Hopfield Networks as Dynamical Systems

Ishank Juneja - Student, IIT Bombay
April, 2020

1. Introduction

Inspired by the neuronal circuits in brains, artificial neural networks consist of numerous identical computational units performing simple operations on individual components of an input stimulus.

For instance the units present in neural network architectures commonly used for pattern recognition tasks, like hand-written character recognition, use an affine transformation on their input component. This is followed by a thresholding operation performed using the same unit.

A neural network can be viewed as a directed graph with the units of the network as nodes, and edges representing dependencies between nodes. That is, the inputs to a node will be the results from other nodes, each weighted by the respective edge weight.

Many of the effective deep learning architectures used for sophisticated pattern recognition tasks, such as computer vision, are purely feed-forward in nature. That is, the nodes in the network are arranged into a hierarchy of layers. Results of intermediate computations are passed from one layer to the next with no feedback from higher layers to any previous layer. A recurrent neural network on the other hand is a network architecture where such feedback connections can exist due to the presence of internal state. That is, the current output of a certain node can depend on both the current inputs and the past inputs and outputs. Mathematically,

$$\mathbf{y_k} = \mathbf{F}(\mathbf{x_k}, \mathbf{x_{k-1}}, \ldots, \mathbf{x_0}, \mathbf{y_{k-1}}, \mathbf{y_{k-2}}, \ldots, \mathbf{y_0}). \quad (1)$$

Where $\mathbf{y_k}$ and $\mathbf{x_k}$ are the vector of outputs and inputs respectively at time step $\mathbf{k}$.

In this paper we will be looking at a model called Hopfield Networks first introduced in (Hopfield, 1982). The Hopfield Network is a recurrent neural network where the states take values in the set $\{0, 1\}^n$. It should be noted here that these are different from Boolean networks where dynamics follow Boolean algebra. The Hopfield model provides us with numerous useful applications such as content addressable (associative) memories, error corrections and approximate solutions to optimization problems. Hopfield networks are interesting from a dynamical systems perspective since they have dynamics governed by an underlying Lyapunov function.

2. Stability of Discrete-Time Systems

Given a time-invariant autonomous discrete time dynamical system with n dimensional state at step k given by $\mathbf{x}(t) \in \mathbf{R}^n$, its update rule will be given by,

$$\mathbf{x}(k+1) = \mathbf{F}(\mathbf{x}(k)). \quad (2)$$

Where $\mathbf{F} : \mathbf{R}^n \mapsto \mathbf{R}^n$ is the next state map of the dynamical system. A fixed point $\mathbf{x_f}$ of the dynamical system is a point in the state space which maps to itself under $\mathbf{F}$. Therefore, all fixed (or equilibrium) points satisfy the relation

$$\mathbf{x_f} = \mathbf{F}(\mathbf{x_f}). \quad (3)$$

Let us assume that $\mathbf{x_f}$ is a fixed point of the dynamical system, i.e. $\mathbf{x_f}$ satisfies the relation 3. Further let $\mathbf{x_o}$ be the state from which the system starts at step $k = 0$. Unlike continuous time dynamical systems, where a unique solution to the differential equation describing system dynamics may or may not exist, the recurrence relation 2 guarantees the existence of a unique state trajectory $\mathbf{s}(k, \mathbf{x_o})$ given start state $\mathbf{x}(0) = \mathbf{x_o}$. Further, we can see that if the system were to start in the state $\mathbf{x_o} = \mathbf{x_f}$, the trajectory in state space would simply be $\mathbf{s}(k, \mathbf{x_f}) = \mathbf{x_f}\ \forall\, k$.

