An introduction to memory and optimisation with Hopfield networks

Ramón Nartallo-Kaluarachchi[1,2]

[1]Mathematical Institute, University of Oxford
[2]Centre for Eudaimonia and Human Flourishing, University of Oxford
ramon.nartallo-kaluarachchi@maths.ox.ac.uk

Abstract

A broadening project for the EPFL course on Statistical physics for optimization and learning (Zdeborova and Krzakala).

In this mini-project, I will explore the area of Hopfield neural networks and discuss their properties in theory, with computational simulations, and with applications to optimisation problems.

Introduction

The study of artificial neural networks with computational capabilities has long been paired with the study of biological circuits, in particular the human brain. The first such study was the work of McCulloch and Pitts who first proposed the idea of a mathematical/computational neural network, modelled after the circuitry of the human brain (17). Their work inspired the development of perceptrons (18) and eventually the paradigm of 'deep learning' (14). As part of this evolution, machine learning has diverged from its neuroscientific roots abandoning many interesting directions in the search for predictive power.

One such avenue is the study of a class of artificial neural networks stemming from statistical physics but with interesting links to the modelling of associative memory in the human brain. Proposed independently by Amari (2) and Little (15), but popularised by Hopfield in his seminal paper (8), are the so-called 'Hopfield' or 'attractor' neural networks.

Hopfield networks

Training

A Hopfield network is a fully connected network of binary (or Ising) units which play the role of neurons. The weights of the network can be trained in such a way to encode persistent attractive states or 'memories'.

Consider a network of N neurons with pairwise interactions given by the weight matrix $\boldsymbol{W}$ where W_{ij} represents the connectivity between neurons i and j. The state of such a system is given by a $N-$dimensional vector with values in $\{\pm 1\}$ (respectively $\{0, 1\}$ for a binary system, but we continue assuming Ising variables).

Given a state $\boldsymbol{v}$, a 'memory', that one wishes to encode, the network can be trained according to a very simple *Hebbian* learning rule,

$$W_{ij} = v_i v_j, \tag{1}$$

with diagonal entries, W_{ii}, set to 0. Once trained, the weights remain fixed. Given some initial state, $\boldsymbol{v}^{[0]}$, the neural network now evolves according to the sequential update rule,

$$v_i \to \begin{cases} 1 & \text{if } \sum_j W_{ij} v_j \geq \theta_i \\ -1 & \text{if } \sum_j W_{ij} v_j < \theta_i \end{cases}, \tag{2}$$

where θ_i is the threshold of the neuron i. The term $\sum_j W_{ij} v_j$ can be considered to be the total input into the neuron i from its neighbours.

The Hebbian learning rule enforces that if two neurons agree in the memory, then $W_{ij} > 0$ and thus $W_{ij} v_j$ will push neuron i towards agreement with v_j. If $W_{ij} < 0$, then the opposite will occur.

Energy

Here we define the 'energy' of the network to be,

$$E = -\frac{1}{2} \sum_{i,j} W_{ij} v_i v_j - \sum_i \theta_i v_i. \tag{3}$$

By writing this in the form,

$$E = -\frac{1}{2} \sum_i v_i (\sum_j W_{ij} v_j - \theta_i), \tag{4}$$

we can see that every neuronal update will decrease our energy function. Iterating the update rule will cause our network to converge to a local energy minimum. As a result,

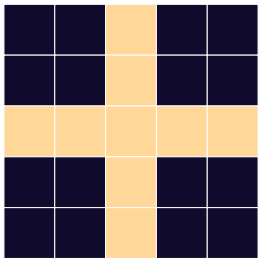

Figure 1: **A 25 neuron memory.** Arranged in a 5×5 grid, we can encode simple pictures as memories in our Hopfield network.

our encoded memory becomes a stable state of the system.

It is important to note that this energy function is precisely the energy function of a spin configuration for the ubiquitous Ising model in statistical physics, drawing an important equivalence between the behaviour of a Hopfield network and spin-glass systems (3).

