IMPLEMENTING BOLTZMANN MACHINES

by

Eugene Wong

Memorandum No. UCB/ERL M90/1

5 January 1990

IMPLEMENTING BOLTZMANN MACHINES

by

Eugene Wong

Memorandum No. UCB/ERL M90/1

5 January 1990

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

IMPLEMENTING BOLTZMANN MACHINES

by

Eugene Wong

Memorandum No. UCB/ERL M90/1

5 January 1990

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Implementing Boltzmann Machines

Eugene Wong

University of California at Berkeley

1. Introduction

Consider a discrete-time Hopfield net [HOP82] where the state $X_i(t)$ at node i takes values ± 1 and satisfies a transition equation of the form

$$X_i(t+1) = \operatorname{sgn}\left[\sum_{j=1}^{n} w_{ij} X_j(t) + \theta_i\right] \tag{1.1a}$$

or

$$= X_i(t) \tag{1.1b}$$

We assume that at each t only one node can change state (i.e., satisfy (1.1a)) but that every node satisfies it infinitely often as $t \to \infty$.

Now define an energy function

$$V(x) = -\left[\frac{1}{2}\sum_{i,j} w_{ij} x_i x_j + \sum_i \theta_i x_i\right] \qquad x \varepsilon \{-1, 1\} \tag{1.2}$$

and assume that the weights w_{ij} satisfy the conditions: $w_{ii} = 0$, $w_{ij} = w_{ji}$ for all i and j. Under these assumptions, it is easy to show that at every t we have

$$V(X(t+1)) - V(X(t)) = [1 - \operatorname{sgn}(\Delta_i(t))]\, \Delta_i(t) \tag{1.3}$$

for some i, where

$$\Delta_i(t) = X_i(t)\left[\sum_j w_{ij} X_j(t) + \theta_i\right] \tag{1.4}$$

Equation (1.3) indicates that $V(X(t))$ never increases, and decreases for at most a finite number of values of t. Thus, there exists a $t_0 < \infty$ such that

$$V(X(t)) - V(X(t_0)) = 0 \qquad \text{for all } t \geq t_0$$

which implies that for every i and every $t \geq t_0$,

$$\Delta_i(t) \geq 0 \quad \text{and} \quad X_i(t) = X_i(t_0)$$

Since every component of X is tested infinitely often, $V(X(t_0))$ cannot be decreased by a change in any single component of $X(t_0)$. Thus, $X(t_0)$ is a *local minimum* of $V(x)$.

As a tool for optimization, Hopfield nets suffer from two flaws: local minimum and restricted $V(x)$, while enjoying an implementation advantage. These nets are no mere algorithms, but models that are readily implementable as hardware. Our goal in this paper is to remove the defects while preserving the implementation advantage.

Global minimization can be achieved through simulated annealing [KIR83] which is based on the following idea. Suppose that instead of satisfying (1.1), $X(t)$ is a stochastic process with a distribution of the form

$$\text{Prob}(X(t) = x) = \frac{1}{Z} e^{-\frac{1}{T}V(x)}, \qquad x \varepsilon \{-1, 1\}^n \tag{1.5}$$

which is known as the *Gibbs distribution*. Further, suppose that the parameter T (interpreted as *temperature*) is reduced slowly so that the distribution of $X(t)$ remains approximately Gibbs, i.e.,

$$\text{Prob}(X(t) = x) \approx \frac{1}{Z(t)} e^{-\frac{1}{T(t)}V(x)}$$

for all t. Then, denoting the global minimum by V_m, we have

$$\text{Prob}(X(t) = x) \approx \frac{e^{-\frac{1}{T(t)}[V(x) - V_m]}}{\sum\limits_x e^{-\frac{1}{T(t)}[V(x) - V_m]}}$$

$$\rightarrow \begin{cases} 0, & V(x) \neq V_m \\ \frac{1}{K}, & V(x) = V_m \end{cases} \qquad \text{as } T(t) \rightarrow 0$$

where K is the number of values of x that attain the global minimum.

It is well known that a process with an equilibrium Gibbs distribution can be generated as a stationary Markov chain. In [ACK85] a neural network with states forming such a process is called a *Boltzmann machine*, and in [ALS87] it is proposed to approximate a Boltzmann machine by injecting noise in a Hopfield net. Our first objective in this paper is to show that an exact, not approximate, construction of Boltzmann machine can be achieved by injecting noise in a Hopfield net, but that this has to be done in a precise way that we shall describe in the sequel.

