Pruning versus clipping in neural networks

Steven A. Janowsky
Department of Physics, Harvard University, Cambridge, Massachusetts 02138
(Received 30 November 1988)

The number of interconnections in a neutral network is reduced by eliminating the "weakest" bonds. The performance is then improved by reapplying the learning algorithm.

I. INTRODUCTION

The study of neural networks has gained a great deal of attention in the physics literature as of late.[1-3] While the initial motivation behind the study of neural networks was based on attempts to understand the functioning of the brain,[4-8] the analogy with spin glasses[7,9] brought physicists into the fold. Hopfield's work[10] revitalized interest in the use of such networks as computation and memory devices. Although neural networks have many other uses,[11] here we will consider only the memory aspect of spin-glass-like neural networks.

The standard spin-glass neural network, where spins take the values ± 1, is fully connected, with every neuron connected to every other neuron. For almost all implementations this becomes a problem as the number of neurons grows, since the number of connections grows as the number of neurons squared. This has provoked investigation of sparse networks, with lesser connectivity.[1,2,12,13]

There are several methods of reducing the connectivity of a network. Of course one wants to find a way to reduce the number of bonds without significantly degrading the performance of the network.

The simplest method of reducing the number of bonds is to delete bonds at random. While seemingly crude, this works reasonably well if the number of deleted bonds (or the number of stored memories) is kept small.[13] For example, Sompolinsky[2] shows that randomly removing half of the bonds of the Hopfield model leaves it with the ability to store 60% of the memories of the undiluted model.

The problem with random dilution is that bonds are removed without regard for their importance. In what we call "clipping," weaker bonds are removed prior to strong bonds. This is similar to Morgenstern's "zero model."[13] So, for example, in a network that was clipped at 50% the smallest (in absolute value) half of the bonds would all be set to zero.

Clipping gives us a means of choosing which bonds should be deleted, but what about the remaining bonds? There is no *a priori* reason why their initial values should remain optimal after the clipping process. Perhaps more learning could take place on the subset of unclipped bonds, in a sense "tweaking" their values to perform better.

We use the term "pruning" to describe this process of relearning after clipping, in analogy with gardening where one cuts back a tree and then allows it to grow within that context.

II. THE LEARNING ALGORITHM

We take as our neural network model an Ising model undergoing zero-temperature parallel relaxation—at each step in the recall process

$$S_i(t+1) \rightarrow \mathrm{sgn}\left[\sum_{j=1}^{N} J_{ij} S_j(t)\right] . \tag{1}$$

By "learning" we refer to an algorithm for determining the matrix $\underline{J}$ such that the chosen memories are stable point attractors of (1). We chose to use the algorithm of Diederich and Opper,[14] an iterative algorithm suitable for the pruning process. This algorithm and other similar algorithms[15-17] share certain features: they store correlated patterns; they are local, allowing parallel implementation; and they are iterative, allowing new memories to be added (or in our case allowing old memories to be relearned).

The Diederich and Opper algorithm can be summarized by the following equation:

$$\delta J_{ij} = \frac{1}{N-1} S_i^{\nu} S_j^{\nu} \Theta\left[1 - S_i^{\nu} \sum_{k=1}^{N} J_{ik} S_k^{\nu}\right], \quad i \neq j . \tag{2}$$

Here δJ_{ij} is the incremental change in the connection matrix, $\mathbf{S}^{\nu} = (S_1^{\nu}, \ldots, S_N^{\nu})$ is a pattern to be stored, and Θ is the Heaviside function. Equation (2) is cycled through for all patterns ν until all the δJ_{ij} are zero. This leads to the condition

$$S_i^{\nu} \sum_{j=1}^{N} J_{ij} S_j^{\nu} \geq 1 \tag{3}$$

for all patterns, so that they are stable under the recall process (1).

Our pruning algorithm begins with the iterative application of Eq. (2). When the algorithm converges, the matrix is ready for clipping. The bonds $\{J_{ij}\}$ are sorted by absolute value, and those with the smallest magnitudes have their values set to zero. Then more bonds are set to zero, in order of increasing magnitude, until a predetermined fraction of the bonds have been clipped.

After clipping, we are ready for reapplication of the learning algorithm. Bonds that are zero remain zero. Nonzero bond strengths are incremented according to a

variant of the equation (2)—instead of $\delta J_{ij}=(N-1)^{-1}\times\cdots$, we have $\delta J_{ij}=N_i^{-1}\times\cdots$, where N_i is the number of (nonzero) bonds feeding in to neuron i: $N_i=|\{j|J_{ij}\neq 0\}|$. For a fully connected network, $N_i=N-1$.