Definition 1 *A fixed point $\mathbf{x_f}$ of a discrete time dynamical system is said to be stable if for each $\epsilon > 0$, there exists $\delta = \delta(\epsilon)$ such that,*

$$||\mathbf{x_o} - \mathbf{x_f}|| < \delta(\epsilon) \Rightarrow ||\mathbf{s}(k, \mathbf{x_o}) - \mathbf{x_f}|| < \epsilon\ \forall\, k \geq 0. \quad (4)$$

Next we look at another property present in the fixed points of Hopfield Networks,

Definition 2 *A fixed point $\mathbf{x_f}$ is said to be attractive if there exists an η_o such that*

$$||\mathbf{x_o} - \mathbf{x_f}|| < \eta_o \Rightarrow \mathbf{s}(k, \mathbf{x_o}) \to \mathbf{x_f} \text{ as } k \to \infty. \quad (5)$$

If the state $\mathbf{x}(k)$ of the system were to lie in some finite or countable set instead of $\mathbf{R}$, then the norm $||\cdot||$ in definitions 1 and 2 will be replaced by a Hamming distance metric. Next we define the notion of asymptotic stability of a fixed point,

Definition 3 *The fixed point $\mathbf{x_f}$ is said to be asymptotically stable if it is both stable and attractive.*

Lyapunov's Second Method

Now that the definitions of stability have been established, we look for a sufficient condition for a fixed point to be stable. Lyapunov's second method (also known as Lyapunov's direct method), provides a tool to define a notion of stability of equilibrium points of both continuous and discrete time dynamical systems. In this paper we focus on the latter.

Theorem 1 *Given that a discrete-time dynamical system has a fixed point $\mathbf{x_f}$, the fixed point will be stable if $\exists$ a scalar valued function (called a Lyapunov function) $V(\mathbf{x}(k))$ such that in a neighberhood around $\mathbf{x_f}$,*

$$V(\mathbf{x}) \geq 0 \tag{6}$$

$$V(\mathbf{x}) = 0 \iff \mathbf{x} = \mathbf{x}_f, \tag{7}$$

and the system dynamics are such that,

$$V(\mathbf{x}(k+1)) \leq V(\mathbf{x}(k)) \; \forall \, k \geq 0. \tag{8}$$

3. Hopfield Networks

Both computing circuits designed by humans and neuronal circuits in our brains consist of numerous identical units interacting with one another. However, unlike meticulously planned computing circuits, biological neuronal circuits are evolutionary in nature, so they don't involve precise planning. Keeping this in mind, Hopfield's work in the 1980s asked the question whether some of the properties of a large number of interacting identical neuronal units are *emergent* from the collective. In this paper, we look at such a possible origin for the stability of memories.

As mentioned the Hopfield Network model in consideration has a binary state associated with each neuron unit. We assume that the binary state takes a value of either 1 or 0. Further the state of every individual node is updated randomly and asynchronously with a mean update rate of W. The model is inspired from biological circuits and properties that exist in spite of asynchrony have interesting implications for biology.

Let there be a total of n units in our network and further let each unit be indexed by an index $i \in \{1, 2, \ldots, n\}$. The directed edge e_{ij} connecting node j to node i has a weight T_{ij} associated with it. If the edge e_{ij} does not exist, we take $T_{ij} = 0$, in particular, we assume that self edge e_{ii} does not exist. Lastly, we assume that there are no inputs to any of the units besides those coming in from other nodes. If the net input to the i^{th} node is given by $h_i(k) = \sum_{j \neq i} T_{ij} x_j(k)$, then the state x_i of the i^{th} unit is updated as per

$$x_i(k+1) = \begin{cases} 1 & \text{if } h_i(k) > 0 \\ x_i(k) & \text{if } h_i(k) = 0 \\ 0 & \text{if } h_i(k) < 0 \end{cases} \tag{9}$$

Thus, each neuron randomly checks whether it is above or below a threshold and updates its state accordingly. Although quite simplified, this update rule is inspired from biology where a large enough input to a neuron ensures that it *spikes* and in the absence of a such an input a neuron remains silent.

A corollary of arbitrary and stochastic asynchronous updates is that the probability of two nodes i and j updating their states x_i and x_j at the same time step is exactly 0. To perform computations with a general topology, the network dynamics need to be followed in detail to work out the future states. However by associating a Lyapunov function with the system we can understand the progression of its dynamics without explicitly computing its state trajectory.

