Appears in *Proceedings of the 1996 IEEE International Conference on Neural Networks*, pp.82-87, Washington D.C., June 1996.

Revising Bayesian Network Parameters Using Backpropagation

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

ABSTRACT

The problem of learning *Bayesian networks* with hidden variables is known to be a hard problem. Even the simpler task of learning just the conditional probabilities on a Bayesian network with hidden variables is hard. In this paper, we present an approach that learns the conditional probabilities on a Bayesian network with hidden variables by transforming it into a *multi-layer feedforward neural network (ANN)*. The conditional probabilities are mapped onto weights in the ANN, which are then learned using standard backpropagation techniques. To avoid the problem of exponentially large ANNs, we focus on Bayesian networks with *noisy-or* and *noisy-and* nodes. Experiments on real world classification problems demonstrate the effectiveness of our technique.

1. Introduction

Bayesian networks are increasingly being used for representing probabilistic knowledge [8]. Their strong grounding in probability theory makes them a particularly attractive formalism for representing knowledge. However, they suffer from the knowledge acquisition problem. Not only is it difficult to formulate the underlying structure of the network, it is especially difficult to specify the conditional dependencies between the variables in precise numeric terms.

The problem of learning Bayesian networks from data has attracted a lot of attention in the recent years. Many researchers have studied this problem and offered interesting solutions. One approach [2, 1, 3, 9] is to use a scoring metric to hill climb through a space of possible Bayesian networks to find one that is most probable given the data. However, a drawback with this approach is that it becomes too expensive when there are hidden variables, i.e. variables that cause some observations but are not themselves observed. This technique also requires that a total ordering on the variables be specified.

Schwalb [12] proposed using connectionist methods to learn the conditional probabilities on a Bayesian network inductively, given its structure and data. By mapping the given Bayesian network onto a neural network with SIGMA-PI nodes, this technique can learn the conditional probabilities associated with the network (represented by link weights in the corresponding neural network) using standard backpropagation techniques [5]. 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.

In this paper, we describe BANNER (BAyesian Networks NEural Revision), a system that learns the conditional probabilities in networks with *noisy-and* and *noisy-or* nodes by mapping such a network onto a *multi-layered feed-forward neural network (ANN)* and refining the weights using standard backpropagation techniques. This, again, enables the learning of conditional probabilities even in the presence of hidden variables. However, by concentrating just on *noisy-and* and *noisy-or* nodes, we have avoided the problem of intractably large networks faced by Schwalb [12].

BANNER is an extension of RAPTURE, described in [4]. While RAPTURE is concerned with applying symbolic and connectionist techniques to revise certainty factor rule bases, we address the issue of doing the same with Bayesian networks.

Although, in this paper we describe only our research into learning conditional probabilities, this is but a first step towards our goal of using this technique to revise the structure of the network inductively. The main goal of our research is to address the problem of inductively revising knowledge represented as a Bayesian network. Theory revision is an active area of research in the field of machine learning and various algorithms have been proposed for knowledge refinement [13, 7, 11, 4]. We aim to adapt these algorithms to revise Bayesian networks.

Our proposed technique for inductively learning the conditional probabilities on a Bayesian network works by first mapping the given Bayesian structure onto a multi-layered feed-forward neural network, then training the neural network on the given data using the standard gradient descent backpropagation algorithm, and finally mapping the trained neural network back into a Bayesian network. In the following subsections, we describe these steps in more detail.

2. The Noisy-or and the Noisy-and Models

The specification of a general Bayesian networks is combinatorial in the fan-in of the nodes. It requires the specification, for each variable, of the conditional probabilities of the variable given all possible combinations of values of its parents. Thus, for a network where all variables are binary-valued, a variable with n parents would require 2^n conditional probabilities to be specified. Thus, mapping a general Bayesian network into an ANN by mapping each conditional probability onto a weight in the neural network [12] becomes infeasible for even modestly large Bayesian networks.

The *noisy-or* and the *noisy-and* models of Bayesian networks [8] avoid this problem providing a way to compute the conditional probability of a variable given a combination of values of its parents from just the conditional probabilities of the variable given the value of each of its parents in isolation.