This relearning reestablishes condition (3), which had been violated after the clipping process. Thus all memories are again stable states of the network, and presumably radii of convergence are increased as well.

III. COMPUTER SIMULATIONS

Computer simulations were performed to compare the performance of the clipped and pruned networks. Networks of 225, 400, and 900 neurons were tested as fully connected networks as well as at clipping and pruning levels of 40%, 60%, and 80%. The simulations were done on a Sun-4 supermicrocomputer, with the programs written in the C programming language.

Each network was presented with a certain number of randomly chosen patterns, which were then learned by the network (with the appropriate learning algorithm). Then the same patterns were degraded by random noise, and were used as initial conditions for the network to recall. The number of successful recalls at each noise level was then recorded. An example data run is shown in Fig. 1.

To evaluate effectively the various learning algorithms,

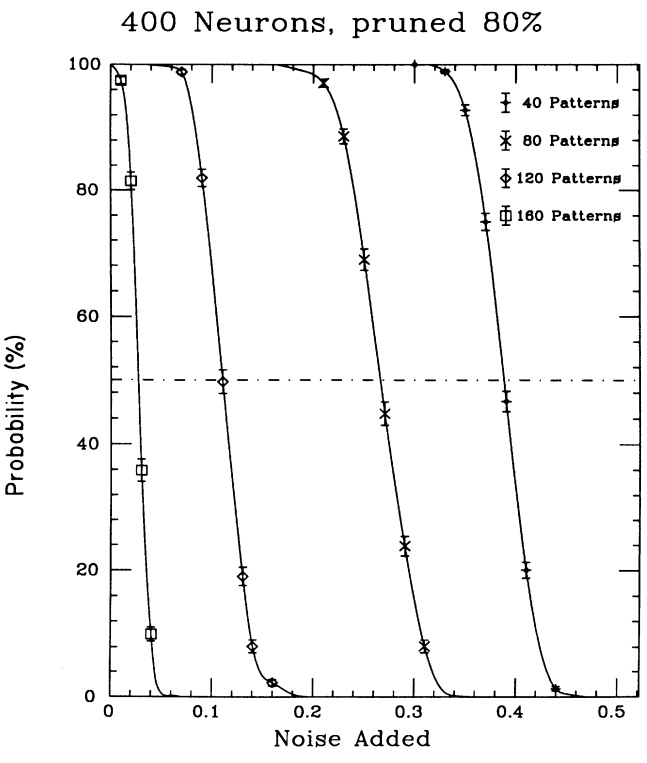

FIG. 1. Plot of the probability of a degraded memory being properly recognized vs the amount of added noise. The measure of noise is the fraction of bits flipped. Error bars represent statistical uncertainty. The dot-dashed line indicates recognition 50% of the time.

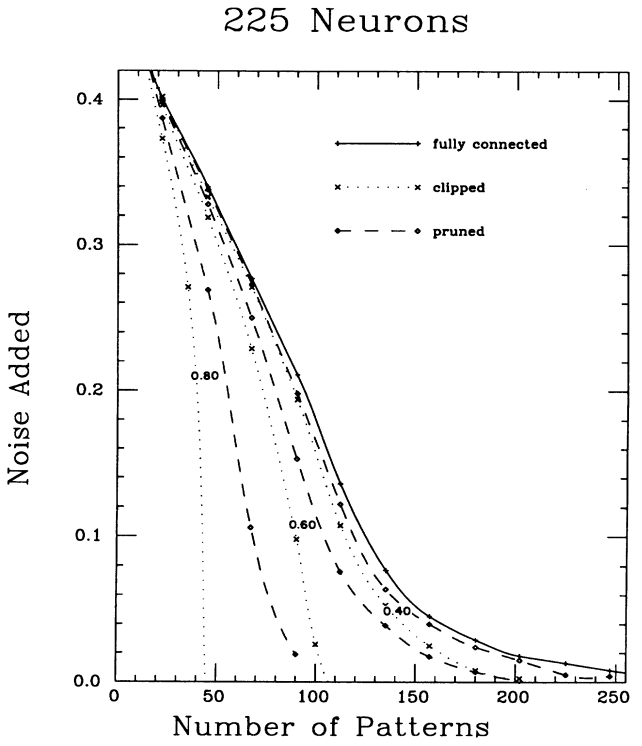

FIG. 2. Plot of noise level causing 50% recognition vs number of patterns, for 225 neurons. From left to right the curves represent clipped 80%, pruned 80%, clipped 60%, pruned 60%, clipped 40%, pruned 40%, and fully connected.

