Attractor neural networks

Bruno A. Olshausen

May 26, 1998

Abstract

This handout describes recurrent neural networks that exhibit so-called "attractor dynamics." The principles governing these networks were first described by John Hopfield in the early 1980's, who showed that recurrent neural networks with symmetric connections could be thought of as having basins of attraction. Such networks serve as a useful physical metaphor for associative memory, and they form the basis of many other models from perception (Marr-Poggio stereo algorithm) to navigation (Zhang's "neural compass").

Up to now we have discussed information processing and learning primarily as it occurs in feedforward networks, where information flows directly from one layer of neurons to the next without feedback (Figure 1*a*).[1] In the brain, the existence of such network connectivity (such as between the retina and LGN) is rare, and what is by far more common are networks with feedback (Figure 1*b*). Given the fact that such recurrent networks are ubiquitous in the brain, it is important to understand something about their behavior. Here we shall examine the behavior of networks with feedback under certain simplifying assumptions.

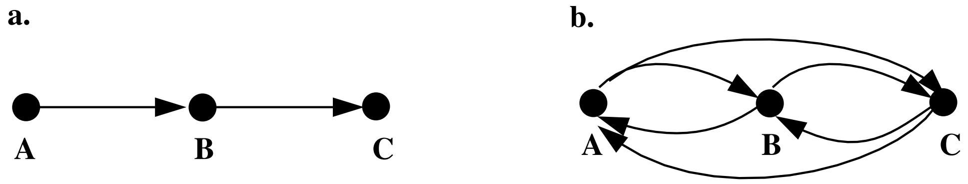

Figure 1: Feedforward (*a*) vs. recurrent (*b*) networks.

The type of network we shall is examine is where the neurons are connected pairwise to each other, as shown below for a five neuron network. We shall use V_i to denote the activity state of neuron i in the network, and T_{ij} to denote the strength of the connection from neuron j to neuron i. Each neuron can take on one of two

[1]The exception perhaps is the back-propagation algorithm, where error signals are fed backwards through the network. But even here, such feedback signals are used only for learning and the do not effect the activity state of the intermediate layers.

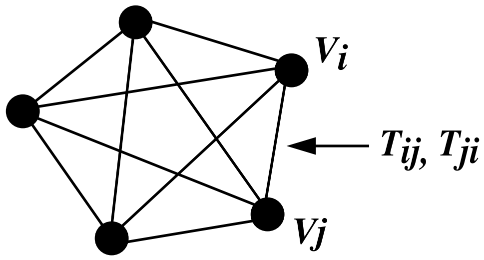

Figure 2: A recurrent network with 5 neurons. All connections are two-way.

states, $+1$ or -1, and each neuron changes its state, V_i, according to the following rule:

$$\begin{aligned} V_i &\to 1 \quad \text{if} \quad \sum_{j \neq i} T_{ij} V_j > 0 \\ V_i &\to -1 \quad \text{if} \quad \sum_{j \neq i} T_{ij} V_j < 0 \end{aligned} \tag{1}$$

Thus, each neuron takes a weighted sum of the activities of other neurons in the network, and flips positive or negative depending upon the current summed activity received from other neurons in the network. If a unit changes state, then it will likely change the state of other units in the network, which may in turn cause the same unit to change state again. For example, let us consider the dynamics of the two neuron network in Figure 3. For this simple network, you should be able to convince

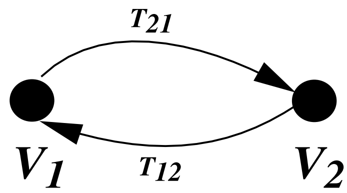

Figure 3: .

yourself without too much trouble that it will continually oscillate if the connections are anti-symmetric, $T_{ij} = -T_{ji}$, while it will come to rest at the state $+1, +1$ or $-1, -1$ if the connections are symmetric, $T_{ij} = T_{ji}$. However, for a network with five units (or 5 million), such a straightforward intuitive analysis becomes impossible. In order to know whether a network will ever settle into a state of equilibrium, and if so how the choice of weights T_{ij} affects the equilibrium state, we must invoke the notion of an *energy function* that governs the dynamics of the network.

Energy function

The idea of introducing an energy function to understand the dynamics of recurrent networks is not unlike the way physicists use energy functions to understand the dynamics of a physical system. Each state of activity in the network, $\mathbf{V}$, is assigned a corresponding energy, just like any state of particles in a physical system can be thought of as possessing a combination of potential and kinetic energy at any point in time. In our case though, the dynamics will be dissipative, meaning that the energy will decrease with time. If the energy function we use to govern the dynamics of the network has a lower bound, then we can show that the system will eventually come to rest.

The energy function that Hopfield (1982) introduced to govern the dynamics of pairwise recurrently connected networks, such as in Figure 2, is of the form

$$E = -\frac{1}{2}\sum_i \sum_{j \neq i} T_{ij} V_i V_j \tag{2}$$

Intuitively, one can see here that the energy will be lowest (i.e., the sum on the right will be highest) when the product of activities of each pair of units matches the connection between them. For example, if T_{ij} is positive, then the energy will be lowest when V_i and V_j have the same sign. Importantly, the energy has a lower bound, because the V_i are always just +1 or -1 and T_{ij} is finite. Thus, if we can show that the dynamics as specified in equation 1 always reduces the energy (or keeps it the same), then the system will for sure come to a stop at some state.

