Unpublished Technical Note, January 1997

Parameter Revision Techniques for Bayesian Networks with Hidden Variables: An Experimental Comparison

Sowmya Ramachandran, Raymond J. Mooney
Department of Computer Sciences,
University of Texas,
Austin, TX 78712-1188

Email : sowmya@cs.utexas.edu

February 22, 1997

Multiple Submission Statement: We intend to submit a version of this paper to The Third International Conference on Knowledge Discovery and Data Mining, 1997 (KDD-97). If this paper is accepted for publication at UAI-97 we will withdraw it from KDD-97.

Abstract

Learning *Bayesian networks* inductively in the presence of hidden variables is still an open problem. Even the simpler task of learning just the conditional probabilities on a Bayesian network with hidden variables is not completely solved. In this paper, we present an approach that learns the parameters of a Bayesian network composed of *noisy-or* and *noisy-and* nodes by using a gradient descent back-propagation approach similar to that used to train neural networks. For the task of causal inference, it has the advantage of being able to learn in the presence of hidden variables. We compare the performance of this approach with the *adaptive probabilistic networks* technique on a real-world classification problem in molecular biology, and show that our approach trains faster and learns networks with higher classification accuracy.

1 Introduction

Specification of a Bayesian network requires both the specification of the structure of the network, as well as a set of parameters associated with the variables in the network. Thus, the task of learning a Bayesian network can be divided into two subtasks: one of learning the structure of the network, and the second of determining the parameters. Within the general framework of inducing Bayesian networks, we can envision the following scenarios.

1. *Known structure, fully observable*: In this scenario, the structure of the network is given and assumed to be completely correct, and the data includes observations of all the variables in the network. The task here is to learn the parameters of the network from data.

2. *Known structure, hidden variables*: Here again, the structure of the network is given and assumed to be correct. The data, however, does not include observations of every variable in the network. The variables whose observations are not specified in the data are called *hidden variables*. Again, the task is to induce the parameters of the network. However, the task is more complicated in this case due to the presence of hidden variables.

3. *Unknown structure, fully observable*: In this case, neither the structure, nor the parameters of the network are known. We can, however, assume that the only variables in the network are those that are included in the data. The task here is to learn both the structure and the parameters of the network.

4. *Unknown structure, hidden variables*: This is the most general learning problem where the structure of the network is unknown and there are hidden variables.

The first of these is fairly straightforward. A common approach is to use the maximum likelihood estimates for the parameters, which in the case of no hidden variables, reduces to a function of the relative frequencies of occurrences of the values of the variable (Spiegelhalter & Lauritzen, 1990).

The problem of learning the parameters for a network with a given structure, in the presence of hidden variables, has also received some attention. Many statistical techniques like *Gibbs sampling* (Geman & Geman, 1984) and EM (Dempster, Laird, & Rubin, 1977; Lauritzen, 1995) can be used in the context of Bayesian networks. APEM (Thiesson, 1995) is a statistical technique that combines EM and gradient descent approaches to learn the parameters of a network. APN (Russell, Binder, Koller, & Kanazawa, 1995) is an approach that optimizes the probability of the data given the network using a gradient descent algorithm.

The learning problem addressed by Cooper and Herskovits (1992) falls in the third category. Their technique is to use a scoring metric to hill climb through a space of possible Bayesian networks to find one that is the most probable given the data. A number of variations and improvements to this approach have since been proposed (Buntine, 1991; Heckerman, Geiger, & Chikering, 1994; Provan & Singh, 1994).

The fourth scenario above, namely that of learning a Bayesian network with hidden variables and an unknown structure, is by far the most difficult and the least studied. Most of the above techniques could be adapted to discover hidden variables, but at a great cost involving brute force search. Connolly (1993) has proposed using clustering techniques (Fisher, 1987) to discover hidden variables. However, this technique can only learn tree structured networks. Figure 1 summarizes these learning scenarios and the techniques that handle them. (Heckerman, 1995) provides a good tutorial on the state-of-the-art with respect to learning Bayesian networks.

Thus, while researchers have a grasp on some aspects of learning Bayesian networks, the problem of inducing Bayesian networks with unknown structures and hidden variables still poses a tough challenge. Within the field of machine learning, theory revision techniques like those proposed by Ourston and Mooney (1990), Towell, Shavlik, and Noordewier (1990), Mahoney and Mooney (1993) have been successful in addressing similar issues for learning various kinds of traditional rule bases. Theory revision is based on the idea that, when only limited data is available, biasing an inductive learner with an existing imperfect knowledge base can improve learning by focusing the search through the hypothesis space. The more complex the domain, the more the advantage of such

Observability \ Prior Knowledge	Known Structure	Unknown Structure
fully observable	**Maximum Likelihood Estimates** *[Spiegelhalter and Lauritzen, 1990)*	**K2 and variations.** *[Cooper and Herskowits, 1992]*
hidden variables	**Gibb's Sampling** *[Geman and Geman, 1984]* **EM** *[Dempster et al., 1977]* **Adaptive Probabilistic Networks** *[Russell et al., 1995]* **APEM** *[Thiesson, 1995]*	**Tantra** *[Connolly, 1993]* *Can only learn tree–structured networks.* *?*

Figure 1: Bayes-Net Learning Scenarios

a bias. There has been some research on revising Bayesian networks. Lam and Bacchus (1994) have a technique for incrementally refining a Bayesian network using the Minimum Description Length principle (Rissanen, 1978). Buntine (1991) has proposed a technique for revising a Bayesian network efficiently, using scoring metrics similar to that proposed by Cooper and Herskovits (1992). However, neither of these techniques can revise networks with hidden variables.

