

# CS 256: Neural Computation Lecture Notes

## Chapter 5: Hopfield Networks, Part II

- Stability of the Hopfield network.
- Solving the Traveling Salesman Problem with a Hopfield network.
- Digression: The Elastic Net Algorithm

Copyright © Robert R. Snapp 2008.

Compiled by  $\text{\LaTeX}$  at 5:44 PM on March 5, 2008.

# References

1. Robert J. McEliece, Edward C. Posner, Eugene R. Rodemich, and Santosh S. Venkatesh, “The Capacity of the Hopfield Associative Memory,” *IEEE Transactions on Information Theory*, **33**(4), 1987, pp. 461–482.
2. Santosh S. Venkatesh, *The Mathematical Foundations of Neural Networks*, 1996, (unpublished course notes).
3. John J. Hopfield and D. W. Tank, “‘Neural’ Computation of Decisions in Optimization Problems,” *Biological Cybernetics*, **52**, 1985, pp. 141–153.
4. John J. Hopfield and D. W. Tank, “Computing with Neural Circuits,” *Science*, **233**, 1986, pp. 625–629.
5. R. Durbin and D. Willshaw, “An analogue approach to the travelling salesman problem using an elastic net method,” *Nature*, **326**, 1987, pp. 689–691.
6. R. Durbin, R. Szeliski, and A. Yuille, “An analysis of the elastic net approach to the traveling salesman problem,” *Neural Computation*, **1**(3), 1989, pp. 348–358.

# The stability of the Hopfield network

Assume that a Hopfield memory is created from  $m$  independent random vectors  $\mathbf{u}^1, \dots, \mathbf{u}^m$ , chosen uniformly over  $\{-1, +1\}^n$ . Each memory component,  $u_i^{(k)}$ , for  $i = 1, \dots, n$  and  $k = 1, \dots, m$  is consequently a random variable with probability distribution,

$$\mathbb{P} \left\{ u_i^{(k)} = 1 \right\} = \mathbb{P} \left\{ u_i^{(k)} = -1 \right\} = \frac{1}{2}.$$

Following the outer-product rule, the weights (which are also random), and bias, are computed as,

$$w_{i,j} = \sum_{p=1}^m \left( u_i^{(p)} u_j^{(p)} - \delta_{i,j} \right), \quad \text{and,} \quad \theta_i = 0. \quad (1)$$

We will now initialize the network to the state defined by the  $k$ -th memory,  $\mathbf{u}^k$ , and test the stability of the  $i$ -th unit: Letting

$$u'_i = \text{sgn} \left( \sum_{j=1}^n w_{i,j} u_j^{(k)} - \theta_i \right), \quad (2)$$

does  $u'_i = u_i^{(k)}$ , for  $k = 1, \dots, m$ , and  $i = 1, \dots, n$ . If so, then all  $m$  memories are said to be stable.

## The stability of the Hopfield network (cont.)

After combining Eqns. (1) and (2) from the previous slide,

$$\begin{aligned} u'_i &= \operatorname{sgn} \left( \sum_{j=1}^n \sum_{p=1}^m \left( u_i^{(p)} u_j^{(p)} - \delta_{i,j} \right) u_j^{(k)} - \frac{1}{2} \right), \\ &= \operatorname{sgn} \left( \sum_{j \neq i} \sum_{p=1}^m u_i^{(p)} u_j^{(p)} u_j^{(k)} \right), \end{aligned}$$

where the restriction on the first summation account for the zero-valued diagonal elements in the weight matrix. We now multiply both sides by  $u_i^{(k)}$  to obtain

$$\begin{aligned} u_i^{(k)} u'_i &= u_i^{(k)} \operatorname{sgn} \left( \sum_{j \neq i} \sum_{p=1}^m u_i^{(p)} u_j^{(p)} u_j^{(k)} \right), \\ &= \operatorname{sgn} \left( u_i^{(k)} \sum_{j \neq i} \sum_{p=1}^m u_i^{(p)} u_j^{(p)} u_j^{(k)} \right), \end{aligned}$$

as  $-\operatorname{sgn}(x) = \operatorname{sgn}(-x)$  if  $x \neq 0$ . The last condition is always valid if  $m(n-1)$  equals an odd integer. If this product is even, the condition  $x \neq 0$  could be enforced by setting  $\theta_i = 1/2$  above. We shall however finesse this technicality for the moment.

