Mathematical Memory Machines*

Ramón Nartallo-Kaluarachchi AMIMA, University of Oxford

I would like you to picture, in your imagination, an animal. A four-legged, tail-wagging animal. One that barks, is fond of bones and definitely not fond of cats. Right now, you are probably picturing a dog.

How is it that, from incomplete or noisy information, our brains are able to converge on the correct thought? At some point, probably early in life, our brain encoded the persistent memory of 'dog'. It encoded this memory in such a way that, given sufficient initial information, it converges automatically towards it. This phenomena has been termed by neuroscientists as *associative memory* because the partial initial information is 'associated' with the persistent memory.

From a mathematical perspective, the brain's tendency to converge to a persistent state from a sufficiently close initial point is reminiscent of an *attractor* in a dynamical system [1]. Such an analogy raises the question: could a dynamical system, using a brain-inspired model, be designed in such a way to encode a persistent memory?

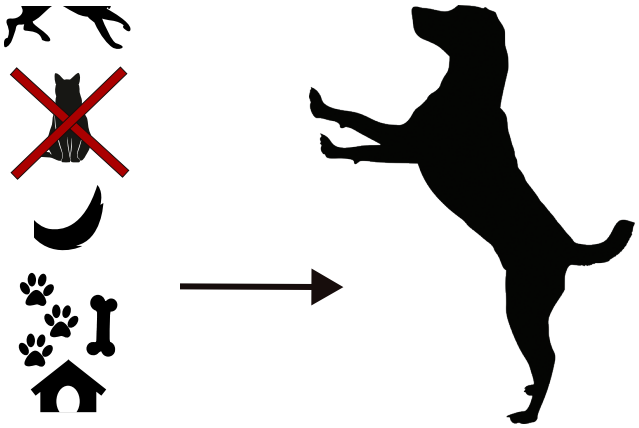

Figure 1: Associative memory. From partial, associated information, the human brain is able to converge to a persistent memorised concept, in this case, a dog. Image: Freepik and Pixabay.

Building a memory machine

To build such a system, we will begin by taking direct inspiration from the brain and consider a network of interconnected 'neurons'. At a given instant, each neuron can either be active, with value $+1$, or inactive, with value -1. Assuming a network made up of N such neurons, the *state* of the network is given by $(x_1, \dots, x_N)$, a vector displaying the activity of each neuron $x_i \in \{\pm 1\}$.

A *memory* can be encoded as a network state, a particular binary pattern of length N. Two neurons i and j are connected with strength W_{ij}, which is analogous to the synaptic weight connecting two biological neurons. Therefore, the total input into a neuron i is given by the weighted sum of inputs from its neighbours,

$$\sum_j W_{ij} x_j.$$

This helps us to define a *dynamical system* for our neural network.

At each discrete time step, one neuron is picked uniformly at random from the network and then updated in the following fashion:

$$x_i \to \begin{cases} +1, & \text{if } \sum_j W_{ij} x_j > 0, \\ -1, & \text{if } \sum_j W_{ij} x_j < 0, \end{cases}$$

i.e. if the total input to a neuron is positive, it becomes active, whereas if it is negative, the neuron becomes inactive.

Supposing that learning in the brain occurs through the modification of synaptic weights, we would like to encode a given memory $(x_1^*, \dots, x_N^*)$ by selecting corresponding network weights, (W_{ij}), thereby *training* the network.

A famous principle in neuroscience is the so-called *Hebbian learning rule*, which claims, in simple terms, that neurons that are active simultaneously are connected [2]. Inspired by this, we assign our weights according to the Hebbian rule:

$$W_{ij} = x_i^* x_j^*,$$

i.e. if the neurons 'agree' in the memory, they are connected positively, whereas if they disagree, they are connected negatively. If two neurons are connected positively, then updates will push them towards agreement, and if they are connected negatively, they will be pushed towards disagreement. To show that this weight assignment will cause the memory to become an attractive state, we define the following *energy* function:

$$E = -\frac{1}{2} \sum_{i,j} W_{ij} x_i x_j.$$

The pre-factor $1/2$ accounts for each interaction being counted twice in the sum, but this only plays a role when the *threshold* for a neuron is different from 0, which is not the case considered here, but we include it by convention. Due to the choice of the Hebbian weights, every time that we update the network, this energy can only decrease, which means that, eventually, the network will converge to a local energy minimum.