In addition, we note that the encoded memory is not the only stable state. Classical results by Sherrington and Kirkpatrick showed that such Ising models, with random symmetric coupling, have a plethora of stable states (12).

Basic implementation and simulations

In order to implement this model, we make a few simple choices. Firstly, following convention, we set the thresholds of each neuron to be 0. Next, we can select between asynchronous and synchronous updating.

Asynchronous updating is the more biologically plausible option where, at every step, a single neuron is chosen to be updated, either randomly or in a given order. Conversely, neurons can be updated simultaneously.

We proceed with asynchronous updates, updating a neuron chosen uniformly at random.

Learning a simple pattern

For simplicity of visualisation, we consider 25 neurons arranged in a 5×5 grid where memories can be simple binary 'pictures'.

The first example we consider is a simple 'cross' memory, as shown in Fig. 1. We can run the network from different initial conditions and assess whether, or how fast, it converges.

As shown in Fig. 3, the proximity of the initial state to the memory partially determines if the system converges to the correct pattern, and how fast. We can see that if we start close to the memorised pattern, the system converges quickly, whereas if we start slightly further away, this takes longer. However, if we start sufficiently far away from the memory, we can converge to a different stable state. In the language of dynamical systems, if we initialise the system within the 'basin of attraction' of some stable state, then we will eventually converge to it.

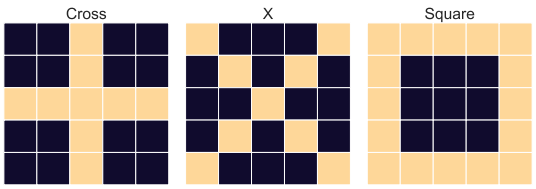

Figure 2: **Encoding multiple memories**. Here we consider three patterns that we can encode as memories simultaneously in our Hopfield network.

In this case, the 'inverted cross' happens to be stable. These non-memorised stable patterns are called 'spurious states'. One such spurious pattern is the negation of the encoded memory. We can show this by noting that a state, $\boldsymbol{u}$ is stable if the update causes no change i.e.

$$u_i = \sum_j W_{ij} u_j, \tag{5}$$

$$u_i = \sum_j v_i v_j u_j, \tag{6}$$

where $\boldsymbol{v}$ is the memory (assuming 0 threshold). We notice that $u_i = -v_i$ satisfies this condition.

Learning multiple patterns

A Hopfield network is capable of encoding more than a single memory. In order to encode multiple patterns, we modify training such that the weights are now set according to,

$$W_{ij} = \frac{1}{k} \sum_{l=1}^{k} v_i^{[l]} v_j^{[l]}, \tag{7}$$

where $v^{[1]}, ..., v^{[k]}$ are k memories to be encoded in the network.

We illustrate this by encoding three patterns in our Hopfield network. The patterns: 'Cross', 'X' and 'Square' are shown in Fig. 2 and are encoded in the network simultaneously.

Fig. 4 shows trajectories from the multi-pattern network converging to each of the three memories or a spurious stable state from different initial patterns. Again, we see that a negated pattern is a spurious stable state. In general, any linear combination of an odd number of memories is a stable pattern (7).

