Local learning in probabilistic networks with hidden variables

Stuart Russell,"John Binder, Daphne Koller, +Keijl Kanazawa
Computer Science Division
University of California
Berkeley, CA 94720, USA

Abstract

Probabilistic networks which provide compact descriptions of complex stochastic relationships among several random variables are rapidly becoming the tool of choice for uncertain reasoning in artificial intelligence We show that networks with fixed structure containing hidden variables can be learned automatically from data using a gradient-descent mechanism similar to that used in neural networks We alio extend the method to networks with intensionally represented distributions, including networks with continuous variables and dynamic probabilistic networks Because probabilistic networks provide explicit representations *of* causal structure human experts can easily contribute pnor knowledge to the training process, thereby significantly improving the learning rate *Adaptive probabilistic networks* (APNs) may soon compete directly with neural networks as models in computational neuroscience as well as in industrial and financial applications

1 Introduction

Intelligent systems whether biological or artificial require the ability to make decisions under uncertainty using the available evidence Several computational models exhibit some of the required functionality For example, *neural networks* which represent complex input/output relations using combinations of simple nonlinear processing elements are a familiar tool in AI and computational neuroscience *Probabilistic networks* (also called *belief networks* or *Bayesian networks)* are a more explicit representation of the joint probability distribution characterizing a problem domain providing a topological description of the causal relationships among variables

Computational models in AI are judged by two main criteria ease of creation and effectiveness in decision making (Cognitive science and neuroscience add the criterion of biological plausibility) Some computational models are associated with learning algorithms that construct specific models automatically from data, adapting to reality rather than to an expert s conception thereof Neural networks for example use a localized gradient-descent scheme to learn the model from the data The resulting ease of construction and the biological plausibility of this approach have contributed significantly to the popularity of neural networks The drawbacks ol current learning schemes include the need for large amounts of training data and the incomprehensibility of the resulting models Furthermore many of the computational models that are associated with a learning algorithm are not the most effective models for decision making Probabilistic networks on the other hand, perform well in complex decision-making domains such as medical diagnosis but have usually required a good deal of construction effort

In this paper we present a new learning algorithm for probabilistic networks that is effective even when some of the variables are *hidden*—that is their values are not observable This makes probabilistic network;, competitive with neural networks in terms of ease of creation In fact because probabilistic networks have a precise local semantics it is quite possible for human experts or other computational systems to provide prior knowledge to the learning process thereby reducing the need for training data Moreover the output of the learning process is comprehensible to humans

The paper begins with a basic introduction to probabilistic models in AI We then present the following results

- Derivation of a gradient-descent learning algorithm for probabilistic networks with hidden variables, where the gradient can be computed locally by each node using information that is available in the normal course of probabilistic network calculations
- Extensions of the algorithm to handle mtensionally represented distributions (such as noisy-OR nodes) continuous variables and dynamic probabilistic networks representing temporal processes
- Experimental demonstration of the algorithm on small and large networks showing a dramatic improvement in learning rate resulting from inclusion of hidden variables
- Experimental demonstration of the extended algorithm applied to a dynamic probabilistic network

We conclude that adaptive probabilistic networks (APNs) may provide an excellent tool for scientists and engineers in building complex models from noisy data Our results also motivate the use of a much broader class of models satisfying

'This research was supported by NSF gram 1RI 9058427 (PY1)
[1] Daphne Roller was supported by a University of California President s Postdoctoral Fellowship and an NSF Postdoctoral Associateship in Experimental Science

the basic requirements of computational neuroscience than is commonly considered

2 Probabilistic networks

Systems based on probability theory now dominate the fields of expert systems and speech recognition, and are making rapid progress in language understanding and computer vi won Here we provide only a brief introduction For a thorough treatment see Pearl [1988]

Probability theory views the world as a set of random variables X_1 X_n each of which has a domain of possible values For example in describing cancer patients the variables LungCancer and Smoker can each take on one of the values True and False The key concept in probability theory is the point probability distribution, which specifies a probability tor each possible combination of values for all the random variables Given this distribution one can compute any desired probability given any combination ot evidence For example given observations and test results one can compute the probability that the patient has lung cancer

Unfortunately, an explicit description of the joint distri bution requires a number of parameters that is exponential in n the number of variables Probabilistic networks derive their power from the ability to represent conditional independences among variables, which allows them to take advantage of the locality of causal influences Intuitively a variable is independent of its indirect causal influences given its direct causal influences In Figure 1 for example the outcome ot the X-ray does not depend on whether the patient is a smoker given that we know that the patient has lung cancer If each variable has at most k other variables that directly influence it then the total number of required parameters is linear in n and exponential in k This enables the representation ot quite large problems For example the CPCS network [Pradhan et al 19941 contains 448 variables and compares well with the world s leading diagnosticians in internal medicine

