Chapter 4

DESIGNING HIGH ORDER RECURRENT NETWORKS FOR BAYESIAN BELIEF REVISION

Ashraf M. Abdelbar

Department of Computer Science
American University in Cairo

I. INTRODUCTION

The Hopfield neural network has been used for a large number of optimization problems, ranging from object recognition [Lin *et al.*, 1991] to graph planarization [Takefuji and Lee, 1989] to concentrator assignment [Tagliarini and Page, 1987]. However, the fact that the Hopfield energy function is of quadratic order limits the problems to which it can be applied. Sometimes, objective functions which cannot be reduced to Hopfield's quadratic energy function can still be reasonably approximated by a quadratic energy function. For other problems, the objective function must be modeled by a higher-order energy function. Examples of such problems include the angular-metric TSP [Aggarwal *et al.*, 1997] and belief revision, which is our subject here.

In this chapter, we describe high-order recurrent neural networks and provide an efficient implementation data structure for sparse high-order networks. We then describe how such networks can be used for Bayesian belief revision, an important problem in diagnostic reasoning and in commonsense reasoning. We begin by introducing belief revision and reasoning under uncertainty.

II. BELIEF REVISION AND REASONING UNDER UNCERTAINTY

A. REASONING UNDER UNCERTAINTY

Humans exhibit the ability to assimilate and reason with information that is incomplete, contradictory, or subject to change. For example, most men have no trouble comprehending that, when their wives ask their opinion on a new hairstyle, they are expected to both "be honest" and "say that they love it." Similarly, if a man is driving his car, believing the car to be in fourth gear, and finds the car unable to accelerate, he is able to consider the possibility that the car is in fact in second gear. Reasoning with uncertainty is the branch of artificial intelligence that is concerned with modeling this facet of human cognition.

Conventional first-order logic is inadequate for this task. Statements are either known to be true, known to be false, or not known to be one way or the other. Further, once a statement is known to be true (or false), it stays true (or false) forever. One approach to reasoning with uncertainty is to use higher-order nonmonotonic logic [Ginsberg, 1987; Marek and Truszczynski, 1993; Reiter, 1987; Shoham, 1987]. For example, modal logic [Konyndyk, 1986; Popkorn, 1994] augments predicate logic with modal operators which take whole sentences as arguments. With modal logic, it is possible, for example, to distinguish between a statement which is false but has the potential of being true (such as Johnny is a straight-A student) and a statement which is by necessity false (such as Johnny has three eyes and six legs).

Another approach is to use numerical representations of uncertainty which may or may not be based on the probability calculus. Methods in this school can frequently be formalized in terms of belief functions. A belief function, $BEL(A)$, measures the degree to which all the evidence we have supports the hypothesis A. For this reason, numerical approaches to uncertainty are sometimes also referred to as the theory of evidence [Yager *et al.*, 1994]. Belief functions are usually defined to have a value within the interval [0, 1]. In addition, a plausibility function is defined as

$$PLAUS(A) = 1 - BEL(\neg A), \tag{1}$$

and measures the degree to which our belief in $\neg A$ leaves room for belief in A. Note that unlike probability functions,

$$BEL(A) + PLAUS(A) \neq 1. \tag{2}$$

The only requirement is that

$$BEL(A) + PLAUS(A) \leq 1. \tag{3}$$

If for a particular hypothesis A, $BEL(A) = 0$ and $PLAUS(A) = 1$, this indicates complete ignorance. While if $BEL(A) = 1$ and $PLAUS(A) = 1$, or $BEL(A) = 0$ and $PLAUS(A) = 0$, this indicates with absolute certainty that A is true, or that A is false, respectively. The most popular non-probabilistic approach to belief functions is the Dempster-Shafer theory [Dempster, 1967; Kofler and Leondes, 1994; Shafer, 1976; Shafer, 1986; Shafer and Logan, 1987]. It is also possible to define belief functions in terms of fuzzy sets [Zadeh, 1979; Zadeh, 1994, Zadeh and Kacprzyk, 1992; Zimmerman, 1991] or in terms of rough sets [Pawlak, 1991; Pawlak, 1992; Pawlak *et al.*, 1995].

Belief functions which are defined probabilistically are called *Bayesian belief functions*. A Bayesian belief function $BEL(A)$ is defined as

$$BEL(A) = P(A|\mathcal{E}), \tag{4}$$

where $\mathcal{E}$ denotes the available evidence. Pearl [1988] gives examples of how Dempster-Shafer belief functions can lead to counterintuitive reasoning. He argues convincingly that probability can be considered a "faithful guardian of common sense." Lindley [1987] contends that the probability calculus is "the only satisfactory description of uncertainty. " However, there have been two primary criticisms of probabilistic approaches to representing uncertainty. The first is that there is no way to distinguish between complete ignorance and complete uncertainty. With Dempster-Shafer belief functions,

$$\begin{aligned} BEL(A) &= 0 \\ PLAUS(A) &= 1 \end{aligned} \tag{5}$$

indicate total ignorance; the evidence we have gives us no reason to believe A nor to disbelieve A. However,

$$BEL(A) = PLAUS(A) = 0.5 \tag{6}$$

indicate total uncertainty; the evidence we have provides equal support to A and $\neg A$. However, in the Bayesian formalism,

$$P(A|\mathcal{E}) = P(\neg A|\mathcal{E}) = 0.5, \tag{7}$$

indicates that A and $\neg A$ are equally likely given $\mathcal{E}$; this could be because $\mathcal{E}$ supports both hypotheses equally or because it provides support to neither. The other criticism of the Bayesian formalism has historically been that the need to consider probabilistic dependencies makes probability calculations unfeasible. However, this has largely changed with the advent of Bayesian belief networks, which provide a natural and concise graphical representation of probabilistic dependencies.

B. BAYESIAN BELIEF NETWORKS

Incarnations of Bayesian belief networks seem to have been around for some time. They have been called influence diagrams, knowledge maps, and causal diagrams. However, Judea Pearl had been largely credited with standardizing and popularizing Bayesian belief networks with his 1988 book [Pearl, 1988]. Given a set of random variables representing events or hypotheses in a given problem domain, a Bayesian belief network is a triple (V, E, P), where V is a set of nodes such that each node is identified with a domain variable, (V, E) specify a directed acyclic graph (DAG), and P is a set of probability distributions which specify for each node $\upsilon \in V$ the probability of each possible instantiation of υ given each possible instantiation of υ's parents $\pi(\upsilon)$, and such that (V, E) is a *minimal independency map* of the domain variables. This requirement that a Bayesian belief network's underlying DAG be a minimal independency map of the problem domain is what allows computations on belief networks to be greatly simplified.