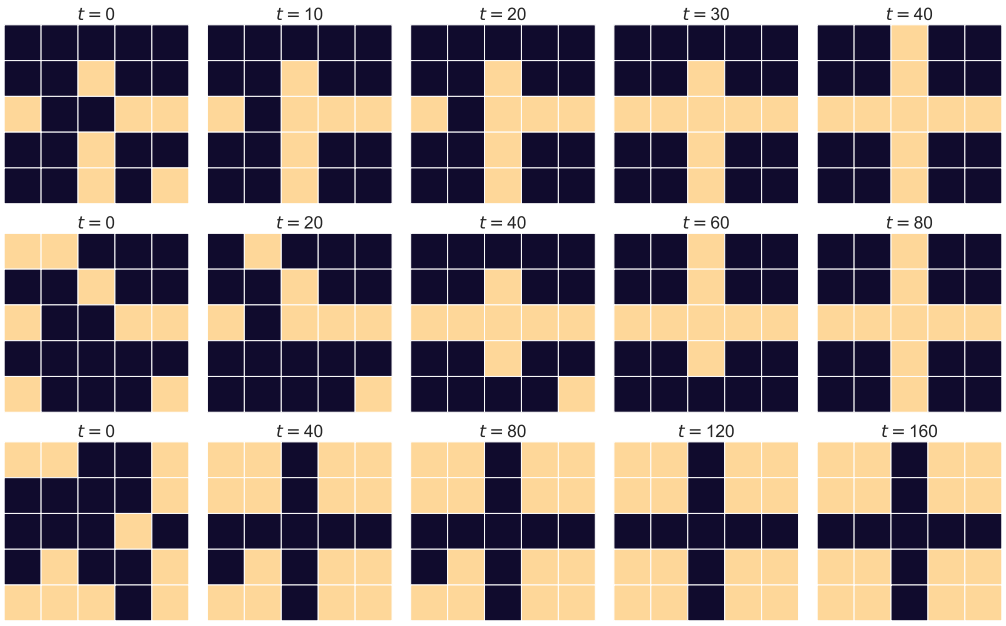

Figure 3: **Trajectories of the 'cross' network:** *Row 1:* As the initial state is close to the encoded memory, the system converges quickly to the cross. *Row 2:* As the initial state is further from the encoded memory, the system takes longer to eventually converge. *Row 3:* As our initial state was too different from the cross, we do not converge to the encoded memory, but instead, to the another stable state.

