Baffes, P. T., & Mooney, R. J. (1996). A novel application of theory refinement to student modeling. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 403–408 Portland, OR.

Brunk, C., & Pazzani, M. (1995). A lexically based semantic bias for theory revision. In *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 81–89 San Francisco, CA. Morgan Kaufman.

Buntine, W. (1991). Theory refinement on Bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 52–60.

Friedman, N., Goldszmidt, M., Heckerman, D., & Russell, S. (1997). Challenge: What is the impact of Bayesian networks on learning?.. pp. 10–15 Nagoya, Japan.

Friedman, N. (1997). Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 125–133 Nashville, Tennessee. Morgan Kaufmann Publishers.

Friedman, N., & Goldszmidt, M. (1996). Building classifiers using Bayesian networks. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 1277–1284.

Heckerman, D. (1995). A tutorial on learning Bayesian networks. Tech. rep. MSR-TR-95-06, Microsoft Research, Redmond, WA.

Kohavi, R., Becker, B., & Sommerfield, D. (1997). Improving simple Bayes. In *Proceedings of the European Conference on Machine Learning.*

Kwoh, C.-K., & Gillies, D. (1996). Using hidden nodes in Bayesian networks. *Artificial Intelligence*, *88*(1-2), 1–38.

Lam, W., & Bacchus, F. (1994). Using new data to refine a Bayesian network. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 383–390.

Mahoney, J. J., & Mooney, R. J. (1994). Comparing methods for refining certainty-factor rule bases. In *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 173–180 New Brunswick, NJ.

McClelland, J. L., & Rumelhart, D. E. (1988). *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises.* The MIT Press, Cambridge, MA.

Mitchell, T. (1997). *Machine Learning.* McGraw-Hill, New York, NY.

Mooney, R. J. (1997). Integrating abduction and induction in machine learning. In *Working Notes of the IJCAI-97 Workshop on Abduction and Induction in AI*, pp. 37–42 Nagoya, Japan.

Opitz, D. W., & Shavlik, J. W. (1993). Heuristically expanding knowledge-based neural networks. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 512–517 Chamberry, France.

Ourston, D., & Mooney, R. J. (1994). Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, *66*, 311–344.

Pazzani, M., & Brunk, C. (1993). Finding accurate frontiers: A knowledge-intensive approach to relational learning. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pp. 328–334 Washington, D.C.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, Inc., San Mateo,CA.

Pradhan, M., Provan, G., Middleton, B., & Henrion, M. (1994). Knowledge engineering for large belief networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 484–490 Seattle, WA.

Provan, G. M., & Singh, M. (1994). Learning Bayesian networks using feature selection. In *Proceedings of the Workshop on Artificial Intelligence and Statistics*, pp. 291–300 New York. Springer-Verlag.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess end games. In Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (Eds.), *Machine Learning: An Artificial Intelligence Approach.* Morgan Kaufmann, Los Altos, CA.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo,CA.

Quinlan, J. (1990). Learning logical definitions from relations. *Machine Learning*, *5*(3), 239–266.

Ramachandran, S. (1998). *Theory Refinement of Bayesian Networks with Hidden Variables.* Ph.D. thesis, University of Texas, Austin, TX. Also appears as Artificial Intelligence Laboratory Technical Report AI 98-265 (see http://www.cs.utexas.edu/users/ai-lab).

Ramachandran, S., & Mooney, R. J. (1996). Revising Bayesian networks parameters using backpropagation. In *International Conference on Neural Networks: Plenary, Panel and Special Sessions*, pp. 82–87 Washington D.C., USA.

Ramoni, M., & Sebastiani, P. (1997). Learning Bayesian networks from incomplete databases. In Geiger, D., & Shenoy, P. (Eds.), *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence.* Morgan Kaufmann Publishers, Inc.

Russell, S., Binder, J., Koller, D., & Kanazawa, K. (1995). Local learning in probabilistic networks with hidden variables. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pp. 1146–1152 Montreal, Canada.

Thiesson, B. (1995). Accelerated quantification of Bayesian networks with incomplete data. In Fayyad, U. M., & Uthurusamy, R. (Eds.), *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pp. 306–11. AAAI Press.

Towell, G. G., & Shavlik, J. W. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence*, *70*, 119–165.