Hopfield Networks

Thomas Quella (Thomas.Quella@unimelb.edu.au)

1 Overview and Summary

In this note we sketch a simple class of neural networks that was proposed by Hopfield in 1982 [1] and that received a lot of attention since then.[1] This note is mainly based on Chapter 42 of Ref. [2] and Chapter 13 of Ref. [3].

Hopfield networks can be thought of as "ensembles of computing units" [3] with complex bidirectional interactions and this makes them amenable to methods of statistical physics. This connection leads to the technically and conceptually important use of an energy function in describing the behaviour of Hopfield networks. From a biological perspective, Hopfield networks realize the paradigm that no global synchronization of computing elements is required. The Hopfield network still fulfills its task if the individual units perform their calculations asynchronously (i.e. one by one in random order) like in the brain. This also makes the model attractive from the perspective of massive parallelization. From a conceptual perspective, Hopfield networks are interesting since they are examples of feedback networks, i.e. the output feeds back again as inputs. For pedagogical reasons, we will mostly restrict ourselves to what is called the binary case (with two possibilities for every input/output) in this talk and simply state results for the continuous case where appropriate.

Hopfield networks are early examples of associative (content-addressable) memories according to the following

Informal Definition. *An associative memory is a function that takes a pattern as an input and returns a pattern. Memories are patterns that are stable, i.e. attractive fixed points under iteration of the function. This means that memories that are slightly corrupted can still be retrieved correctly.*

Remarks.

- *Note the difference to an address-based memory (in ordinary computers) where all information is lost if the storage location of the memory is lost.*
- *Due to the stability property, associative memories may be used for pattern completion and error correction, provided the missing/wrong part is sufficiently small.*

A schematic view of an associative memory explaining these properties is provided in Figure 1. Hopfield networks can also be used to solve certain types of optimization problems.

2 From single neurons to a neural network

General aspects. In every neural network there are four basic points to consider, namely

1. the *architecture*, i.e. how neurons are connected
2. the *activity rule*, i.e. how neurons respond to activation (stimuli)
3. the *learning rule*, i.e. how the neural network adapts its behaviour over time

An illustration of some basic networks can be found in Figure 2.

[1] The paper is cited more than 20 000 times.

Memories to be stored

```
moscow--------russia
lima------------peru
london-------england
tokyo----------japan
paris---------france
berlin-------germany
canberra---australia
```

Pattern completion and error correction

Input	**Output**
`berlin-:::::::::::::`	`berlin-------germany`
`::::::::::-australia`	`canberra---australia`
`tokio----------yapan`	`tokyo----------japan`
`london------scotland`	`london-------england`

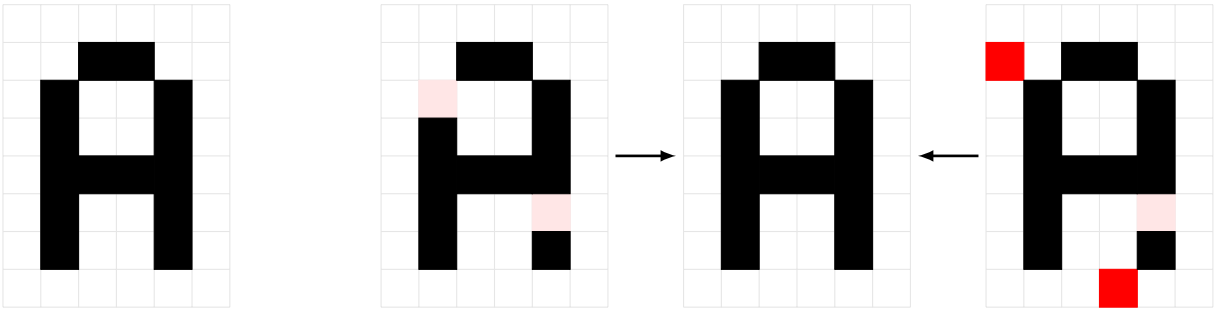

Figure 1: The basic working of an associative memory: Storage of memories, pattern completion and error correction. Examples adapted from [2].