In the absence of any known fixed points of the system, instead of the Lyapunov function $V(\mathbf{x})$, we can look for an *energy function* $E(\mathbf{x})$ that only has the monotonicity property 8. Subsequently when a fixed point $\mathbf{x_f}$ is identified, its stability can be shown by obtaining a Lyapunov function by shifting the reference level of $E(\mathbf{x})$ to the value $E(\mathbf{x_f})$, i.e. the energy function evaluated at the fixed point. To be precise, given a monotonically decreasing $E(\mathbf{x})$, we choose,

$$V(\mathbf{x}) = E(\mathbf{x_f}) - E(\mathbf{x}). \tag{10}$$

Next we examine when $E(\mathbf{x})$ exists for a network.

Theorem 2 *If the edge weights are symmetric, that is if $T_{ij} = T_{ji}$ and if there are no self edges, meaning $T_{ii} = 0$, then the network admits an energy function of the form,*

$$E(\mathbf{x}(k)) = -\frac{1}{2} \sum_{1 \leq i,j \leq n} x_i(k) T_{ij} x_j(k), \tag{11}$$

where x_i is the i^{th} component of the n dimensional state $\mathbf{x}$.

Proof of Theorem 2

The difference in energy between two successive time steps, ΔE is given by,

$$\Delta E = E(\mathbf{x}(k+1)) - E(\mathbf{x}(k)) \tag{12}$$

$$= \frac{1}{2} \sum_{i,j} \Big[x_i(k) T_{ij} x_j(k) - x_i(k+1) T_{ij} x_j(k+1) \Big] \tag{13}$$

Since updates are asynchronous, only one unit changes at a time, let us assume it has index l. Then, using the symmetry of the weights matrix, ΔE is given by,

$$\Delta E = -(x_l(k+1) - x_l(k)) \sum_{j \neq l} T_{lj} x_j(k) \tag{14}$$

Using the system dynamics specified in equation 9 and the fact that each component of the state lies in $\{0, 1\}$, it is easy to see that the expression in 14 will be non-positive, thereby making the function $E(\mathbf{x})$ a valid candidate for establishing stability of fixed points. Also, since only a single component of the network state is updated at a time, and the mean update rate of all the units is known to be a fixed value W, any fixed point of such a dynamical system will be attractive as per the definition 2. In the next section we shall see how we can place fixed points and use the Hopfield Network as an associative memory.

4. Constructing an Associative Memory

An associative or content addressable memory is an information retrieval system that given some input, returns the address of the memory closest to the input data. Hence an associative memory can be seen as a system capable of recall based on partial or even slightly corrupted information. For instance consider the original and corrupted versions of the letter 'T' in figure 1. A well designed associative memory would be able to store numerous letters and would match the corrupted 'T' to the true letter when queried with the corrupted version.

Another application of an associative memory could be the complete recall of the information associated with an entity using only a few of its attributes. As an example, given the name of a person, an associative memory would be able to quickly provide information like their place of residence and favourite color. Next we look at how the weights matrix T of a Hopfield network can be chosen in a way that these interesting properties of an associative memory emerge.

At any time step k, the state of the network consists of the vector $\mathbf{x}(k)$ encoding n bits of information. If the matrix T is symmetric with all its diagonal elements as 0, then from theorem 2, we know that a fixed point $\mathbf{x}_f$, will be stable and attractive to all trajectories with start state $\mathbf{x}_o$ lying in a certain δ neighborhood of $\mathbf{x}_f$.