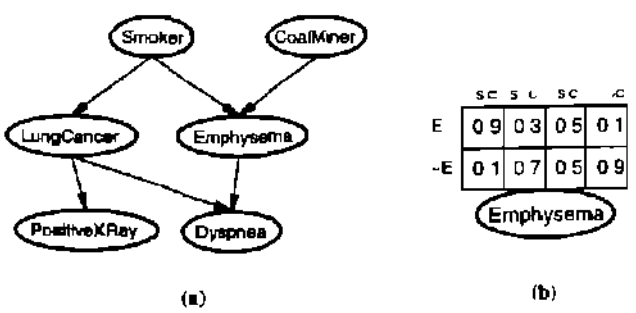

Figure 1 (a) A Simple probabilistic network showing a proposed causal model (b) A node with associated conitional probability table The table gives the conditional probability of each possible value of the variable Emphysema, given each possible combination of values of the parent nodes Smoker and Coalminer

Formally a probabilistic network is defined by a directed acyclic graph together with a conditional probability table (CPT) associated with each node (see Figure 1)' Each node represents a random variable The CPT associated with variable X specifies the conditional distribution P(X | Parents(X)) The arcs encode probabilistic dependence in the sense that each variable must be conditionally independent of its nondescendants in the graph given its parents This constraint implies that the network provides a complete representation of the joint distribution through the following equation

$$P(X_1 \quad X_n = \prod_{I=I}^{n} P(x, \mid \text{Parents}(X,)) \qquad (1)$$

where $P(X_1 \quad x_n)$ is the probability of a particular combina tion of values for $X\backslash \quad X_n$

Once a network has been constructed inference algorithms operate on it to calculate probabilities for query variables given values for evidence variables It is important to note that the distinction between evidence and query variables is entirely flexible—any variable can be set and any variable can be queried The best exact inference algorithms typically use a transformation to Markov random fields [Launtzen and Spiegelhaller 1988] Stochastic approximation algorithms using Monte Carlo simulation have also been developed [Pearl 1988] Although the general inference problem is likely to be of exponential complexity in the worst case large networks are often solvable in practice Massive parallelism can easily be applied particularly with simulation algorithms

3 Learning probabilistic networks

How can probabilistic networks be learned from data? There are several variants of this question The structure of the network can be known or unknown and the variables in the network can be observable or hidden in all or some of the data points (The latter distinction is also described by the terms complete data and incomplete data)

The case of known structure and fully observable Variables is the easiest In this case we need only learn the CPT entries Since every variable is observable each data case can be pigeonholed into the CPTentries corresponding to the values of the parent variables at each node Simple Bayesian updating then computes posterior values for the conditional probabilities based on Dirichlet priors fOlesen et al 1992 Spiegelhalterer al 19911

The case of unknown structure and fully observable variables has also received some attention In this case the problem is to reconstruct the topology of the network—a discrete optimization problem usually solved by a greedy search in the space of structures [Cooper and Herskovits 1992 Heckerman et al 1994] For any given proposed structure the CPTs can be reconstructed as described above The resulting algorithm-, are capable of recovering fairly large networks from large data sets with a high degree ot accuracy

In this paper we are mostly concerned with problems in which the structure is fixed but some variables are hidden [2]

[1]We have described the simplest form of network Networks can include continuous as well as discrete variables provided the representation of the conditional density function is finite CPTs can be represented implicitly by paramelenzed lunctions instead ot explicit tables

[2]We also note that an algorithm for learning CPTs on a fixed structure with hidden variables can be applied to the general case of hidden vanables and unknown structure by wrapping a structural search algorithm around it. However the structural search algonlhm must be more powerful than Lhose described above for the fully observable case since it may need to introduce new hidden variables

This case often occurs 1 n practice since causal structure is a lot easier to elicit from experts than numbers, whereas data cases are unlikely to contain values for all the relevant variables For example the causal connections between diseases and their symptoms are often known and medical records can easily provide a large number of data cases But the medical records are not typically complete data points the actual disease is often not observed directly we rarely have results for all possible clinical tests and so on Furthermore causal models often contain variables that are sometimes inferred but never observed directly such as syndromes in medicine