The larger goal of our research is to use a theory revision approach to learn Bayesian networks with hidden variables. From the perspective of the four learning scenarios outlined earlier, this problem lies somewhere between scenarios 2 and 4. Thus, we assume that the learner is given a Bayesian network that may be incomplete or incorrect. We also assume that the learner is provided with data that may not include all of the variables in the network. The task is to use the data to improve the predictive accuracy of the network, modifying both it's parameters and structure (adding hidden variables if needed).

Since general Bayesian networks are impractical for many large problems because the size of the conditional probability tables grows exponentially in the fan-in of a node, we focus on the problem of learning networks with *noisy-or* and *noisy-and* nodes (Pearl, 1988; Pradhan, Provan, Middleton, & Henrion, 1994). These are specialized models for representing dependencies that only require a linear number of parameters. We specifically chose these models because they are close to *logical ors* and *logical ands*, thus making it possible to use knowledge expressed in the form of logical rules as an initial theory. Using such nodes, a knowledge base originally expressed as rules can be mapped to an analogous Bayesian network and refined to improve its accuracy. Many existing knowledge bases are written in the form of rules, and many experts have become comfortable with this formalism. However, results in theory revision show that the accuracy of such rule bases can be dramatically improved by mapping them to a representation that employs some form of uncertain reasoning or numerical summing of evidence (Towell et al., 1990; Baffes & Mooney, 1993; Mahoney & Mooney, 1993). This approach also provides a very straight-forward way of biasing a Bayesian network learner with some prior knowledge.

An essential component of such a theory revision approach is an efficient technique that can revise the parameters of a Bayesian network composed of *noisy-or/and* nodes in the presence of hidden variables. In this paper, we present BANNER, a technique that revises the parameters of a network by using gradient descent to minimize the mean squared-error between the measured and computed values of certain output variables. In addition to being restricted to *noisy-or/and* nodes, it only works on problems where the inference is causal (i.e. from cause to effect). Thus, it can only learn networks whose root causes are specified as evidence, and whose leaves are specified as outputs. Furthermore, it can only be applied to networks in which every loop involves an instantiated piece of evidence, i.e. an input. The reason for these restrictions will become clear when we present the details of the algorithm.

We then compare the performance of BANNER with two other techniques for parameter revision on a real-world learning problem of DNA promoter recognition (Towell et al., 1990). These are:

1. **Adaptive Probabilistic Networks (APN)** (Russell et al., 1995): This technique uses gradient descent to learn a network that optimizes the likelihood of the given data being generated by the network. This is a general algorithm that does not place many restrictions on the kinds of distributions it can learn. In fact, this is one of the reasons why we picked this algorithm for comparison.
2. **Conditional APN:** APN learns to optimize the likelihood of the entire data being generated by the network. However, as pointed out by Friedman and Goldszmidt (1996), such an approach may not lead to the best classifier. This is an important consideration when a Bayesian network is being learned to serve as a classifier. So, we have implemented a system called C-APN which uses APN to learn a network that is optimized to estimate the probability of certain class variables given some evidence. Since it is based on APN, it is just as general.

The approach is also compared to a "naive" Bayesian classifier and a previous rule-based revision system that has the currently best known performance on this task (Mahoney, 1996).

We begin with a description of the details of BANNER, APN and C-APN. This is followed by a discussion of the experimental evaluation of these techniques. We conclude with a discussion of where we hope to take our research in the future.

2 Overview of BANNER

The learning algorithm used by BANNER is analogous to the standard backpropagation algorithm used to train a multi-layered feedforward network (McClelland & Rumelhart, 1988). It uses gradient descent to minimize the mean-squared error between the measured and computed values of certain output variables. The algorithm is as follows:

1. Initialize the parameters of the network either randomly or based on some prior knowledge.
2. For every example in the training data
   (a) Place the evidence on the network and propagate the beliefs through the network.
   (b) Compute the mean squared-error at the output nodes.
   (c) Back-propagate the gradient and the errors from the output nodes to the evidence nodes.

(d) Modify the parameters based on the errors and gradients.

3. Repeat Step 2 for several cycles until a desired level of training performance is reached. Each cycle through the entire data set is called an *epoch*.

In the following subsections, we discuss the forward propagation and the backpropagation phases. This is followed by a discussion of step 3, i.e. the criteria used to decide when to stop training.