## Stability of the Hopfield network (cont.)

We now reorder the double summation, and treat the term resulting from  $p = k$  separately:

$$\begin{aligned} u_i^{(k)} u'_i &= \operatorname{sgn} \left( u_i^{(k)} \sum_{j \neq i} u_i^{(k)} u_j^{(k)} u_j^{(k)} + u_i^{(k)} \sum_{p \neq k} \sum_{j \neq i} u_i^{(p)} u_j^{(p)} u_j^{(k)} \right), \\ &= \operatorname{sgn} \left( Q_i^{(k)} \right), \end{aligned}$$

where,

$$Q_i^{(k)} = n - 1 + \sum_{p \neq k} \sum_{j \neq i} u_i^{(k)} u_i^{(p)} u_j^{(p)} u_j^{(k)}.$$

(We identify the  $n - 1$  term above as the *signal term*, and the second term (in red) as the *noise* or *crosstalk* term.) Now observe that the  $i$ -th component of the  $k$ -th memory will be correctly recalled if  $u'_i = u_i^{(k)}$ , or equivalently if  $u_i^{(k)} u'_i = 1$ . The latter will certainly occur if  $Q_i^{(k)} > 0$ . Consequently,

$$\mathcal{E}_0 \triangleq \bigcap_{k=1}^m \bigcap_{i=1}^n [Q_i^{(k)} > 0], \quad (3)$$

represents the event that all  $m$  memories,  $\mathbf{u}^1, \dots, \mathbf{u}^m$  are stable.

## Stability of the Hopfield network (cont.)

We now seek a relation between  $m$  and  $n$  that will ensure that event  $\mathcal{E}_0$  occurs with near certainty. That is given any  $\delta > 0$ , and  $n$ , we can choose  $m$  so that

$$\mathbb{P}\{\mathcal{E}_0\} \geq 1 - \delta.$$

From the definition of a probability measure (see the notes on *Probability Theory*),

$$\begin{aligned} \mathbb{P}\{\mathcal{E}_0\} &= 1 - \mathbb{P}\{\overline{\mathcal{E}_0}\} \\ &= 1 - \mathbb{P}\left\{\bigcup_{k=1}^m \bigcup_{i=1}^n \overline{[Q_i^{(k)} > 0]}\right\} && \text{DeMorgan's Law} \\ &= 1 - \mathbb{P}\left\{\bigcup_{k=1}^m \bigcup_{i=1}^n [Q_i^{(k)} \leq 0]\right\} \\ &\geq 1 - \sum_{k=1}^m \sum_{i=1}^n \mathbb{P}[Q_i^{(k)} \leq 0] && \text{Boole's inequality} \\ &= 1 - mn \mathbb{P}[Q_n^{(m)} \leq 0]. && \text{Symmetry over } i \text{ and } k. \end{aligned}$$

The last term in the final expressions can now be identified as  $\delta$ . We still need however to estimate  $\mathbb{P}[Q_n^{(m)} \leq 0]$ .

## The stability of the Hopfield network (cont.)

Recall that,

$$\begin{aligned} Q_n^{(m)} &= n - 1 + \sum_{p=1}^{m-1} \sum_{j=1}^{n-1} u_n^{(m)} u_n^{(p)} u_j^{(p)} u_j^{(m)}, \\ &= n - 1 + \sum_{\ell=1}^N X_\ell, \end{aligned}$$

where  $N = (m - 1)(n - 1)$ , and

$$X_\ell \triangleq u_n^{(m)} u_n^{(P(\ell))} u_{J(\ell)}^{(P(\ell))} u_{J(\ell)}^{(m)},$$

where for  $\ell = 1, \dots, N$ ,

$$J(\ell) \triangleq 1 + (\ell - 1) \pmod{n - 1} < n,$$

$$P(\ell) \triangleq 1 + \left\lfloor \frac{\ell - 1}{n - 1} \right\rfloor < m.$$

Since  $J(\ell) < n$  and  $P(\ell) < m$ , the factors of  $X_\ell$  are mutually independent, and thus,

$$\mathbb{P}\{X_\ell = 1\} = \mathbb{P}\{X_\ell = -1\} = \frac{1}{2}.$$

Are  $X_1, X_2, \dots, X_N$ , really independent? Explain.