The fixed-structure, hidden-variable case has been studied by several researchers The earliest work of which we are aware is thai by Golmard and Mallet [1991] who describe an algonthm for learning in tree-structured networks The general case of directed acyclic graphs was addressed by Launtzen 11991 1995] (See also the discussions in ISpiegelhalter *et al* 1993 Olesen *et al* 1992 Spiegelhalter and Cowell, 1992]) These papers describe the application of the EM (Expectation Maximization) algorithm [Dempster *et al* 1977] to probabilistic networks EM like gradient descent, finds local maxima on the likelihood surface defined by the network parameters Launtzen notes some difficulties with the use of EM for this problem and suggests gradient descent as a possible alternative Thiesson is currently undertaking direct comparison of the performance of the two approaches A third possible approach is to use Gibbs sampling [Heckerman personal communication] Buntine 11994] in the course of a general mathematical analysis of structured learning problems also suggests that one could use generalized network differentiation for learning probabilistic networks with hidden variables

As mentioned above the gradient-descent approach for belief network learning is closely related to neural network learning an analogy observed by Neal [19921 Neal derives an expression for the likelihood gradient in sigmoid networks using stochastic simulation and uses it to show that the Boltzmann Machine (a variety of neural network) is a special cast, of a probabilistic network The Helmholtz machine ' [Dayan *et al* in press] is a hybnd of neural network and probabilistic network ideas It restricts the kinds of probability distributions that can be represented in an attempt to retain the linear-time execution property of neural networks

One might ask why the known-structure hidden vanable problem cannot be reduced to the fully observable case by eliminating the hidden variables using marginalization (averaging out) There are two reasons for this First it is not necessarily the case that any particular vanable is hidden in all the observed cases (although we do not rule this out) Second networks with hidden variables can be more *compact* than the corresponding fully observable network (see Figure 2) In general if the underlying domain has significant local structure, then with hidden vanables it is possible to take advantage of that structure to find a more concise representation for the joint distribution on the observable vanables This in turn makes it possible to learn from fewer examples

Before describing the details of our solution we will explain the task in more detail The algorithm is provided with a network structure and initial (randomly generated) values for theCPTs It is presented with a set D of data cases D_1 D_m We assume that the cases are generated independently from some underlying distribution In each data case values are

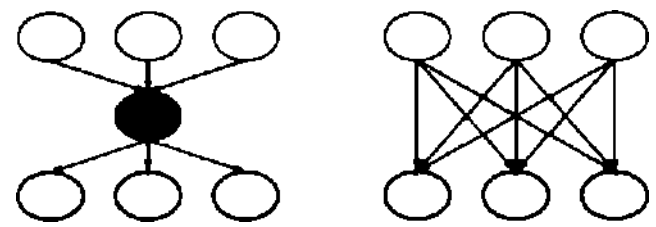

Figure 2 (a) A probabilistic network with a hidden vanable la belled *H* (*H* is two valued and the other vanables are three valued) The network requires 45 independent parameters (b) The corre sponding fully observable network which requires 168 parameters

given for some subset of the vanables this subset may differ from case to case The object is to find the CPT parameters w that best model the data We adopt a Bayesian notion of best More specifically we assume that each possible setting of w is equally likely *a prion* so that the *maximum likelihood* model is appropriate This means that the arm is to maximize $P_w(D)$ the probability assigned by the network to the observed data when the CPT parameters are set t o w [1]

4 Gradient-descent algorithms

Our approach is based on viewing the probability $P_W(D)$ as a function of the CPT entnes w This reduces the problem to one of finding the maximum of a multivanate nonlinear function Algonthms for solving this problem typically take small steps on the surface whose coordinates are the parameters and whose height is the value ot the function trying to get to the highest point on the surface In fact it turns out to be easier to maximize the log-likelihood function I*n* $P_m(D)$ Since the two functions are monotonically related maximizing one is equivalent to maximizing the other

The simplest variant of this approach and the one we use is *gradient descent* (also known as "hill-climbing) At each pointw itcomputesVw the *gradient* vector of partial denva lives with respect to the CPT entries The algorithm then takes a small step in the direction of the gradient Naively this would be to the point w + o V_w where *a* is the step-size parameter However we need to be more careful We ac tually want to maximize $P_W(D)$ subject to the constraint that w consists of conditional probability values which must be between 0 and 1 Furthermore in any CPT the entnes corre sponding to a particular conditioning case (an assignment of values to the parents) must sum to 1 Standard results show that taking a step in the direction V_w and then renormalizing to the constrained surface achieves the same effect In par ticular when an edge of the parameter space is reached this algorithm will have the effect of following it The algonthm terminates when a local maximum is reached that is when the renormalized gradient is zero

[3]Compare this to the neural network task minimize $E_W(D)$ the sum of squared differences between observed and predicted data val ues when the network weights arc set to w It has been pointed out that both maximum likelihood and neural network methods some umes find local maxima at extreme values of their parameters which can cause problems It is possible that such problems can be avoided by carrying through the analysis for nonuniform pnors

