Journal of Statistical Physics, Vol. 62, Nos. 3/4, 1991

Finite-State Neural Networks. A Step Toward the Simulation of Very Large Systems

G. A. Kohring

Received May 10, 1990; final July 24, 1990

Neural networks composed of neurons with $Q_{\mathcal{N}}$ states and synapses with $Q_{\mathcal{J}}$ states are studied analytically and numerically. Analytically it is shown that these finite-state networks are much more efficient at information storage than networks with continuous synapses. In order to take the utmost advantage of networks with finite-state elements, a multineuron and multisynapse coding scheme is introduced which allows the simulation of networks having 1.0×10^9 couplings at a speed of 7.1×10^9 coupling evaluations per second on a *single* processor of the Cray-YMP. A local learning algorithm is also introduced which allows for the efficient training of large networks with finite-state elements.

KEY WORDS: Neural networks; multi-spin coding; replica method; finite-state networks; learning algorithms.

1. INTRODUCTION

Through the work of many different research groups, it has been established that neural networks can in theory perform elementary cognitive operations such as associative memory[1] and generalization.[2] These results are very encouraging for biological studies; however, to study ever more realistic and complex networks, computer simulations of very large systems will be required. Furthermore, the adaptation of biological computational methods for artificial neurocomputing may require the construction of some novel types of computing devices[3]; but the feasibility of such neurocomputational approaches must first be demonstrated via simulations, and perhaps even modest implementations, on conventional com-

[1] Höchstleistungsrechenzentrum an der KFA Jülich, D-5170 Jülich, Germany.

0022-4715/91/0200-0563$06.50/0

puters. For both of these reasons, one needs the ability to handle very large neural networks as fast and efficiently as possible.

For the sake of concrete discussions, we consider here the class of neural network models known as "attractor neural networks" (ANN),[4] the prototypical network of this class being the Little-Hopfield model.[5] Attractor neural networks are extremely useful for analytical and numerical discussions because of the relative simplicity with which their performance can be quantified.

When simulating very large systems, the elementary variables must be chosen with care so that only the essentials are included. Now, many researchers consider the firing rate of cortical neurons to be a continuous function of the membrane voltage (see, e.g., ref. 6). However, given the noisy character of the individual neurons, it is implausible that an arbitrarily small change in the neuron firing rate is meaningful. Rather, if two different firing rates are to be significant, then the difference between the two firing rates must be greater than that resulting from random noise. Since the neuron firing rate is bounded above,[7] this naturally leads to the concept of neurons with only a finite number of firing states. A similar argument holds when considering the plasticity of the synaptic connections. Hence, networks of finite-state elements may be more relevant for biological studies than the continuously varying elements which are often used.

To quantify one aspect of the performance of ANN models composed of neurons with $Q_{\mathcal{N}}$ states and synapses with $Q_{\mathcal{J}}$ states, consider the efficiency with which information is stored. For a network with N neurons, the storage of P random patterns is accomplished with an efficiency given by

$$\begin{aligned} \mathcal{E}(Q_{\mathcal{N}}, Q_{\mathcal{J}}) &= \frac{\text{total number of bits to be stored}}{\text{total number of bits used for storage}} \\ &= \frac{PN \ln Q_{\mathcal{N}}}{N^2 \ln Q_{\mathcal{J}}} \\ &\xrightarrow{N \to \infty} \alpha(Q_{\mathcal{N}}, Q_{\mathcal{J}}) \frac{\ln Q_{\mathcal{N}}}{\ln Q_{\mathcal{J}}} \end{aligned} \tag{1.1}$$

where $\alpha(Q_{\mathcal{N}}, Q_{\mathcal{J}}) = \lim_{N \to \infty} P/N$. One of the main results of the analytic investigations in Section 2 is that this efficiency of information storage is severly degraded whenever $Q_{\mathcal{J}} \gg Q_{\mathcal{N}}$. Thus, the most commonly studied ANN model, $Q_{\mathcal{N}} = 2$ with continuous synapses, is also the most inefficient in terms of information storage.

Given the increased storage efficiency for finite-state networks, a simulation algorithm to maximize these benefits is needed. In Section 3, a

single-bit handling algorithm known as multineuron coding[8,9] is introduced, and shown to achieve the above theoretical storage limits. As an additional bonus, this storage algorithm allows the construction of updating algorithms that are up to 35 times faster than the conventional, one-word, one-neuron algorithms. Indeed, the algorithm discussed in Section 3 has been run on a Cray-YMP using a network with 1.0×10^9 couplings and it achieved an updating speed of 7.1×10^9 couplings per second (7.1 Gcops), compared with a speed of only 0.2 Gcops using conventional algorithms.