The definition of an independency map, or *I-map*, is based on the notion of conditional independence. If X, Y, and Z are disjoint sets of random variables,

X and Y are conditionally independent given Z, written $I(X, Z, Y)$ if for any possible instantiations of x, y, and z, of X, Y, and Z, respectively,

$$P(x|y \wedge z) = P(x|z), \tag{8}$$

whenever

$$P(y \wedge z) > 0. \tag{9}$$

Furthermore, if D is a DAG and X, Y, and Z are three disjoint sets of nodes of D, Z is said to *d-separate* X and Y, denoted $<X, Z, Y>$, if every path from a member of X to a member of Y is blocked by a member of Z. If D is a DAG where each node of D represents a random variable, then D is an independency map if $I(X, Z, Y)$ is implied by $<X|Z|Y>$. Finally, a DAG is a minimal 1-map if the removal of any edge from the DAG renders the DAG no longer an I-map.

x	y	z	v	P(v | x,y,z)
T	**T**	**T**	**T**	0.34
T	**T**	**T**	**F**	0.66
T	**T**	**F**	**T**	0.80
T	**T**	**F**	**F**	0.20
T	**F**	**T**	**T**	0.23
T	**F**	**T**	**F**	0.77
T	**F**	**F**	**T**	0.65
T	**F**	**F**	**F**	0.35
F	**T**	**T**	**T**	0.95
F	**T**	**T**	**F**	0.05
F	**T**	**F**	**T**	0.55
F	**T**	**F**	**F**	0.45
F	**F**	**T**	**T**	0.10
F	**F**	**T**	**F**	0.90
F	**F**	**F**	**T**	0.25
F	**F**	**F**	**F**	0.75

Figure 1. Example of a local probability distribution with redundancies.

Based on the requirement that a Bayesian belief network be an I-map, the joint probability of any instantiation $\mathcal{A}$ of the nodes of a belief network (V, E, P) can be computed according to

$$P(\mathcal{A}) = \prod_{\upsilon \in V} P(\mathcal{A}(\upsilon)|\mathcal{A}(\pi(\upsilon))). \tag{10}$$

The correctness of this equation relies on the network being an I-map; while the minimality requirement enforces conciseness.

For a belief network (V, E, P), the set of probability distributions P specifies for each node υ the probability of every possible instantiation of υ given every possible instantiation of $\pi(\upsilon)$. Thus, if υ is a binary-valued node and has two parents, x and y, which are also binary-valued, then υ's probability distribution would be similar to Figure 1. However, Figure 1 contains some redundant information since for any given fixed instantiation $\mathcal{I}$ of $\pi(\upsilon)$,

$$P(\upsilon = \mathbf{T}|\mathcal{I}(\pi(\upsilon))) = 1 - P(\upsilon = \mathbf{F}|\mathcal{I}(\pi(\upsilon))) \qquad (11)$$

Therefore, it is sufficient for binary-valued nodes to specify the probability of one truth assignment for each possible instantiation of the parents, as shown in Figure 2. In general, for a discrete node with k possible instantiations, it is necessary and sufficient to specify $k - 1$ probabilities for each possible instantiation of the parents.

x	y	z	v	P(v | x,y,z)
T	**T**	**T**	**T**	0.34
T	**T**	**F**	**T**	0.80
T	**F**	**T**	**T**	0.23
T	**F**	**F**	**T**	0.65
F	**T**	**T**	**T**	0.95
F	**T**	**F**	**T**	0.55
F	**F**	**T**	**T**	0.10
F	**F**	**F**	**T**	0.25

Figure 2. Example of a local probability distribution without redundancies.

C. BELIEF REVISION

Suppose a grocery store clerk sees a man come into his store carrying a gun. Based on the observed evidence ("man in store with gun"), the clerk may develop the belief that he is about to be the victim of a robbery. If the evidence set is augmented with the observation that the "gunman" is carrying a policeman's badge, the clerk may then revise his belief.

Formally, belief revision is the problem of finding the most plausible explanation for the current evidence at hand. This has applications in many areas of AI. For example, in natural language understanding, the evidence would be the natural language text and the possible explanations would be the possible meanings of the text [Charniak and Shimony, 1994; Hobbs *et al.*, 1993]; in medical diagnosis, the evidence would be the symptoms and lab results and the explanations would be the possible diagnoses [Shachter, 1986]. For Bayesian belief networks, belief revision is the problem of finding the most probable explanation for a given set of evidence. In other words, given a set of observances, represented as a partial assignment $\mathcal{E}$ to a subset of the network variables, the objective is to find the network assignment $\mathcal{A}$ which maximizes the conditional probability $P(\mathcal{A}|\mathcal{E})$. Because it maximizes the posterior probability, $\mathcal{A}$ is called the *maximum a posteriori* assignment and the belief revision problem on Bayesian belief networks is often called the MAP assignment problem, or simply the MAP problem.

From Bayes' theorem we know that

$$P(\mathcal{A}|\mathcal{E}) = \frac{P(\mathcal{A})P(\mathcal{E}|\mathcal{A})}{P(\mathcal{E})}, \qquad (12)$$

and since $P(\mathcal{E})$ is constant, maximizing $P(\mathcal{A}|\mathcal{E})$ is equivalent to maximizing $P(\mathcal{A})$ subject to the constraint that $\mathcal{E} \subseteq \mathcal{A}$. If $\mathcal{E}$ is empty, then we are simply interested in finding the network assignment with highest unconditional probability $P(\mathcal{A})$.

D. APPROACHES TO FINDING MAP ASSIGNMENTS

An important indicator of complexity for a belief network is whether it is *singly-connected* or *multiply-connected*. A singly-connected network is a network in which, for any pair of nodes, there is only one directed path connecting them. An alternative definition is that it is a network in which the underlying undirected graph is also acyclic. A multiply-connected network is a network in which there is more than one directed path connecting at least one pair of nodes.

