3.1 Associative Memory

One important thing to pay attention to when using Hopfield nets as an associative memory is that there are two equivalent sets of state variables, namely $U_i$ and $V_i$. When we say that we want to “store a pattern” or “make a pattern stable” or “have a minimum of energy at a pattern” we always have to be sure we are clear on whether we mean a pattern in $U$ space or in $V$ space. For all the work we have done in lecture we work with the $V$ space variables; thus when we say “store a pattern” we always mean “store a pattern of $V_i$’s”.

- So when we create our weight matrix according to the formula:

$$T = \sum_{k=1}^{m} (P_k)(P_k)^T$$

(except zeros on the diagonals)

My routine weight.m is in Appendix.

This encourages the vectors $P_k$ to be stable $V$ space states of the system (not stable $U$ space states). When we check to see if the patterns were “retrieved” we should check to see where the system converged to in $V$ space. That is why the function plotumpd displayed a movie of the $V_i$’s not the $u_i$’s. A reminder: the above rule is sometimes called a version of the Hebb rule because if we imagine a learning rule $\Delta T_{ij} = V_iV_j$ as in Hebbian learning and we further imagine that the network activity is being driven by some external stimuli (for example retinal images) then the $T$ given above is what we end up with when the weights converge (up to some scale factor).

One last comment about the self-connections. Setting the diagonal elements of $T$ to zero is not required for the stability analysis, but in fact it is better in terms of spurious states to leave these connections out. A quick intuition about why this might be so comes from realizing that the self connections cannot distinguish between patterns. For any pattern at all, $< V_iV_i > = 1$ and so the self connections will only “stick” us in the state we are currently in, but will not help us place the minima where we need them.

- Two more important comments: the associative memory works much better if it is run in the high gain limit, in our case $\beta \approx 10$ works nicely. It should be certainly run with the gain high enough to allow multiple stable states, in other words with $\beta > 1$. Finally, you will have noticed that if a certain state $V$ is stable then its negative $-V$ is also stable. This is correct behaviour, to be expected, and easy to explain if you think it through and remember that $T_{ij}$ is symmetric.

- When you played with the associative memory you will have noticed that pattern correlations really kill you. In particular you should have noticed that $m = 2$ was the limit for “successfully storing” the original patterns. It is important to clearly define what we mean by “successfully storing”; here we take it to mean that the original patterns are stable states regardless of additional spurious states. (Actually if you chose the patterns “1”, “2”, and “0” then you could get $m = 3$ to work.) With these low $m$ values successful retrieval is possible with relatively high levels of noise. Again it is important to be clear on what “successful retrieval” means; here we take it to mean that at least 90% of the time when a certain level of noise was added the original pattern was the stable state into which the system settled. Using these definitions, with the digits example it was possible to get signal-to-noise ratios\(^1\) as low as about -10dB for $m = 2$ and between 2dB and 8dB for $m = 3$.

\(^1\)The signal to noise ratio in decibels is $10 \log_{10}(\sigma^2_{signal}/\sigma^2_{noise})$.
and still retrieve the original pattern 90% of the time. For higher $m$ the system did not have stable states at the stored patterns but instead a single spurious minimum. This minimum occurred at the point in the state space that represented the correlations between patterns, essentially at the thresholded mean pattern. The theoretical capacity of a 100 neuron network is something like 14 patterns so we seem to be falling way short at $m = 2$; but remember this capacity is calculated assuming perfectly uncorrelated patterns.

### 3.2 Graph Bipartitioning

- We are asked to minimize the number of edges between the two groups, which we will call $\epsilon$. Since $V_i V_j = 1$ if vertex $i$ and vertex $j$ are in the same group, and $-1$ otherwise, then $\frac{1}{2}C_{ij}(1 - V_i V_j) = 1$ iff vertex $i$ and vertex $j$ are in different groups and there is an edge between them; the expression is 0 otherwise. (We are considering the high-gain limit, where $V_i = \pm 1$, but cannot be anything in between.) This allows us to write:

$$
\epsilon = \sum_i \sum_j \frac{1}{2}C_{ij}(1 - V_i V_j) .
$$

However, we also require that there be an equal number of vertices in each group. That gives rise to the constraint $\gamma = 0$ where:

$$
\gamma = \sum_i V_i .
$$

So a reasonable energy function would be:

$$
L = \epsilon + K \gamma^2 = \sum_i \sum_j \frac{1}{2}C_{ij}(1 - V_i V_j) + K(\sum_i V_i)^2
$$