## The stability of the Hopfield network (cont.)

Thus,

$$\begin{aligned}\mathbb{P} \left\{ Q_n^{(m)} \leq 0 \right\} &= \mathbb{P} \left\{ n - 1 + \sum_{\ell=1}^N X_\ell \leq 0 \right\} \\ &= \mathbb{P} \left\{ \sum_{\ell=1}^N X_\ell \leq -(n - 1) \right\}\end{aligned}\tag{4}$$

From the notes on *Probability Theory*, we learned that Chernoff's bound, for a symmetric random walk ( $\mathbb{P}\{X_\ell = 1\} = \mathbb{P}\{X_\ell = -1\} = 1/2$ ), with  $\epsilon > 0$ , yields for the left tail event,

$$\mathbb{P} \left\{ \sum_{\ell=1}^N X_\ell < -N\epsilon \right\} \leq e^{-\epsilon^2 N/2}.$$

Comparing the above event with that in Eqn. (4), yields,

$$\epsilon = \frac{n - 1}{N} = \frac{n - 1}{(m - 1)(n - 1)} = \frac{1}{m - 1}.$$

Consequently,

$$\mathbb{P} \left\{ Q_n^{(m)} \leq 0 \right\} = \mathbb{P} \left\{ \sum_{\ell=1}^N X_\ell \leq -(n - 1) \right\} \leq e^{-(n-1)/(2(m-1))} \approx e^{-n/(2m)}.$$

## The stability of the Hopfield network (cont.)

Thus

$$\mathbb{P}\{\mathcal{E}_0\} \leq 1 - mne^{-n/2m} = 1 - \delta.$$

Iterative substitution on the equation

$$mne^{-n/2m} = \delta \iff m = \frac{n}{2(\log m + \log n - \log \delta)}$$

yields,

$$m = \frac{n}{4 \log n} \left( 1 + \frac{\log \log n + \log 4\delta}{2 \log n} + \dots \right).$$

Consequently, for any  $\delta > 0$ , then for sufficiently large  $n$ , the probability of event  $\mathcal{E}_0$ , that all memories  $\mathbf{u}^1, \dots, \mathbf{u}^m$  are stable fixed points, satisfies

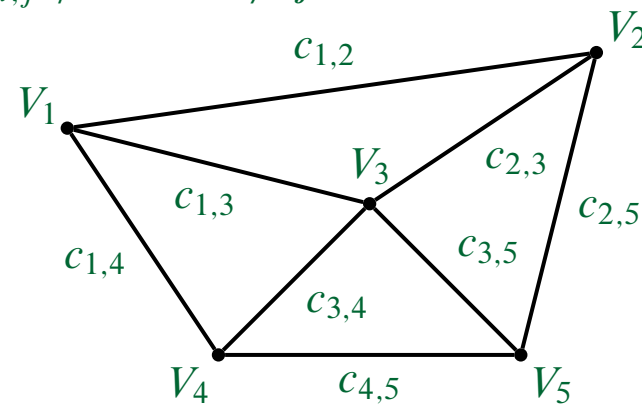
$$\mathbb{P}(\mathcal{E}_0) \geq 1 - \delta, \text{ if, } m < \frac{n}{4 \log n}.$$

McEliece et al., define the *asymptotic capacity* of the Hopfield networks as the *largest* rate of growth  $C(n)$ , such that if  $m < C(n)$ , then the probability of the event that every one of the fundamental (random) memories  $\mathbf{u}^1, \dots, \mathbf{u}^m \in \mathbb{B}^n$  is stable can be set as close to *one* as desired by choosing  $n$  to be sufficiently large. (Likewise, if  $m > C(n)$  then the probability of this event can be set as close to *zero* as desired.) Furthermore, they obtain  $C(n) = n/(4 \log n)$ . (They additionally study the of convergence of partially corrupted memories to their correct values.)

# The Traveling Salesman Problem

In the *Traveling Salesman Problem* (or TSP), one seeks the round-trip tour of minimal cost that visits each vertex in a connected graph exactly once. Formally a graph  $G = (\mathcal{V}, \mathcal{E})$  is designated by a pair of sets, where  $\mathcal{V} = \{V_1, \dots, V_N\}$  denotes the set of vertices, and  $\mathcal{E}$ , the set of edges of the form  $e_{i,j} = (V_i, V_j)$  i.e., the edge between vertices  $V_i$  and  $V_j$ . Each edge  $e_{i,j} \in \mathcal{E}$  is then assigned a cost  $c_{i,j} \geq 0$ . (Here we will assume that the edges are *not* directional, so that  $e_{i,j}$  and  $e_{j,i}$  are equivalent, and thus,  $c_{i,j} = c_{j,i}$ .) The square cost matrix is completed by assigning  $c_{i,i} = 0$ , and  $c_{i,j} = +\infty$  if  $e_{i,j} \notin \mathcal{E}$  and  $i \neq j$ .

