Random Neural Networks : a Beginning

P.McGuire G.Sonnenberg D.Harley G.Littlewort
Dr. J.Rafelski

10 December 1989

1 Introduction

John Clark states,"The first objective of neural-network computer simulation is to produce a net which, after an initial instantaneous stimulus to a selected set of neurons, remains active and does not display catastrophic behavior. The second objective of computer simulation is to produce interesting cycling behavior."

In our initial experiments on the computer, simulating small neural networks (N=200, M=30), we have quite consistently been able to fulfill the first objective, but we have yet to reach the second objective. We find it quite easy, via brainwashing, to quell a random net's urge to die or to become epileptic. However, we still need to polish our simulation if we hope to see any interesting phenomena, such as cyclic modes.

Nevertheless, our network simulation, together with its interactive environment, its tracking capabilities and our powerful Apollo workstation, are quite keen to polishing. The network's matrix of connection strengths is a sparse matrix; the connection matrix carries around a lot of trivial, but important, information which does not change in time (the zeroes). Since we are interested in how a random network evolves to a multistable network, we used lists of connections with their synaptic strengths and a pointer to that list, rather than the sparse connection matrix with all of its constant zeroes. For a net with average connectivity per neuron, m, using lists rather than matrices cuts down on the memory requirement by a factor of (1 - 2m), which for a typical net (N = 200, m = 0.15) that we modelled means that we only need 30% of the memory that we would need if we used matrices. Experts tell me, though I sometimes doubt them, that lists will save much internal processing time per unit of network time.

2 The Simple Model

We take a network of N neurons in discrete time t, each with a fixed number, M = Nm, of incoming connections of quasirandom connection strengths. A signal from one neuron to a second neuron contributes a certain amount of activity to the second neuron, where this amount ofactivity is the connection strength from the first neuron to the second neuron. The activity of a neuron decays by a certain percentage each time step. In time, each neuron accumulates signals from all the presynaptic neurons, and then decides whether or not to fire a signal to all the postsynaptic neurons, by comparing its activity to a threshold value , t, of activity. If the neuron fires, it cannot fire again for R time units and that neuron's activity goes to zero, but if the neuron doesn't fire, it continues to accumulate input pulses of activity.

The number of neurons and the number of incoming connections per neuron is fixed, say at N = 200 neurons, and M = 30 connections. The range of connection strength is integral, ranging from -128 to 127. The number of negative synapses, called the inhibitory fraction, is fixed at h = 0.3. The amount of decay of neuronal activity, the refractive period, and the threshold are globally constant, but we can vary them in time with our interactive environment.

Initially, we arbitrarily give the M incoming connection strengths per neuron random values and to random presynaptic neurons, and then we tell the net that an arbitrary percentage of its neurons are going to fire, so that on the next time step, all the postsynaptic neurons receive a boost of activity, and the initially firing ones go into their refractive state. Invariably, the net would go into trivial behavior, like epilepsy or death, if the connection strengths could not adapt themselves.

The net adapts itself via brainwashing to become more interesting and probably more multistable. When the net is brainwashing itself, it looks for correlation between pre- and postsynaptic neurons, and punishes the connection between correlated neurons. This means that if a neuron, by firing, causes one of its postsynaptic neurons to fire, we decrease the connection weight of that synapse, by a typical globally fixed percent, bw = 1%. We can vary the brainwashing parameter with time. After each step of brainwashing, all of the synaptic weights are renormalized, so as to retain nearly the same sum.

3 Initial Experiments

In our first experiment, we measured how the network simulation scales with the number of neurons, for a fixed connectivity. We found that for $N < 300$ neurons, that the time for each step is proportional to $N^{0.176}$ and for $N > 300$ neurons, that the time for each step is proportional to $N^{1.75}$. We did this experiment before we put the renormalizing procedure into the program, so maybe the results are no longer valid.

In our second experiment, we measured how the network simulation scales with connectivity, for N = 100 and N = 500. We found that the time for each step is linear, with $T = 0.072 + 0.378m$ for N = 100, and $T = 0.181 + 0.922m$ for N = 500. This makes intuitive sense since the length of our list is proportional to m, but the slopes should be proportional to N^2, which is not the case.