2.1 Forward propagation for noisy-or and noisy-and nodes

A *noisy-or* node in a Bayesian network is a generalization of a logical *or*. As in the case of the logical *or*, an event X is presumed to be false if all the conditions that cause X are false. However, unlike a logical *or*, if one of the causes of the event X is true, it does not necessarily imply that X is definitely true. Each condition C_i causing the event X can be thought of as having an associated inhibitory influence which is active with a probability q_{ix}. Thus, if C_i is the only cause of X that is true, then X is true with a probability $(1 - q_{ix})$. Moreover, the likelihood of X is a monotonic function of the number of its causal conditions that are true. The parameter $c_{ix} = 1 - q_{ix}$ is the degree to which an isolated cause C_i of an event X can endorse the event.

Given some evidence, the c_{ix} associated with each link to a node X from each of its parents, and the belief measures of all the parents of X, there is a simple equation for calculating the degree of belief that X is true. Under the assumption that all the evidence in the network is causally upstream of the node, the degree of belief in a node X is given by:

$$Bel(x) = \begin{cases} \prod_i (1 - c_{ix}\pi_{iX}) & \text{if } x = 0 \\ 1 - \prod_i (1 - c_{ix}\pi_{iX}) & \text{if } x = 1 \end{cases} \tag{1}$$

where π_{iX} is the degree of belief in the truth of the i-th parent of X. Thus, the number of parameters that have to be specified for a *noisy-or* node is linear in its fan-in. Given some data, the learning task is to learn these parameters such that the prediction accuracy of the network is optimized.

A *noisy-and* node is the dual of a *noisy-or* node. It is a generalization of a logical *and*. Each causal link between C_i and X has associated with it a parameter c_{ix} that specifies the degree to which disproving the event C_i disproves X. The belief measure of a *noisy-and* node X, given some evidence, the c_{ix} associated with each link to X from its parents, and the belief measures of all the parents of X, is given by

$$Bel(x) = \begin{cases} 1 - \prod_i (1 - c_{ix}(1 - \pi_{iX})) & \text{if } x = 0 \\ \prod_i (1 - c_{ix}(1 - \pi_{iX})) & \text{if } x = 1 \end{cases} \tag{2}$$

where π_{iX} is the degree of belief in the truth of the i-th parent of X. Here again, the assumption is that all the evidence in the network is causally upstream of the node.

Note that, since the belief that a variable is true and the belief that the variable is false are complementary, learning a good estimation of one leads to a good estimation of the other. Thus, our problem reduces to that of learning a network that best estimates the probability that the output variables are true given some evidence.

2.2 Backpropagation

Once the mean-squared error is computed for the output nodes, it is propagated back through the network. The gradient to be applied to each of the parameters is computed based on these errors. The mean squared-error function is defined as:

$$E[\mathbf{w}] = 1/2 \sum_{i\mu} (\zeta_i^\mu - O_i^\mu)^2 \tag{3}$$

where ζ_i^μ is the value of the i-th output node for pattern μ of the data and O_i^μ is the actual value of the i-th output variable for the same pattern.

The gradient is computed by incorporating the belief functions for *noisy-or/and* nodes (Equations 1 and 2) into the error function above and taking its partial derivative with respect to the parameters. This results in the following gradients for the *noisy-or* and *noisy-and* nodes:

$$\Delta c_{ji} = \begin{cases} \eta \delta_i O_j \prod_{k \neq j} (1 - c_{ki} O_k) & \textit{(for Noisy-or)} \\ -\eta \delta_i (1 - O_j) \prod_{k \neq j} (1 - c_{ki}(1 - O_k)) & \textit{(for Noisy-and)} \end{cases} \tag{4}$$

where
η is the learning rate,
δ_i is the error propagated back from the output nodes to node i,
O_j is the computed belief that the jth parent of i is true, and
c_{ji} is the parameter on the link between node i and its jth parent. The error that a node i propagates to its parent j is:

$$\delta_j = \begin{cases} -\delta_i c_{ji} \prod_{k \neq j} (1 - c_{ki} O_k) & \textit{(for Noisy-or)} \\ \delta_i c_{ji} \prod_{k \neq j} (1 - c_{ki}(1 - O_k)) & \textit{(for Noisy-and)} \end{cases} \tag{5}$$

where
δ_i is the error at node i,
O_j is the belief that the j parent of i is true, and
c_{ji} is the parameter on the link node j to node i.

The computations presented here and in the previous subsection lead us to the reason why BANNER is limited to causal inference and allows only certain kinds of loops. Since the gradients are based on Equations 1 and 2 which were derived based on the assumption that the inference is causal, they would not be appropriate for diagnostic inference. Moreover, the functions for forward propagation shown above are no longer applicable in the presence of loops, unless the loop contains an instantiated variable that can break the loop. Since in the case of BANNER, all the evidence is placed at the roots of the network, and the roots of a network can break loops when instantiated, BANNER is applicable to networks in which every loop contains an instantiated evidence node.

2.3 Stopping Criteria

Knowing when to stop training is one of the crucial aspects of a gradient descent training algorithm. If the network is not trained enough, then it will not learn the training data effectively and hence may not be able to generalize well. However, if it is trained for too long, it may over-fit the training data and therefore lose out on generalization. In order to avoid overfitting, we use 10-fold internal cross-validation to determine when to stop training. Thus, each set of training data is further split into internal training and test sets (in the ratio 90% to 10%). The network is trained on the reduced training set, and the generalization performance on the internal test set is monitored at the end of each epoch. This is done for ten different internal splits of the training data, and the average generalization error is computed for each epoch. The number of training epochs that shows the least average generalization error is picked as the number of epochs needed to train the network on the full training set.