If we wish to store a state vector $\mathbf{x}^s \in \{0, 1\}^n$ in a network, we can choose the weights matrix as per the following rule,

$$T_{ij} = \begin{cases} (2x_i^s - 1)(2x_j^s - 1) & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{15}$$

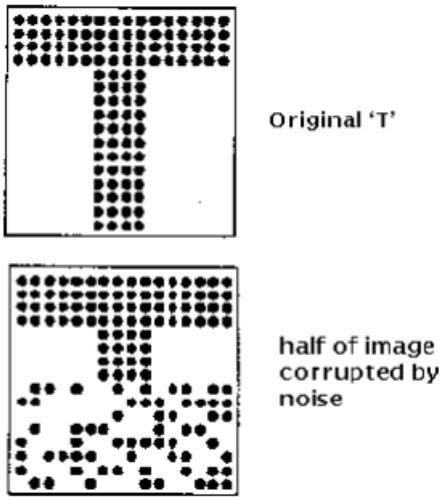

Figure 1. Mapping of stimulus to correct memory. Image taken from (Rosen)

Using these weights, we can show that x^s will be a fixed point of the network as per the definition 3. Let the state of the network at the k^{th} time step be $\mathbf{x}^s$

$$x_i(k+1) = \begin{cases} 1 & \text{if } \sum_{j \neq i} T_{ij} x_j^s > 0 \\ x_i(k) & \text{if } \sum_{j \neq i} T_{ij} x_j^s = 0 \\ 0 & \text{if } \sum_{j \neq i} T_{ij} x_j^s < 0 \end{cases} \tag{16}$$

$$\sum_{j \neq i} T_{ij} x_j^s = (2x_i^s - 1) \sum_{j \neq i} (2x_j^s - 1) x_j^s \tag{17}$$

From the right hand side of 17 it can be seen that,

$$\sum_{j \neq i} T_{ij} x_j^s \geq 0 \text{ when } x_i^s = 1 \tag{18}$$

$$\sum_{j \neq i} T_{ij} x_j^s \leq 0 \text{ when } x_i^s = 0 \tag{19}$$

Using 18 and 19, it is clear that when the matrix T is defined as per 15, $\mathbf{x}^s$ becomes a fixed point of the network. We should note here that $\mathbf{x}^s$ is not the only fixed point associated with the network, in particular $\mathbf{x} = \mathbf{0}$ is always a fixed point under the dynamics 9. Thus we have constructed a content addressable memory and have stored a single entry $\mathbf{x}^s$ into it. $\mathbf{x}^s$ could, for instance be the original letter 'T' in figure 1, and when a corrupted or incomplete version of the letter is presented as the initial state $\mathbf{x_o}$ to the network, it would converge onto the original 'T' in memory.

To store more memories in our associative memory, we need to incorporate a large number of fixed points into our network. Let us say we wish to store m memories, each of which is an n bit vector $\mathbf{x}^s$, indexed by s. A weights matrix T complying with this requirement can be achieved using

the following,

$$T_{ij} = \begin{cases} \sum_{s=1}^{m} (2x_i^s - 1)(2x_j^s - 1) & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{20}$$

We wish to check if our programmed memories do indeed correspond to fixed points. Assuming that the state of the network has reached the programmed memory with index s', the net input to the i^{th} unit will be given by

$$\sum_{j \neq i} T_{ij} x_j^{s'} = \sum_s (2x_i^s - 1) \Big[\sum_{j \neq i} (2x_j^s - 1) x_j^{s'} \Big] \tag{21}$$

Typically information that needs to be stored as a memory goes through some pre-processing in both artificial and biological systems. If the result is a binary string, then we expect each of the n components to be either 0 or 1 with equal probability. If this is not the case, that is, if $\mathbf{P}(x_i = 1) > \mathbf{P}(x_i = 0)$ then the asymptotic equipartition property would allow us to perform compression on the set of such n dimensional vectors. Thus, random binary vectors simulate memories generated through efficient pre-processing.

If we treat each component of a memory $\mathbf{x}^s$ and that of the current state $\mathbf{x}^{s'}$ as independent binary $\{0, 1\}$ random variables with $\mathbf{P}(x_i = 1) = \mathbf{P}(x_i = 0)$, then the bracketed term in 21 has an expected value of 0 when $s \neq s'$ and of $(n-1)/2$ when $s = s'$. Thus, we can write the expected or average value of the input to the i^{th} unit as,

$$\Big\langle \sum_{j \neq i} T_{ij} x_j^{s'} \Big\rangle = (2x_i^{s'} - 1)(n-1)/2 \tag{22}$$

The expression in 22 is positive when $x_i^{s'} = 1$ and negative when $x_i^{s'} = 0$, therefore once any of the memories programmed into the network is reached, it is quite probable that the state will not be updated again. For a more quantitative analysis using the central limit theorem, the reader is referred to (Latham, 2018).