Now, speed and storage efficiency are fine, but finite-state networks must still be able to function as associative memory devices. It is well known from studying the case of $Q_{\mathcal{N}} = 2$ with continuous synapses that the basins of attraction become arbitrarily small as $\alpha \to 2$.[10] In fact, already at $\alpha \approx 0.5$, any given pattern cannot, on average, be recalled, if the input pattern contains more than 10% errors. If one takes this 10% error criterion as defining an effective α_e, then one needs to ask how large is α_e for finite-state networks? Since there is no known method for calculating the basins of attraction analytically, they must be determined numerically. This, of course, requires the need of a learning rule to train the network. A local learning rule well suited for this purpose is discussed in Section 4. There it is also discussed how a network with $Q_{\mathcal{N}} = 2$ and $Q_{\mathcal{J}} = 3$, reaches the 10% error criterion at $\alpha_c \approx 0.31$. In terms of $\mathcal{C}$, this defines an effective storage efficiency which is far superior to that obtainable with continuous couplings.

It is obvious from the above discussion that this paper is arranged as follows: Section 2 examines analytically the efficiency with which finite-state networks store information; Section 3 then looks at an algorithm for the fast updating of such networks; Section 4 takes up the question of learning for finite-state ANN models; and the final section concludes with a comparison of some of the different networks discussed in the paper.

2. NEURONS AND SYNAPSES WITH A FINITE NUMBER OF STATES

Consider a single-layer neural network of N formal neurons $\{S_i(t)\}$ $(i = 1,..., N)$, each of which can take on $Q_{\mathcal{N}}$ distinct states $\{\eta_i\}$ $(i = 1,..., Q_{\mathcal{N}})$. (Biologically, as noted above, this represents $Q_{\mathcal{N}}$ distinct neuron firing rates.) These neurons are to be connected by N^2 synaptic couplings J_{ij}, which are allowed only $Q_{\mathcal{J}}$ discrete states. (For simplicity consider only two-neuron interactions. The extension to the more general case of multineuron interactions[11] is straightforward.) The network is then

allowed to evolve under zero-temperature (noise-free) dynamics, which can be either parallel or asynchronous:

$$S_i(t+1)=\begin{cases}\eta_1 & \text{if } \theta^-(\eta_1)\leqslant h_i(S[t])\leqslant\theta^+(\eta_1)\\ \vdots & \\ \eta_{Q_\mathcal{N}} & \text{if } \theta^-(\eta_{Q_\mathcal{N}})\leqslant h_i(S[t])\leqslant\theta^+(\eta_{Q_\mathcal{N}})\end{cases} \tag{2.1}$$

where $h_i(S[t])=\sum_{j\neq i}J_{ij}S_j(t)$, and the thresholds $\{\theta^{\pm}(\eta)\}$ are *a priori* defined constants.

If one wants to store P patterns in this network, i.e., if one wants to make P states $\{\xi_i^\mu\}$ ($\mu=1,\ldots,P$) fixed points of the above dynamics, then for random patterns (correlated patterns can be treated analogously), the efficiency with which this can be done is given by Eq. (1.1). For continuous couplings and two-state neurons, $\alpha=2$,[12] and $\mathscr{E}=0$. From an information storage point of view this is extremely inefficient. As noted in the introduction, this efficiency can be increased by considering finite-state networks. (In reality one does not have an infinite number of synaptic states available, since this would require an infinite number of molecules; hence, strictly speaking, it is rater meaningless in this contex to speak of "continuous synapses.") Determining $\mathscr{E}(Q_\mathcal{N}, Q_\mathcal{J})$ for finite-state networks requires a determination of $\alpha(Q_\mathcal{N}, Q_\mathcal{J})$.

As in the case of continuous couplings,[13] determining $\alpha(Q_\mathcal{N}, Q_\mathcal{J})$ can be done by calculating the total volume of the synaptic space V_T which solves the fixed-point constraints and then looking for the largest value of P such that this volume is nonzero. V_T is given by

$$V_T=\frac{1}{Z}\sum_J\prod_{i,\mu}B(\theta^-(\xi_i^\mu), h_i(\xi^\mu), \theta^+(\xi_i^\mu))\,\delta\left(\sum_{j\neq i}(J_{ij})^2-N\right) \tag{2.2}$$

where

$$h_i(\xi^\mu)=\frac{1}{(N\sigma)^{1/2}}\sum_{j\neq i}J_{ij}\xi_j^\mu \tag{2.3}$$

$\sigma=(1/Q_\mathcal{N})\sum_i\eta_i^2$, $B(a, x, b)$ is the barrier function defined by $B(a, x, b)\equiv\int_a^b dy\,\delta(y-x)$, and $Z\equiv\sum_J\delta(\sum_{j\neq i}(J_{ij})^2-N)$. The delta function simply defines the scale for the J_{ij}. (Any scale can be used so long as the results of calculations with different scales are renormalized before comparison.) In order to get $\alpha(Q_\mathcal{N}, Q_\mathcal{J})$, one must average V_T over all possible sets of patterns $\{\xi_i^\mu\}$. However, the patterns form a set of quenched, random variables and one cannot average quantities such as V_T over quenched variables; rather, one must average only extensive quantities.[21] The entropy per coupling $\mathscr{S}=-(1/N)\ln V_T$ is such an extensive quantity.