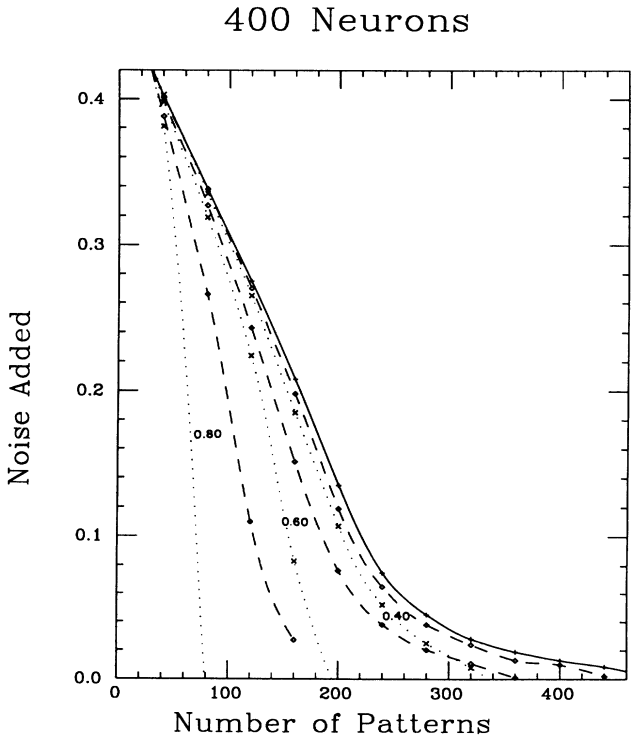

FIG. 3. Plot of noise level causing 50% recognition vs number of patterns, for 400 neurons. The legend is the same as in Fig. 2.

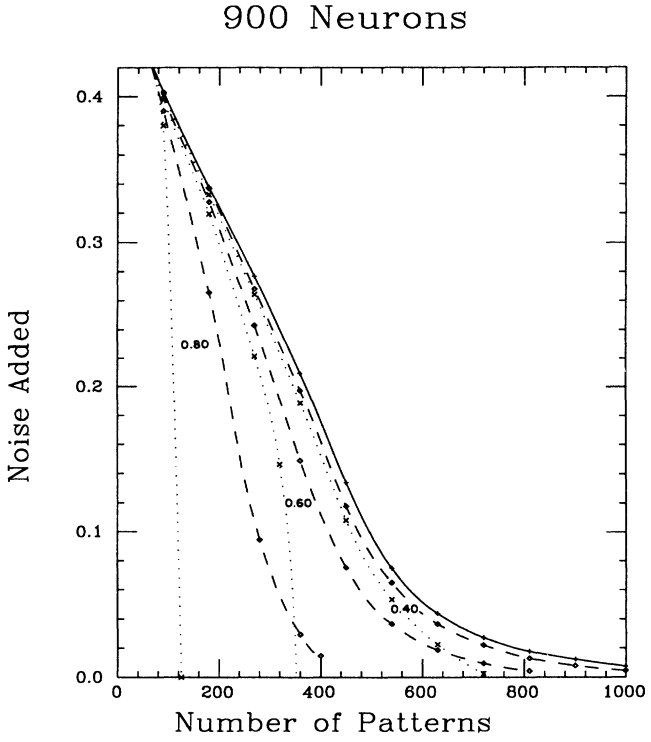

FIG. 4. Plot of noise level causing 50% recognition vs number of patterns, for 900 neurons. The legend is the same as in Fig. 2.

it is necessary to condense the data from its form in Fig. 1. As a useful tool for comparison, we chose the amount of noise required so that degraded patterns converge half of the time—this noise level is easily determined with high accuracy because of the near-linear behavior around the 50% level (see Fig. 1). In fact the slope increases as the number of neurons increases (indicative of the onset of a phase transition), so for large systems the exact convergence fraction used is irrelevant. Each network (with stored patterns) is now represented by a single point; these points are graphed in Figs. 2–4. It is clear from the graphs that the pruned networks significantly outperform the clipped ones.

In addition to comparing the algorithms qualitatively, we can examine some quantitative indicators. From Figs. 2–4 we determine the "capacity" of each algorithm as the number of patterns that can be stored so that a noise level of 0.10 gives 50% convergence. There is nothing special about the level 0.10; it is chosen for convenience.