$c_{i,j}$	1	2	3	4	5
1	0	$c_{1,2}$	$c_{1,3}$	$c_{1,4}$	$+\infty$
2	$c_{1,2}$	0	$c_{2,3}$	$+\infty$	$c_{2,5}$
3	$c_{1,3}$	$c_{2,3}$	0	$c_{3,4}$	$c_{3,5}$
4	$c_{1,4}$	$+\infty$	$c_{3,4}$	0	$c_{4,5}$
5	$+\infty$	$c_{2,5}$	$c_{3,5}$	$c_{4,5}$	0



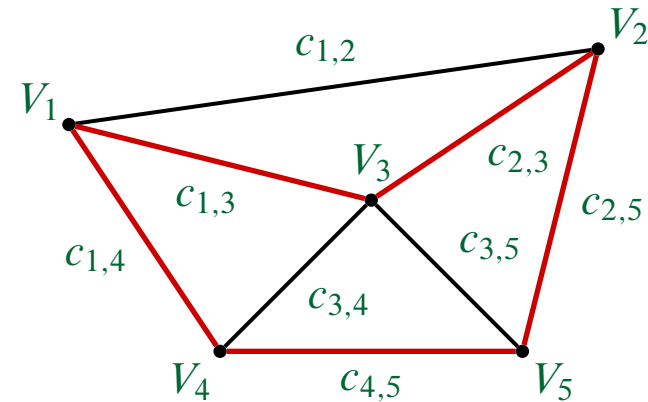
A tour that visits each vertex exactly once and returns to the initial vertex can be represented by the ordering  $\langle V_{\pi(1)}, V_{\pi(2)}, \dots, V_{\pi(N)} \rangle$ , where  $\pi = \langle \pi(1), \pi(2), \dots, \pi(N) \rangle$  denotes a permutation of the integers  $1, 2, \dots, N$ . The cost of this tour would thus be

$$C_{\pi} = \sum_{i=1}^{N-1} c_{\pi(i), \pi(i+1)} + c_{\pi(N), \pi(1)}.$$

# The Traveling Salesman Problem

The tour defined by the permutation  $\pi$  can also be represented by an incidence matrix in which each row represents a distinct vertex in  $\mathcal{V}$ , and each of the  $N$  columns represents a position in the ordering. Thus for example, a particular tour of length over  $\mathcal{V} = \{V_1, \dots, V_5\}$  can be represented as

Vertex	Epoch				
	1	2	3	4	5
$V_1$	1	0	0	0	0
$V_2$	0	0	1	0	0
$V_3$	0	1	0	0	0
$V_4$	0	0	0	0	1
$V_5$	0	0	0	1	0



Note that exactly one element in each row and column are set to 1, the rest to 0. The above matrix represents the tour  $V_1, V_3, V_2, V_5, V_4$  and then back to  $V_1$ , incurring a cost of  $c_{1,3} + c_{2,3} + c_{2,5} + c_{4,5} + c_{1,4}$ .

# Hopfield and Tank's Solution

First represent each matrix element in the incidence matrix by a bit  $v_{i,r} \in \{0, 1\}$  which equals 1 if the tour visits vertex  $V_i$  at time  $r$ , and 0 otherwise. Thus, a graph with  $N$  vertices, demands a network with  $n = N^2$  bits, with  $1 \leq i, r \leq N$ .

Vertex	Epoch				
	1	2	3	4	5
$V_1$	$v_{1,1}$	$v_{1,2}$	$v_{1,3}$	$v_{1,4}$	$v_{1,5}$
$V_2$	$v_{2,1}$	$v_{2,2}$	$v_{2,3}$	$v_{2,4}$	$v_{2,5}$
$V_3$	$v_{3,1}$	$v_{3,2}$	$v_{3,3}$	$v_{3,4}$	$v_{3,5}$
$V_4$	$v_{4,1}$	$v_{4,2}$	$v_{4,3}$	$v_{4,4}$	$v_{4,5}$
$V_5$	$v_{5,1}$	$v_{5,2}$	$v_{5,3}$	$v_{5,4}$	$v_{5,5}$

There are  $2^{N^2}$  states of the matrix, but only  $N! \approx N^{N+1/2} e^{-N} \sqrt{2\pi}$  valid tours. We require exactly one active unit in each column and in each row. Thus, we construct,

$$E_C = \frac{A}{2} \sum_{i=1}^N \sum_{r=1}^N \sum_{s \neq r} v_{i,r} v_{i,s} + \frac{B}{2} \sum_{r=1}^N \sum_{i=1}^N \sum_{j \neq i} v_{i,r} v_{j,r} + \frac{C}{2} \left( \sum_{i=1}^N \sum_{r=1}^N v_{i,r} - N \right)^2 \geq 0,$$

where  $A$ ,  $B$ , and  $C$  are positive coefficients. Verify that every state of the matrix that defines a valid tour, satisfies  $E_C = 0$ .