3 APN and C-APN

Having discussed BANNER in detail, we now move on the other two learning algorithms under consideration. Russell et al. (1995) proposed APN as a technique to learn the parameters of a Bayesian network with hidden variables. It uses gradient descent to optimize $\ln P_w(D)$, i.e. the log of the probability assigned by the network to the data when the parameters are set to w. Equation 6 shows the gradient. APN was shown to be effective in learning the parameters of a network that models the task of car insurance risk estimation. However, that example did not involve *noisy-or* nodes, which is what this comparison is concerned with. Moreover, their experiment used artificial training data generated from an actual network built for this task. Our experiments evaluate performance of the techniques on a problem set where the training data is generated independent of the network being learned. Although, the authors do not report any results on *noisy-or* models, they point out how their technique can be extended to compute gradients for *noisy-or* parameters. Our implementation of APN, built on top of IDEAL (Srinivas & Breese, 1993), can handle *noisy-or* and *noisy-and* nodes in addition to general discrete-valued, probabilistic nodes. [1]

$$\frac{\partial \ln P_w(D)}{\partial w_{ijk}} = \sum_{l=1}^{m} \frac{P_w(x_{ij}, u_{ik} \mid D_l)}{w_{ijk}} \tag{6}$$

where
m is the number of training instances,
x_{ij} is the jth possible assignment to variable X_i,
u_{ik} is the kth possible value assignment to the parents of X_i,
w_{ijk} is the probability that variable X_i takes on its jth possible assignment given that its parents U_i take on their kth possible assignment.

While APN optimizes the probability assigned by the network to the data as a whole, sometimes it may be desirable to optimize the *classification* accuracy of the network, specially when the Bayesian network is being trained to be a classifier. Friedman and Goldszmidt (1996) have experimentally demonstrated the advantage of using this approach to maximize classification accuracy. Specifically, the learning algorithm should try to learn a network that best estimates the probability distribution of the class variables conditioned on some evidence or attributes. Thus, the function to be optimized is $\ln P_w(C \mid E)$, which is the log of the conditional distribution found in the data of certain class variables C, conditioned on some evidence E. In order to see how APN can be used to optimize this metric, note that by applying Bayes law we get

$$\ln P_w(C \mid E) = \sum_{l=1}^{m} \ln P_w(C_l \mid E_l) \tag{7}$$

[1] When contacted, the authors of APN were unable to release their implementation.

$$= \sum_{l=1}^{m} (\ln P_w(C_l, E_l) - \ln P_w(E_l))$$

where m is the number of training instances. Therefore,

$$\frac{\partial \ln P_w(C \mid E)}{\partial w_{ijk}} = \sum_{l=1}^{m} \left(\frac{\partial \ln P_w(C_l, E_l)}{\partial w_{ijk}} - \frac{\partial \ln P_w(E_l)}{\partial w_{ijk}} \right) \tag{8}$$

Thus, for each training example l, the gradient to optimize $\ln P_w(C_l \mid E_l)$ can be computed by first computing the gradient wrt. $\ln P_w(C_l, E_l)$ using APN, and then computing the gradient wrt. $\ln P_w(E_l)$ using APN, and taking their difference. This is the procedure adopted by C-APN, which can also handle *noisy-or*, *noisy-and* as well as general probabilistic nodes.

4 Experimental Evaluation

The standard practice in the field of learning Bayesian networks is to generate data from a pre-specified network and then try to use the data to re-learn the original network. The performance of the learning technique is evaluated by directly comparing the learned network to the original. However, in real-world applications the "correct" network is not known. Recently, there has been growing interest in using evaluation functions such as classification accuracy to demonstrate the effectiveness of techniques for learning Bayesian networks (Friedman & Goldszmidt, 1996; Russell et al., 1995). In our experiment, we follow the standard methodology used to evaluate machine learning methods, i.e. that of studying the effect of training by determining the classification accuracy of the trained network on a separate test set. We use a real-world classification problem of DNA promoter recognition (Towell et al., 1990) to evaluate the learning algorithms. This problem comes with an initial domain theory (rule base) provided by a domain expert and independent data for use in training and testing. As described below, we use the initial domain theory to provide the structure and initial parameters for a Bayesian network. Our experiments also compare the performance of BANNER with those of APN, C-APN, and a naive Bayesian classifier on this problem. Naive Bayes learns a simple single-layer Bayes net and has been shown to perform quite well on many real-world classification problems (Langley, Iba, & Thompson, 1992; Friedman & Goldszmidt, 1996).

4.1 DNA Promoter recognition

Figure 2 shows the initial expert rules for recognizing a DNA promoter sequence. There are 57 input features called nucleotides, each of which can take on one of four values, A, G, T and C. The target class, *promoter*, predicts whether or not the input DNA sequence indicates the start of a new gene.