For singly-connected networks, Pearl [Pearl, 1986; Pearl, 1988] has developed an algorithm, based on message passing, for finding the optimal MAP in linear time. The problem of finding MAPs on multiply-connected networks is NP-hard [Shimony, 1994] and even approximating the optimal MAP within a constant factor is **NP**-hard [Abdelbar and Hedetniemi, 1998]. Existing methods for finding exact MAPs on multiply-connected networks all have exponential complexity in the worst case. Simulated annealing [Abdelbar and Hedetniemi, 1997; Abdelbar and Attia, 1999; Geman and Geman, 1984], genetic algorithm [Abdelbar and Hedetniemi, 1997; Abdelbar and Attia, 1999; Rojas-Guzman and Kramer, 1993; Rojas-Guzman and Kramer, 1994], and integer programming approaches [Abdelbar, 1998; Abdelbar, 1999; Santos, 1994; Santos and Santos, 1996] for the problem are currently being investigated.

III. HOPFIELD NETWORKS AND MEAN FIELD ANNEALING

A. OPTIMIZATION AND THE HOPFIELD NETWORK

A recurrent neural network is one whose underlying topology of inter-neuronal connections contains at least one cycle. The Hopfield network [Hopfield, 1982; Hopfield, 1984] is perhaps the best known network of this class. The underlying topology of a Hopfield network is a graph: each weighted connection is either a binary connection T_{ij} between two neurons i and j or a unary connection I_i involving a single neuron i. A neuron in the Hopfield network is governed by

$$\frac{du_i}{dt} = \sum_{j \neq i} T_{ij} V_j + I_i, \tag{13}$$

and

$$V_i = g(u_i), \tag{14}$$

where g is a (typically sigmoidal) activation function. The Hopfield network is a member of the Cohen-Grossberg [Cohen and Grossberg, 1983] family of dynamical systems. Under the requirements that the Tij matrix be symmetric and with a zero-diagonal, and that the activation function g be monotonically non-decreasing and with a sufficiently high slope, the neurons of a Hopfield network will tend towards a collective state which minimizes the energy function

$$E = \sum_{i=1}^{n} I_i V_i - \sum_{i=1}^{n} \sum_{j<i} T_{ij} V_i V_j. \tag{15}$$

The Hopfield network is used for optimization by constructing the T_{ij} and I_i connections such that the minimum points of the energy function correspond to the optimal solutions of the problem at hand. Many optimization problems can be described by an objective function that is to be minimized or maximized and a set of constraints that must be satisfied. These two components of the energy function can be constructed separately and then superimposed to form the overall energy function:

$$E = E^{obj} + \beta E^{cons}, \tag{16}$$

where E^{cons} and E^{obj} represent the energy functions corresponding to the constraints and objectives, respectively, and β is a manually-tuned scaling constant.

Certain types of constraints are encountered so frequently in the context of different problems that special design rules have been developed for their handling. Tagliarini *et al.*'s [Tagliarini *et al.*, 1991] k-out-of-n rule deals with the case where it is desired to select exactly k out of an ensemble of n neurons.

In the energy function

$$E = \left(k - \sum_i V_i\right)^2 + \left(\sum_i V_i (1 - V_i)\right), \tag{17}$$

the first term is minimized when the sum of the V_i's is k and the second term when all the V_i's have digital values. With some algebraic manipulation, it is easy to see that equation (17) has the same minimum points as

$$E = -\frac{1}{2} \sum_i \sum_{j \neq i} (-2) V_i V_j - \sum_i V_i (2k - 1). \tag{18}$$

Correspondingly, the k-out-of-n rule prescribes that we assign T_{ij} to (-2) for every pair of neurons and assign I_i to $(2k - 1)$ for every neuron in the ensemble. This design rule can be applied to simultaneous constraints and the energy functions produced by each application of the rule can be superimposed to produce the overall E^{cons} [Page and Tagharini, 1988].

B. BOLTZMANN MACHINE

Like the Hopfield network, the Boltzmann machine [Hinton and Sejnowski, 1986] can be used for optimization. The energy function of the Boltzmann machine is the same as that of the Hopfield network but, unlike the deterministic Hopfield network, the Boltzmann machine employs stochastic neurons. The activation level u_i of a neuron i in the Boltzmann machine is computed according to equation (13) as in the Hopfield network, but the output V_i of a neuron i is a binary-valued random variable with distribution:

$$P(V_i = 1) = \frac{1}{1 + e^{-\frac{u_i}{T}}}, \tag{19}$$

where T is a parameter known as the *temperature*. Initially, the temperature is set to a relatively high value and then, over time, it is gradually decreased according to some *annealing schedule*.

Note that when T is close to infinity, the probability is close to 0.5 regardless of the value of u_i; this corresponds to a random walk through weight space. On the other hand, when T is very low, network behavior becomes very similar to the discrete version of the Hopfield network.

The choice of annealing schedule for a Boltzmann machine is central to network performance. A well-known theoretical result by Geman and Geman [1984] holds that if the rate of decay of the temperature is no faster than logarithmic, the network is guaranteed to eventually converge to a global optimum. However, this schedule is very slow in practice and is rarely used. A commonly used schedule, first proposed by Kirkpatrick *et al*. [1983], is to reduce the temperature by a fixed fraction f after every iteration,

$$T^{new} = T^{old} * f. \tag{20}$$

C. MEAN FIELD ANNEALING

Mean field theory [Peterson and Hartman, 1989] can be used to obtain a deterministic approximation to the Boltzmann machine. In this variation, the output V_i of a neuron i is deterministically approximated to be

$$V_i = \tanh\left(\frac{u_i}{T}\right), \tag{21}$$

where T is the annealing temperature. This mean field approximation, often called the deterministic Boltzmann machine, has been observed to produce faster convergence than the stochastic Boltzmann machine [Peterson and Anderson, 1987].