Averaging of $\mathscr{S}$ over the $\{\xi_i^\mu\}$ can be done in a well-defined manner as discussed in refs. 14 and 23 for the case $Q_{\mathscr{N}} = Q_{\mathscr{J}} = 2$. The extension to the more general case represented here is straightforward and results in the following expression for $\mathscr{S}$:

$$\mathscr{S} = -\alpha(Q_{\mathscr{N}}, Q_{\mathscr{J}}) \frac{1}{Q_{\mathscr{N}}} \sum_{i=1}^{Q_{\mathscr{N}}} \int_{-\infty}^{\infty} \mathscr{D}t \ln \left\{ \int_{(\theta^+(\eta_i) + r^{1/2}t)/(1-r)^{1/2}}^{(\theta^+(\eta_i) + r^{1/2}t)/(1-r)^{1/2}} \mathscr{D}y \right\} - \frac{\zeta r}{2} - \frac{\omega}{2}$$
$$- \int_{-\infty}^{\infty} \mathscr{D}v \ln \left\{ \sum_J \exp\left[-\frac{\omega + \zeta}{2} J^2 + \zeta^{1/2} v J \right] \right\} \tag{2.4}$$

where

$$\mathscr{D}x = \frac{dx}{(2\pi)^{1/2}} e^{-x^2/2} \tag{2.5}$$

The values of ζ, ω, and r are fixed by the saddle-point conditions $\partial\mathscr{S}/\partial\zeta = \partial\mathscr{S}/\partial\omega = \partial\mathscr{S}/\partial r = 0$. Here r can be physically interpreted as the average overlap between sets of J_{ij} which solve the fixed-point constraints of (2.3). ζ is the conjugate variable to r, and ω ensures that the scale of the couplings is correctly fixed. For the case $Q_{\mathscr{N}} = Q_{\mathscr{J}} = 2$, Krauth and Mézard conjectured that, since the entropy $\mathscr{S}$ cannot be negative, the maximum value of α is found when $\mathscr{S}$ vanishes.[14] Although this conjecture has not been rigorously established, it is physically plausible and has given reliable results for the case $Q_{\mathscr{N}} = Q_{\mathscr{J}} = 2$. Hence, this conjecture will be invoked here and shown to yield reasonable results. Unfortunately, the location of the maximum value of α cannot, in general, be found analytically, but can be found easily on a small computer.

Figure 1 shows a plot of $\mathscr{E}$ vs. $Q_{\mathscr{J}}$ for $Q_{\mathscr{N}} = 2$ as found from the above equations. The neuron states are chosen to be $\eta_1 = -1$, $\eta_2 = +1$, and the thresholds are defined as $\theta^-(-1) = -\theta^+(+1) = -\infty$ and $\theta^+(-1) = \theta^-(+1) = 0$. The synaptic states are chosen as $\{-(Q_{\mathscr{J}}-1)/2, \ldots, (Q_{\mathscr{J}}+1)/2\}$ when $Q_{\mathscr{J}}$ is odd and $\{-Q_{\mathscr{J}}/2, \ldots, Q_{\mathscr{J}}/2\}$ (omitting the zero state) when $Q_{\mathscr{J}}$ is even. This choice for the thresholds provides optimal performance. From Fig. 1 it can be seen that the most efficient information storage occurs for a small number of synaptic states; in fact, for two-state synapses, $\mathscr{E}$ is about 14 times larger than for continuous synapses on a 32-bit computer. Note that $\mathscr{E}$ falls off logarithmically with increasing $Q_{\mathscr{J}}$, indicating that α, a measure of the maximum number of patterns which can be stored, saturates rather quickly. It can also be seen in the figure that the efficiency of networks made from synapses with an odd number of states is slightly greater than what one would expect by interpolating between the points for networks made from synapses with an even number of states.

This is most probably due to the presence of a synaptic state with zero strength. Apparently, allowing some synapses to be zero can slightly improve the storage efficiency. Previous researchers have discussed setting an *a priori* defined fraction of the synapses to zero and they also find performance improvements.[22] The difference between the former approaches and the present one is one of quenching the synapses versus annealing the synapses.

Figure 1 also shows solutions for $Q_{\mathcal{N}}=4$. Here, the neuron states were chosen to be $\eta_1=-2$, $\eta_2=-1$, $\eta_3=+1$, $\eta_4=+2$, and the thresholds were defined as $\theta^-(-2)=-\theta^+(+2)=-\infty$, $\theta^+(-2)=\theta^-(-1)=-\theta^+(+1)=-\theta^-(+2)=-0.57$, and $\theta^+(-1)=\theta(+1)=0$. The synaptic states were kept the same. Again, these values of the thresholds were chosen so as to provide optimal performance. Once more it is seen that the efficiency of information storage decreases logarithmically from the maximum value of $\mathcal{E}\approx 0.97$ at $Q_{\mathcal{J}}=2$. And again, the synapses with an odd number of states perform slightly better than expected.