Table I lists the capacities of the networks, as well as the capacity per neuron and per bond. In terms of capacity per bond, pruning is up to 2.7 times more effective than the fully connected system, and 1.5 times better than the best clipped system. One remarkable observation is how well the capacity scales with the number of neurons. In fact, Figs. 2–4 are almost exact copies of each other. The discrepancies at maximal density for the clipped systems appear to be artifacts of using 50% recognition as a criterion for data point selection.

TABLE I. We compare the memory capacity of the different learning algorithms. Capacity is defined as the number of memories that can be stored so that 50% can be properly recalled when 10% of the bits are incorrect.

	Memory capacity	Capacity/neuron	10^{-3} (Capacity/bond)
225 Neurons			
fully connected	124	0.55	2.46
clipped 40%	112	0.50	3.70
clipped 60%	89	0.40	4.41
clipped 80%	40	0.18	3.97
pruned 40%	119	0.53	3.94
pruned 60%	105	0.47	5.21
pruned 80%	68	0.30	6.75
400 Neurons			
fully connected	221	0.55	1.38
clipped 40%	202	0.51	2.11
clipped 60%	152	0.38	2.38
clipped 80%	70	0.18	2.19
pruned 40%	211	0.53	2.20
pruned 60%	183	0.46	2.87
pruned 80%	122	0.31	3.82
900 Neurons			
fully connected	500	0.56	0.62
clipped 40%	462	0.51	0.95
clipped 60%	335	0.37	1.04
clipped 80%	125	0.14	0.77
pruned 40%	475	0.53	0.98
pruned 60%	415	0.46	1.28
pruned 80%	275	0.31	1.70

IV. DISCUSSION

Almost any type of procedure designed to reduce the number of interconnections within a neural network will cause degradation of network performance. Minimizing that degradation can be achieved in two ways: prevention and repair.

Here we have used a combination of both methods. We prevent excessive degradation by removing the "smallest" bonds, and we repair the damage by reapplying the learning algorithm. The computer simulations show that this is effective for our choice of learning algorithm. Such ideas, however, are not limited to this particular choice. For example, one could consider a network where the number of bonds is fixed, but the topology is variable. Small, unnecessary bonds would be removed and replaced elsewhere. The learning algorithm would be continually reapplied until an optimal configuration was reached. We believe that iterative learning algorithms that allow such adjustments to take place deserve further investigation.

ACKNOWLEDGMENTS

I would like to thank Ken Berryman, Mario Inchiosa, and Arthur Jaffe for their many contributions that made this work possible. The work of S.A.J. was supported in part by National Science Foundation Grants No. DMS-88-58073 and No. PHY/DMS 86-45122.

[1]J. S. Denker, Physica **22D**, 216 (1986).
[2]H. Sompolinsky, in *Heidelberg Colloquium on Glassy Dynamics,* Vol. 275 of *Lecture Notes in Physics* (Springer-Verlag, Berlin, 1987), pp. 485-527.
[3]H. Sompolinsky, Phys. Today **41** (12), 70 (1988).
[4]W. S. McCulloch and W. Pitts, Bull. Math. Biophys. **5**, 115 (1943).
[5]F. Rosenblatt, Psychol. Rev. **65**, 386 (1958).
[6]M. Minsky and S. Papert, *Perceptrons* (MIT, Cambridge, MA, 1969).
[7]W. A. Little, Math. Biosci. **19**, 101 (1974).
[8]T. Kohonen, *Self-Organization and Associative Memory* (Springer-Verlag, Berlin, 1984).
[9]K. Binder and A. P. Young, Rev. Mod. Phys. **58**, 801 (1986).
[10]J. J. Hopfield, Proc. Natl. Acad. Sci. U.S.A. **79**, 2554 (1982).
[11]*Neural Networks for Computing (Snowbird, Utah, 1986),* in Proceedings on the Conference on Neural Networks for Computing, AIP Conf. Proc. No. 151, edited by John S. Denker (AIP, New York, 1986).
[12]A. Canning and E. Gardner, J. Phys. A **21**, 3275 (1988).
[13]I. Morgenstern, in *Heidelberg Colloquium on Glassy Dynamics,* Vol. 275 of *Lecture Notes in Physics* (Springer-Verlag, Berlin, 1987), pp. 399–427.
[14]S. Diederich and M. Opper, Phys. Rev. Lett. **58**, 949 (1987).
[15]B. M. Forrest, J. Phys. A **21**, 245 (1988).
[16]G. Pöppel and U. Krey, Europhys. Lett. **4**, 979 (1987).
[17]E. Gardner, J. Phys. A **21**, 257 (1988).