where $K$ is a constant yet to be determined. Let’s consider the minima of this function. First, $L > 0$ since both $\epsilon$ and $\gamma^2$ must be positive. Note that the two terms are in conflict: $\epsilon$ achieves 0 when all vertices are in the same group, while $\gamma$ achieves 0 when both groups are the same size. $K$ determines the trade-off between the terms. If we make $K$ big enough, we can insist that the constraint $\gamma = 0$ is satisfied. How big is “enough”? Since $\epsilon \leq \sum_{ij} C_{ij} = E$, if we set $K = E$ then satisfying the constraint is always more important than reducing the number of undesired edges. Now, (in the high-gain limit) if $\gamma \neq 0$ then $\gamma^2 \geq 4$, so

$$
L \geq \epsilon + 4E \geq 4E .
$$

Therefore, any configuration where $\gamma = 0$ will be better than any other configuration where $\gamma \neq 0$, no matter what $\epsilon$ is in either configuration. And of the ones where $\gamma = 0$, the lowest $\epsilon$ gives the lowest $L$. We conclude that the global minimum of $L$ is a solution to the graph bipartitioning problem.

Whenever we have a quadratic energy function on $V_i \in \{-1, +1\}$, we can try to minimize it using a Hopfield net whose (high gain) Lyapunov function is:

$$
L = -\frac{1}{2} \left( \sum_i \sum_j T_{ij} V_i V_j \right) - \left( \sum_i I_i V_i \right) .
$$

To determine $T_{ij}$ and $I_i$ from the previous equation for $L$, we must expand the squared sum and compare coefficients (and ignore constant terms, including $V_i V_i$). This gives us:

$$
T_{ij} = C_{ij} - K \quad \text{for } i \neq j ,
$$

$$
I_i = 0 .
$$
Note that since $C_{ij} = 0$ in a graph without loops, $T_{ii} = 0$; also note that since $C_{ij} = C_{ji}$, the network is symmetric, so $T_{ij} = T_{ji}$ and Hopfield’s Lyapunov analysis applies.

• In this particular question:

$$C = \begin{bmatrix}
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 
\end{bmatrix}$$

And I choose

$$T_{ij} = C_{ij} - 7$$
$$\beta = 20$$
$$\eta = 0.01$$
$$\text{iters} = 100$$

Then the result will be good in $\{1, 2, 3, 4, 5\} / \{6, 7, 8, 9, 10\}$

A Hopfield net with these connection strengths will try to minimize $L$, but it may end up in a local minimum. Sadly, for our choice of $K$, every configuration where $\gamma = 0$ is a local minimum, since changing a single $V_i$ will lead to $\gamma \neq 0$ and thus a higher energy. This means that in addition to the global minimum $\{1, 2, 3, 4, 5\} / \{6, 7, 8, 9, 10\}$ where $\epsilon = 5$, there are local minima like $\{1, 2, 5, 9, 10\} / \{3, 4, 6, 7, 8\}$ where $\epsilon = 16$. This is not good. We might hope that the global minimum, being the deepest, will have enough influence at the origin (where we start the network) to tilt the gradient in its direction, but this is not guaranteed.

As a side note, remember that any simulation of the continuous dynamics will have finite gain, for which the Lyapunov function will be

$$L = -\frac{1}{2} \sum_{i,j} (C_{ij} - K)V_i V_j + \sum_i \int_0^{V_i} g^{-1}(V) dV$$

and the dynamic equation will be

$$\tau \frac{\partial u_i}{\partial t} = -u_i + \sum_j C_{ij} V_j - K \sum_j V_j$$

where $g(s) = \tanh(\beta s)$. One approach to ameliorating the effects of local minimum is to slowly increase the gains $K$ and $\beta$ during the simulation, so that the high-gain limit is achieved only at the very end — but this isn’t guaranteed to work either. You can’t beat an NP-complete problem!  

Appendix

weight.m:

\footnote{A tricky point could be raised, namely that although in practice we see Hopfield nets converging rapidly, we have no theoretical result telling us that it is always so. Thus it is conceivable, in the light of the NP-completeness results, that we could come up with a network scheme which always finds the global minimum... but which takes an exponential time to settle.}
function [t] = weight(p,m)

N = size(p,1);
t = zeros(N,N);

for i=1:m
    t=t+p(:,i)*p(:,i)';
end

for i=1:N
    t(i,i)=0;
end