Lastly, Fig. 1 gives for comparison the solutions obtain when $Q_{\mathcal{N}}=3$. Here, the neuron states were chosen to be $\eta_1=-1$, $\eta_2=0$, $\eta_3=+1$, and the thresholds were defined as $\theta^-(-1)=-\theta^+(+1)=-\infty$ and $\theta^-(0)=-\theta^+(+1)=0.52$. The synaptic states were the same as for cases of even $Q_{\mathcal{J}}$. It can easily be seen that networks composed of neurons with three states are slightly less efficient than those composed of neurons with two states. This seems to be an effect due to the presence of a zero neuron state,

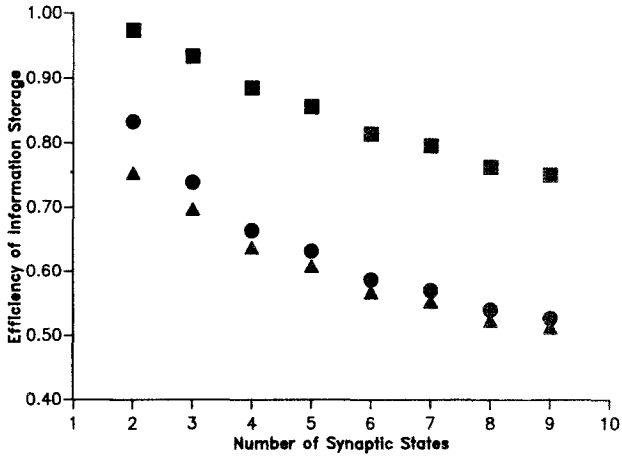

Fig. 1. Efficiency of information storage as a function of the number of synaptic states. The circles are for networks with two-state neurons, the squares for networks with four-state neurons, and the triangles for networks with three-state neurons.

since, as $Q_{\mathcal{N}}$ becomes large, the efficiencies of networks with an odd number of neuron states converges to that of networks with an even number of neuron states.

There are three special limits when the maximum value of α can be found exactly: (1) $Q_{\mathcal{S}} \to \infty$ with $Q_{\mathcal{N}}$ fixed, (2) $Q_{\mathcal{N}} \to \infty$ with $Q_{\mathcal{S}}$ fixed, and (3) $Q_{\mathcal{N}}, Q_{\mathcal{S}} \to \infty$ with $Q_{\mathcal{N}}/Q_{\mathcal{S}} \to$ const. The first case yields the simple relationship $\mathscr{E} \propto 1/\ln Q_{\mathcal{S}}$. For the second and third cases, $\mathscr{E}(Q_{\mathcal{N}}, Q_{\mathcal{S}})$ depends only upon θ $(\eta_{Q_{\mathcal{N}}})$ and the width of the threshold region, $\Delta\theta = |\theta^+(\eta_i) - \theta^-(\eta_i)|$ $(i = 2, \ldots, Q_{\mathcal{N}} - 1)$ [assuming $\theta^-(\eta_1) = -\theta^+(\eta_Q) = -\infty$]. When $\Delta\theta \propto Q_{\mathcal{N}}^{-a}$ and $\theta^-(\eta_{Q_{\mathcal{N}}}) \propto Q_{\mathcal{N}}^{1-a}$, then, in cases 2 and 3, $r \to 0$ and the following equation is easily obtained:

$$\mathscr{E}(Q_{\mathcal{N}}, Q_{\mathcal{S}}) \approx \frac{\ln Q_{\mathcal{N}}}{a \ln Q_{\mathcal{N}} + Q_{\mathcal{N}}^{2(1-a)} \times \text{const}} \tag{2.6}$$

Hence, for cases 2 and 3, if $a \geqslant 1$, then $\mathscr{E} \to 1/a$ and if $a < 1$, then $\mathscr{E} \to 0$. Thus, with $a = 1$, $\mathscr{E}$ asymptotically approaches maximum efficiency. It would be interesting if the exponent a could be related to the gain parameter in analog neural networks,[6] so that one could interpolate between the two approaches.

For simulation purposes, not only is the efficiency of information storage important, but the absolute number of bits needed is also of importance. On a computer with B bits per word, a fully connected network with N neurons and N^2 couplings requires $O(BN^2)$ bits when both the couplings and neurons are real variables. On the other hand, a network of two-state neurons and synapses requires only $O(N^2)$ bits. With a 32-bit machine, this amounts to using 32 times fewer bits. In order to realize this advantage in the absolute storage requirements, a fast algorithm is needed for the updating of networks in which single bits are used to describe single neurons and/or synapses. Such an algorithm is presented in the next section.