IV. HIGH ORDER RECURRENT NETWORKS

A High Order Recurrent Network (HORN) is a recurrent network whose underlying topology is a hypergraph, i.e., it allows weighted hyperedges which connect more than two neurons. The degree of a hyperedge is the number of neurons it connects; the order of a HORN is the largest hyperedge degree in the topology. We will use the notation $T^{(d)}_{i_1 \dots i_d}$ to denote the weight of the dth-degree edge connecting neurons $i_1 \dots i_d$. A HORN is symmetric if

$$T^{(d)}_{i_1 \dots i_d} = T^{(d)}_{i_{h(1)} \dots i_{h(d)}}, \tag{22}$$

for any permutation h of the integers $1 \dots d$. By default, a HORN is assumed to be symmetric unless otherwise specified.

A kth-order HORN minimizes a kth-order energy function [Pinkas, 1995]:

$$\begin{aligned} K = &- \sum_{1 \le i_1 < i_2 < \dots < i_k \le n} T^{(k)}_{i_1 \dots i_k} V_{i_1} \dots V_{i_k} \\ &- \sum_{1 \le i_1 < i_2 < \dots < i_{k-1} \le n} T^{(k-1)}_{i_1 \dots i_{k-1}} V_{i_1} \dots V_{i_{k-1}} \\ &- \dots - \sum_{1 \le i \le n} T^{(1)} V_i, \end{aligned} \tag{23}$$

where n is the number of neurons. For example, the energy function of a fourthorder HORN has the form:

$$\begin{aligned} E = &-\sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} \sum_{l=k+1}^{n} T^{(4)}_{ijkl} V_i V_j V_k V_l - \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} T^{(3)}_{ijk} V_i V_j V_k \\ &- \sum_{i=1}^{n} \sum_{j=i+1}^{n} T^{(2)}_{ij} V_i V_j - \sum_{i=1}^{n} T^{(1)}_i V_i. \end{aligned} \tag{24}$$

If S_d denotes the set of all sequences $j_1, \dots, j_d$, such that $1 \le j_a \le n$, for $a = 1, \dots, d$, and $j_a \ne j_b$ if $a \ne b$, then, each neuron is governed by

$$\frac{du_i}{dt} = \sum_{d=1}^{k} \sum_{s \in \mathcal{S}_d, i \in s} T^{(d)}_s \prod_{j \in s, j \ne i} V_j. \tag{25}$$

In the new notation of this section, equation (13) would be expressed as:

$$\frac{du_i}{dt} = \sum_{j \ne i} T^{(2)}_{ij} V_j + T^{(1)}_i. \tag{26}$$

The relationship between the output V_i and the activation level u_i of a neuron i can follow the form either of the Hopfield network, of the stochastic Boltzmann machine, or of the mean field theory Boltzmann machine.

In a 1995 paper, Gadi Pinkas [Pinkas, 1995] shows that it is possible to transform a k^{th}-order network to a strongly equivalent quadratic-order network with an increase in the number of neurons. Although it is often more efficient in practice to simulate the high-order networks directly, Pinkas' transformation provides an important theoretical foundation for HORNs because of the Hopfield network's relationship to the Cohen-Grossberg family.

We will now briefly review the Pinkas transformation. Given a k^{th}-order network, each k^{th}-order connection $T^{(k)}_{i_1 \ldots i_k}$ is replaced by a number of lower-order connections in a manner that depends on the sign of the weight of $T^{(k)}_{i_1 \ldots i_k}$.

If the weight of $T^{(k)}_{i_1 \ldots i_k}$ is positive, then it is replaced by $(k + 1)$ second- and first-order connections and a new hidden neuron is added. If we let h denote the new hidden neuron, then the $(k + 1)$ connections are created as follows:

1. For every $j = 1, \ldots, k$, a connection $T^{(2)}_{i_j h}$ is added and its weight is set according to

$$T^{(2)}_{i_j h} = 2T^{(k)}_{i_1 \ldots i_k}. \tag{27}$$

2. A connection $T^{(1)}_{h}$ is added and its weight is set according to

$$T^{(1)}_{h} = -(2k - 1)T^{(k)}_{i_1 \ldots i_k}. \tag{28}$$

If, on the other hand, the weight of $T^{(k)}_{i_1 \ldots i_k}$ is negative, then $(k - 2)$ new hidden neurons are needed. If we let $h_3 \ldots h_k$ denote the $(k - 2)$ new hidden neurons, then the low-order connections are created as follows:

1. For every $j = k, \ldots, 3$, perform the following steps:

   (a) For $\ell = 1, \ldots, j - 1$, a new connection $T^{(2)}_{i_\ell h_j}$ is added and its weight is set according to

$$T^{(2)}_{i_\ell h_j} = -2T^{(k)}_{i_1 \ldots i_k}. \tag{29}$$

   (b) A new connection $T^{(2)}_{i_j h_j}$ is created and its weight is set according to

$$T^{(2)}_{i_j h_j} = 2T^{(k)}_{i_1 \ldots i_k}. \tag{30}$$

   (c) A connection $T^{(1)}_{h_j}$ is created and its weight is set according to

$$T^{(1)}_{h_j} = (2j - 3)T^{(k)}_{i_1 \ldots i_k}. \tag{31}$$

2. A connection $T^{(2)}_{i_1 i_2}$ is created and its weight is set according to

$$T^{(2)}_{i_1 i_2} = T^{(k)}_{i_1 \ldots i_k}. \tag{32}$$

In this manner, an arbitrary k^{th}-order network can always be converted to a strongly equivalent second-order network.

V. EFFICIENT DATA STRUCTURES FOR IMPLEMENTING HORNS

The most common implementation of a Hopfield network stores the T_{ij} and I_i connections in two-dimensional and one-dimensional arrays, respectively. To fire a neuron, say neuron i, we can use

```
delta_u = I [i];
for j = 1 to n do
        if j ≠ i then
          delta_u += T [i,j] * V [j];
od ;
```

For a HORN with dense connections, a similar approach can be adopted, using a d-dimensional array for each $T^{(d)}$. To fire a neuron, we would then have $k - 1$ nested loops, which means a computational complexity of $O(n^k)$ to fire all n neurons once.

However, for some applications such as belief revision, the HORN's of interest are sparsely-connected, that is, the majority or even the vast majority of possible connections have a connection weight of 0. For such HORNs, we propose the following data structure, which is meant to be very fast at the expense of redundance in storage.