By moving in the direction of the gradient this simple algorithm executes a greedy hill-climbing procedure A variety of techniques can be used to speed up this process, such as Polak-Ribiere conjugate gradient methods Variants of this basic technique are the standard approach for training the parameters (weights) of a neural network Our results along with the results of Buntine and Neal demonstrate a very close connection between neural networks and probabilistic networks Our results are a significant extension of Neal s result since they apply to any probabilistic network

5 Local computation of the gradient

The usefulness of gradient descent depends on our ability to compute the gradient efficiently This is one of the main keys to the success of gradient descent in neural networks There *back-propagation* is used to compute the gradient of the function encoded by the neural network with respect to the network parameters (i e , the weights on the links) The existence of a simple local algorithm for training the network allows one to use the same network and algorithms for both training and inference Furthermore the similarity to real biological processes lends a certain plausibility to the entire neural-network paradigm

We now show that a similar phenomenon occurs in probabilistic networks In fact for probabilistic networks, no back-propagation is needed The gradient can be computed locally by each node using information that is available in the normal course of probabilistic network calculations In our derivation we will use the standard notation Wrjk to denote a *specific CPT entry the probability that variable X, takes on its* jlh possible value assignment given that its parents U, take on their kth possible value assignment

$$w_{ijk} \equiv P(X_i = x_{ij} \mid \mathbf{U}_i = \mathbf{u}_{ik}) \tag{2}$$

First we show that we can compute the contribution of each case to the gradient separately and sum the results

$$\begin{aligned}
\frac{\partial \ln P_{\mathbf{w}}(\mathbf{D})}{\partial w_{ijk}} &= \frac{\partial \ln \prod_{l=1}^{m} P_{\mathbf{w}}(D_l)}{\partial w_{ijk}} \qquad \text{(independent cases)} \\
&= \sum_{l=1}^{m} \frac{\partial \ln P_{\mathbf{w}}(D_l)}{\partial w_{ijk}} \\
&= \sum_{l=1}^{m} \frac{\partial P_{\mathbf{w}}(D_l)/\partial w_{ijk}}{P_{\mathbf{w}}(D_l)} \qquad (3)
\end{aligned}$$

Now the aim is to find a simple local algorithm for computing each of the expressions $\frac{\partial P_{\mathbf{w}}(D_l)/\partial w_{ijk}}{P_{\mathbf{w}}(D_l)}$ In order to get an expression in terms of local information we introduce X_i and $\mathbf{U}_i$ by averaging over their possible values

$$\begin{aligned}
&\frac{\partial P_{\mathbf{w}}(D_l)/\partial w_{ijk}}{P_{\mathbf{w}}(D_l)} \\
&= \frac{\frac{\partial}{\partial w_{ijk}} \left(\sum_{j',k'} P_{\mathbf{w}}(D_l \mid x_{ij'}\ \mathbf{u}_{ik'}) P_{\mathbf{w}}(x_{ij'}\ \mathbf{u}_{ik'}) \right)}{P_{\mathbf{w}}(D_l)} \\
&= \frac{\frac{\partial}{\partial w_{ijk}} \left(\sum_{j',k'} P_{\mathbf{w}}(D_l \mid x_{ij'}\ \mathbf{u}_{ik'}) P_{\mathbf{w}}(x_{ij'} \mid \mathbf{u}_{ik'}) P_{\mathbf{w}}(\mathbf{u}_{ik'}) \right)}{P_{\mathbf{w}}(D_l)}
\end{aligned}$$

For our purposes, the important property of this expression is that w_{ijk} appears only in linear form In fact w_{ijk} appears only in one term in the summation the term for $j' = j$ $k' = k$ For this term $P_{\mathbf{w}}(x_{ij'} \mid \mathbf{u}_{ik'})$ is just w_{ijk} Hence

$$\begin{aligned}
\frac{\partial P_{\mathbf{w}}(D_l)/\partial w_{ijk}}{P_{\mathbf{w}}(D_l)} &= \frac{P_{\mathbf{w}}(D_l \mid x_{ij}\ \mathbf{u}_{ik}) P_{\mathbf{w}}(\mathbf{u}_{ik})}{P_{\mathbf{w}}(D_l)} \\
&= \frac{P_{\mathbf{w}}(x_{ij}\ \mathbf{u}_{ik} \mid D_l) P_{\mathbf{w}}(D_l) P_{\mathbf{w}}(\mathbf{u}_{ik})}{P_{\mathbf{w}}(x_{ij}\ \mathbf{u}_{ik}) P_{\mathbf{w}}(D_l)} \\
&= \frac{P_{\mathbf{w}}(x_{ij}\ \mathbf{u}_{ik} \mid D_l)}{P_{\mathbf{w}}(x_{ij} \mid \mathbf{u}_{ik})} \\
&= \frac{P_{\mathbf{w}}(x_{ij}\ \mathbf{u}_{ik} \mid D_l)}{w_{ijk}} \qquad (4)
\end{aligned}$$