A *noisy-or* node in a Bayesian network is a generalisation of a logical *or*. As in the case of the logical *or*, an event E is presumed to be false if all the conditions that cause E are false (i.e. P(E)=0). However, unlike a logical *or*, if one of the causes of the event E is true, it does not necessarily imply that E is definitely true. Each condition C_i causing the event E can be thought of as having an associated inhibitory influence which is active with a probability q_i. Thus, if C_i is the only cause of E that is true, then E is true with a probability $(1 - q_i)$. Moreover, the likelihood of E is a monotonic function of the number of its causal conditions that are true. The parameter $c_i = 1 - q_i$ is the degree to which an isolated cause C_i of an event E can endorse the event.

Given some evidence, the c_i associated with each link in a network and the belief measures of all the parents of a node in the network, there is a simple equation for calculating the degree of belief that the node 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

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

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.

A *noisy-and* node is the dual of a *noisy-or* node. It is a generalisation of a logical *and*. Due to space limitations, we will not elaborate this further except to mention that each causal link between C_i and E has associated with it a parameter c_i that specifies the degree to which disproving the event C_i disproves E. The belief measure of a *noisy-and* node X, given some evidence, the c_i associated with each link in the network and the belief measures of all the parents of the node, is given by

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

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.

3. Neural Network Mapping

The structure of the ANN is identical to the structure of the given Bayesian network. *Noisy-or* and *noisy-and* nodes in the Bayesian network are respectively mapped onto *noisy-or* and *noisy-and* units in the neural network. The causal links between the nodes of the network are mapped onto connections between the corresponding units in the ANN. The weight on each link corresponds to the c_i associated with the link in the corresponding Bayesian network (previous section). Such a direct mapping facilitates the recovery of the Bayesian network from a trained ANN.

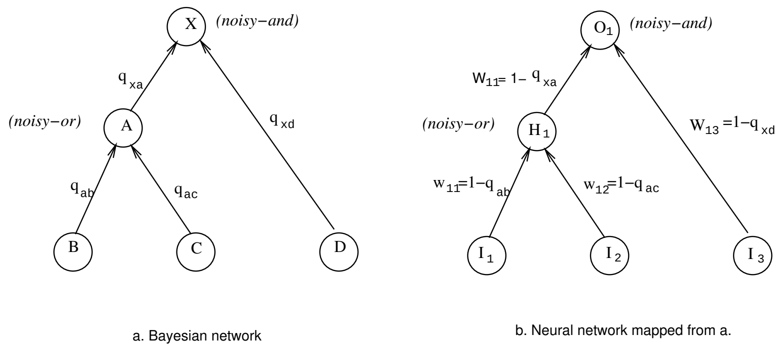

Fig. 1: Mapping from a Bayesian network into a neural network

Thus, the Bayesian network shown in Figure 1a is mapped on to an ANN shown in Figure 1b. The output of the *noisy-or* and *noisy-and* units are computed using the following functions:

$$activation(Unit_i) = \begin{cases} 1 - \prod_j (1 - w_{ij} O_j) & (Noisy\text{-}or) \\ \prod_j (1 - w_{ij}(1 - O_j)) & (Noisy\text{-}and) \end{cases}$$

where O_j is the activation of the jth unit feeding into unit i, and w_{ij} is the weight of the link between unit i and the jth unit feeding into it.

This function computes the degree of belief in the truth of the variable represented by the unit. Note that, since the only evidence placed in the network is on the input variables, the assumption that all evidence in the network is causally upstream of all the hidden and output variables holds.

4. The Training Phase

Once the Bayesian network is transformed into an ANN as described in the previous section, the problem of learning the parameters (the c_i's) for the Bayesian network is transformed into a problem of learning the weights in the ANN. This is achieved by initialising the weights randomly and training them using standard backpropagation techniques.

The data used for training the network should consist of a set of patterns that specify the value for the input and the output variables in the network. In a Bayesian network, input variables are the *sources* in the DAG representing the network (i.e those variables that have no incoming links). The *sinks* in the DAG are the output variables. The values for the hidden variables do not have to be specified in the data.

To train the network, the gradient descent algorithm requires a learning rule. The learning rules for the *noisy-or* and *noisy-and* units are given by the following functions:

$$\Delta w_{ij} = \begin{cases} \eta \delta_i O_j \prod_{k \neq j} (1 - w_{ik} O_k) & (Noisy\text{-}or) \\ -\eta \delta_i (1 - O_j) \prod_{k \neq j} (1 - w_{ik}(1 - O_k)) & (Noisy\text{-}and) \end{cases}$$

where η is the learning rate, δ_i is the error propagated back from the output units to unit i, O_j is the activation of the jth unit feeding into i, and w_{ij} is the weight of the link between unit i and the jth unit feeding into it.