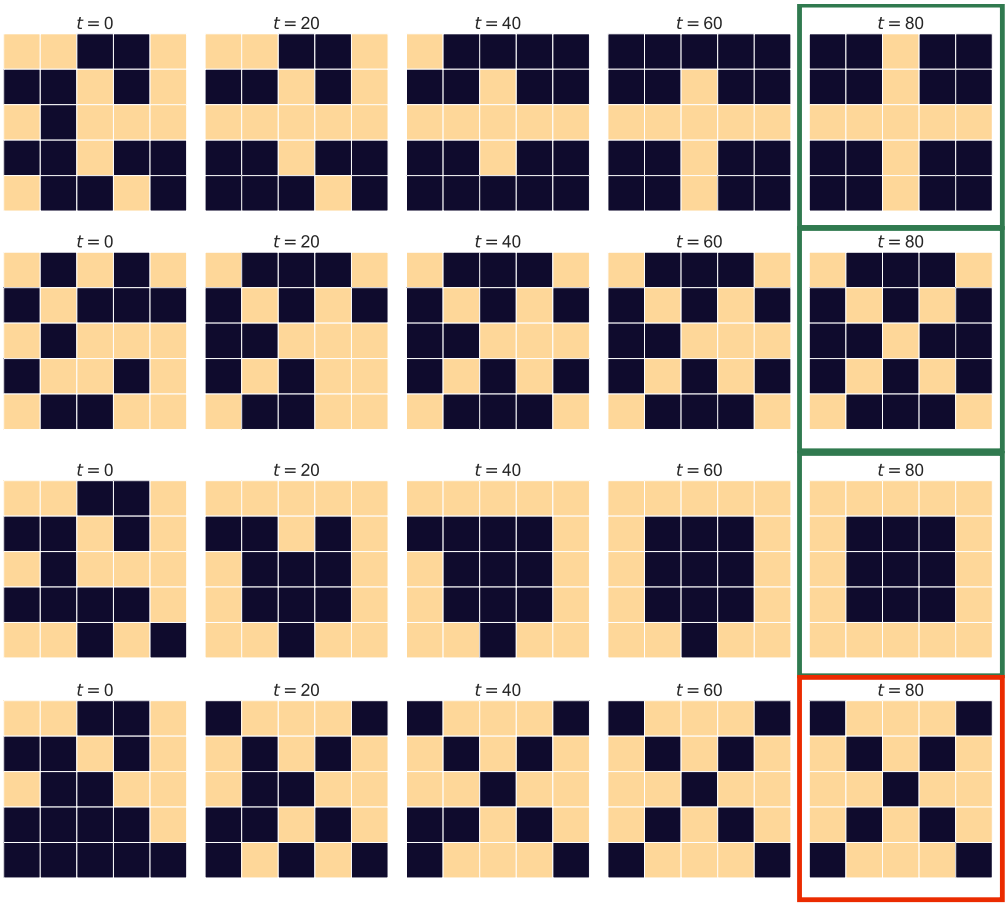

Figure 4: **Trajectories of the multi-memory network:** *Row 1:* The system converges to the cross. *Row 2:* The system converges to the X. *Row 3:* The system converges to the square. *Row 4:* The system converges to a spurious stable state.

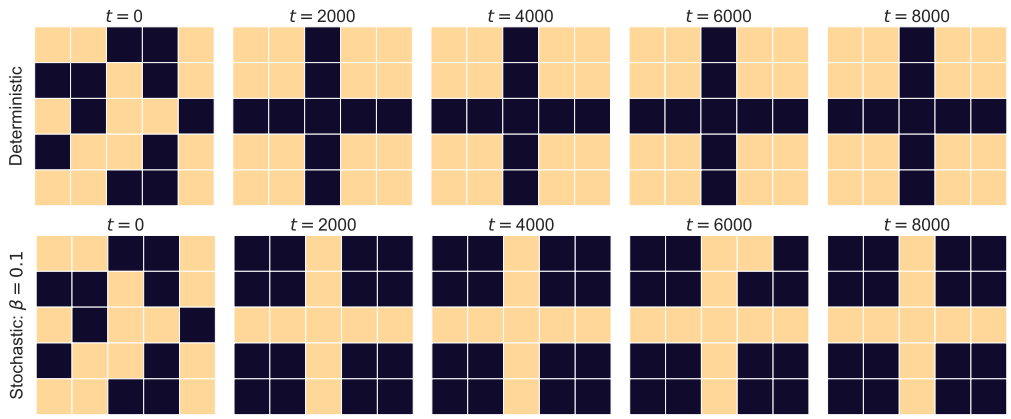

Figure 5: **Deterministic and stochastic 'cross' network:** *Row 1:* The deterministic network converges to a spurious pattern. *Row 2:* The stochastic network with $\beta = 0.1$ is able to retrieve the memory, but can leave the stable state due to fluctuations.

Stochastic Hopfield networks

From our simulations, we have shown that Hopfield networks have the ability to encode and retrieve memories. However, the problem of spurious patterns makes these networks unreliable as computational systems. In order to mitigate the prevalence of these spurious patterns, we turn to stochastic dynamics.

Systems like the Ising model are typically used as a mathematical description of magnets, with each neuron, in our case, being a magnetic spin that can either point 'up' (+1) or 'down' (-1).

When modelling magnetic systems with the Ising model, one must consider thermal fluctuations which are a consequence of the magnet being coupled to a thermal reservoir. Whilst the 'field', provided by the interactions, aims to align the spins, the thermal fluctuations disrupt this process via random flipping of the spins. This causes the dynamics to become stochastic. We can employ an equivalent dynamic in our Hopfield network.

Mathematically, the stochastic effect of thermal fluctuations can be modelled using Glauber dynamics (6). At each step, after selecting a neuron to update, the value of that neuron is updated stochastically according to,

$$P(v_i = \pm 1) = \frac{1}{1 + \exp(\mp \beta \sum_j W_{ij} v_j)}, \qquad (8)$$

where β is the inverse of the thermodynamic temperature T which controls the prevalence of the thermal fluctuations.