This last equation allows us to piggyback the computation ot the gradient on the calculations of posterior probabilities done in the normal course of probabilistic network operation Essentially any standard probabilistic network algorithm when executed with the evidence D_1 will compute the term $P_{w,}(x_{rl}\ u_{rk} \mid D_1)$ as a by-product We are therefore able to use a standard commercial package (Hugin) for the required inference calculations

The gradient vector can now be obtained as follows We run an inference algorithm on each data case D_1 separately computing $P_w(X_{sl}\ u_{lk} \mid D\mid)$ for each *tjk* in the process We then sum these expressions over the different data cases / and divide by w_{rjk} This is then used as outlined above Section 7 describes results obtained from our implementation

6 Extensions for generalized parameters

Our analysis above applies only to networks where there is no relation between the different parameters (CPT entries) in the network Clearly this is not always the case If we do a particular clinical test twice the parameters associated with these two nodes in the network should probably be the same (even though the results can differ) In many situations the causal influences on a given node are related so that more compact representations than an explicit CPT are called tor Viewing a CPT as a function from the parent values U_{ik} and the child value x_{rj} to the number $P(X_1\text{-}x_{tj} \mid U_r = U_{rk})$, it is often reasonable to describe this function intensionally using a small number of parameters For example, we may choose to describe this function as a neural network In other contexts we might have more information about the structure of this function A *noisy or model* for example encodes our belief that a number of diseases all have an independent chance of causing a certain symptom We then have a parameter A, describing the probability that disease *i* in isolation causes the sympiom The probability ot the symptom appearing given a combination of diseases is fully determined by these parameters It the symptom node has *k* parents the CPT for the node can De described using *k* rather than 2^k parameters (assuming that all nodes are binary-valued) Using noisy-or nodes can make an otherwise intractably large network practical For example the CPCS network mentioned above has only 8 254 parameters but would require 133 931 430 parameters if the CPTs were defined by explicit tables

Given that we want our network to be defined using parameters that are different from the CPT entries themselves, we would like to learn these parameters from the data Our basic algorithm remains unchanged rather than doing gradient ascent over the surface whose coordinates are the CPT entries

we do gradient ascent over the surface whose coordinates are these new parameters The only issue we need to address is the computation of the gradient with respect to these parameters As we now show our analysis can easily be extended to this more general case using a simple application of the *Chain Rule* for derivatives Technically assume that the network is defined using some vector of parameters A whose values we are trying to adjust Each CPT entry w_{yk} can be viewed as a function $w_{yk}(x)$ Assuming these functions are differentiable we obtain the following

$$\frac{\partial \ln P_{\mathbf{w}}(\mathbf{D})}{\partial \lambda_m} = \sum_{i,j,k} \frac{\partial \ln P_{\mathbf{w}}(\mathbf{D})}{\partial w_{ijk}} \times \frac{\partial w_{ijk}}{\partial \lambda_m} \qquad (5)$$

Our analysis above shows how the first term in each product can be easily computed as a by-product of any standard probabilistic network algorithm The second term requires only a simple function application

The ability to learn intensionally represented probabilistic networks confers many advantages First as we argued certain networks are simply impractical unless we reduce the size of their representation in this way This is even more important when learning such networks since learning each CPT entry separately would almost certainly require an unreasonable amount of training data This is another instance where our algorithm is able to utilize prior knowledge in the right way to speed up the learning process But even more importantly this ability allows us to learn networks that otherwise would not fit into this framework For example as we mentioned above, probabilistic networks can also contain continuous-valued nodes The 'CPT" for such nodes must be intensionally defined, for example as a Gaussian distribution with parameters for the mean and the variance [Launtzen and Wermuth 1989] Equation 5 gives us the fundamental tool needed for learning such networks

Perhaps the most important application of Equation 5 is in learning *dynamic probabilistic networks* (DPNs) i e , networks that represent a temporal stochastic process Such networks are typically divided into *time slices* where the nodes at each slice encode the state at the corresponding time Figure 3 shows the coarse structure of a generic DPN The CPTs for a DPN include a *state evolution model* which describes the transition probabilities between states and a *sensor model,* which describes the observations that can result from a given state Typically one assumes that the CPTs in each slice do not vary over time The same parameters therefore will be duplicated in every time slice in the network In this case we can show that Equation 5 simplifies out to the sum of the gradients corresponding to the different instances of the parameter (Section 7 demonstrates the application of this algorithm to a simple example) As a way of modelling a partially observable process DPNs compete directly with hidden Markov models (HMMs) DPNs allow the decomposition of the hidden state into several variables potentially revealing additional structure in the process being modelled and improving inductive performance Intuitively a DPN represents *n* bus of state information using *O(n)* state variables whereas an HMM uses $0(2^n)$ states If the state evolution model can be described compactly in terms of the the CPTs for the state variables we would expect DPNs to outperform HMMs on problems with large state spaces