If the connections are symmetric, i.e., $T_{ij} = T_{ji}$, then we can show that the dynamics in (1) reduces the energy as follows. Let us consider the change in energy induced by making a positive change in the value of a single unit V_k (i.e., by changing it from -1 to +1, or keeping it the same), while keeping the values of all the other units fixed. Thus, we have

$$\begin{aligned}
\Delta E &= E^{new} - E^{old} && (3)\\
&= -\frac{1}{2}\sum_i \sum_{j \neq i} T_{ij} V_i^{new} V_j^{new} + \frac{1}{2}\sum_i \sum_{j \neq i} T_{ij} V_i^{old} V_j^{old} && (4)
\end{aligned}$$

Now since only the kth unit is changing its value, the difference between $\mathbf{V}^{new}$ and $\mathbf{V}^{old}$ will be zero for all components except V_k. So, we need to separate out in the sum the terms that depend on V_k and the ones that don't

$$\begin{aligned}
E &= E_k + E_{others} && (5)\\
E_k &= -\frac{1}{2}\left[\sum_{i \neq k} T_{ik} V_i V_k + \sum_{j \neq k} T_{kj} V_k V_j\right] && (6)\\
&= -\sum_{i \neq k} T_{ki} V_k V_i && (7)\\
E_{others} &= -\frac{1}{2}\sum_{i \neq k} \sum_{j \neq i,k} T_{ij} V_i V_j && (8)
\end{aligned}$$

where (7) follows from (6) since the connections are symmetric. Now since the E_{others} term will cancel when we take the difference between E^{new} and E^{old}, we are left only with the difference due to E_k:

$$\begin{aligned} \Delta E &= E_k^{new} - E_k^{old} && (9) \\ &= -\sum_{i \neq k} T_{ki} V_k^{new} V_i + \sum_{i \neq k} T_{ki} V_k^{old} V_i && (10) \\ &= -\sum_{i \neq k} T_{ki} (V_k^{new} - V_k^{old}) V_i && (11) \\ &= -\Delta V_k \sum_{i \neq k} T_{ki} V_i \,. && (12) \end{aligned}$$

Now let us consider what the last equation says in light of the dynamics in (1). If the sum on the right is positive, $\sum_{i \neq k} T_{ki} V_i > 0$, then equation 1 says that we should flip V_k to the positive state (or if it is already positive then keep it the same). Thus, ΔV_k will be positive or zero. If the sum is positive, and ΔV_k is positive or zero, then ΔE must necessarily be negative or zero. Vice versa when the sum is negative. Therefore, if we let the network simply follow the dynamics as given in equation 1, then the energy will always decrease until it has reached bottom, at which point the system will come to equilibrium and the units will stop changing value.

Note that the critical assumption we needed to make here is that the connections are symmetric, $T_{ij} = T_{ji}$. Without making this assumption then it is not possible to show the system will have fixed points. In fact, for certain asymmetric weight settings the system will follow a trajectory or sequence - i.e., $\mathbf{V}^{(1)} \rightarrow \mathbf{V}^{(2)} \rightarrow \mathbf{V}^{(3)}$. Such a system may be useful in motor control, or in other situations where it would be useful to have specific activity sequences stored. For a network with symmetric connections though, the dynamics will converge to so-called "basins of attraction," or fixed points where the system comes to rest. These networks have been used as models of associative memory, in which the basins correspond to memories. If you start the system in some state that is near one of the memories that has been formed (for example, by providing only partial information, or a "key" on the inputs), then the other information associated with that state will be recalled on the other units of the network when the system comes to equilibrium. In order to make this system work usefully as a memory though, we need some way of storing memories, or basins, where we want them by appropriately setting the connection strengths T_{ij}. This we address next.

Setting the weights

Let us say there is a certain pattern, $\mathbf{V}^{\alpha}$, that we wish to store as a basin of attraction in the network. One way of doing this is through the Hebbian prescription,

$$T_{ij} = V_i^{\alpha} V_j^{\alpha} \qquad (13)$$

That is, if V_i^{α} and V_j^{α} have the same sign, then we set T_{ij} positive, and if V_i^{α} and V_j^{α} have opposite signs, then we set T_{ij} negative. We can show that this state will now form a fixed point in the network dynamics, since the summed input for the ith unit in response to an initial state $\mathbf{V}^\beta$ will be

$$\begin{aligned}
U_i &= \sum_{j\neq i} T_{ij} V_j^\beta && (14)\\
&= \sum_{j\neq i} V_i^\alpha V_j^\alpha V_j^\beta && (15)\\
&= V_i^\alpha \sum_{j\neq i} V_j^\alpha V_j^\beta && (16)
\end{aligned}$$

So if $\mathbf{V}^\alpha = \mathbf{V}^\beta$ then the sum $(\sum_{j\neq i}(V_j^\alpha)^2)$ will be positive and U_i will have the same sign as V_i, so V_i won't change state and the network will thus stay put.

What if we want to store multiple memories? Then we will need to form multiple basins of attraction at the states $\mathbf{V}^1, \mathbf{V}^2, \mathbf{V}^3$, etc. where we wish to put the memories. This we can do in a similar fashion to the rule above via superposition:

$$T_{ij} = \sum_\alpha V_i^\alpha V_j^\alpha \tag{17}$$

in which case the sum in (16) becomes

$$U_i = \sum_\alpha V_i^\alpha \sum_{j\neq i} V_j^\alpha V_j^\beta \,. \tag{18}$$

If the patterns we wish to store as memories have few elements in common, then the cross-terms $\sum_{j\neq i} V_j^\alpha V_j^\beta$ will tend to zero for $\alpha \neq \beta$ and U_i will still have the same sign as V_i. But as we attempt to store more patterns in the network, then the similarity between them will necessarily increase and memories will begin to degrade - i.e., the system will no longer have its basins of attraction at the desired locations. Thus, the system has a certain *capacity*, and for a Hopfield network (like we are discussing here) it is about 15% of the number of neurons in the network. But if the patterns stored as memories are similar to each other, then the capacity will be somewhat less. This you will verify through simulation.