Fig. 5 shows how the stochastic network is able to, in this case, retrieve the encoded cross memory, whilst the deterministic network converges to a spurious pattern. However, notice that we run the networks for a very long time and that the stochastic network can leave the stable state due to the fluctuations. Figure 6 shows how the non-monotonic decrease in energy for the stochastic network allows the system to jump between local minima which is not possible in the deterministic system. This stochastic learning paradigm is known as 'Boltzmann learning' (1). Combining the stochastic dynamics with mean-field theory allows for the calculation of the approximate ratio between the maximum number of memories that can be encoded and the number of nodes in the network, which is given in Ref. (7) as $\alpha \approx 0.138$.

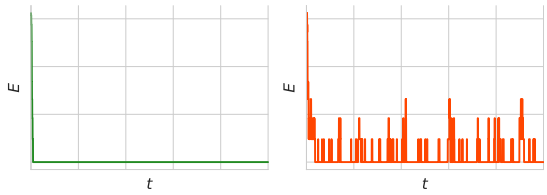

Figure 6: **Energy of the Hopfield Network.** *Left:* The energy of the deterministic model decreases monotonically to a local minimum, but that minimum was a spurious pattern. *Right:* The energy of the stochastic model decreases but not monotonically. The stochasticity allows it to jump between local minima.

Continuous Hopfield networks

In addition to the discrete systems described above, one can define a continuous-time Hopfield network with a graded (non-binary) response (9). Borrowing from spiking models of neurons, we define a linear dynamical system that is then

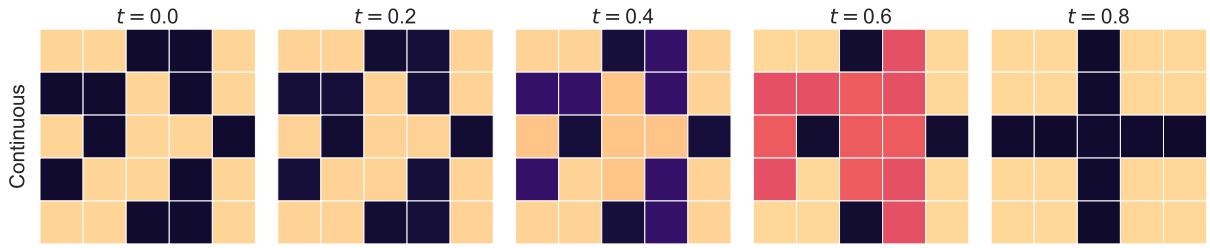

Figure 7: **Continuous evolution of the 'cross' network.** With the continuous-time dynamics, the Hopfield network still converges to the memorised pattern, the cross, but its response is now graded and no longer discrete. Here $\Delta t = 0.08$ and all other constants are set to 1.

passed through a non-linear, sigmoidal activation function,

$$\frac{du_i}{dt} = \frac{1}{C_i}\left(\sum_j W_{ij}v_j - \frac{u_i}{R_i} + I_i\right), \tag{9}$$

$$v_i = g(u_i), \tag{10}$$

where $v_i(t)$ is the graded output of neuron i at time t, C_i is the capacitance, R_i is the resistance and I_i is the current flow of neuron i, all of which are positive constants. The function g is a monotonically increasing sigmoidal function that goes between two fixed bounds, v_i^-, v_i^+, such as,

$$g(v) = \tanh(v), \tag{11}$$

$$= \frac{e^v - e^{-v}}{e^v + e^{-v}}, \tag{12}$$

which goes between -1 and 1. Similarly to before, we define the energy function,

$$E = -\frac{1}{2}\sum_{i,j} W_{ij}v_iv_j + \sum_i \frac{1}{R_i}\int_0^{v_i} g^{-1}(v)\,dv + \sum_i I_iv_i. \tag{13}$$