We use an array of records, where each record holds three fields:

```
degree : integer ;
neurons: array [1..k] of integer;
weight : real ;
```

The size of the array of records is set to the maximum number of non-zero connections in the entire network, which we will denote as m. We then duplicate this array n times. This gives us the following declaration, which is illustrated in Figure 3:

```
Conn :  array [1..n,1..m] ;
```

For each neuron i, the one-dimensional array `Conn [i]` holds the connections in which i participates. Each record `Conn [i,j]` stores the specifications of one connection in which i participates. The array `Conn [i,j]. neurons` holds the neurons which participate in the connection **not including neuron *i* itself**. Since the entire `Conn [i]` deals with connections which involve i, there is no need to include i in the `neurons` array of each record; in addition the exclusion of i makes it possible to avoid including an if-statement between lines 4 and 5 in the pseudo-code below.

To fire neuron i, we can use

```
1. delta_u = 0 ;
```

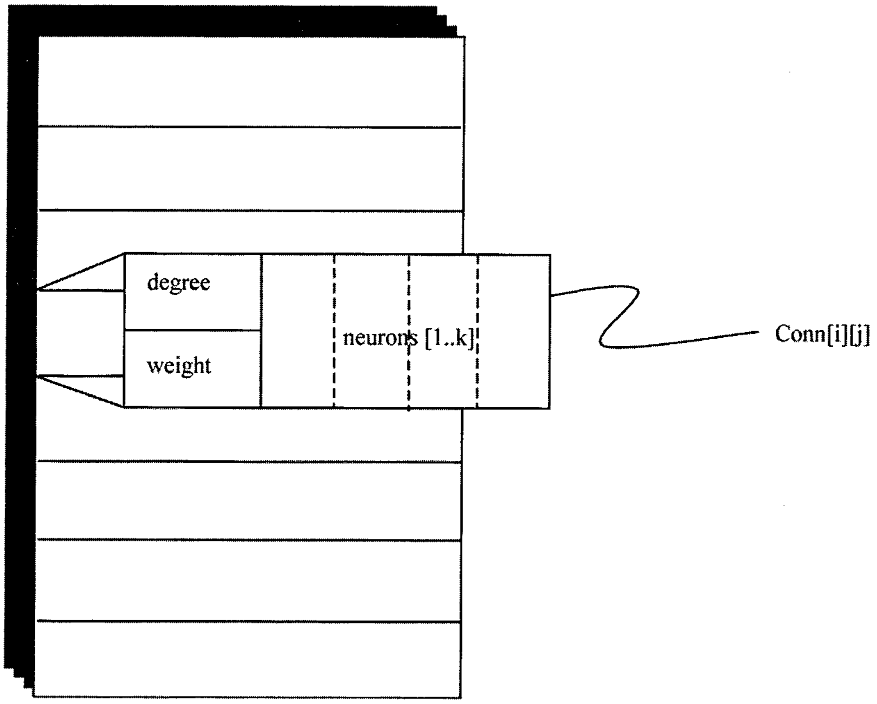


Figure 3. Each record `Conn [i][j]` describes the j th connection in which neuron `i` participates.

```
for j = 1 to m [i] do
        factor = Conn [i,j].weight ;
        for a = 1 to Conn [i,j].degree - 1 do
            factor *= V [Conn [i,j].neurons[a]];
        od ;
        delta_u += factor ;
od ;
```

We can now fire a neuron in $O(mr)$, where r is the average connection degree of the HORN, and all neurons can be fired in $O(nmr)$.

VI. DESIGNING HORNS FOR BELIEF REVISION

Bayesian belief networks are themselves essentially connectionist structures (and interestingly they meet most generic definitions of a neural network). Let us go back for a moment to the example probability distribution table shown in Figure 1. Consider for example the sixth line of this table. This line associates a probability of 0.77 with the combination of four hypotheses: that x is true, that y

is false, that z is true, and that v is false. Each line in the probability table of a belief network node with in-degree d connects $d + 1$ hypotheses. For this reason, a Bayesian belief network with a maximum in-degree of k will require a HORN of order $(k + 1)$.

Here, we present an algorithm for constructing a HORN for a given Bayesian belief network with discrete-valued (not necessarily binary) variables and a given evidence set.

Let $B = (U, E, P)$ and $\mathcal{E}$ be a Bayesian belief network and associated evidence set, respectively. For $U = \upsilon_1, \ldots, \upsilon_n$, let $\mathcal{D}(\upsilon_i)$ be the finite domain from which variable υ can be instantiated. We will assume that, for each $\upsilon \in U$, each distribution P_υ is in the form of a table, where each line ℓ in the table is in the form of a set of assignments $\{(x \rightarrow r) | x \in \{\upsilon\} \cup \pi(\upsilon), r \in \mathcal{D}(x)\}$, and we will let $P(\ell)$ denote the probability associated with line ℓ.

We construct a neural network with $\Sigma_{\upsilon \in U}|\mathcal{D}(\upsilon)|)$ neurons: a neuron υ_r is associated with every $r \in \mathcal{D}(\upsilon)$ for every $\upsilon \in U$. For each $\upsilon \in U$ and for each $\ell \in P_\upsilon$, we create a connection

$$T^{(d)}_{i_1 i_2 \ldots i_d} = \log P(\ell), \tag{33}$$

where $d = |\ell|$, and $i_a = x_r$ where $(x \rightarrow r) \in \ell$, for $a = 1, \ldots, d$.

Let the evidence be represented as a set of assignments. For each assignment $(x \rightarrow r) \in \mathcal{E}$ where $r \in \mathcal{D}(x)$:

1. Let x_r be the neuron corresponding to the instantiation r of x. Since the evidence requires x to take the value r, we can consider V_{x_r} to be clamped to 1. Therefore, we can replace every connection $T^{(d)}_{i_1 i_2 \ldots i_d}$ such that $i_a = x_r$ for some $a \in 1,\ldots, d$, with the connection $T^{(d-1)}_{i_1 \ldots i_{a-1} i_{a+1} \ldots i_d}$ letting the new connection retain the weight of the removed connection.
2. We can now remove neuron x_r (which now corresponds to a fact rather than a hypothesis) from the network and permanently assign V_{x_r} to 1.
3. For each $\mathcal{S} \in (\mathcal{D}(x) - \{r\})$, let x_s, be the neuron corresponding to the instantiation s of x. Since the evidence requires x to take on a value different from s, we can consider V_{x_s} to be clamped to 0. Therefore, we can prune every connection $T^{(d)}_{i_1 i_2 \ldots i_d}$ such that $i_b = x_s$ for some $b \in 1, \ldots, d$.
4. We can now remove neuron x_s, and permanently assign V_{x_s} to 0.