Our second objective is to generalize $V(x)$ to an arbitrary function. We show that even in the general case $X(t)$ can still be realized by a network (though no longer a Hopfield net). Finally, we discuss alternative architectures that are suitable for implementing the resulting network.

2. Boltzmann Machine

Boltzmann under machine under is a binary-state and discrete-time neural network whose states form a stationary Markov chain $X(t)$ with an equilibrium Gibbs distribution. Such a Markov chain can be constructed by controlling its transitions as follows:

(a) At time t, select one component, say $X_i(t)$, for possible change.

(b) Compute the energy change that would result from a state change $X_i(t) \to -X_i(t)$, viz.,

$$\Delta_i(t) = 2X_i(t)\,[\sum_j w_{ij}\,X_j(t) + \theta_i] \tag{2.1}$$

(c) Set

$$\begin{aligned} X_i(t+1) &= -X_i(t) && \text{with probability } \pi(\Delta_i(t)) \\ &= X_i(t) && \text{with probability } 1 - \pi(\Delta_i(t)) \end{aligned} \tag{2.2}$$

The quantity π is known as the *acceptance probability* (accepting a change). In [KES89] it is shown that an acceptance probability of the form

$$\pi(\Delta) = e^{-\frac{\Delta}{2T}} f(|\Delta|) \tag{2.3}$$

suffices to ensure that $X(t)$ will have a steady state distribution given by a Gibbs distribution (1.3). It is also shown in [KES89] that

$$\pi(\Delta) = \min(1, e^{-\frac{\Delta}{T}}) \tag{2.4}$$

is optimal in the sense of maximizing the speed of reaching equilibrium.

We note that for $T = 0$, (2.4) reduces to the following:

$$\begin{aligned} \pi(\Delta) &= 1 \quad \Delta < 0 \\ &= 0 \quad \Delta > 0 \end{aligned} \tag{2.5}$$

and (2.2) can then be written as

$$\begin{aligned} X_i(t+1) &= X_i(t)\,\mathrm{sgn}[\Delta_i(t)] \\ &= X_i(t)\,\mathrm{sgn}\{X_i(t)\,[\sum_j w_{ij}\,X_j(t) + \theta_i]\} \\ &= \mathrm{sgn}[\sum_j w_{ij}\,X_j(t) + \theta_i] \end{aligned} \tag{2.6}$$

which is just (1.1a). Thus, for $T = 0$, a Boltzmann machine is indeed a Hopfield net.

For $T > 0$, we hypothesize that the state transition equation (2.2) for a Boltzmann machine is re-expressible as

$$X_i(t+1) = X_i(t)\,\mathrm{sgn}[\Delta_i(t) - Z] \tag{2.7}$$

where Z is a random variable with a probability density function $p_Z(z)$. From (2.7), we have

$$\begin{aligned} \pi(\Delta_i(t)) &= \mathrm{prob}(X_i(t+1) = -X_i(t)) \\ &= \mathrm{prob}(\Delta_i(t) < Z) \\ &= \int_{\Delta_i(t)}^{\infty} p_Z(z)\,dz \end{aligned} \tag{2.8}$$

Comparing (2.8) with (2.2) shows that if we set

$$\int_{\Delta}^{\infty} p_Z(z)\,dz = \pi(\Delta) \tag{2.9}$$

then (2.7) is indeed equivalent to (2.2) and our hypothesis is confirmed. We can now differentiate (2.9) to get

$$p_Z(z) = -\frac{d}{dz}\pi(z) \tag{2.10}$$

which prescribes the distribution of Z in terms of the acceptance probability π.

We have thus shown that a Boltzmann machine can be implemented by injecting noise $Z(t)$ in a Hopfield net so that the transition equation is given by

$$X_i(t+1) = \operatorname{sgn}[\sum_j w_{ij} X_j(t) + \theta_i - X_i(t)Z(t)] \tag{2.11}$$

where $Z(t)$ is independent for different t's (i.e., a white noise) but not Gaussian. Instead, it distribution is governed by the choice of the acceptance probability through (2.10). Both (2.10) and (2.11) represent new results.

Fro the example given by (2.4), we have

$$p_Z(z) = e^{-\frac{z}{T}} 1(z) \tag{2.12}$$

where $1(z)$ is the unit-step function. A random variable with this distribution can be generated as

$$Z = Z_1^2 + Z_2^2$$

where Z_1 and Z_2 are independent $N(0, T/2)$ random variables. We note, however, no allowable choice of the acceptance probability would yield a Z that is Gaussian.