Next, we note that,

$$\frac{dE}{dt} = -\sum_i \frac{dv_i}{dt}\left(\sum_j W_{ij}v_j - \frac{u_i}{R_i} + I_i\right), \tag{14}$$

$$= -\sum_i C_i \frac{dv_i}{dt}\frac{du_i}{dt}, \tag{15}$$

$$= -\sum_i C_i \left(\frac{dv_i}{dt}\right)^2 g^{-1'}(v_i). \tag{16}$$

Due to the monotonic increasing of g^{-1}, we have that $\frac{dE}{dt} \leq 0$. As a result, the system will again tend towards a local energy minimum. In this case, we can call this function a Lyapunov function for the system.

Fig. 7 shows trajectories of a continuous Hopfield network as it retrieves the cross memory. Despite the sigmoidal gain function, the output of the neuron is still graded and continuous so during convergence, the output can vary from the Ising (or binary) values, yet, eventually, the system arrives very close to the Ising values $\{\pm 1\}$.

Optimisation with Hopfield networks

Beyond retrieving simple memories, Hopfield networks can have practical applications in the solution of challenging optimisation problems (10).

In order to encode an optimisation problem, the cost function must be written in such as way that it coincides with the energy function of the Hopfield network. As a result, the solutions to the optimisation problem are encoded as minima of the energy function and the Hopfield network will converge to them.

The travelling salesman problem

The travelling salesman problem (TSP) is a classical problem in optimisation that is simple to state yet computationally challenging.

Given n cities and the pairwise distances between them, the TSP is to find a 'tour', a non-repeating sequence of the cities, that includes each one exactly once, that minimises the total distance travelled. Finding the *optimal* solution to the TSP is np-complete (4).

Heuristic solution with Hopfield networks In order to solve this optimisation problem with a Hopfield network, one must create an intuitive mapping between tours of the TSP and the output of the network. One such mapping is given by a Hopfield network with $N = n^2$ neurons (where n is the number of cities).

Again, we consider the neurons arranged in a square grid, however not for visualisation in this instance. The location m of a given city n in the tour is represented by a 1 in $m-$th column of the $n-$th row in the network output. For

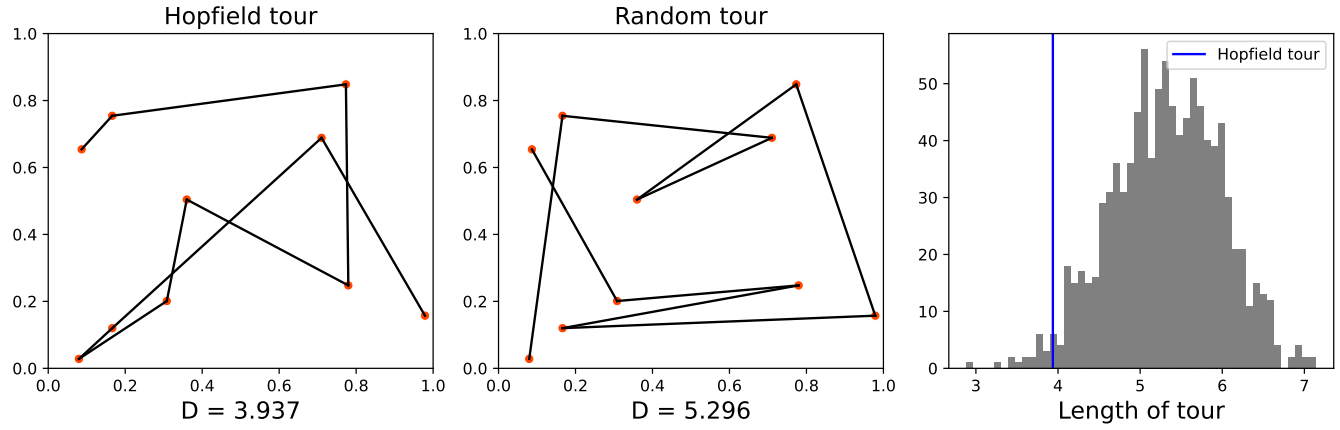

Figure 8: **Travelling salesman with Hopfield networks.** *Left:* The Hopfield tour obtained for a 10 city problem. *Center:* A random tour on a 10 city problem. *Right:* A distribution of 1000 random tour lengths for a 10 city problem. The Hopfield tour is far shorter than the median random tour but still not globally optimal.