In this manner, we construct a connection corresponding to every line in every probability table in the belief network. Let these connections constitute E^{obj}.We illustrate the E^{obj} connections with a small numerical example. Consider the binary-valued belief network shown in Figure 4. Figure 5 shows the `Conn [i]` arrays that would be constructed for the evidence $\{(x \rightarrow \mathbf{T})\}$ assuming the neuron associations $(V_1 : w_T, V_2 : w_F, V_3 : x_T, V_4 : x_F, V_5 : y_T, V_6 : y_F, V_7 : z_T, V_8 : z_F)$.

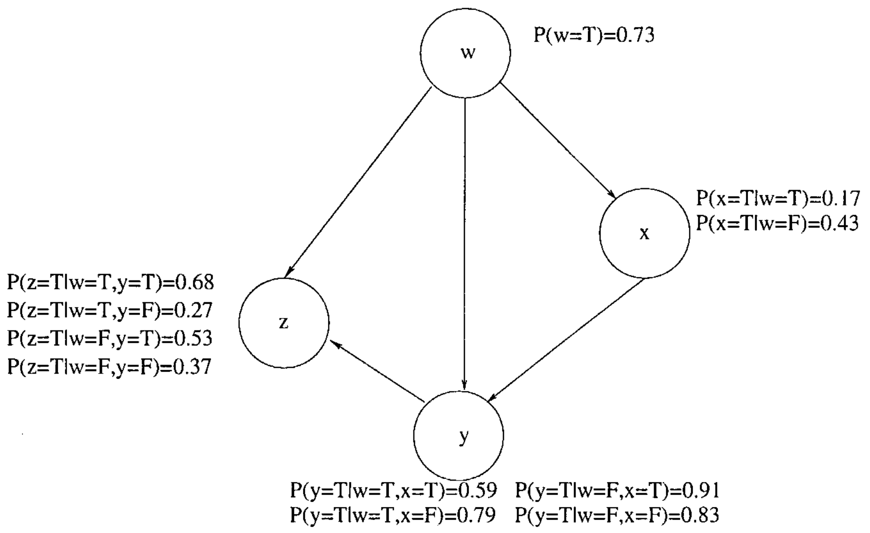


Figure 4. A small belief network

An assignment $\mathcal{V}$ to the HORNs V vector will induce a belief network assignment $\mathcal{A}$,

$$\mathcal{A} = \{(\upsilon \rightarrow r) | \upsilon \in U, \ V_{\upsilon_s} = 1\}, \qquad (34)$$

under the two constraints that all V_i's have digital values and that exactly one of $\{V_{\upsilon_r} \mid r \in \mathcal{D}(\upsilon)\}$ is equal to 1 for every $\upsilon \in U$. An assignment $\mathcal{V}$ is said to be feasible if it induces a belief network assignment $\mathcal{A}$.

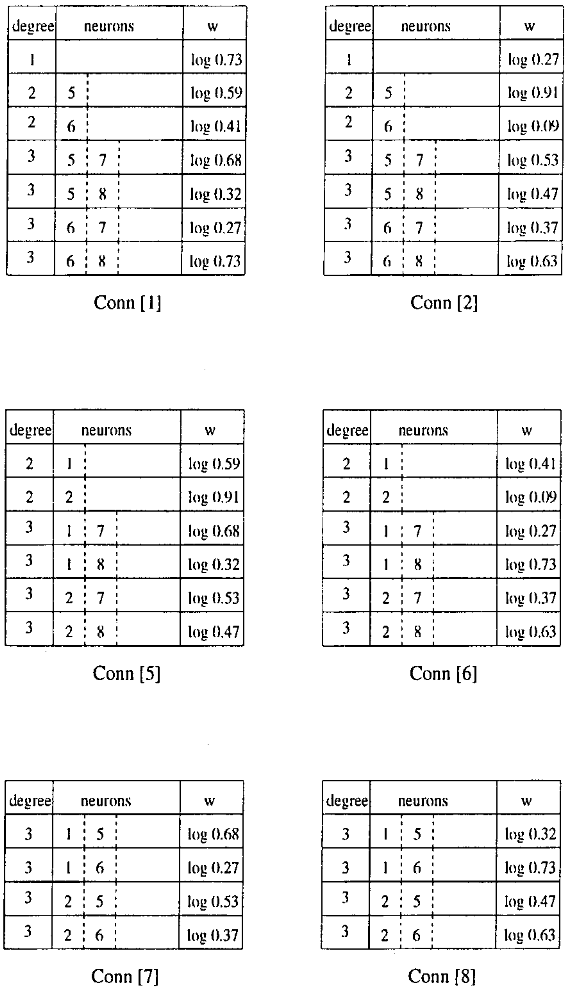
Theorem: Let $\mathcal{V}_1$ and $\mathcal{V}_2$ be two feasible neural network assignments such that $\mathcal{V}_1$ yields a lower value for E^{obj} than $\mathcal{V}_2$. Then, if $\mathcal{A}_1$ and $\mathcal{A}_2$ are the two belief network assignments induced by $\mathcal{V}_1$ and $\mathcal{V}_2$, respectively, then $\mathcal{P}(\mathcal{A}_1 | \mathcal{E}) > P(\mathcal{A}_2|\mathcal{E})$.