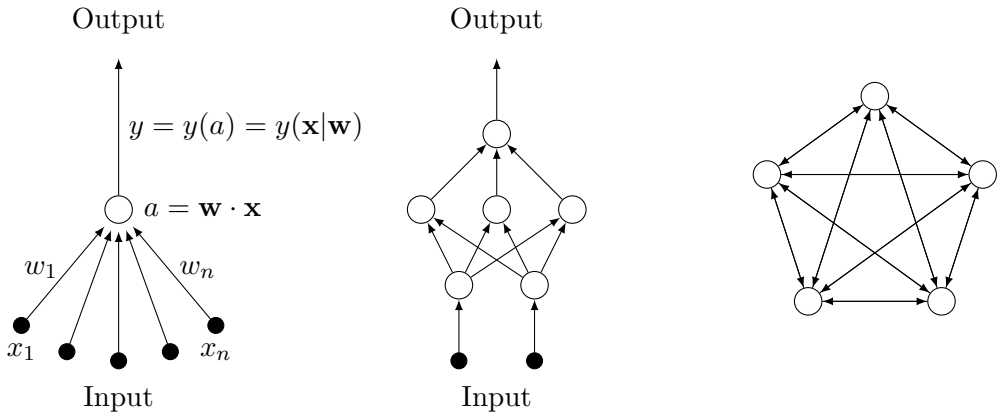

Figure 2: The mathematical model for a single neuron, a feedforward network, a feedback network (here: a Hopfield network).

Single neurons. A single neuron can be modelled as a function taking a number of inputs $\mathbf{x} = (x_1, \ldots, x_n)$ to a single output $y = y(\mathbf{x}|\mathbf{w})$, also known as the *activity*. The function $y(\mathbf{x}|\mathbf{w})$ is generally non-linear (in both $\mathbf{x}$ and $\mathbf{w}$) and depends on *weights* $\mathbf{w} = (w_1, \ldots, w_n)$ that are used to weight the importance of different inputs. It is custom to assume that all the variables x_i and the function y take values in $\{\pm 1\}$ for a discrete model and the interval $[-1, 1]$ for a continuous model. Alternatively, one may work with the set $\{0, 1\}$ and the interval $[0, 1]$. The first choice is inspired by physics (spin up/down), the second by computer science (bit on/off).

The architecture of a single neuron is displayed in Figure 2. The activity rule for a single neuron consists of two steps

1. Calculate the neuron's *activation* $a = \sum_i w_i x_i = \mathbf{w} \cdot \mathbf{x}$.
2. Calculate the neuron's *activity* $y = y(a) = y(\mathbf{x}|\mathbf{w})$ as a function of the activation a.

Sometimes a special weight w_0 is included which is known as the *bias* (since it is, by definition, associated with a constant activity $x_0 = 1$). In practice, the activity function y is taken from one of the classes depicted in Figure 3, depending on the precise nature of the neuron's possible activity values.

Remark. *It is essential that the activity function $y(a)$ is* non-linear *since this is what enables networks of several layers to be more powerful than a single layer.*

Remark. *Single neurons have a* capacity *of two bits per weight. This corresponds to the number of patterns that can be stored in a single neuron (by adjusting the weights $\mathbf{w}$).*

Neural networks. If neurons are connected to each other, with outputs of individual neurons used again as inputs for others (or themselves), one speaks about a neural network. Neural networks come in various forms and architectures, depending on the precise problems they are meant to address. In what follows, we first give a brief outline of some of the basic characteristics and then focus on the exploration of a special type of network, the Hopfield network.

Definition. *A* feedforward network *is a network whose underlying graph is directed and acyclic. A* feedback network *is a network that is not a feedforward network.*

A feedback network thus has cycles. These may occur in the form of cycles of directed connections (including self-connections) or undirected (bidirectional) connections.