example, given a 5 city problem with cities, A, B, C, D, E, the tour $BADCE$ would be given by network output,

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}. \tag{17}$$

As a result, a valid tour can only have a single 1 in each row/column, a restriction that we encode into our energy function. In order for a Hopfield network to 'solve' this problem, the network must be described by an energy function in which the minimum energy state corresponds to the optimal path. We design a network with weights given by,

$$\begin{aligned} W_{Xi,Yj} = & - A\delta_{XY}(1-\delta_{ij}) \\ & - B\delta_{ij}(1-\delta_{XY}) \\ & - C \\ & - Dd_{XY}(\delta_{j,i+1}+\delta_{j,i-1}). \end{aligned} \tag{18}$$

Here we are using double indices to account for the network state having a matrix output as demonstrated in equation (17) and δ is the Kronecker delta ($\delta_{ij} = 1$ if $i = j$ or 0 otherwise). Each term in this weight assignment enforces a different criterion for the tour. The first term enforces inhibitory connections in each row whilst the second enforces inhibitory connections in each column and the third represents global inhibition. These terms will push the system towards valid tour outputs. The final term is the 'data' term that takes into account the distance between cities.

Whilst this problem can be solved with either the discrete or continuous Hopfield network, in general, the continuous network performs better (10). This Hopfield network with continuous dynamics is equivalent to the dynamical system with equations,

$$\begin{aligned} \frac{du_{Xi}}{dt} = & -\frac{u_{Xi}}{\tau} - A\sum_{j\neq i} V_{Xj} - B\sum_{Y\neq X} V_{Yi} \\ & - C\left(\sum_Y\sum_j V_{Yj} - \tilde{n}\right) \\ & - D\sum_Y d_{XY}(V_{Y,i+1}+V_{Y,i-1}), \end{aligned} \tag{19}$$

where the neuronal state u_{Xi} is run through a non-linear sigmoidal function of the form,

$$V_{Xi} = g(u_{Xi}) = \frac{1}{2}(1+\tanh(u_{Xi}/\hat{u})). \tag{20}$$

In order to converge to both a valid and relatively short tour, the parameters must be chosen carefully. Here we follow Hopfield and Tank (10) and take,

$$\begin{aligned} A = B = 500 \qquad & C = 200 \qquad & \\ D = 500 \qquad & \hat{u} = 0.01 \qquad & \tilde{n} = 15. \end{aligned} \tag{21}$$

In addition, the initial state of the network can also affect convergence to a valid tour. The network has sufficient information, in the data term, to find an optimal solution without biasing in the initialisation. Therefore, we choose a minimum-bias initial condition u_0 such that $\sum_X\sum_j V_0 = n$, the desired total sum of a valid tour. However, as pointed out by Hopfield and Tank (10), this unbiased starting point prevents the network from choosing between the $2n$ equivalent optimal tours - the same tour with different starting points. Therefore we must break the symmetry by adding a

small amount of noise to the initial condition,

$$\tilde{u_0} = u_0 + \delta u_0, \tag{22}$$

where $\delta u_0 \sim U(-\frac{u_0}{10}, \frac{u_0}{10})$, which causes the network to choose between the tours.

We implement this method with 10 cities randomly, uniformly placed in the square $[0, 1] \times [0, 1]$ and consider the Euclidean distance between them. Figure 8 shows the typical tour obtained by the Hopfield network on a 10 city problem compared to a random tour. The Hopfield tour is much shorter than the average random tour, but still not globally optimal indicating that it can converge to spurious local minima depending on the initial condition.