Figure 3 shows a portion of the Bayesian network corresponding to this theory. All the logical *and*s in the domain theory are mapped onto *noisy-and* nodes and all the logical *or*s are mapped onto *noisy-or* nodes. Each 4-valued input feature has been converted into *four* binary-valued features.[2] This network was translated into a neural network as described earlier. The initial

[2] We use binary-valued nodes because the noisy-or and noisy-and nodes are binary-valued. However, there have been recent extensions that allow multi-valued noisy-or nodes. BANNER cannot handle these kinds of extended noisy-or nodes.

promotor <– contact, conformation

contact <– minus_35, minus_10

minus_35 <– (P–36 T), (P–35 T), (P–34 G), (P–33 A), (P–32 C).

minus_35 <– (P–36 T), (P–35 T), (P–34 G), (P–32 C), (P–31 A).

minus_10 <– (P–14 T), (P–13 A), (P–12 T), (P–11 A), (P–10 A), (P–9 T).

minus_10 <– (P–13 T), (P–12 A), (P–10 A), (P–8 T).

minus_10 <– (P–12 T), (P–11 A), (P–7 T).

conformation <– (P–47 C), (P–46 A), P(–45 A), (P–43 T), (P–42 T), (P–40 A)
(P–39 C), (P–22 G).

conformation <– (P–45 A), (P–44 A), (P–41 A).

conformation <– (P–49 A), (P–44 T), (P–27 T), (P–22 A), (P–18 T), (P–16 T),
(P–15 G), (P–1 A).

conformation <– (P–45 A), (P–41 A), (P–28 T), (P–27 T), (P–23 T), (P–21 A),
(P–17 T), (P–4 T).