Definition. *A Hopfield network is a neural network which is* fully connected *through symmetric, bidirectional connections described by weights $w_{ij} = w_{ji}$ (where i and j refer to the neurons). Self-connections are forbidden, i.e. $w_{ii} = 0$.*

As we will explain in the following, Hopfield networks can be used as an associative memory and in optimization problems. There are close relationships to the physics of the Ising model and, in fact, one of the most important notions in the context of a Hopfield model is that of an energy function. For later convenience we define the weight matrix $W = (w_{ij})$ consisting of row vectors $\mathbf{w}_i = (w_{ij})$.

In order to analyze the dynamics of a Hopfield network we need to specify an activity rule which evolves a current state $\mathbf{x}$ of activities to a new one after one unit of time. There are essentially two possibilities, synchronous or asynchronous updates.[2]

[2]One could also consider models with continuous time but these are beyond the scope of this introduction.

Asynchronous updates

1. Choose a neuron i
2. Compute its activation $a_i = \sum_j w_{ij} x_j = \mathbf{w}_i \cdot \mathbf{x}$
3. Determine its (new) activity $x_i = \mathrm{sgn}(a_i)$ (discrete case) or $x_i = \tanh(a_i)$ (continuous case)
4. Repeat

The neuron may be chosen at random or following a fixed sequence.[3] Asynchronous updates only change a single component of $\mathbf{x}$ at a time.

Synchronous updates

1. Compute all activations $a_i = \sum_j w_{ij} x_j = \mathbf{w}_i \cdot \mathbf{x}$
2. Determine all (new) activities $x_i = \mathrm{sgn}(a_i)$ (discrete case) or $x_i = \tanh(a_i)$ (continuous case)
3. Repeat

In this case all components of $\mathbf{x}$ are updated at the same time.

Besides the updates used above, one could use various types of stochastic updates that are based on Monte-Carlo ideas.

Remark. *If the diagonal is not fully zero there is not necessarily a stable state (fixed point), e.g. $W = -\mathbb{I}$ generates all possible states (in the asynchronous case) since at each time step one (random) component is flipped. Similarly, there may be non-trivial orbits if W is not symmetric.*

Remark. *A sequence of asynchronous updates of neurons $1, \ldots, N$ is generally not equivalent to one synchronous update.*

In the presence of non-trivial bias, it is convenient to treat the weights w_{0i} and w_{i0} on a different footing since the associated activities $x_0 = 1$ are never updated. We set $w_{00} = 0$ as well as $\mathbf{w}_0 = \theta$ use this to write the matrix W in block form

$$W = \begin{pmatrix} w_{00} & w_{01} & \cdots & w_{0n} \\ w_{10} & w_{11} & \cdots & w_{1n} \\ & & & \\ w_{n0} & w_{n1} & \cdots & w_{nn} \end{pmatrix} = \begin{pmatrix} 0 & \theta \\ \theta^T & \mathbf{W} \end{pmatrix} . \tag{1}$$

The update rules now read (by abuse of notation)

$$a_i = \mathbf{w}_i \cdot \mathbf{x} - \theta_i . \tag{2}$$

When calculating the activity $y(a_i)$ we see that the (negative of the) bias essentially functions as a threshold value for the neuron to trigger. We call θ the threshold vector.

3 The energy function and convergence

Definition. *Consider a discrete Hopfield network with weight matrix $\mathbf{W}$ and threshold vector θ. The* energy functional *associated with this network is*

$$E(\mathbf{x}|W, \theta) = -\frac{1}{2}\mathbf{x}^T \mathbf{W} \mathbf{x} + \theta \cdot \mathbf{x} . \tag{3}$$

[3] In both cases it needs to be made sure that all neurons are chosen an infinite number of times to ensure convergence (of pme).