3. A FAST NUMERICAL SIMULATION ALGORITHM

For any neural network simulation, the quantity which must be repeatedly calculated is the local field $h_i(S[t])$,

$$\begin{aligned} h_i(S[t]) &= \sum_{j \neq i} J_{ij} S_j(t) \\ &= \sum_j J_{ij} S_j(t) - J_{ii} S_i(t) \\ &= H_i(t) - J_{ii} S_i(t) \end{aligned} \tag{3.1}$$

Updating the entire network once involves calculating $H_i(t)$ at every site,

i.e., $O(N^2)$ arithmetic operations. Therefore, it is fitting to concentrate on calculating $H_i(t)$ as fast and efficiently as possible. This can be done by generalizing the multineuron coding algorithm developed for the Hopfield model.[8,9]

Consider the case $Q_{\mathcal{N}} = Q_{\mathcal{J}} = 2$. In this case, only one bit is needed to describe a single neuron or synapse. If the two neuron states are $\eta_1 = -1$ and $\eta_2 = +1$ and the synapses are likewise, then the following correspondence can be made between the computer bits and the neuron states: $0 \Leftrightarrow -1$ and $1 \Leftrightarrow 1$. Since conventional computers operate on words with B bits in one word, it is possible to store B neurons or synapses in one computer word. Denote such a word by $\sigma(w)$ for the neurons and $\lambda(i, w)$ for the synapses; then Eq. (3.1) can easily be calculated by using the following expression to calculate $H_i(t)$:

$$H_i(t) = N - 2 \sum_w \text{POPCNT}(\lambda(i, w) \otimes \sigma(w)) \tag{3.2}$$

where $\otimes$ symbolizes the XOR (excluded or) function, and POPCNT counts the number of bits set equal to one in its argument. Since the summation unavoidably includes the diagonal term J_{ii}, this term should be subtracted as indicated in Eq. (3.1) if it is not desired. Equation (3.2) requires only $O(N^2/B)$ arithmetic calculations, because B coupling evaluations are made each time the XOR and POPCNT commands are executed.

This procedure has been implemented on the Cray-YMP/832 using Cray-standard Fortran. A fully connected network with $N = 32{,}000$ neurons (1.024×10^9 couplings) ran at a speed of 7.1×10^9 coupling evaluations per second (7.1 Gcops) on a *single* processor. Using a standard algorithm and the same amount of memory, one can simulate a network with only $N \approx 4000$ neurons (1.6×10^7 couplings). Furthermore, a conventional one-word, one-neuron algorithm runs at 0.2 Gcops on a *single* processor, or a about 35 times slower than the present algorithm. (On all eight processors of the Cray-YMP/832, the present algorithm would run at about 50 Gcops and the conventional algorithm would have a speed of about 1.5 Gcops.)

Starting from this basic algorithm, the extension to cases of larger $Q_{\mathcal{N}}$ or $Q_{\mathcal{J}}$ is straightforward. For example, at $Q_{\mathcal{N}} = 4$ and $Q_{\mathcal{J}} = 2$ (or, by symmetry, $Q_{\mathcal{N}} = 2$, $Q_{\mathcal{J}} = 4$) one can store the four states of $Q_{\mathcal{N}}$ in two bits on two different words, e.g., $\sigma(w)$ and $\varrho(w)$. If $\eta_1 = -2$, $\eta_2 = -1$, $\eta_3 = +1$, $\eta_4 = +2$, then one bit in $\sigma(w)$ is used to indicate the absolute value and one bit in $\varrho(w)$ to indicate the sign. $H_i(t)$ is then calculated as

$$H_i(t) = N + \sum_w \text{POPCNT}(\varrho(w)) - 4 \sum_w \text{POPCNT}(\varrho(w) \wedge (\lambda(i, w) \otimes \sigma(w)))$$
$$- 2 \sum_w \text{POPCNT}(\neg \varrho(w) \wedge (\lambda(i, w) \otimes \sigma(w))) \tag{3.3}$$

where $\wedge$ symbolizes the logical AND function and $\neg$ symbolizes the logical NOT function. Such a program requires N^2+2N bits ($2N^2+N$ bits for $Q_{\mathcal{N}}=2$, $Q_{\mathcal{J}}=4$), and it runs at approximately 2.5 Gcops. Although the speed is considerably reduced from that obtainable with $Q_{\mathcal{N}}=Q_{\mathcal{J}}=2$, it still runs ten times faster than if real words had been used.

Such constructions can be continued for larger values of $Q_{\mathcal{N}}$ and $Q_{\mathcal{J}}$, although at the cost of diminishing returns. The speed of the program falls off rather rapidly, leaving low memory usage as the only advantage to using the above algorithm. But with the high storage efficiencies achieved for small values of $Q_{\mathcal{N}}$ and $Q_{\mathcal{J}}$, there may be little reason to consider using larger values.