Figure 2: DNA Promoter Recognition - Initial Domain Theory

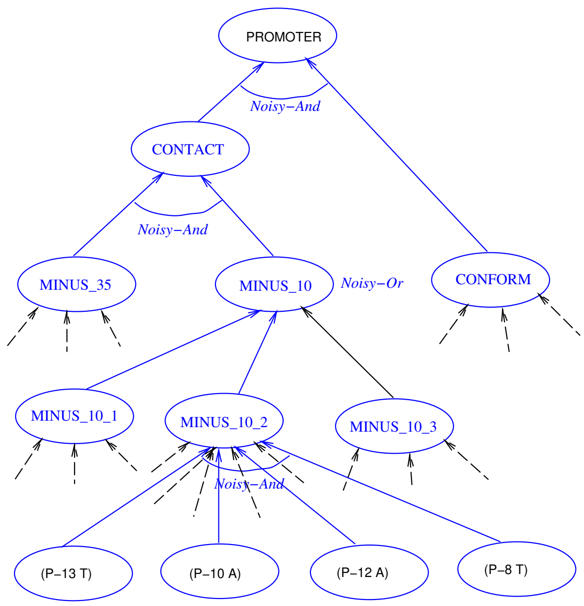

Figure 3: DNA Promoter Recognition - Bayesian Network

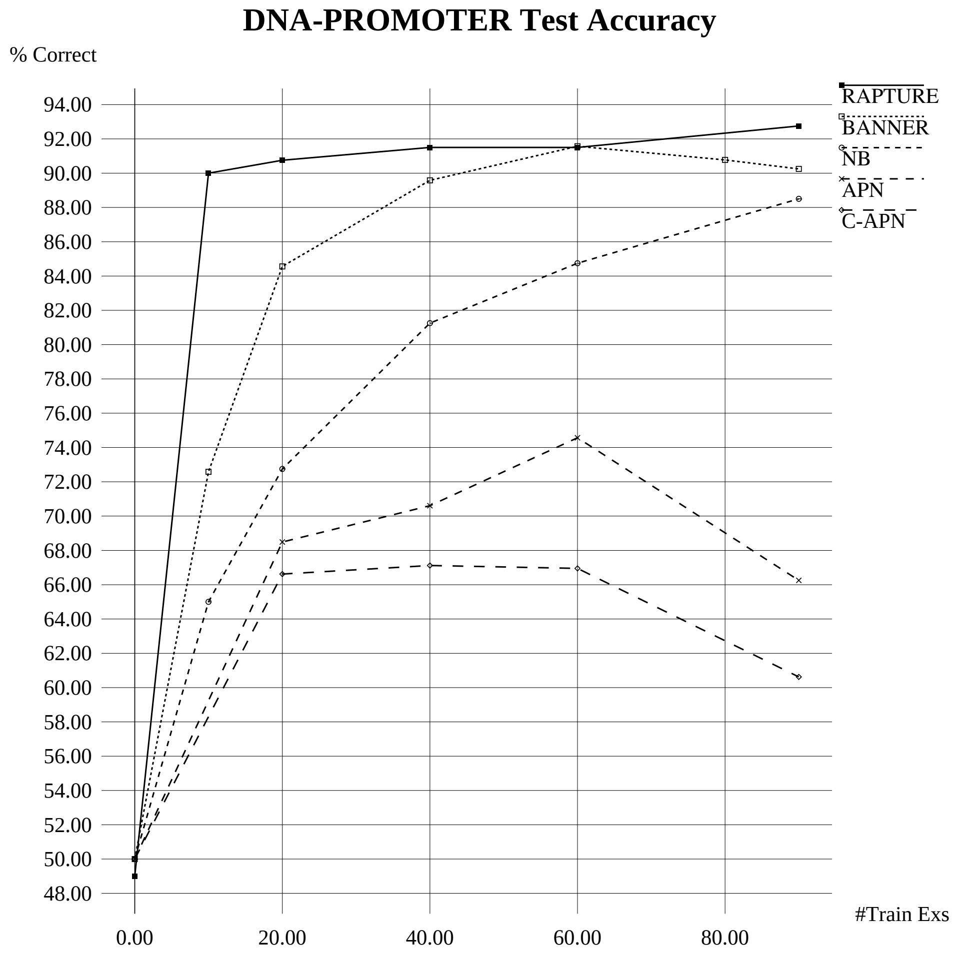

Figure 4: DNA Promoter Recognition - Predictive Accuracy

weights were all set to random values between to 0.90 and 0.99 to mimic the initial logical theory. Some randomization is necessary to break the symmetry for better convergence.

The data consisted of 106 patterns (53 positive and 53 negative examples). All the input variables (*P-50 ... P7*) were completely known and specified. The values of the output variable (*promoter*) was also given for each pattern. The values for the hidden variables (e.g. *minus-10)* were not given. The data was split into training and test sets of varying sizes. Figure 4 shows the resulting learning curve. This graph is a plot of the average accuracy of the network at classifying DNA strings over 25 different random training/test splits. As mentioned earlier, BANNER used 10-fold internal cross-validation on the training set to determine the stopping point for each train-test split. However, we were unable to do this with APN and C-APN due to time limitations. Therefore, for these two techniques, all networks were trained uniformly to 300 epochs. Also, due to time limitations, the current results for APN and C-APN are averages over only 10 trials.

The learning curves clearly demonstrate that our technique is successful at learning a network with a high classification accuracy. They also show that BANNER outperforms APN and C-APN significantly. It also performs better than naive Bayes. RAPTURE (Mahoney & Mooney, 1993, 1994) is a system that uses a connectionist approach to revise certainty-factor rule bases. Among a number of theory-revision systems that have been evaluated on this problem, RAPTURE currently has the best performance. Thus, although BANNER is effective in achieving high classification accuracy, there is still some room for improvement. However, BANNER generates a theory in the form of a Bayesian network, which has more clearly defined semantics than a certainty-factor rule base.

4.2 Discussion of Results

There are two issues that we hoped to address with our experiment. The first issue is: given a classification problem to be modeled as a Bayesian network of a given structure, which of the three techniques produces better results (measured in terms of classification performance on unseen cases)? For the problem discussed here, BANNER produces better networks than APN or C-APN. However, this difference in accuracy on the test data could be in part due to a difference in accuracy on training data (Figure 5). APN and C-APN show relatively poor training accuracy, which is reflected in their generalization performance. An obvious solution is to let the networks train longer until they converge on the training data. However, both APN and C-APN were found to be very slow, which made it infeasible to train the networks longer. For instance, it took 15 hours (real-time) for C-APN to train a network for 300 epochs on 90 example on an UltraSparc, whereas it only took 90 seconds for BANNER to do the same. This is also the reason why it was infeasible to perform internal cross-validation to determine the stopping points for APN and C-APN. Of course, these algorithms are slowed down to some extent because of the underlying Bayesian network simulator that our implementation uses. However, real time aside, BANNER was also found to converge with many fewer epochs than APN or C-APN. For instance, BANNER needed an average of 30 epochs to converge on 90 examples, whereas APN and C-APN could achieve less than 80% training accuracy even when trained up to 300 epochs. Since both approaches are performing gradient descent, the results imply that the error-space navigated by BANNER is more well-behaved (less plateaus and local minima) than that navigated by APN. By restricting itself to certain kinds of network structures, BANNER is able to directly exploit the computations involved in the Bayesian inference process, which in turn exploit the fact that *noisy-or/and* parameters combine linearly. This could also partly explain why BANNER is able converge much more quickly than APN or C-APN.

The second issue is whether or not it is better to train a network specifically for classification in order to get better classification performance. Our experiments do not yet provide a completely definitive answer on this issue. While BANNER, which trains for classification accuracy does perform better than APN, C-APN seems to do worse. However, it should be noted that C-APN performs worse than APN on the training data, which could partly explain its poorer generalization. It would be interesting to compare APN and C-APN on a smaller problem where training time would not be an issue.

5 Related Work

The problem of learning the parameters of a Bayesian network from data has attracted attention in the recent years. Several researchers have studied this problem and offered interesting solutions. However, previous methods have limitations that prevent them from being effective on problems like those considered in this paper.

