Hopfield networks

We are interested in building a network of neurons that has a large number of fixed points. There are at least two reasons such a network might be useful. First, it can serve as a memory – if you want to store something in your brain for a while in the absence of input, pretty much be definition you need an attractor network. Second, you often want to recall something based on partial information. For instance, you might see somebody's face and want to recall other information about them (name, friend or enemy, etc.).

In principle we should do this with spiking neurons – and such networks have been built. However, that's complicated. So to illustrate the main idea, we'll consider about the simplest possible network: binary units with a discrete time update rule. We'll assume there are N units, denoted x_i, each of which can take on the values $+1$ or -1. We'll assume they are updated according to the rule

$$x_i(t+1) = \text{sign}\left(\sum_{j=1}^{N} J_{ij} x_j(t)\right). \tag{1}$$

There are three main ways to implement this set of equations:

1. Asynchronous update: a unit, i, is chosen randomly, and it is updated according to Eq. (1).
2. Serial update: the units are updated in order, and periodically: $1, 2, ..., N, 1, 2, ..., N, ...$
3. Synchronous update: the term inside the parentheses, $\sum_j J_{ij} x_j$, is computed, and that's used to simultaneously update all the units.

In some regimes it doesn't matter which scheme you use, but asynchronous update is easiest to analyze, so that's what we'll focus on.

While this update rule is massively simplified compared to real networks, it does retain some semi-realistic features: if input is large enough the neuron "spikes" ($x_i = +1$); otherwise the neuron is "silent" ($x_i = -1$). The reason for the quote marks is that we're not really going to think of $+1$ as active and -1 as silent: as we'll see, at any time, about half the neurons take on the value $+1$; much more than in a real network.

Lyapunov function for the Hopfield network. The first observation is that if the weight matrix is symmetric ($J_{ij} = J_{ji}$), its diagonal elements are zero ($J_{ii} = 0$), and updates are asynchronous, this system of equations admits a Lyapunov function – a function that cannot increase under Eq. (1). We're not going to use that fact, so you can skip this on first reading. It's important mainly for historical reasons.

The Lyapunov function, which we'll call E (for energy), is

$$E(t) = -\frac{1}{2}\sum_{ij} x_i(t) J_{ij} x_j(t) \tag{2}$$

The difference in energy on subsequent times steps, denoted ΔE, is given by

$$\Delta E \equiv E(t+1) - E(t) = -\frac{1}{2}\sum_{ij} x_i(t+1) J_{ij} x_j(t+1) + \frac{1}{2}\sum_{ij} x_i(t) J_{ij} x_j(t)\,. \tag{3}$$

Because we're using asynchronous update, only one unit changes at a time. For definiteness, assume it's unit k. Noting that $x_i(t+1) = x_i(t)$ unless $i = k$, the change in energy is

$$\Delta E = -\frac{1}{2} \sum_{j \neq k} \big(x_k(t+1) - x_k(t)\big) J_{kj} x_j(t) - \frac{1}{2} \sum_{j \neq k} x_j J_{jk} \big(x_k(t+1) - x_k(t)\big) \,. \tag{4}$$

We didn't need the fact that $J_{ii} = 0$ to derive this expression. That's because $x_i^2 = 1$ regardless of the value of x_i, so the diagonal elements of the weight matrix don't contribute to any time variation in the energy, and so they don't contribute to ΔE. Using the fact that the weight matrix is symmetric, this simplifies to

$$\Delta E = -\big(x_k(t+1) - x_k(t)\big) \sum_{j \neq k} J_{kj} x_j(t) \,. \tag{5}$$

Inserting the update rule, Eq. (1), into this expression gives us

$$\Delta E = -\big(\text{sign}(h_k(t)) - x_k(t)\big) h_k(t) \tag{6}$$

where

$$h_k(t) \equiv \sum_{j \neq k} J_{kj} x_l(t). \tag{7}$$

Here we did use the fact that $J_{ii} = 0$; otherwise, the term h_k that appears inside the sum would have a contribution from J_{kk}.

There are three possibilities: $h_k(t) = 0$, $h_k(t) > 0$ and $h_k(t) < 0$. If $h_k(t) = 0$, the right hand side of Eq. (6) is zero, and the energy doesn't change. If $h_k(t) > 0$, then either $x_k(t) = +1$, in which case the energy doesn't change, or $x_k(t) = -1$, in which case the energy goes down. Similarly, if $h_k(t) < 0$, the energy either stays the same or goes down. Thus, no matter what the value of $h_k(t)$, the energy does not increase.

The fact that the energy is non-decreasing means that the system must eventually go to a local energy minimum, at which point the x_i can't change any more. This means Eq. (1) admits at least one fixed point. The trick is to get multiple fixed points, and so have some control over them.

The Hopfield weight matrix. John Hopfield, in a seminal 1982 paper (*PNAS* **79**:2554-2558, 1982), considered the following weight matrix

$$J_{ij} = \frac{1}{N} \sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu \tag{8}$$

where the ξ_i^μ are drawn *iid* from the following simple distribution,

$$\xi_i^\mu = \begin{cases} +1 & \text{probability } 1/2 \\ -1 & \text{probability } 1/2 \,. \end{cases} \tag{9}$$

As we'll see below, the ξ_i^μ turn out to be stored memories. Note that we have not made the restriction $J_{ii} = 0$, so there's no guarantee that there will be a Lyapunov function. But in the large N limit there is. Note also that the factor of $1/N$ is unnecessary, as were taking the sign, but it's convenient.