To borrow again from dynamical systems, this energy function acts similarly to a *Lyapunov function*. By writing the energy in the form

$$E = -\frac{1}{2} \sum_i x_i \sum_j W_{ij} x_j,$$

we can see that it can only decrease over subsequent updates as after the update

$$x_i \sum_j W_{ij} x_j \to 1.$$

Furthermore, the energy of the memory,

$$E^* = -\frac{1}{2} \sum_{i,j} 1,$$

is a global minimum.

* Graham Hoare Prize 2024 winning article

Remembering and misremembering

To visualise the network converging to a memory, we can organise the neurons into a square grid, so that memories and network states correspond to simple pixelated 'pictures'. For example, we consider the Ω-memory in Figure 2 (left) represented by the activity of a 36-neuron network. We can train the network weights to encode this memory and then begin the dynamics from different initial states to see if the system converges to the desired pattern.

Whilst the memory is an energy minima, it is not the only one. As a result, if the network starts from an initial condition too far away from the memory, outside of its *basin of attraction*, it could converge to an *unintended* pattern, one that we did not encode. Figure 2 (right) shows the network evolution from two initial states. In the top row, the network converges to the correct memorised Ω-memory, whilst in the second row, it converges to an unintended pattern. In this case, the unintended pattern is the *inverse* of the memory i.e. $x_i = -x_i^*$. This inverse pattern is always an unintended energy minima as

$$E = -\frac{1}{2}\sum_{i,j}(-1 \cdot -1) = E^*.$$

To better understand how our network remembers, and why it sometimes misremembers, we can consider the *energy landscape* of the trained network. To do this, we take a simple 2×2 grid where there are 16 possible network states and we train the network on the memory shown in Figure 3 (left). For each of the system states, we can measure the energy and project our 16 patterns into two dimensions by varying the top row along the x-axis and the bottom row along the y-axis, as shown in Figure 3.

Each panel of the 16 represents a distinct network state. We can see that three different energy values emerge, the lowest of which corresponds to the memorised pattern, but also to the inverse pattern. We begin in one square of this 4×4 plot and, as a single neuron is updated at each time step, the state can move only vertically or horizontally (with periodic boundaries) from darker squares to lighter (or equally dark) squares until, eventually, the network converges to one of the yellow squares, either *remembering* or *misremembering* the encoded memory.

For this example, there are no possible trajectories that avoid converging to a yellow square. Thus, convergence to the global minimum, either the memory or its inverse, is guaranteed. In larger networks, spurious *local* minima can emerge where the network becomes stuck at some other incorrect pattern.

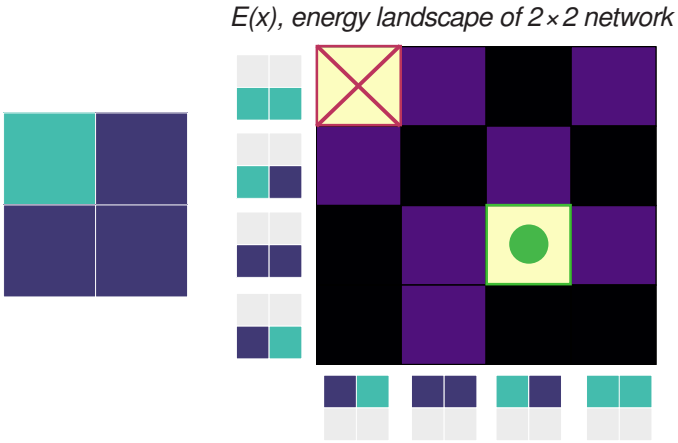

Figure 3: Energy landscape. We train our network, evaluate the energy of each possible configuration, and get two minima for the correct pattern (green circle) and its inverse (red cross).

Memorising multiple patterns

Beyond a single pattern, the same network can be trained to encode multiple memories simultaneously. Given a set of k-memories, $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k\}$, in a network with N neurons, we modify the selected weights to be

$$W_{ij} = \frac{1}{k}\sum_{l=1}^{k} x_{l,i}x_{l,j},$$

where $x_{l,i}$ is the value of the ith neuron in the lth memory.