## Hopfield and Tank's Solution (cont.)

To distinguish between the  $N!$  valid tours, we introduce

$$E_T = \frac{D}{2} \sum_{i=1}^N \sum_{j=1}^N \sum_{r=1}^N c_{i,j} v_{i,r} (v_{j,r+1} + v_{j,r-1}),$$

where sums are taken modulo  $N$ . (Note that  $E_T$  equals the total cost of the tour.)

Now let

$$E(\{v_{i,r}\}) = E_C + E_T = -\frac{1}{2} \sum_{i=1}^N \sum_{r=1}^N \sum_{j=1}^N \sum_{s=1}^N v_{i,r} w_{i,r;j,s} v_{j,s} + \sum_{i=1}^N \sum_{r=1}^N \theta_{i,r} v_{i,r} + E_0$$

where,

$$\begin{aligned} w_{i,r;j,s} &= -A\delta_{i,j}(1 - \delta_{r,s}) - B(1 - \delta_{i,j})\delta_{r,s} - C - Dc_{i,j}(\delta_{s,r+1} + \delta_{s,r-1}) \\ \theta_{i,r} &= -CN \\ E_0 &= \frac{CN^2}{2}. \end{aligned}$$

N.B.,  $w_{i,r;j,s} = w_{i,r;j,s}$  and  $w_{i,r;i,r} = -C$ , for all  $1 \leq i, r, j, s \leq N$ .

Thus the above can be simulated using a Hopfield network with the above parameters, assuming a threshold function  $H(x) \triangleq I_{[0,+\infty)}(x) = (\text{sgn}(x) + 1)/2$ . Alternatively, let  $v_{i,r} = (u_{i,r} - 1)/2$ , and derive new weights and biases. Random restarts, vs. stochastic relaxation.

## Digression: Elastic Net Algorithm

Here we restrict the Traveling Salesman problem to vertices that lie on a two-dimensional plane, and let  $\mathbf{V}_i \in \mathbb{R}^2$  denote the *position* of vertex  $V_i$ , for  $i = 1, \dots, N$ . We also assume that the cost matrix  $c_{i,j} = \|\mathbf{V}_i - \mathbf{V}_j\|$  is determined by the Euclidean distance between pairs of vertices.

An *elastic net* is defined to be a closed deformable loop, in which  $M > N$  nodes are embedded, like beads on an elastic bracelet. Let  $\mathbf{Y}_j \in \mathbb{R}^2$ , for  $j = 1, \dots, M$ , denote the current positions of these nodes. This algorithm is designed so that the path defined by the ordered sequence of nodes  $\langle \mathbf{Y}_1, \dots, \mathbf{Y}_M \rangle$ , will define the tour. (Since  $M > N$  it is possible, and indeed desirable, for degeneracies among the nodes.)

Nodes are updated according to the rule,  $\mathbf{Y}_j(t + 1) = \mathbf{Y}_j(t) + \Delta\mathbf{Y}_j(t)$ , where

$$\Delta\mathbf{Y}_j(t) = \alpha \sum_{i=1}^N w_{i,j} (\mathbf{X}_i - \mathbf{Y}_j(t)) + \beta K (\mathbf{Y}_{j+1}(t) - 2\mathbf{Y}_j(t) + \mathbf{Y}_{j-1}(t)),$$

and,

$$w_{i,j} = \frac{e^{-\|\mathbf{X}_i - \mathbf{Y}_j\|^2/2K^2}}{\sum_{k=1}^M e^{-\|\mathbf{X}_i - \mathbf{Y}_k\|^2/2K^2}},$$

and  $\alpha$  and  $\beta$  are positive constants, and  $K > 0$  is a scale parameter.

## Elastic Net Algorithm (cont.)

The algorithm can be described as optimization by gradient descent,

$$\mathbf{Y}_j = -K \frac{\partial E}{\partial \mathbf{Y}_j},$$

with respect to the energy function,

$$E(\mathbf{Y}_1, \dots, \mathbf{Y}_M) = -\alpha K \sum_{i=1}^N \log \left( \sum_{j=1}^M e^{-\|\mathbf{x}_i - \mathbf{Y}_j\|^2 / 2K^2} \right) + \beta \sum_{j=1}^M \|\mathbf{Y}_j - \mathbf{Y}_{j+1}\|^2.$$

The last term represents Hooke's law for the potential energy an elastic spring,  $\frac{1}{2}kx^2$ .