Learning the parameters is fairly straightforward when all the variables of the network are represented in the data. A common approach is to use the maximum likelihood estimates for the

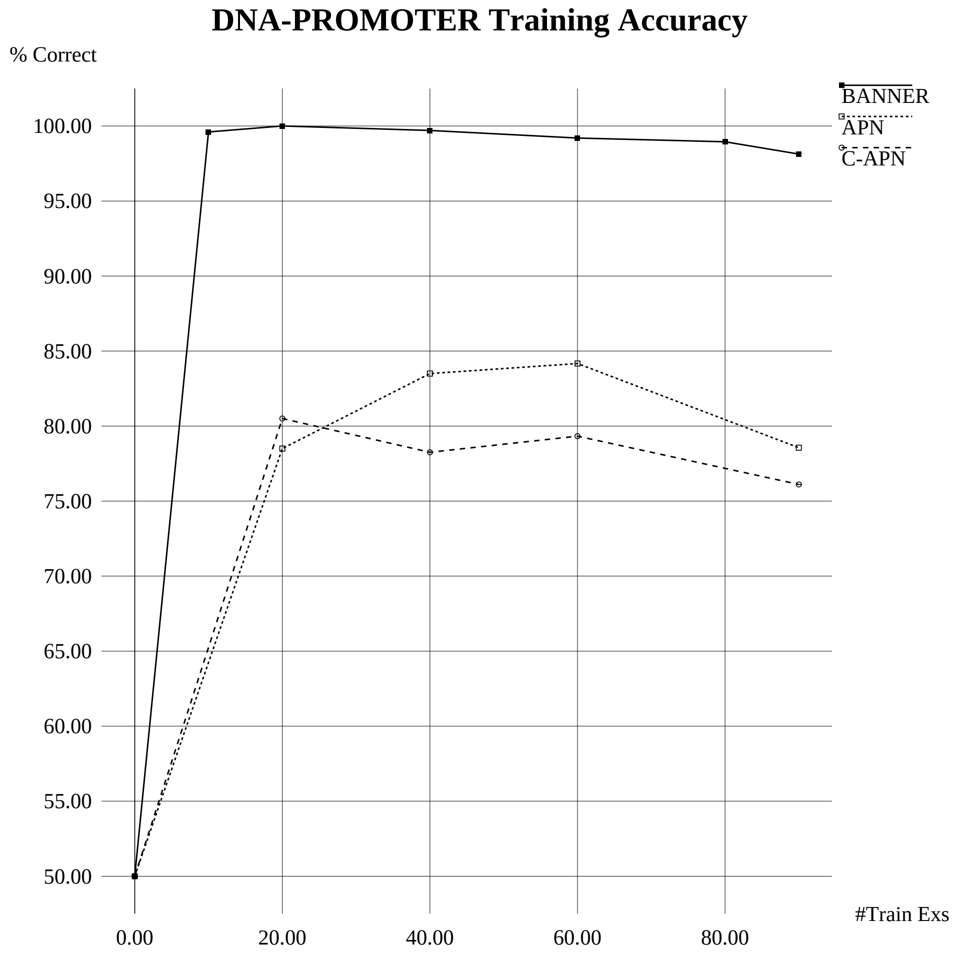

Figure 5: DNA Promoter Recognition - Training Accuracy

parameters, which in the case of no hidden variables, reduces to a function of the relative frequencies of occurrences of the values of the variable.

In the case of data that is missing some values, approximation methods like *Gibbs Sampling* (Geman & Geman, 1984) and EM (Dempster et al., 1977; Lauritzen, 1995) have been proposed. Both these methods require some initialization of the parameters and data for the missing variables. The complete data is then sampled to compute new values for the parameters. These steps are repeated until some convergence criteria is met. The goal of these methods is to optimize the likelihood of the data given the network. APEM (Thiesson, 1995) is a technique that combines traditional EM with gradient descent methods for faster convergence. Preliminary experiments with using APEM[3] on the DNA promoter recognition problem did not yield encouraging results (Ramachandran & Mooney, 1996). Although APEM could improve the accuracy of the network, it could only achieve about 65% accuracy on the test sets. However, since APEM does not currently handle noisy-or or noisy-and nodes, these nodes were modeled by general nodes with appropriate initial parameters. This results in the system being less constrained and having to learn significantly more parameters.

Musick (1994) uses a statistical technique for induction of the parameters of a Bayesian network assuming that the structure is given. Each parameter is represented as a distribution rather than

[3]We used the implementation of APEM provided by Bo Thiesson.

as a point value. The focus of this research is less on induction of networks and more on inference techniques for Bayesian networks with parameters specified as distributions. He does address the question of inventing hidden variables, but the solution proposed works only for very limited cases.

One of the early connectionist approaches to learning the parameters of a Bayesian networks is that reported in Neal (1992). It also uses the *noisy-or* approximation of a Bayesian node. However, since it uses stochastic networks similar to the Boltzmann machine, simulation of the network involves allowing the network to settle down to an equilibrium for each pattern observed. This is expensive and slows down learning dramatically. We use a forward propagation algorithm which results in significantly faster training.

Schwalb (1993) addresses the problem of learning the parameters of a given Bayesian network by mapping it onto a neural network with SIGMA-PI nodes and learning the conditional probabilities associated with the network (represented by link weights in the corresponding neural network) using standard backpropagation techniques (McClelland & Rumelhart, 1988). This has the advantage that it is able to learn the conditional probabilities even in the presence of hidden variables. However, the size of the neural network is combinatorial in the number of parents a node has in the corresponding Bayesian network, making the technique infeasible for even modestly large networks such as that for the DNA promoter recognition problem. Kwoh and Gillies (1996) have recently proposed a technique similar to BANNER for learning the parameters of a network with general discrete-valued, probabilistic nodes. Like BANNER, it performs gradient descent to minimize mean squared-error with respect to some output variables. Since, like BANNER they use the belief propagation computations for formulating the gradients, their technique faces the same limitations in terms on the kinds of network structures it can handle. As such, their computations are only valid for tree-structured networks and do not handle *noisy-or/and* nodes.