Thus, while a Boltzmann machine can be implemented bu injecting noise in a Hopfield net, the noise must be multiplied by the state before added to the input as prescribed by (2.11), and the noise cannot be Gaussian.

3. Generalizing the Energy Function

Any function $V(x)$ on the hypercube $\{-1, 1\}^n$ admits a multilinear expansion of the form:

$$V(x) = -\sum_{\underline{k}} w_{\underline{k}} x_{k_1} x_{j_2} \cdots x_{k_m} + V_0 \tag{3.1}$$

where $\underline{k} = (k_1, j_2, \cdots, j_m)$ denotes an ordered subset of $\{1, 2, \cdots, n\}$ and the summation is taken over all $2^n - 1$ such ordered subsets. Henceforth, we shall write $x_{\underline{k}}$ to denote the product

$x_{k_1} x_{j_2} \cdots x_{j_m}$. For a $V(x)$ given by (3.1), we can now generalize (2.1) to read

$$\Delta_i(t) = 2 \sum_{\underline{k} \varepsilon i} w_{\underline{k}} X_{\underline{j}}(t) \tag{3.2}$$

With Δ thus generalized, a stationary Markov chain with an equilibrium Gibbs distribution for a general $V(x)$ can again be constructed using (2.7) and (2.11). However, $X_i(t)\,\Delta_i(t)$ is no longer linear in $X(t)$. Rather, we have

$$\begin{aligned} X_i(t+1) &= \operatorname{sgn}[X_i(t)\,\Delta_i(t) - X_i(t) Z(t)] \\ &= \operatorname{sgn}[\sum_{\underline{k} \varepsilon i} w_{\underline{k}} X_{k_1}(t) \cdots X_i(t) \cdots X_{j_m}(t) - X_i(t)\, Z(t)] \end{aligned} \tag{3.3}$$

Even at zero temperature (hence zero noise), (3.3) is not realizable as a Hopfield net. To realize (3.3) as a network, we shall develop several alternative architectures based on the first of the equations in (3.3).

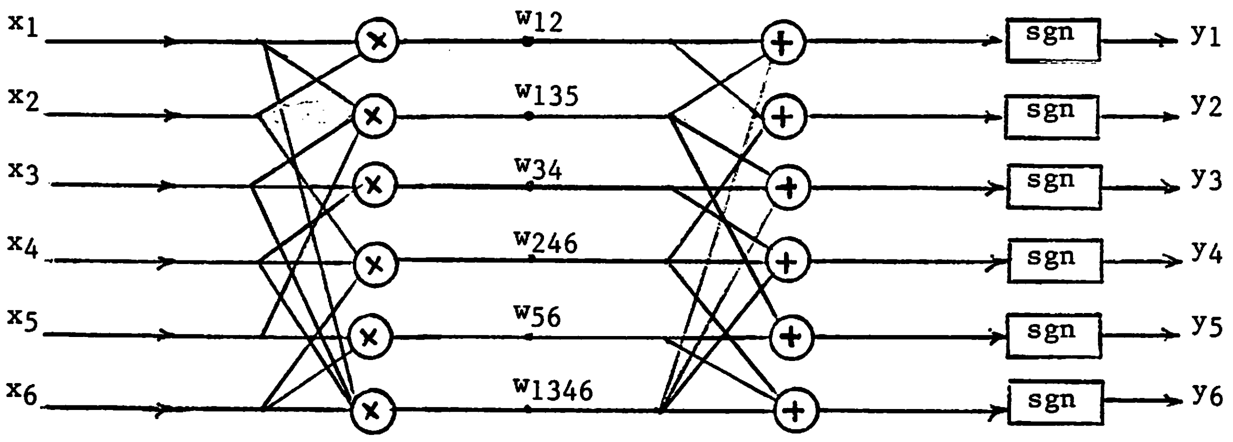
4. Alternative Network Architectures

We begin by rewriting the state transition equation as follows:

$$X_i(t+1) = Y_i(t)\, X_i(t) \tag{4.1}$$

$$\begin{aligned} Y_i(t) &= \operatorname{sgn}[\Delta_i(t) - Z(t)] \\ &= \operatorname{sgn}[\sum_{\underline{k} \varepsilon i} w_{\underline{k}} X_{\underline{k}}(t) - Z(t)] \end{aligned} \tag{4.2}$$

An obvious network structure to implement (4.2) is to make use of multipliers to compute $w_{\underline{k}} X_{\underline{k}}$ and adders to yield Y_i. An example is given in Fig. 1.



$$V(x) = w_{12}x_1x_2 + w_{34}x_3x_4 + w_{56}x_5x_6 + w_{135}x_1x_3x_5 + w_{246}x_2x_4x_6 + w_{1346}x_1x_3x_4x_6$$

Figure 1. Analog Network Realization

It is apparent that only the adders in the last stage represent analog operations. The rest is entirely digital. This means that a considerable reduction in interconnect-complexity can be realized by using random-access memory and indexing. In such an arrangement, the physical connections are replaced by logical connections. A hybrid architecture comprising both analog and digital circuits is shown in Figure 2.