In the multi-memory network, the connection strength between two neurons is simply the mean of its assigned value in each of the single-memory networks. Each memory will become an energy minimum with its unique basin of attraction. If you start in the basin of attraction of a particular memory, you will converge to it.

Figure 4 shows how a network can encode three memories simultaneously, namely the plus, cross and square memories, and that convergence depends on the initial network state. The maximum number of memories that a network can encode, its *capacity* C, increases with the number of neurons with the approximate growth rate [3]:

$$C \approx \frac{N}{2\log_2 N}.$$

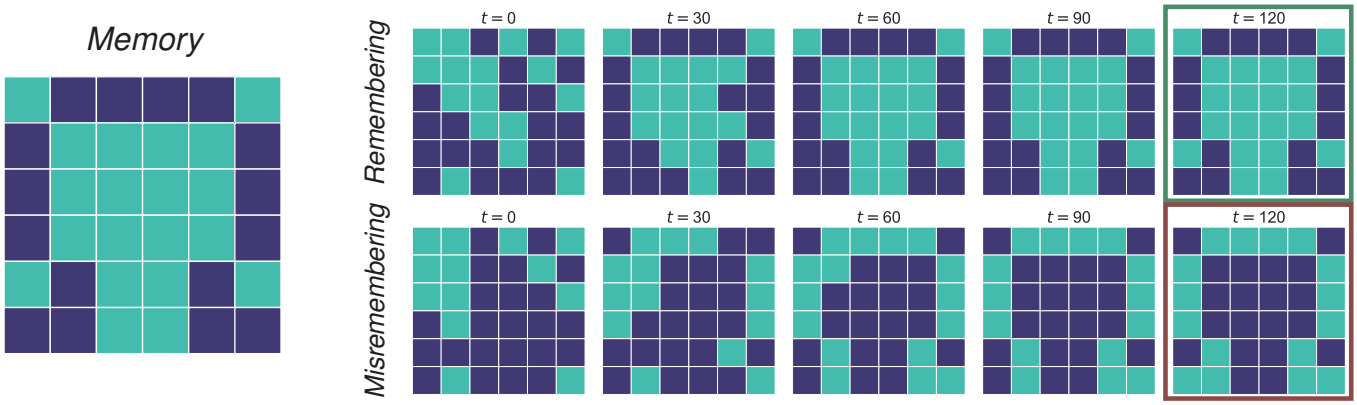

Figure 2: Remembering and misremembering. Depending on the initial condition, the network can either converge to the encoded memory or to an unintended energy minima.