If p is not too large compared to N, there are fixed points close to the ξ_i^ν. To see why, we'll let $x_i(t) = \xi_i^\nu$ and compute $x_i(t+1)$,

$$x_i(t+1) = \text{sign}\left(\frac{1}{N}\sum_{\mu,j} \xi_i^\mu \xi_j^\mu \xi_j^\nu\right) = \text{sign}\left(\xi_i^\nu + \frac{1}{N}\sum_{\mu\neq\nu,j} \xi_i^\mu \xi_j^\mu \xi_j^\nu\right). \tag{10}$$

To get a handle on the sum over μ and j, note that the terms in this sum are random and uncorrelated – each term is either $+1$ or -1, with probability 1/2. Therefore, by the central limit theorem, the sum (without the factor $1/N$) is a Gaussian random variable with a variance equal to the number of terms in the sum, which is $N(p-1)$. Thus, the variance of the whole term (including the factor $1/N$) is $(p-1)/N$, which, for large p, is close to p/N. We thus have have

$$x_i(t+1) = \text{sign}\left(\xi_i^\nu + (p/N)^{1/2}\eta_i\right) \tag{11}$$

where η_i is a zero mean, unit variance Gaussian random variable. If $p \ll N$, meaning the number of stored memories is much smaller than the number of neurons. No matter how small p is, in the large N limit (with p/N fixed) there will always be a few "bit flips". But that doesn't change the basic story: ξ_i^ν is an approximate fixed point.

When p is on the same order as N, the analysis becomes a lot harder. In 1987 – five years after Hopfield published his classic paper – Sompolinsky and colleagues showed that memories exist so long as $p < 0.138N$, but when p exceeds this value there's complete blackout and all memories vanish (Amit, Gutfreund and Sompolinsky, *Ann. Phys.* **173**:30-67, 1987). They used the replica method; a simpler derivation, using mainly pedestrian methods, was given in "Introduction to the theory of neural computation," (Hertz, Krogh and Palmer, Addison Wesley, 1991).

This would seem to be good news: even modest networks in mammals can easily exceed one million neurons, so it would seem to be easy to store lots of memories; certainly enough to explain human memory. However, there's a problem: the weight matrix given in Eq. (8) has all-all connectivity, meaning every unit is connected to every other unit. This is not the case in real brains; each neuron connects to somewhere between 1,000 and 10,000 others. We'll consider this more realistic connectivity next. We'll find that memory capacity is greatly reduced.

<u>Realistic connectivity.</u> Let's assume that each neuron in our network connects to, on average, K others; for simplicity we'll assume that those K other neurons are chosen randomly. In that case, the weight matrix (the equivalent of Eq. (8)) becomes

$$J_{ij} = \frac{\chi_{ij}}{K}\sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu \tag{12}$$

where

$$\chi_{ij} \equiv \begin{cases} 1 & \text{probability } K/N \\ 0 & \text{probability } 1-K/N\,. \end{cases} \tag{13}$$

Again we'll assume $x_i = \xi_i^\nu$ and estimate the capacity – the number of memories, p, for which there aren't too many bit flips. With this setting, $x_i(t)$ is given by

$$x_i(t+1) = \text{sign}\left(\frac{1}{K}\sum_{\mu,j}\chi_{ij}\xi_i^\mu\xi_j^\mu\xi_j^\nu\right) = \text{sign}\left(\xi_i^\nu\frac{1}{K}\sum_j\chi_{ij} + \frac{1}{K}\sum_{\mu\neq\nu,j}\chi_{ij}\xi_i^\mu\xi_j^\mu\xi_j^\nu\right). \tag{14}$$

The term $K^{-1}\sum_j \chi_{ij} \approx 1$, with corrections that are $\mathcal{O}(1/\sqrt{K})$. We'll ignore those corrections and focus on the second term, which has the potential to be much larger. There are about $K(p-1)$ terms in the sum, each of which is either $+1$ or -1, and they are all uncorrelated. Thus, the variance of the sum is $K(p-1)$, and the variance when the factor of $1/K$ is included is $(p-1)/K \approx p/K$. Consequently, we may write

$$x_i(t+1) = \text{sign}\left(\xi_i^\nu + (p/K)^{1/2}\eta_i\right). \tag{15}$$

So long as p is small compared to K, there's a fixed point very close to ξ_i^ν. And, based on the results for all-all connectivity, we wouldn't be surprised if the capacity of the network – the number of memories it can store – is proportional to K.

That's the case, but it's a bit problematic; K simply isn't that big. Even if $K = 10,000$, multiplying that by 0.138, the scaling factor for the all-all case, the network can store only 1,380 memories. One possible fix is to build sparse memories: rather than letting half the ξ_i^μ be equal to $+1$, in a sparse system only a factor f are equal to $+1$. This increases the capacity; it scales more or less like K/f (Tsodyks and Feigel'man *Europhys. Lett.* **6**:101-105, 1988). If one makes f very small – on the order of $1/N$, then we regain the $\mathcal{O}(N)$ capacity that we saw in the all-all case. Unfortunately, in realistic spiking networks in which there's a background state – a state where all the neurons fire at low rate – f must be greater than some threshold (Roudi and Latham, *PLoS Comp. Biol.* **3**:679-1700, 2007). In simulations, that threshold was about 0.2. And the capacity was not very high; the number of memories that could be stored was about $0.05K$. For $K = 10,000$, that's only 500 memories! Not nearly enough to explain the memory capacity of humans.

The conclusion, then, is that we have no idea how to build networks of spiking neurons that exhibit a large number of attractors. OK, that's not exactly true: one could make a high capacity attractor by decoupling neurons, and making each of them bistable. In that case, for a network of N neurons, there would be 2^N attractors. However, that wouldn't be so useful; what we want is a set of attractors with reasonable basins of attraction, so that we can fill in missing information based on partial cues. For that the neurons need to be coupled, and it's the coupling that makes things hard. This is an active area of research.