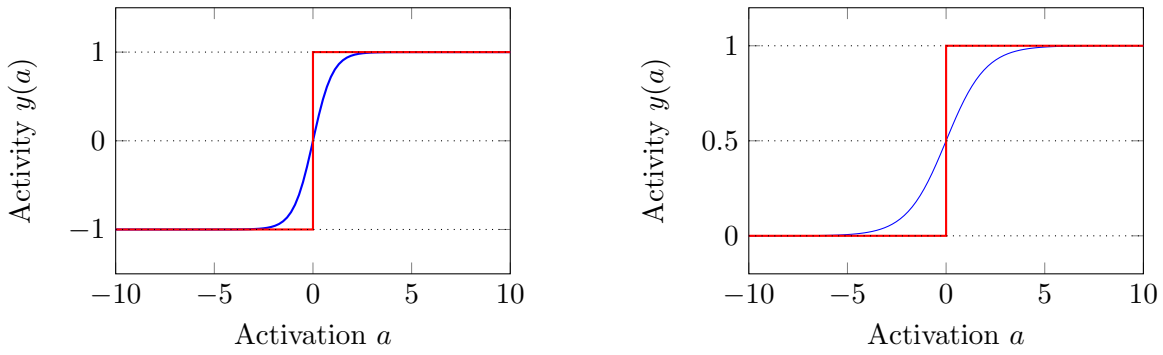

Figure 3: Different types of deterministic activity functions for individual neurons: (a) Sigmoid function $y(a) = \tanh(a)$ and sign function $y(a) = \mathrm{sgn}(a)$. (b) Sigmoid function $y(a) = 1/(1 - e^{-a})$ and step function $y(a) = H(a)$.

Proposition. *A Hopfield network with n units and asynchronous dynamics, which starts from any given network state, eventually reaches a stable state at a local minimum of the energy function.*

Proof. We will show that the energy strictly dicreases under each non-trivial update. Since the energy is bounded from below and there are only a finite number of configuration this process will end in a local minimum at some point where no flip of a single neuron's state leads to a lower energy. Let us imagine that neuron k is chosen for an update. If the activation $a_k = \mathbf{w}_k \mathbf{x} - \theta_k$ has the same sign as x_k there is no non-trivial update. Otherwise the neuron will flip to $x'_k = -x_k$ (while all others retain their value) and we may assume that a_k, x'_k and $-x_k$ all have the same sign. We then calculate (using the symmetry $w_{ij} = w_{ji}$)

$$\begin{aligned}
\Delta E &= E(\mathbf{x}'|W,\theta) - E(\mathbf{x}|W,\theta) \\
&= -\frac{1}{2}\sum_{i,j} x'_i w_{ij} x'_j + \sum_i \theta_i x'_i + \frac{1}{2}\sum_{i,j} x_i w_{ij} x_j - \sum_i \theta_i x_i \\
&= -\sum_j x'_k w_{kj} x'_j + \sum_j x_k w_{kj} x_j + \theta_k (x'_k - x_k) \; . \qquad (4)
\end{aligned}$$

With $w_{kk} = 0$ and $x'_j = x_j$ for $j \neq k$ this simplifies to

$$\Delta E = -(x'_k - x_k)\Big[\sum_j w_{kj} x_j - \theta_k\Big] = -2 x'_k a_k < 0 \; . \qquad (5)$$

Hence the energy decreases and this decrease is strict whenever $a_k \neq 0$. □

Remark. *We are safe to assume $a_k \neq 0$ in the previous proof by assuming that the neuron keeps its current state if $a_k = 0$ (even though this is not quite what is expressed by the sign function originally).*

Remark. *The previous Proposition relies on the asynchronicity of updates, the absence of self-connections and the fact that weights are symmetric. If any of these conditions is violated, convergence will not hold for general starting states.*

Remark. *A similar statement holds for the continuous case. In this case the energy function is more complicated and reads*

$$E(\mathbf{x}|W,\theta) = -\frac{1}{2}\mathbf{x}^T \mathbf{W} \mathbf{x} - \sum_i H_2^{(e)}\big(\frac{1}{2}(1 + x_i)\big) \; , \qquad (6)$$

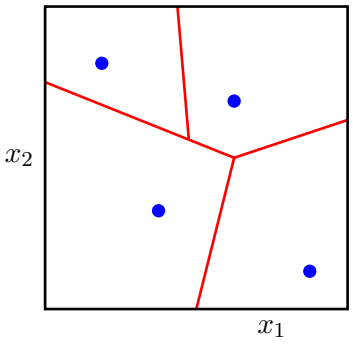

Figure 4: Schematic picture of the dynamics of a Hopfield network. Depicted is the space of possible states, together with fixed points (stable and unstable) and regions of attraction.