4. LEARNING IN FINITE-STATE NETWORKS

An obvious approach to learning in systems composed of finite-state neurons and synapses is to use a truncated version of Hebbian couplings. With $Q_{\mathcal{N}}=Q_{\mathcal{J}}=2$, one obtains the so-called "clipped-synapses" model, which has been studied in detail and yields results similar to the Hopfield model.[15] All such one-step learning rules are known to give far from optimal performance[16] and one would like to construct networks which perform near the optimal performance discussed in Section 2. Forrest used an exact enumeration scheme to find the optimal couplings in his studies of a $Q_{\mathcal{N}}=Q_{\mathcal{J}}=2$ network on a nearest-neighbor, two-dimensional square lattice.[17] But when the number of couplings per neuron grows like N, such a learning procedure becomes impractical.

One method for learning which has produced good results for fully connected, $Q_{\mathcal{N}}=Q_{\mathcal{J}}=2$ networks of up to a few hundred neurons is that of simulated annealing.[18] This method is very general; however, it is nonlocal and somewhat time consuming. Although it is desirable to have a local learning rule such as those which exist for networks with continuously varying synapses[12] (see ref. 19 for a review), that does not seem to be possible. Instead, one can easily create a local learning rule which is "guided" by a real vector.

Consider the problem of making the states $\{\xi_i^\mu\}$ stable states of a network with $Q_{\mathcal{N}}=Q_{\mathcal{J}}=2$. In this case, the fixed-point constraints of Eq. (2.3) reduce to[12]

$$h_i^\mu=\frac{1}{\sqrt{N}}\sum_{j\neq i}\xi_i^\mu J_{ij}\xi_j^\mu\geqslant\kappa,\qquad\forall i \text{ and } \mu \tag{4.1}$$

where κ is a constant which determines the size of the basins of attraction. A simple local learning rule which must be executed sequentially over the patterns *and* nodes can be defined as follows:

1. Calculate an error mask ε_i^μ for pattern μ:

$$\varepsilon_i^\mu = \begin{cases} 1 & \text{if } \kappa \geq h_i^\mu \\ 0 & \text{if } \kappa < h_i^\mu \end{cases} \tag{4.2}$$

2. Change a real *vector* T_j by $\Delta T_j = \varepsilon_i^\mu \xi_i^\mu \xi_j^\mu (\kappa - h_i^\mu)$.
3. Change J_{ij} to $J_{ij} = \text{sgn}(T_j)$.
4. Repeat steps 1–3 until condition (4.1) is satisfied.

This learning algorithm is an extension of the learning algorithms used with continuous coupling networks (see ref. 19 for a review). It is simple to implement and requires only an additional real vector of length N. Since the storage requirements of the program grow like $O(N^2)$ bits, the additional NB bits needed for this vector are not normally important. Furthermore, this algorithm is easily extendible to the more general situation of $Q_A, Q_B > 2$.

Unlike the case of continuous synapses, this algorithm cannot be proven to converge to a solution of (4.1), even when such a solution exists. For this reason, it will be demonstrated via numerical simulations that the learning algorithm presented here does perform quite reasonably.

Figure 2 shows a plot of the learning time of the above algorithm as a function of κ at $N = 4096$ and $\alpha = 0.25$ for $Q_A = Q_B = 2$, $\alpha = 0.31$ for $Q_A = 2$, $Q_B = 3$, and $\alpha = 0.5$ for $Q_A = 2$ with continuous synapses. In this

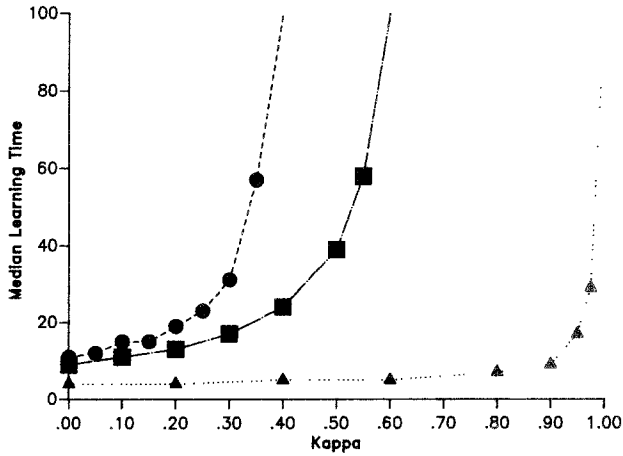

Fig. 2. Median learning time on an $N = 4096$ network. The circles are for $Q_A = Q_B = 2$ at $\alpha = 0.25$, the squares for $Q_A = 2$, $Q_B = 3$ at $\alpha = 0.31$, and the triangles for $Q_A = 2$, Q_B continuous at $\alpha = 0.5$.