Proof: Maximizing $\mathcal{P}(\mathcal{A}|\mathcal{E})$ is equivalent to maximizing $\mathcal{P}(\mathcal{A})$ under the constraint $\mathcal{E} \subseteq \mathcal{A}$; the containment of $\mathcal{E}$ in $\mathcal{A}$ is guaranteed by the clamping of V_{x_r} to 1 for every $(x \rightarrow r) \in \mathcal{E}$. Every connection in the HORN corresponds to a line in the probability table of some node. Let P_υ be the probability distribution for an arbitrary $\upsilon \in U$. For any assignment $\mathcal{A}$, there is exactly one $\ell \in P_\upsilon$ such that $\ell \in \mathcal{A}$. Therefore, E^{obj} consists of exactly U non-zero terms. Each non-zero connection has a weight

$$T^{(d)}_{i_1 i_2 \ldots i_d} = \log P(\ell), \qquad (35)$$

for some $\ell \in P_\upsilon$ for some $\upsilon \in U$ and such that $\ell \in \mathcal{A}$. This means

$$E^{obj} = -\sum_{\upsilon \in U} \log P(\mathcal{A}(\upsilon) | A(\pi(\upsilon))),$$



Conn [1]

degree	neurons		w
1			log 0.73
2	5		log 0.59
2	6		log 0.41
3	5	7	log 0.68
3	5	8	log 0.32
3	6	7	log 0.27
3	6	8	log 0.73

Conn [2]

degree	neurons		w
1			log 0.27
2	5		log 0.91
2	6		log 0.09
3	5	7	log 0.53
3	5	8	log 0.47
3	6	7	log 0.37
3	6	8	log 0.63

Conn [5]

degree	neurons		w
2	1		log 0.59
2	2		log 0.91
3	1	7	log 0.68
3	1	8	log 0.32
3	2	7	log 0.53
3	2	8	log 0.47

Conn [6]

degree	neurons		w
2	1		log 0.41
2	2		log 0.09
3	1	7	log 0.27
3	1	8	log 0.73
3	2	7	log 0.37
3	2	8	log 0.63

Conn [7]

degree	neurons		w
3	1	5	log 0.68
3	1	6	log 0.27
3	2	5	log 0.53
3	2	6	log 0.37

Conn [8]

degree	neurons		w
3	1	5	log 0.32
3	1	6	log 0.73
3	2	5	log 0.47
3	2	6	log 0.63

Figure 5. The Conn arrays that result from applying the algorithm to the belief network shown in Figure 4 with $\mathcal{E} = \{(x \rightarrow \mathbf{T})\}$

	min	max	avg
Nodes	10	20	13.33
Maximum in-degree	2	4	3.095
Maximum out-degree	3	7	4.905
Average degree	2.2	3.87	3.11
Total number of entries in all probability tables	50	216	121.33
Ratio of number of evidence nodes to total nodes	0	0.9	0.5738
Number of neural network iterations	10	101	51.381

Figure 6. Summary of experimentation

$$= -\prod_{v \in U} P(\mathcal{A}(v)|A(\pi(v))) = -\log P(\mathcal{A}). \tag{36}$$

Therefore, minimizing E^{obj} is equivalent to maximizing $P(\mathcal{A})$.

What remains is to construct connections, which we will call E^{cons}, to enforce the feasibility constraints. This can be achieved using the standard k-out-of-n design rule. The two components, E^{cons} and E^{obj}, are then combined according to equation (16) with an appropriate choice of weighting constant β.

Using mean field annealing, with the schedule of (20), this technique has found the optimal assignments for a collection of twenty belief networks and evidence sets with the characteristics shown in Figure 6. For each belief network and evidence set, extensive experimentation, however, is required to obtain good values for the three parameters: β, initial temperature T_0, and temperature cooling factor f. Performance is especially sensitive to β and f. It is hoped that heuristics can be developed for automatically setting β according to the probability values of the network. Alternatively, techniques such as genetic optimization could be used to automate the parameter selection problem.

VII. CONCLUSION

Belief revision is the problem of finding the most plausible explanation for a given set of observances. In the context of Bayesian belief networks, belief revision becomes the problem of finding the network assignment $\mathcal{A}$ with maximum posterior probability $P(\mathcal{A}|\mathcal{E})$, where $\mathcal{E}$ is a partial network assignment corresponding to the observed evidence. Exact techniques for multiply-connected belief networks run in time exponential in the size of the network graph's minimum loop-cutset: the smallest set of vertices whose removal renders the network graph acyclic. For multiply-connected networks in which the loop-cutset is small, traditional methods can be used. However, for large heavily-connected networks, other methods are needed.

In this chapter, we began by describing High Order Recurrent Networks (HORNs) and reviewing a transformation which allows HORNs to be transformed to quadratic-order networks with equivalent energy functions. This was followed by the description of an efficient data structure for the software implementation of HORNs. We then showed how HORNs could be used for belief revision on belief networks.

Using the Pinkas transformation described in Section 4, the high order networks produced by our method can be converted to equivalent Hopfield networks and Boltzmann machines; this is of significance because of the potential for the hardware implementation of these networks [Schneider and Card, 1993; Schneider and Card, 1998].

ACKNOWLEDGMENT

I would like to thank my graduate student, Mr. Murad Assaggaf, for invaluable assistance and contributions.

REFERENCES

Abdelbar, A.M. A linear constraint satisfaction approach to Bayesian networks, *Proceedings IEEE International Joint Conference on Neural Networks,* 504, 1999.

Abdelbar, A.M. An algorithm for finding MAPs for belief networks through cost-based abduction, *Artificial Intelligence,* 104, 331, 1998.

Abdelbar, A.M. and Attia, S. Finding most probable explanations using simulated annealing as a genetic operator, *Proceedings World Multiconference on Systemics, Cybernetics and Informatics,* 8, 131, 1999.

Abdelbar, A.M. and Hedetniemi, S.M. Approximating MAPs for belief networks is NP-hard and other theorems, *Artificial Intelligence,* 102, 21, 1998.

Abdelbar, A.M. and Hedetniemi, S.M. A parallel hybrid genetic algorithm simulated annealing approach to finding probable explanations on Bayesian belief networks, *Proceedings IEEE International Conference on Neural Networks,* **I**, 450, 1997.

Aggarwal, A. Coppersmith, D. Khanna, S. Motwani, R. and Schieber, B. The angular-metric traveling salesman problem, *Proceedings Eighth Annual ACM-SIAM Symposium on Discrete Algorithms,* 221, 1997.

Charniak, E. and Shimony, S. Cost-based abduction and MAP explanation, *Artificial Intelligence,* 66, 345, 1994.

Cohen, M.A. and Grossberg, S. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks, *IEEE Transactions on Systems, Man, and Cybernetics,* 13, 815, 1983.

Dempster, A.P. Upper and lower probabilities induced by a multivalued mapping, *Annals of Mathematical Statistics,* 38, 157, 1967.