Our research was initially motivated by previous research on RAPTURE (Mahoney & Mooney, 1993, 1994). While RAPTURE is concerned with applying symbolic and connectionist techniques to revise certainty factor rule bases, we address the same problem for Bayesian networks. Although RAPTURE is successful in its task, it is limited by the fact that certainty factors themselves are not as formally grounded in probability theory as Bayesian networks.

6 Extensions and Future Work

One of the restrictions of BANNER is that it assumes that the intended inference task to be optimized is causal. However, this restriction is not inherent to the learning algorithm. The general technique can be extended to apply to diagnostic (abductive) tasks, although it would involve multiple error propagation passes and computing the gradients would be more complex.

The main aim of this research is to apply the idea of theory revision prevalent in the field of machine learning to facilitate the acquisition of probabilistic knowledge in the form of Bayesian networks. The research reported in this paper is concerned with the parameter revision component. We are developing an algorithm for revising the structure of a Bayesian network with *noisy-or/and* nodes that would augment a network with leak nodes and other buffer nodes to localize inconsistencies in the network (based on the data). This procedure would also help identify and localize the types of revisions (e.g. by adding a parent, removing a parent etc.) that would help remove these inconsistencies. It would then use information gain (Quinlan, 1986) to determine the nodes to be added or deleted from the parent set of a node. Efforts are underway to implement this algorithm, and to study its performance.

7 Conclusion

We have proposed a method for learning the parameters of a Bayesian network composed of *noisy-or* and *noisy-and* nodes in the presence of hidden variables. Our approach uses standard gradient-descent backpropagation techniques to optimize the classification accuracy of such networks for causal inference. The approach is efficient and effective for revising Bayesian networks that are initialized with an existing logical rule base. This makes it particularly appropriate for a range of real-world problems where such rule bases are readily available or easy to build. In particular, the technique was experimentally shown to quickly and effectively revise an existing knowledge base for classifying DNA sequences using real scientific data. By comparison, other more general gradient-descent techniques for revising parameters in the presence of hidden variables (APN and C-APN) were significantly slower to train and unable to efficiently and effectively converge on the training data, thereby limiting their generalization accuracies.

Acknowledgements

We gratefully acknowledge John Binder for answering questions about some of the details of APN. We would like to thank Steffen Lauritzen and Bo Thiesson for making the code for APEM available. Thanks are also due to Mike Hewett, Robert Blumofe and Emery Berger for letting us use their machines for our experiments.

References

Baffes, P., & Mooney, R. (1993). Symbolic revision of theories with M-of-N rules. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pp. 1135–1140 Chambery, France.

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

Connolly, D. (1993). Constructing hidden variables in Bayesian networks via conceptual clustering. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 65–72 Amherst, MA.

Cooper, G. G., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, *9*, 309–347.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, *39*, 1–38.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, *2*, 139–172.

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

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE transactions on Pattern Analysis and Machine Intelligence*, *6*, 721–742.

Heckerman, D. (1995). A tutorial on learning Bayesian networks. Tech. rep. MSR-TR-95-06, Microsoft Research, Advanced Technology Division, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052.

Heckerman, D., Geiger, D., & Chikering, D. M. (1994). Learning Bayesian networks: The combination of knowledge and statistical data. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pp. 293–301 Seattle, WA.

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.

Langley, P., Iba, W., & Thompson, K. (1992). An analysis of Bayesian classifiers. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pp. 223–228.

Lauritzen, S. L. (1995). The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, *19*, 191–201.

Mahoney, J. J. (1996). *Combining Symbolic and Connectionist Learning to Revise Certainty-Factor Rule Bases*. Ph.D. thesis, University of Texas, Austin, TX.

Mahoney, J. J., & Mooney, R. J. (1993). Combining connectionist and symbolic learning to refine certainty-factor rule-bases. *Connection Science*, *5*, 339–364.

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.

Musick, R. C. (1994). *Belief Network Induction*. Ph.D. thesis, University of California at Berkeley.

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, *56*, 71–113.

Ourston, D., & Mooney, R. (1990). Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 815–820 Detroit, MI.

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*.

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

Ramachandran, S., & Mooney, R. J. (1996). Revising Bayesian network parameters using back-propagation. Unpublished.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, *14*, 465–471.

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

Schwalb, E. (1993). Compiling Bayesian networks into neural networks. In *Proceedings of the Tenth International Conference on Machine Learning*, pp. 291–297 Amherst, MA.

Spiegelhalter, D. J., & Lauritzen, S. L. (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks*, *20*, 579–605.

Srinivas, S., & Breese, J. (1993). Ideal: Influence diagram evaluation and analysis in Lisp: Documentation and users' guide. Tech. rep. No. 23, Rockwell International Science Center, Palo Alto: Rockwell.

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., & Noordewier, M. O. (1990). Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp. 861–866 Boston, MA.