Once the network has been trained to a desired accuracy, it can be mapped back into a Bayesian network.

5. Experimental Evaluation

We have evaluated BANNER on two classification problems: DNA promoter recognition [6] and Gene Splice Junction recognition [6]. Each of these has associated with it an initial domain theory that does not have a good prediction accuracy on the data and therefore has to be revised.

For each problem, we created several random splits of the data into training and test sets. One of these data splits was used to determine the stopping point for training. The network was trained until further training did not decrease the *root mean square error* of the network on the training set. The networks for the remaining splits were trained for the same number of epochs as the first one.

In the following subsections, we present the results of the experiments. We also compare the performance of Banner with the performances of other inductive learning and theory-revision systems like Kbann, Id3, Either, Rapture and Backprop. Id3 [10] is a system for inducing decision trees. Either [7] learns and revises propositional Horn-clause theories. Rapture [4] is a system for revising certainty-factor rule bases using neural networks. Kbann [13, 6] revises a logical theory using a hybrid of symbolic and connectionist learning methods.

DNA Promoter Recognition Figure 2 shows the initial logical theory for recognising 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. The logical theory was converted into a Bayesian network such that all the logical *and*s in the domain theory were mapped onto *noisy-and* nodes and all the logical *or*s were mapped onto *noisy-or* nodes. Each 4-valued input feature was converted into *four* binary-valued features[1]. This network was translated into a neural network as described earlier. The initial weights were all set to random values close to 1.0 to mimic the initial logical theory.

```
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).
```

Fig. 2: DNA Promoter Recognition - Initial Domain Theory

The data consisted of 106 patterns (53 positive and 53 negative examples). Figure 3 shows the learning curve determined from this experiment. This graph is a plot of the average accuracy of the network at classifying DNA strings over 25 different random splits. It clearly demonstrates that our technique is successful in improving the accuracy of the network substantially (by about 40 percentage points). The graph also shows the performance of some of the inductive learning algorithms (Id3 and Backprop) and theory revision algorithms (Either, Rapture and Kbann) on the same task. The theory revision systems started out with the same initial theory as Banner(Figure 2), which they subsequently revised to fit the data. Our technique shows a performance that is better than Either and comparable to both Rapture and Kbann, although its learning curve is not as steep.

Splice Junction We also evaluated Banner on the task of learning to recognise the *splice junctions* in a given DNA sequence [6]. There are two kinds of splice junctions: IE sites and EI sites. These form the two output categories. There are 60 input features, each of which can take on the values A, C, G or T. The initial logical domain theory was converted into Bayesian network and subsequently into a neural network

[1] 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. We will extend our technique to accommodate multi-valued features in the future.

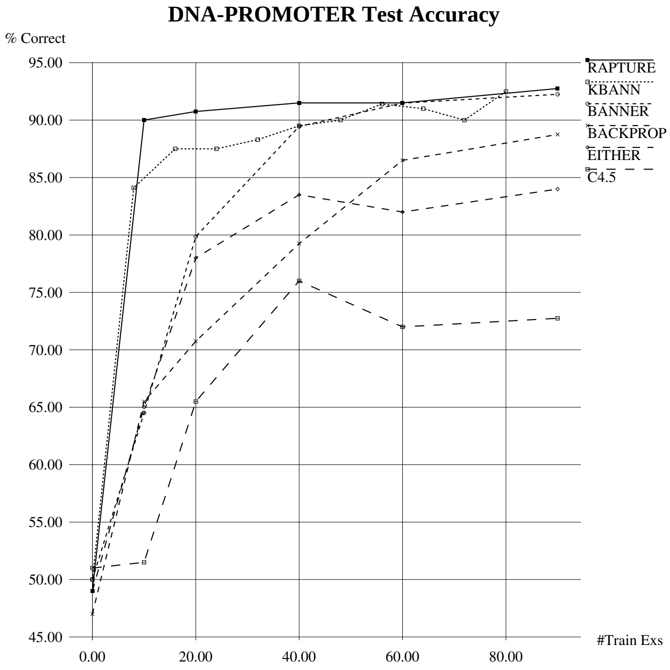

Fig. 3: DNA Promoter Recognition - Prediction Accuracy

as described in the previous subsection. The weights in the neural network were initialised to random values close to 1.0 to mimic the logical theory.