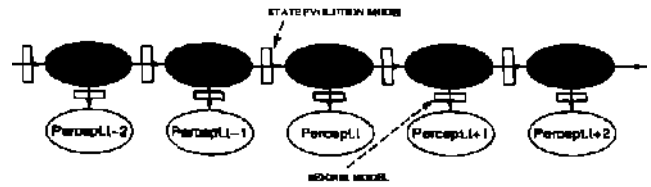

Figure 3 Generic structure of a dynamic probabilistic network In an actual network there may be many state and sensor variables in each Lime slice

7 Experimental results

We report on three experiments The first shows the importance of prestructuring the probabilistic network using hidden variables The second shows the effectiveness of the algo nthm on a large network with many hidden variables The third demonstrates the capability for learning a model of a temporal process with hidden variables

The basic tools we need are a probabilistic inference engine and a gradient-descent algorithm For the former, we use Hugin in the hrst two experiments and our own stochastic simulation system for the third experiment For the latter we have adapted the conjugate gradient algorithm [Price 1992] to keep the probabilistic variables in the legal [0]] range as described above This uses repeated line minimizations (with direction chosen by the Polak-Ribiere method) and a heuristic termination condition to signal a maximum In each exper iment training cases are generated by stochastic sampling from the distribution defined by each network for the observ able variables, new training cases are added incrementally into the existing training set

The performance of the algorithm is measured as a function of the number of training cases (X-axis) Among the possible choices for the performance metric (Y-axis) the most obvious would be the probability *P(D')* assigned by the learned network to a set of test data D' generated from the original net work, or possibly the Kullback-Liebler distance from Lhe true distribution if available However in order to facilitate com parison with traditional algorithms which have fixed inputs and outputs we designate certain observable nodes as out puts ' and measure the ability of Lhe learned network to predict the output values given values for Lhe remaining observable nodes More precisely we measure the mean square error for each output node probability value where the mean is taken over the distribution of input values (In large networks, this is approximated by sampling)

The first experiment uses data generated from the 3-1-3 network in Figure 2(a) We ran three algorithms on this data an APN with the 3-3' structure shown in Figure 2(b) a backpropagation neural network with a maximum-likelihood energy function and an APN with theonginal 3-1-3" struc ture The neural network had three nodes for each of the three-valued nodes in the probabilistic network, and used lo cal coding the number of nodes in the hidden layer was optimized using 10-fold cross-validation The resulls, shown in Figure 4, demonstrate the advantage of using the network structure that includes the hidden node

The second experiment uses data generated from a network for car insurance risk estimation (Figure 5) The network has 27 nodes of which only 15 are observable and over 1400 parameters Three of the observable nodes are designated as

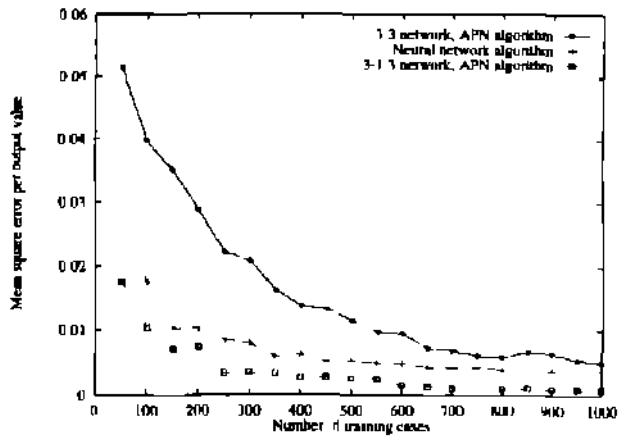

Figure 4 The output prediction accuracy as a function of the number of cases observed for data generated from the network shown *in* Figure 2(a) The three curves are for the APN algorithm using the network structure in Figure 2(b) a back propagation neural network using 10-fold cross validation and the APN algorithm using the correct network structure

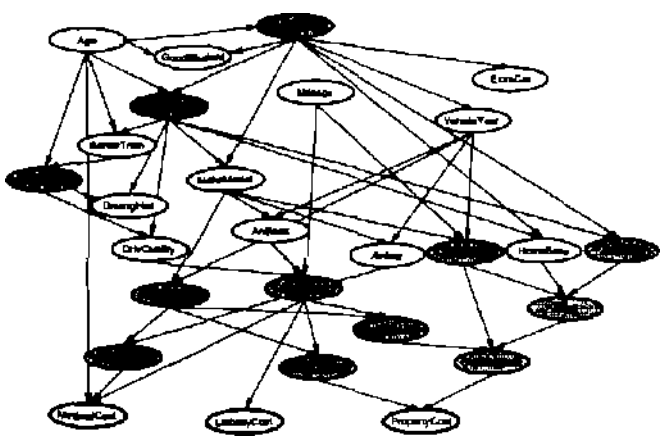

Figure 5 A network for estimating the expected claim costs for a car insurance policyholder Hidden nodes are shaded and output nodes arc shown with heavy lines