where the extra term is given by

$$H_2^{(e)}(q) = -q \log q - (1-q) \log(1-q) \ . \tag{7}$$

It is an amusing fact that this is just the mean field free energy for the associated $\mathbb{Z}_2$ Ising spin model. In that case x_i is not the fundamental spin variable but rather interpreted as the expected magnetization which can take arbitrary values in the interval $[-1, 1]$.

In the theory of general dynamical systems a real-valued function defined on the space of states that is bounded below and always decreases (or stays constant) under the time evolution is called *Lyapunov function.*

Proposition. *If a dynamical system has a Lyapunov function then its dynamics are bound to settle down to a fixed point (which is a local minimum) or a limit cycle (on which the function is constant).*

Remark. *A Lyapunov function divides the state space into different* basins of attraction*, one for each attractive fixed point.*

4 Learning and stability

Training a neural network amounts to finding the values of $\mathbf{w}$ that approximate a desired function best. From a general perspective this can be interpreted as a search in weight space with a suitable value function, e.g. through fixed point iteration using gradients.

In the case of a Hopfield network, the goal of the learning process is to make a set of given states $\mathbf{x}^{(\sigma)}$ stable states of the network (i.e. fixed points under iteration). Instead of using more sophisticated methods, it is common to simply set

$$w_{ij} = \eta \sum_\sigma x_i^{(\sigma)} x_j^{(\sigma)} \ , \tag{8}$$

where η is an arbitrary constant. This is known as Hebbian learning [4]. We will discuss below in which sense this is a good or bad choice. It should be noted though that there are other methods that have a higher degree of reliability. Hebbian learning works best if the memory states that are meant to be learned are nearly orthogonal.

Remark. *One can set $\eta = 1/N$ to prevent the weights from growing with the number of memories.*

Let us briefly discuss possible reasons for failure to store the desired memories:

- Memories are corrupted, i.e. the stable state of the network is slightly off
- Memories might be missing, either because there is no stable fixed point for them or the region of attraction is too small to be useful
- There may be additional fixed points, either corresponding to undesired memories or to variations thereof (mixes, reflections, ...)

Let us consider the problem of how many randomly chosen binary patterns can be stored in a Hopfield network with n neurons. The weights have been described in Eq. (8). Choosing a memory $x^{(l)}$ as the start configuration we can express the weights (for $i \neq j$) as

$$w_{ij} = x_i^{(l)} x_j^{(l)} + \sum_{m(\neq l)} x_i^{(m)} x_j^{(m)} \tag{9}$$

and this gives rise to the activations (since $x_j^{(l)} x_j^{(l)} = 1$)

$$a_i = \sum_{j(\neq i)} w_{ij} x_j^{(l)} = \sum_{j(\neq i)} x_i^{(l)} x_j^{(l)} x_j^{(l)} + \sum_{j(\neq i)} \sum_{m(\neq l)} x_i^{(m)} x_j^{(m)} x_j^{(l)} = (n-1) x_i^{(l)} + \text{noise} \,. \tag{10}$$

The bit i is stable if a_i has the same sign as $x_i^{(l)}$, i.e. if the noise term is sufficiently small. A closer inspection shows that, for large n and N, a_i is a Gaussian random variable with mean $I x_i^{(l)}$ and variance IN. If we try to store $N \approx 0.18n$ patterns in the Hopfield network there is a 1% chance that a specified bit is flipped. Using methods of statistical physics it has been analyzed under which conditions the memories are stable under multiple iterations [5]. In this case one can store up to $0.138n$ patterns, each with an error rate of 1.6%. Generally, there will be competition with random stable states that have lower energies. This ceases to be the case for $N < 0.05n$, i.e. then the desired memories are the lowest energy states.