In our next experiment, we tested the randomness of our random number generator. We found that our home-cooked random number generator was terrible. We know that the number of incoming synapses per neuron is a constant, and we wanted to know the distribution of the number of outgoing synapses per neuron. With our first random number generator, we found that this distribution was sharply peaked at zero outgoing synapses per neuron (for N = 200, M(in)= 30), with several subsidiary peaks at higher numbers of outgoing synapses at a higher number of outgoing synapses per neuron, giving a mean number of outgoing synapses, $\langle M(out) \rangle = 29.95$. The weird distribution did not make sense, so we tried a new random number generator. For N = 300, M(in) = 50, we found that the distribution of outgoing synapses was sharply peaked at $\langle M(out) \rangle = 49.83$, quickly going to zero at M(out) = 30 and M(out) = 70. This random number generator is much better.

Next, we looked at how the distribution of synaptic weights evolves with time due to brainwashing and renormalization, given some initial distribution of weights. We initialized the weights to be 70% excitory and 30% inhibitory, of random magnitude $V_{ij} \leq 40$. However, before brainwashing and before renormalization, the distribution looks very similar to our specifications of $V_{ij} \leq 40$, yet there are long tails of about 25 synapses/synaptic strength, stetching all the way out to $|V_{ij}| = 128$. There is still a bug in the setup routine. However, after we brainwash at bw = 1% and renormalize for t time steps, the distribution of weights evolves to a distribution sharply peaked around $V_{ij} = 0$. I am still unsure what to do with synapses with $|V_{ij}| > 128$. A synaptic weight wishes to step out of bounds if its pre- and postsynaptic neurons are uncorrelated, and therefore, the weight is consistently renormalized to be larger and larger than the active links which are being punished. If we stipulate that synaptic weights which want to increase their magnitude beyond $V_{ij} = 128$, remain at $V_{ij}=128$, then we invariably have a pileup of weights at $V_{ij} = 128$, if we brainwash for a long enough time ($t \geq 50$) for our initial distrbution, with long tails. Maybe if we eliminated the long tails we would not have to worry about this problem of pileup.

In our most recent experiment, we used the Trackertools filter on the Sun workstation to monitor the activity (number of neurons fired) and the eligibility as a function of time. For a non-plastic net of N = 200 neurons, M(in) = 30 connections/neuron, the net quickly goes into epilepsy at t = 150. This was evident by the eligibility going to zero, and the activity becoming nearly constant at A = 175 neurons firing/time step. Then to increase the eligibility and cure the epilepsy, we brainwashed for 50 steps at bw = 1%. A sharp drop of activity ensued, to $A \approx 75$ neurons firing /time step. The eligibility went up to nearly 75% and there was much more variation in net activity, from A = 60 n/s to 100 n/s. Therefore, brainwashing does tend to enrich an otherwise boring random neural network.

4 Conclusion and Future Prospects

We still need to debug, polish and optimize our network simulation before we can expect further interesting results. For example, for nets bigger than N = 300, the simulation runs quickly for a small number of of the first time steps (t(fast) $\approx$ 10 for N = 300, M = 50), and then slows down incredibly, say tenfold. Probably other processes on the workstation begin taking over after a short time, allocating only a short time to the simulation. If we can eliminate this scaling problem, we can go up to bigger nets with ease.

However, we have much flexibility and potential power with our interactive environment and tracking capabilities. If we use the Sun workstation as the tracker and Apollo series 10000 workstation as our calculator, we have very reasonable, though uncalculated, limitations, but many possibilities. Also, with our list method of representing the sparse connection matrix, we save much in memory and computing time, so we can simulate bigger nets more quickly. Also, with the list method, we can add and subtract connections much easier than with the standard matrix method. I do not think that the list method limits us in any new way.

Possibilities for future work abound. We must continue to verify our net. We can continue to look at the statistical mechanics of a single random neural net, perhaps by observing its susceptibility to small changes in initial conditions, or to verify the work of Littlewort, Rafelski, Clark and Winston (the transition to cycling, etc.) Then we might observe the interaction of two weakly connected networks, one random and the other either random or Perceptron-like (layered), and slowly work up to the interaction of N weakly-coupled networks, two of them being layered, to serve as input and output registers. Maybe by then we will understand how to program a computer well enough to simulate and observe the complex processes that the computer should be able to simulate.