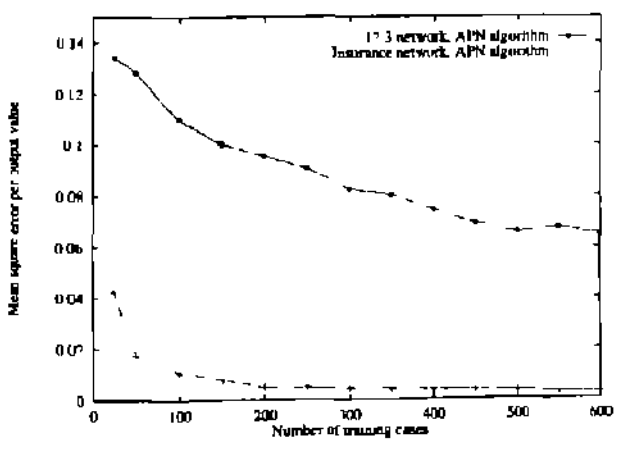

Figure 6 The prediction accuracy as a function of the number of cases observed for data generated from the network shown in Figure 5 The two curves are for the APN algorithm using a 12-3 network and the APN algorithm using the correct structure

outputs ' We ran an APN with the correct structure, and an APN with a 12-1 structure analogous to the 3-3 network in Figure 2(b) The correctly structured APN learns essentially the correct distribution from around 400 cases whereas the 12-3 APN requires many thousands of cases to reach the same level (Figure 6) [4]

The third expenment uses data generated from the dynamic probabilistic network shown in Figure 7 Applying the APN algorithm with the correct network structure and using the chain rule extension from Section 6 we obtain the learning curve shown in Figure 8 For this experiment we used a stochastic simulation algorithm based on likelihood weighting [Shachter and Peot 1989] Because this algorithm pro vides anytime estimates of the required probabilities it suits our purposes very well early in the gradient descent process we need only very rough estimates of the gradient and these can be generated very quickly

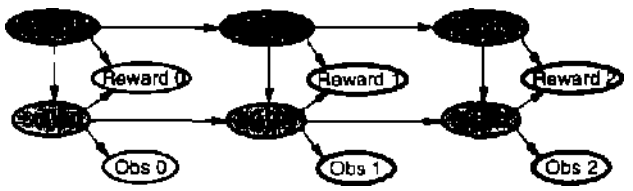

Figure 7 A simple dynamic probabilistic network modelling a partially observable Markov process with reinforcement Hidden nodes arc shaded and output nodes are shown with he ivy lines

8 Conclusions

We have demonstrated a gradient-descent learning algorithm for probabilistic networks with hidden variables that uses localized gradient computations piggybacked on the standard network inference calculations Although a detailed comparison between neural and probabilistic networks, requires more extensive analysis than is possible in this paper one is struck by the fact that the motivations tor the widespread adoption of neural networks as cognitive and neural models—localized learning massive parallelism and robust handling of noisy data—are also satisfied by probabilistic networks Furthermore the precise local semantics of probabilistic networks allows humans or other systems to provide prior knowledge to constrain the learning process We have demonstrated the dramatic improvements that can be achieved by pre structuring

[4] For comparison we applied a two layer neural network learning algonthm to the same data using cross validation to find the best size of hidden layer The neural net s behavior is interesting For all sample sizes it converges to predict just the output proportions observed in the training data independent of the particular inputs The same behavior was found for decision trees with pruning and for k nearest-neighbor learning This suggests that the complex patterns found by the structured APN are not discernible by the knowledge free algorithms One might imagine that a neural network structured similarly to the structured APN would be able to overcome this problem There arc two reasons to doubt this first, structure in a neural network represents deterministic functional dependencies rather than the probabilistic dependencies represented by the APN structure second training sparse neural networks with more than *A* few layers seems to be very difficult although we are currently trying to make it work