figure, the maximum value of κ is signaled by the sharp increase in the learning time. Using $\kappa = 0.35$ for $Q_{\mathcal{N}} = Q_{\mathcal{I}} = 2$ and $\kappa = 0.50$ for $Q_{\mathcal{N}} = 2$, $Q_{\mathcal{I}} = 3$, so that the learning time is not too long, but the network is near optimal, the probability of recall has been calculated and is shown in Fig. 3. The probability of recall for random patterns is calculated by starting the system in some randomly chosen configuration $\{S_j(0)\}$ having an overlap $m^{\mu}(0) \equiv (1/N) \sum_j S_j(0)\, \xi_j^{\mu}$ with pattern μ and then updating the system in parallel, using Eq. (2.1), until a fixed point is reached. Figure 3 shows that at these values of α the systems can indeed function as associative memory devices. Since the maximum value of κ decreases as α increases, larger values of α will have smaller basins of attraction and these systems will no longer function as associative memory devices; hence, $\alpha_c \approx 0.25$ for $Q_{\mathcal{N}} = Q_{\mathcal{I}} = 2$ and $\alpha_c \approx 0.31$ for $Q_{\mathcal{N}} = 2$, $Q_{\mathcal{I}} = 3$. Note that α_c is about four times smaller than the optimal α shown in Fig. 1. This is in rough agreement with the results found for continuous couplings, where the effective maximum $\alpha_c \approx 0.5$ was also found to be about four times smaller than the theoretical maximum.[10]

The learning algorithm presented here is local, simple to implement, and seems to be among the fastest so far discussed in the literature (for alternative learning algorithms see ref. 20). For finite-state networks it is a good starting point for simulation purposes, and so far has shown that such networks perform better than their continuously varying counterparts.

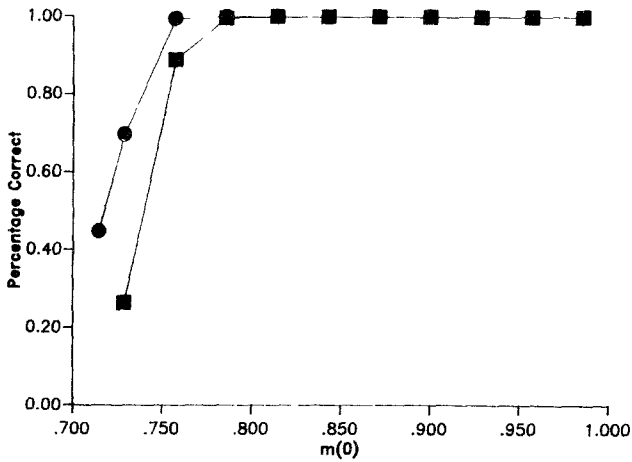

Fig. 3. Probability of retrieval for an $N = 4096$ network. Circles are for $Q_{\mathcal{N}} = Q_{\mathcal{I}} = 2$ at $\alpha = 0.25$, and the squares are for $Q_{\mathcal{N}} = 2$, $Q_{\mathcal{I}} = 3$ at $\alpha = 0.31$.

5. SUMMARY AND CONCLUSIONS

The main results of this paper are summarized in Table I, where the $Q_{\mathcal{N}}=Q_{\mathcal{J}}=2$ network, the $Q_{\mathcal{N}}=2$, $Q_{\mathcal{J}}=3$ network, and the $Q_{\mathcal{N}}=2$ network with continuous couplings are compared when all of these systems are implemented on the 64-bit Cray-YMP. Here it is easily seen that the performance of finite-state networks is superior to that of networks with continuously varying synapses in four of the five categories. The main advantage of synapses with a large number of states seems to be faster learning, but for this price, most of the coupling space is wasted, resulting in low efficiency. It would seem that high storage efficiency can only be achieved at the expense of long learning times. More generally, the calculations of Sections 2 and 3 have shown that whenever $Q_{\mathcal{J}} \gg Q_{\mathcal{N}}$, the network performance is severely degraded in all respects except learning time. For those designing novel computing devices, this can be very important because neurons or synapses with a small number of states are simpler to construct than those with a larger number of states.

Furthermore, with regard to possible neural network device implementations, the present work suggests that there is a great deal of parallelism which can be exploited in conventional computer designs. In the short term, it may be worthwhile to consider computers with a small number of processors, each working with very long words. With the algorithm introduced in Section 3, a speedup equal to the increase in word length is easily achieved if the clock frequency can be held fixed. Such speedups are not so easily obtained on massively parallel machines.

With respect to biology, the present results have indicated the inefficiency of systems with $Q_{\mathcal{J}} \gg Q_{\mathcal{N}}$. It would be of great interest, then, if the situation with respect to the number of neuron firing states and the number of synaptic states could be clarified. This would give a better indication of

Table I. Comparison of Performance Factors for Finite-State Neural Networks Running on a Single Processor of the Cray-YMP[a]

Network type	$\mathscr{E}_{\max}$	$\mathscr{E}_{\text{eff}}$	Bits required	Speed (Gcops)	Learning steps
$Q_{\mathcal{N}}=Q_{\mathcal{J}}=2$	0.833	0.25	N^2	7.1	57
$Q_{\mathcal{N}}=2, Q_{\mathcal{J}}=3$	0.740	0.20	$2N^2$	2.4	39
$Q_{\mathcal{N}}=2, Q_{\mathcal{J}}=2^{64}$	0.031	0.008	$64N^2$	0.2	17

[a] The last column gives the median number of learning steps at $N=4096$ when the network is operating near $\mathscr{E}_{\text{eff}}$. $Q_{\mathcal{J}}=2^{64}$ represents the Cray approximation to continuous synapses.

the relative importance nature has placed upon the speed of learning as opposed to the efficiency of information storage.