The data consisted of 2190 patterns, of which we randomly selected 900 patterns which were then divided into training and test sets. Figure 4 shows the learning curves for BANNER and some of the other inductive learning algorithms. The graph shows that BANNER performs as well as KBANN and BACKPROP. Although RAPTURE performs better than BANNER it should be noted that RAPTURE modifies the structure of the domain theory as well, which BANNER is not yet equipped to do.

6. Conclusion and Extensions

We have proposed a method for learning the conditional probabilities on a Bayesian network inductively by mapping them onto a neural network and using standard backpropagation techniques to fit the data. Since general Bayesian networks lead to exponentially large neural networks, we have focussed on simpler networks with only *noisy-or* and *noisy-and* nodes. Experimental evaluation of our technique on the DNA promoter recognition problem shows that it is comparable to some of the other induction and theory revision methods such as BACKPROP and KBANN. However, since the networks learned by BANNER map directly onto Bayesian networks, their structure and their parameters have more clearly defined semantics than the networks learned by BACKPROP and KBANN.

References

[1] W. Buntine. Theory refinement on Bayesian networks. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 52–60, 1991.

[2] G. G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[3] David Heckerman, Dan Geiger, and David M. Chikering. Learning Bayesian networks: The combination of knowledge and statistical data. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 293–301, Seattle, WA, July 1994.

[4] J. J. Mahoney and R. J. Mooney. Combining connectionist and symbolic learning to refine certainty-factor rule-bases. *Connection Science*, 5:339–364, 1993.

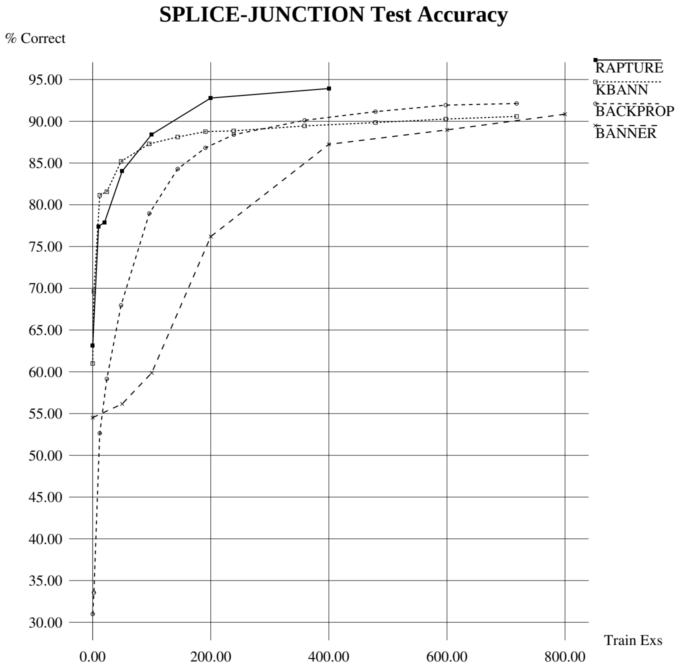

Fig. 4: Splice Junction - Prediction Accuracy

[5] James L. McClelland and David E. Rumelhart. *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises.* The MIT Press, Cambridge, MA, 1988.

[6] M. O. Noordewier, G. G. Towell, and J. W. Shavlik. Training knowledge-based neural networks to recognize genes in DNA sequences. In *Advances in Neural Information Processing Systems*, volume 3, San Mateo, CA, 1991. Morgan Kaufman.

[7] D. Ourston and R. J. Mooney. Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, 66:311–344, 1994.

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

[9] Gregory M. Provan and Moninder Singh. Learning Bayesian networks using feature selection. In *Proceedings of the Workshop on Artificial Intelligence and Statistics*, 1994.

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

[11] B. L. Richards and R. J. Mooney. Automated refinement of first-order Horn-clause domain theories. *Machine Learning*, 19(2):95–131, 1995.

[12] E. Schwalb. Compiling Bayesian networks into neural networks. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 291–297, Amherst, MA, June 1993.

[13] G. G. Towell, J. W. Shavlik, and Michiel O. Noordewier. Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 861–866, Boston, MA, July 1990.