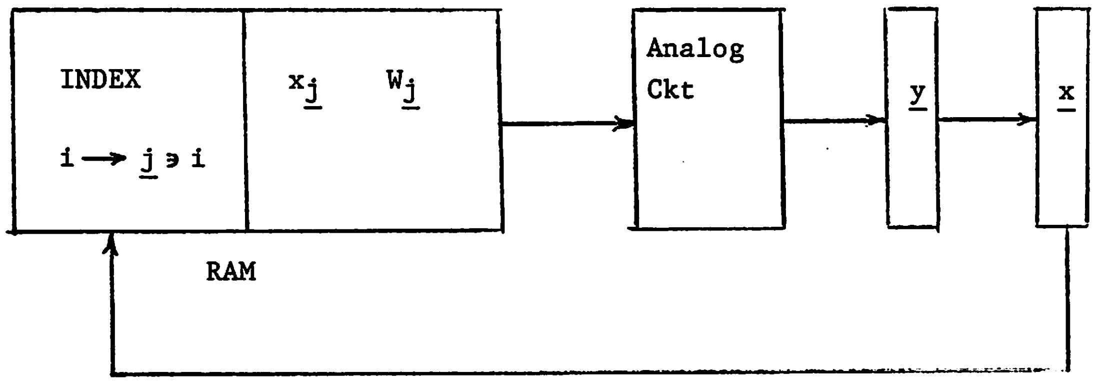


Figure 2. A Hybrid Network

In this arrangement, the network operates as follows:

(a) A component i is chosen at random.

(b) The INDEX is accessed to determine for each $\underline{k} \varepsilon i$ the address where $x_{\underline{k}} w_{\underline{k}}$ is stored.

(c) Each product $x_{\underline{k}} w_{\underline{k}}$ is retrieved and converted to an analog value.

(d) The sum $\sum_{\underline{k} \varepsilon i} X_{\underline{k}} w_{\underline{k}} - Z$ is computed.

(e) Y_i is computed as the "sign" of the sum computed in (d).

(f) If $Y_i = -1$, change the sign of X_i and go to (g), otherwise go to (a).

(g) For each $\underline{k} \varepsilon i$, change the sign of $X_{\underline{k}} w_{\underline{k}}$.

We note that the need to access the set $\{X_{\underline{k}} w_{\underline{k}}\}$ makes the operation slow. However, there is ample opportunity for *pipelining*, which means that the cycles can be overlapped. In this respect, the situation is no different from digital signal processing in general where pipelining is essential for achieving acceptable speeds.

Finally, we note that only analog-add and not analog-multiply is required. If the degree of accuracy needed is not too high, it may well be better to replace the analog adders by digital ones, thereby obviating the need for D/A conversion and affording a further opportunity to eliminate physical interconnections. We would then have an all-digital system that can be implemented in various ways, including ones that use only off-the-shelf components. In this form, the distinction between hardware and software blurs, and the use of a standard DSP (digital signal processing) system for implementation provides an attractive alternative.

Acknowledgement

The research reported herein was supported by U.S. Army Research Office Grant DAAL03-89-K-0128 and U.S. Air Force Office of Scientific Research Contract F49620-87-C-0041.

References

[ACK85] D. H. Ackley, G. W. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," Cognitive Science 9 (1985) 147-169.

[ALS87] J. Alspector and R. B. Allen, "A neuromorphic VLSI learning system," in Advanced Research in VLSI, Proc. 1987 Stanford Conference, P. Losleben ed., MIT Press, Cambridge, 1987, 313-349.

[HOP82] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," Proc. National Academy of Sciences 79 (1982) 2554-2558.

[KES89] G. Kesidis and E. Wong, "Optimal acceptance probability for simulated annealing," to be published in Stochastics and Stochastic Reports.

[KIR83] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," Science 220 (1983) 671-680.