Geman, S. and Geman, D. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Recognition and Machine Intelligence,* 6, 721, 1984.

Ginsberg, M.L. *Readings in Nonmonotonic Reasoning,* Morgan Kauffman, San Mateo, 1987.

Hinton, G.E. and Sejnowski, T.J. Learning and re-learning in Boltzmann machines, in: J.L. McClelland, D.E. Rumelhart, and the PDP Research Group, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* **I**, MIT Press, Cambridge, 282, 1986.

Hobbs, J.R. Stickel, M.E. Appelt, D.E. and Martin, P. Interpretation as abduction, *Artificial Intelligence,* 63, 69, 1993.

Hopfield, J.J. Neurons with graded response have collective computational properties like those of two-state neurons, *Proceedings National Academy of Science,* 81,3088,1984.

Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities, *Proceedings National Academy of science,* 79, 2554, 1982.

Kirkpatrick, S. Gelatt, D. and Vecchi, M. Optimization by simulated annealing, *Science,* 220, 621, 1983.

Kofler, E.T. and Leondes, C.T. Algorithmic modifications to the theory of evidential reasoning, *Journal of Algorithms,* 17, 269, 1994.

Konyndyk, K. *Introductory Modal Logic,* Notre Dame Press, Notre Dame, 1986.

Lin, W-C. Liao, F-Y. Tsao, C-K. and Lingutla, T. A hierarchical multiple-view approach to three-dimensional object recognition, *IEEE Transactions on Neural Networks,* 2, 84, 1991.

Lindley, D.V. The probability approach to the treatment of uncertainty, *Statistical Science,* 2, 17, 1987.

Marek, V.W. and Truszczynski, M. *Nonmonotonic Logic,* Springer-Verlag, Berlin, 1993.

Page, E.W. and Tagliarini, G.A. Algorithm development for neural networks, *Proceedings SPIE Symposium Innovative Science and Technology,* 880, 11, 1988.

Pawlak, Z. Grzymala-Busse, J. Slowinski, R. and Ziark, W. Rough sets, *Communications of the ACM,* 38, 88, 1995.

Pawlak, Z. Rough sets: A new approach to vagueness, in: L. Zadeh, and J. Kacprzyk, eds., *Fuzzy Logic for the Management of Uncertainty,* John Wiley & Sons, New York, 1992.

Pawlak, *Z. Rough Sets,* Kluwer Academic, Dordrecht, 1991.

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

Pearl, J. Fusion, propagation, and structuring in belief networks, *Artificial Intelligence,* 29, 241, 1986.

Peterson, C. and Hartman, E. Explorations of the mean field theory learning algorithm, *Neural Networks,* 2, 475, 1989.

Peterson, C. and Anderson, J.R. A mean field theory learning algorithm for neural networks, *Complex Systems,* 1, 995, 1987.

Pinkas, G. Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge, *Artificial Intelligence,* 77, 203, 1995.

Popkorn, S. *First Steps in Modal Logic,* Cambridge University Press, Cambridge, 1994.

Reiter, R. Nonmonotonic reasoning, *Annual Review of Computer Science,* 147, 1987.

Rojas-Guzman, C. and Kramer, M.A. "GALGO: A Genetic Algorithm decision support tool for complex uncertain systems modeled with Bayesian belief networks," *Proceedings Ninth Annual Conference on Uncertainty in AI,* 368, 1993.

Rojas-Guzman, C. and Kramer, M.A. Remote diagnosis and monitoring of complex industrial systems using a genetic algorithms approach, *Proceedings IEEE International Symposium on Industrial Electronics,* 363, 1994.

Santos Jr., E. A linear constraint satisfaction approach to cost-based abduction, *Artificial Intelligence,* 65, 1, 1994.

Santos Jr., E. and Santos, E.S. Polynomial solvability of cost-based abduction, *Artificial Intelligence,* 86, 157, 1996.

Schneider, C.R. and Card, H.C., Analog hardware implementation issues in deterministic Boltzmann machines, *IEEE Transactions on Circuits and Systems II,* 45, 352, 1998.

Schneider, C.R. and Card, H.C. Analog CMOS deterministic Boltzmann circuits, *IEEE Journal Solid-State Circuits,* 28, 907, 1993.

Shachter, R.D. Evaluating inference diagrams, *Operations Research,* 34, 871, 1986.

Shafer, G. The combination of evidence, *International Journal of Intelligent Systems,* 1, 155, 1986.

Shafer, G. A *Mathematical Theory of Evidence,* Princeton University Press, *Princeton,* 1976.

Shafer, G. and Logan, R. Implementing Dempster's rule for heirarchial evidence, *Artificial Intelligence,* 33, 271, 1987.

Shimony, E. Finding MAPs for belief networks is NP-hard, *Artificial Intelligence,* 68, 399,1994.

Shoham, Y. Nonmonotonic logics: Meaning and utility, *Proceedings International Joint Conference on Artificial Intelligence,* 388, 1987.

Tagliarini, G. and Page, E.W. A neural-network solution to the concentrator assignment problem, *Proceedings IEEE Conference on Neural Information Processing Systems—Natural and Synthetic,* 775, 1987.

Tagliarini, G. Fury-Christ, J. and Page, E.W. Optimization using neural networks, *IEEE Transactions on Computers,* 40, 1347, 1991.

Takefuji, Y, and Lee, K-C. A near optimum parallel planarization algorithm, *Science,* 245, 1221, 1989.

Yager, R. Fedrizzi, M. and Kacprzyk, J. *Advances in the Dempster-Shafer Theory of Evidence,* John Wiley & Sons, New York, 1994.

Zadeh, L. Fuzzy logic, neural networks and soft computing, *Communications of the ACM,* 37, 77, 1994.

Zadeh, L. and Kacprzyk, J. *Fuzzy Logic for the Management of Uncertainty,* John Wiley & Sons, New York, 1992.

Zadeh, L. A theory of approximate reasoning, in: J. Hayes, D. Michie, and L.I. Mikulich, eds., *Machine Intelligence* **9**, Halstead Press, New York, 149, 1979.

Zimmerman, H-J. *Fuzzy Set Theory and Its Applications,* Kluwer Academic, Boston, 1991.