Conclusions and outlook

Hopfield networks provide a simple, interesting and analytically tractable model of computation in neurons with strong links to neuroscience but applications to optimisation (11). Since its conception, the Hopfield network has been repeatedly refined and improved to increase its memory and convergence, such as the introduction of the Storkey learning rule that improved upon the capacity of the original Hebbian rule (20). More recently, the notion of 'dense associative memories' and 'modern Hopfield networks' were introduced which use stronger non-linearities to break the linear relationship between number of neurons and stored memories, allowing for super-linear (13), and eventually exponential (19), increases in capacity. Whilst the connections in the Hopfield network are assumed to be symmetric, connections in the human brain are asymmetric. Such connections cause the brain to operate as a non-equilibrium, time-irreversible complex dynamical system (5; 16). The energy landscape of asymmetric, non-equilibrium Hopfield networks has been shown to vary significantly from that of its equilibrium counterpart raising the possibility for biologically plausible computation out of equilibrium (21). In conclusion, whilst Hopfield networks represent an old model of neural computation, far from the state-of-the-art in machine learning, their intimate, but relatively unexplored, links to statistical mechanics imply a wealth of opportunities for both understanding the brain and yielding interpretable optimisation and learning algorithms.

Code availability

The code used in this project is available at `https://github.com/rnartallo/hopfield`

References

Ackley, D. H., Hinton, G., and Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169.

Amari, S.-I. (1972). Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions*, 21:1197–1206.

Gallavotti, G. (1999). *Statistical Mechanics*. Springer-Verlag.

Garey, M. and Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman.

Gaspard, P. (2007). Time asymmetry in nonequilibrium statistical mechanics. *Advances in Chemical Physics*, 135:83–133.

Glauber, R. J. (1963). Time-dependent statistics of the Ising model. *Journal of Mathematical Physics*, 4:94–307.

Hertz, J. A. (1991). *Introduction To The Theory Of Neural Computation*. CRC Press.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79:2554–2558.

Hopfield, J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the United States of America*, 81:3088–3092.

Hopfield, J. J. and Tank, D. W. (1985). "Neural" computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152.

Hopfield, J. J. and Tank, D. W. (1986). Computing with neural circuits: A model. *Science*, 233(4764):625–633.

Kirkpatrick, S. and Sherrington, D. (1978). Infinite-ranged models of spin-glasses. *Physical Review B*, 17:4384.

Krotov, D. and Hopfield, J. J. (2016). Dense associative memory for pattern recognition. *Advances in Neural Information Processing Systems*, 29:1172–1180.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–444.

Little, W. A. (1974). The existence of persistent states in the brain. *Mathematical Biosciences*, 19(2):101–120.

Lynn, C. W., Cornblath, E. J., Papadopoulos, L., and Bassett, D. S. (2021). Broken detailed balance and entropy production in the human brain. *Proceedings of the National Academy of Sciences of the United States of America*, 118(47).

McCulloch, W. S. and Pitts, W. H. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Minsky, M. and Papert, S. (1969). *Perceptrons*. MIT Press.

Ramsauer, H., Schäfl, B., Lehner, J., Seidl, P., Widrich, M., Gruber, L., Holzleitner, M., Pavlovic, M., Sandve, G. K., Greiff, V., Kreil, D. P., Kopp, M., Klambauer, G., Brandstetter, J., and Hochreiter, S. (2021). Hopfield networks is all you need. *arXiv*.

Storkey, A. (1997). Increasing the capacity of a Hopfield network without sacrificing functionality. *International Conference on Artificial Neural Networks*, pages 451—-456.

Yan, H., Zhao, L., Hu, L., Wang, X., Wang, E., and Wang, J. (2013). Nonequilibrium landscape theory of neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 100(45).

Zdeborova, L. and Krzakala, F. Statistical physics for optimization and learning.