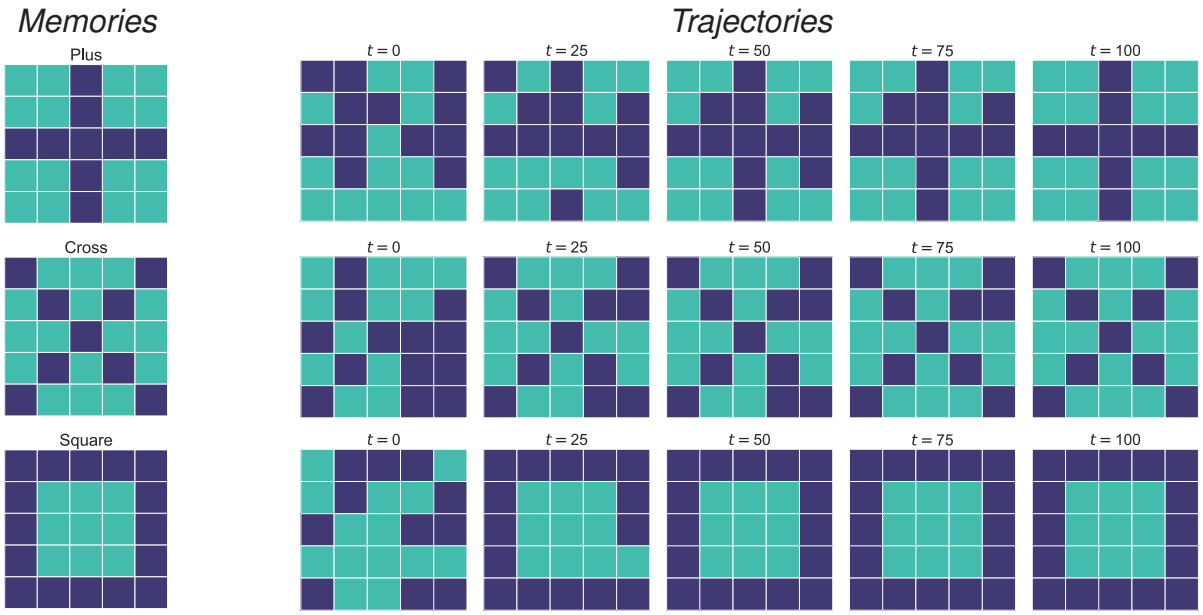

Figure 4: Multi-memory. A network trained to encode three memories: plus, cross and square, simultaneously. Each memory is reachable from its disjoint basin of attraction.

Noisy models of memories and magnets

If libraries and schoolteachers have taught us anything, it is that noise and learning do not mix. Nevertheless, whilst the above formulation of the memory machine is entirely deterministic, an analogous *stochastic* memory machine, a so-called *Boltzmann machine*, can be constructed using the tools of statistical mechanics [4].

First, notice that our memory machine, with its binary variables and interconnected structure, is reminiscent of a cornerstone model of magnetic dipoles known as the *Ising model*. To describe the dynamics of a magnet under heating, one must balance the attractive dynamics of the network with random thermal fluctuations due to heat. Under these constraints, a central result in statistical mechanics tells us that the probability of a given network state is given by *Boltzmann's distribution*:

$$P(x_1, \ldots, x_N) = \frac{1}{Z_\beta} \mathrm{e}^{-\beta E(x_1,\ldots,x_N)},$$

where β is the inverse of the temperature T, which controls the prevalence of the thermal fluctuations, E is the energy of a particular network state and Z_β is a normalising factor. This gives rise to the stochastic *Glauber dynamic* where neurons are updated probabilistically according to the distribution

$$P(x_i = \pm 1) = \left[1 + \exp\left(\mp\beta \sum_j W_{ij} x_j\right)\right]^{-1}.$$

Whilst the energy will no longer decrease monotonically, the encoded memories remain attractive states under this stochastic updating. Furthermore, the inclusion of noise can help the system to escape from unintended minima, thus aiding recall.

Beyond memory with attractor neural networks

Network dynamical systems of this kind have been formalised by mathematicians and computer scientists as *attractor neural networks* [5]. Whilst the learning rule corresponding to memory recall is inspired by the human brain, the mechanism by which the system recalls memories is simply through the minimisation of a particular energy function. This insight opens up an interesting avenue for diverse applications of our so-called memory machines. Simply through an appropriate choice of learning rule, one creates a network dynamical system that converges to the minima of a particular energy function, which can be associated with the solutions to some optimisation problems.

A classic example is the *travelling salesman problem* (TSP) where one aims to find a tour of vertices in a graph that minimises the distance travelled. By using a grid of neurons to represent all possible tours in the system and defining a corresponding energy function, we can derive a learning rule such that our attractor neural network will converge to a heuristic solution to the TSP [6]. In contrast with the example of memory, this system is able to converge to a solution that was not known a priori, more in line with traditional use cases of neural networks and machine learning.

In a scientific landscape that is witnessing the proliferation of gargantuan, black-box learning systems, the beauty and simplicity of the 'mathematical memory machine' represents an ideal of an intuitive, biologically inspired system with immense learning potential.

References

1 Poucet, B. and Save, E. (2005) Attractors in memory, *Science*, vol. 308, no. 5723, pp. 799–800.

2 Hebb, D.O. (1949) *The Organization of Behavior*, Wiley, New York.

3 McEliece, R. et al. (1987) The capacity of the Hopfield associative memory, *IEEE Trans. Inf. Theory*, vol. 33, no. 4, pp. 461–482.

4 Ackley, D.H., Hinton, G.E. and Sejnowski, T.J. (1985) A learning algorithm for Boltzmann machines, *Cogn. Sci.*, vol. 9, no. 1, pp. 147–169.

5 Hopfield, J.J. (1982) Neural networks and physical systems with emergent collective computational abilities, *PNAS*, vol. 79, no. 8, pp. 2554–2558.

6 Hopfield, J.J. and Tank, D.W. (1985) Neural computation of decisions in optimization problems, *Biol. Cybern.*, vol. 52, no. 3, pp. 141–152.