One final point with respect to neural network implementations; since a $Q_{\mathcal{N}} = Q_{\mathcal{J}} = 2$ network has α_e only half of that for a network with continuous couplings, it means that a $Q_{\mathcal{N}} = Q_{\mathcal{J}} = 2$ network with $2N$ neurons can store as many patterns as an N-neuron network with continuous couplings. Moreover, the former network with still require far fewer bits for storage and it will still update about eight times faster. Hence, on conventional computers there is clearly much to be gained by using the algorithms introduced in this paper.

ACKNOWLEDGMENTS

I would very much like to thank D. Stauffer and K. E. Kürten for many helpful conversations related to this work.

REFERENCES

1. D. Amit, H. Gutfreund, and H. Sompolinsky, *Ann. Phys.* **173**:30 (1987); C. M. Newman, *Neural Networks* **1**:223 (1988); T. Kohonen, *Self-Organization and Associative Memory*, 3rd ed. (Springer-Verlag, Berlin, 1989).
2. D. E. Rumelhart and J. L. McClelland, eds., *Parallel Distributed Processing* (MIT Press, Cambridge, 1986); R. Scalettar and A. Zee, *Biol. Cybern.* **58**:193 (1988); K. Hornik, M. Stinchcombe, and H. White, *Neural Networks* **2**:359 (1989); M. Opper, W. Kinzel, J. Kleinz, and R. Nehl, University of Giessen preprint.
3. J. R. Barker, in *Parallel Processing in Neural Systems and Computers*, R. Eckmiller, G. Hartmann, and G. Hauske, eds. (North-Holland, Amsterdam, 1990).
4. D. J. Amit, G. Parisi, and S. Nicolis, *Network* **1**:75 (1990).
5. J. J. Hopfield, *Proc. Natl. Acad. Sci. USA* **79**:2554 (1982); W. A. Little, *Math. Biosci.* **19**:101 (1975).
6. C. M. Marcus, F. R. Waugh, and R. M. Westervelt, *Phys. Rev. A* **41**:3355 (1990), and references therein.
7. M. Abeles, E. Vaadia, and H. Bergman, *Network* **1**:13 (1990).
8. T. J. P. Penna and P. M. C. Oliveira, *J. Phys. A* **22**:L719 (1989).
9. G. A. Kohring, *J. Stat. Phys.* **59**:1077 (1990); *J. Phys. A* **23**:2237 (1990).
10. B. M. Forrest, *J. Phys. A* **21**:245 (1988); J. Krätzschmar and G. A. Kohring, *J. Phys. (Paris)* **51**:223 (1990).
11. P. Peretto and J. J. Niez, *Biol. Cybern.* **54**:53 (1986); C. L. Giles and T. Maxwell, *Appl. Opt.* **26**:4972 (1987); G. A. Kohring, *J. Phys. (Paris)* **51**:145 (1990).
12. T. M. Cover, *Proc. IEEE* **EC14**:326 (1965).
13. E. Gardner, *Eur. Phys. Lett.* **4**:481 (1987); *J. Phys. A* **21**:257 (1988).
14. W. Krauth and M. Mézard, *J. Phys. (Paris)* **50**:3057 (1989).
15. J. L. van Hemmen, *Phys. Rev. A* **36**:1959 (1987); H. Englisch and M. Herrmann, *Studia Biophys.* **132**:145 (1989); D. Bormann, *Z. Phys. B* **79**:307 (1990).
16. I. Kanter, *Phys. Rev. A* **37**:2739 (1988); C. Meunier, D. Hansel, and A. Verga, *J. Stat. Phys.* **55**:859 (1989).

17. B. M. Forrest, *J. Phys. (Paris)* **50**:2003 (1989); see also K. E. Kürten, *J. Phys. (Paris)* **50**:2313 (1989).
18. E. Amaldi and S. Nicolis, *J. Phys. (Paris)* **50**:2333 (1989); H. Köhler, S. Diederich, W. Kinzel, and M. Opper, *Z. Phys. B* **78**:333 (1990).
19. L. F. Abbot, *Network* **1**:105 (1990).
20. K. E. Kürten, *J. Phys. (Paris)*, in press.
21. K. Binder and A. P. Young, *Rev. Mod. Phys.* **58**:801 (1986).
22. K. E. Kürten, *Phys. Lett. A* **129**:157 (1988); A. Canning and E. Gardner, *J. Phys. A* **21**:3275 (1988); E. Koscielny-Bunde, *J. Stat. Phys.* **58**:1257 (1990).
22. E. Gardner and B. Derrida, *J. Phys. A* **21**:271 (1988).