A statistical analysis concerning the storage of N randomly distributed patterns [2, Exercise 42.7] gives the following

Proposition. *Let us assume that the desired patterns are completely stable (no bit-flip occurs), with a total error probability of less than $\epsilon \ll 1$. In this case the maximum number of patterns that can be stored in a Hopfield network with n neurons and Hebbian learning is*

$$N_{max} \sim \frac{n}{4\ln(n) + 2\ln(1/\epsilon)} \tag{11}$$

The number increases if we allow a small amount of corruption of individual memories to occur.

5 Selected applications of Hopfield networks

N rooks problem

Problem. *Place N rooks on a $N \times N$ chessboard such that no figure can take another.*

We label the positions on the board by a pair (i, j) and define $x_{ij} = 1$ if a rook is located at (i, j) and $x_{ij} = -1$ if not. A solution will need to have precisely one rook per row and column. The formulation is more transparent when working with binary numbers $n_{ij} = \frac{1}{2}(1 + x_{ij})$. The number of rooks in row i and the number of rooks in column i is

$$N_i^r = \sum_j n_{ij} \quad \text{and} \quad N_i^c = \sum_j n_{ji} \tag{12}$$

The idea is to give configurations that do not satisfy the desired conditions a high energy compared to the desired ones and hence a natural energy function for this problem is

$$E_{\text{rooks}} = \sum_i (N_i^r - 1)^2 + \sum_i (N_i^c - 1)^2 \,. \tag{13}$$

The energy vanishes if and only if $N_i^r = N_i^c = 1$ for all i. Since $E \geq 0$, this function thus has global minima at the desired locations. Obviously, one can easily determine the associated matrix W and threshold vector θ (thresholds are all -1 and weights are -2 along rows and columns and 0 otherwise).

Traveling salesman problem (TSP)

Problem. *Given a set of N cities together with their mutual distances d_{ij} find the shortest tour that visits each city precisely once.*

We use one index for the city and one for its position in the tour. In other words, $n_{ij} = 1$ if city i is the j^{th} destination visited and $n_{ij} = 0$ if not (where $n_{ij} = \frac{1}{2}(1 + x_{ij})$ as previously). This interpretation relies on having a "valid configuration" with the matrix (n_{ij}) only having a single 1 in each row and column and the other entries being 0. This can be enforced in the same way as in the N rooks problem.

On the other hand, the weights also need to encode the cost function that we try to minimize. We thus need to have another term in the energy function which, for a valid tour, is proportional to the total distance. This can be achieved by

$$E_{TSP} = E_{\text{rooks}} + \xi \sum_{i \neq j} d_{ij} \sum_l n_{il} n_{jl+1} \tag{14}$$

The parameter ξ may be used to tune the relative importance of "having a valid tour" and "minimizing the distance". For large problems there tend to be local minima that do not correspond to valid tours. This can be rectified but the presentation of the corresponding methods is beyond the scope of this introduction. A detailed analysis of the performance of the Hopfield network approach to the Travelling Salesman Problem can be found in [6].

The Traveling Salesman Problem illustrates a general issue faced in optimization: Complex combinatorial problems produce an energy function with an exponentially large number of local minima. In the physics literature such systems are known under the name "spin glass".

References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences 79* (1982) 2554–2558, `https://www.pnas.org/content/79/8/2554.full.pdf`.

[2] D. J. C. MacKay, *Information Theory, Inference & Learning Algorithms.* Cambridge University Press, New York, NY, USA, 2002.

[3] R. Rojas, *Neural Networks: A Systematic Introduction.* Springer-Verlag, Berlin, Heidelberg, 1996.

[4] D. Hebb, *The organization of behavior: A neuropsycholocigal theory.* John Wiley & Sons Inc, 1949.

[5] D. J. Amit, H. Gutfreund, and H. Sompolinsky, "Storing infinite numbers of patterns in a spin-glass model of neural networks," *Phys. Rev. Lett. 55* (1985) 1530–1533.

[6] S. V. Aiyer, M. Niranjan, and F. Fallside, "A theoretical investigation into the performance of the hopfield model," *IEEE transactions on neural networks 1* (1990) 204–215.