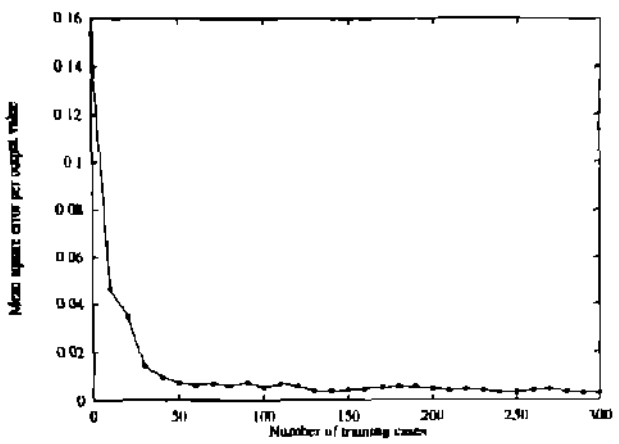

Figure 8 The prediction accuracy as a function of the number of cases observed for data generated from the network shown in Figure 7 The curve is for an APN algorithm using the correct structure

the network especially using hidden variables Theoretical analysis of the sample complexity of learning probabilistic networks is an obvious next step We would also like to investigate the use of APNs for classification rather than density estimation this can be done by altering the optimization goal to minimize the error on specified variables [Spiegelhalter and Cowell 1992] Detailed empirical comparisons with EM Gibbs and neural network methods are urgently needed

The existence of localized gradient descent algorithms for both adaptive probabilistic networks and back-propagation neural networks is no accident In other work we have established general conditions under which any distributed computational system is amenable to local learning (see also [Buntine, 1994]) Such results suggest that the class of abstract models considered in computational neuroscience can be broadened considerably

References

[Buntine 19941 Wray L Buntine Operations for learning with graphical models Journal of Artificial Intelligence Research 2 December 1994

[Cooper and Herskovits 1992] G Cooper and E Herskovits A Bayesian method for the induction of probabilistic networks from data Machine Learning 9 309-347 1992

[Deyan er al, in press] Peter Dayan Geoffrey E Hinton Radford M Neal and Richard S Zemel The helmholtz machine Neural Computation in press

[Dempster el at 1977] A Dempster, N Lajrd and D Rubin Maximum likelihood from incomplete data via the EM algorithm Journal of the Royal Statistical Society 39 (Series B) 1-38 1977

[Golmard and Mallet 1991] J-L Golmard and A MaJlet Learning probabilities in causal trees from incomplete databases Revue d Intelligence Artificielle 5 93-106 1991

[Heckerman et al 1994] D Heckerman D Geiger and M Chickering Learning Bayesian networks The combination of knowledge and statistical data Technical Report MSR-TR-94-09, Microsoft Research, Redmond, Washington 1994

[Launtzen and Spiegelhalter 1988] Steffen L Launtzen and David J Spiegelhalter Local computations with probabilities on graphical structures and their application to expert systems Journal of the Royal Statistical Society B 50(2) 157-224 1988

lLauntzen and Wermuth 1989] S L Launtzen and N Wermuth Graphical models for associations between variables some of which are qualitative and some quantitative Annate of Statistics 17 31-57 1989

[Launtzen 1991] Steffen L Lauritzen The EM algorithm for graphical association models with missing data Technical Report TR 91-05 Department of Statistics Aalborg University 1991

[Launtzen 1995] S L Launtzen The EM algorithm for graphical association models with missing data Computational Statistics and Data Analysis 19 191-201 1995

[Neal 1992] R M Neal Connectionist learning of belief networks Artificial Intelligence, 56 71-113 1992

[0lesen et al 1992] K G Olesen, S L Launtzen and F V Jensen aHUGIN A system for creating adaptive causal probabilistic networks In Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence (UA/ 92) Stanford California 1992 Morgan Kaufmann

[Pearl 1988] Judea Pearl Probabilistic Reasoning in Intelligent Systems Networks of Plausible Inference Morgan Kaufmann San Mateo California 1988

[Pradhan et al 1994] M Pradhan G M Provan B Middleton, and M Hennon Knowledge engineering for large belief networks In Proceedings of Uncertainty in Artificial Intelligence Seattle, Washington 1994 Morgan Kaufmann

[Price 1992] William H Price Numerical Recipes in C Cambridge University Press Cambidge 1992

[Shachter and Peot 1989] R D Shachter and M A Peol Simulation approaches to general probabilistic inference on belief networks In Proceedings ofthe Fifth Conference on Uncertainty in Artificial Intelligence (UAI 89) Windsor Ontano 1989 Morgan Kaufmann

[Spiegelhalter and Cowell 1992] D J Spiegelhalter and R G Cowell Learning in probabilistic expert systems In J M Bernardo Berger J O Dawid A P and Adnan F M Smith, editors Bayesian Statistics 4 1992

[Spiegelhalter et al, 1993] D Spiegelhalter P Dawid S Launtzen and R Cowell Bayesian analysis in expert systems Statistical Science 8 219-282 1993