As long as stochastic effects are small, the essence of reaching a certain fixed point on starting from a closely related start state by flowing down the energy landscape of 11 remains. A caveat here is that the flow in the phase space towards a certain memory is not entirely deterministic but rather is a statistical choice between the states that its current state most resemble.

An advantage of constructing an associative memory in this manner is that it has the potential to be extremely efficient in terms of its access time compared to the results that are obtained through traditional hardware.

The associative memory described could also be used as a tool for error and erasure correction among a set of binary strings. Once the network is initialized with the corrupted binary string as its start state, the network can be made to follow its usual dynamics, while keeping the nodal states known to be correct as static.

5. Extensions of the Hopfield Model

In this section we look at extensions of the Hopfield model that have been mentioned in (Hopfield, 2007) but have not been elaborated on or justified mathematically prior to this work as far as the author is aware.

External Inputs to Network

To make the model presented in Section 2 more general, external time invariant inputs can be added to every unit alongside the existing inputs from other nodes. That is, if the external input to the i^{th} node is I_i, the net input to it will be $h_i(k) = I_i + \sum_{j \neq i} T_{ij} x_j(k)$. The change to the dynamics introduced by I_i can be captured by replacing the default threshold of 0 in equation 9 with $-I_i$. For this model, the stability of a fixed point can be established by using the below energy function,

$$E(\mathbf{x}(k)) = -\sum_{1 \leq i \leq n} I_i x_i(k) - \frac{1}{2} \sum_{1 \leq i,j \leq n} x_i(k) T_{ij} x_j(k) \tag{23}$$

For this energy, ΔE when the l^{th} node is updated will be

$$\Delta E = -(x_l(k+1) - x_l(k)) \Big(I_l + \sum_{j \neq l} T_{lj} x_j(k) \Big), \tag{24}$$

which is always non positive. For this new model, a fixed point cannot be chosen using the rule 15 in section 4 since the internal state alone is no longer responsible for updates.

Asymmetric Weight Matrix

Until this point, we have made the assumption that the matrix T is symmetric, that is $T_{ij} = T_{ji}$. Here, we relax this assumption by considering an augmented weight matrix S with $S_{ij} = \lambda_i T_{ij} \mu_j$. T continues to be a symmetric matrix with T_{ij} as defined in either 15 or 20, and $\lambda, \mu \in \mathbf{R}^n$. As we shall see, we further require each component of λ and μ to be positive.

Borrowing from 11, we can define the following energy function for this model,

$$E(\mathbf{x}(k)) = -\frac{1}{2} \sum_{1 \leq i,j \leq n} \frac{x_i(k)}{\lambda_i} S_{ij} \frac{x_j(k)}{\mu_j}. \tag{25}$$

On substituting the definition $S_{ij} = \lambda_i T_{ij} \mu_j$ into 25, we get an expression for ΔE identical to 14 from Section 3. Next we verify that the technique to choose fixed points described in Section 4 still holds good. Rewriting equation 21 using the new weights we get,

$$\sum_{j \neq i} S_{ij} x_j^{s'} = \lambda_i \sum_s (2x_i^s - 1) \Big[\sum_{j \neq i} \mu_j (2x_j^s - 1) x_j^{s'} \Big]. \tag{26}$$

Under the assumption that for every component of a binary string, the values of 0 and 1 are equally likely,

$$\Big\langle \sum_{j \neq i} S_{ij} x_j^{s'} \Big\rangle = \lambda_i (2x_i^{s'} - 1) \sum_{j \neq i} \frac{\mu_j}{2}. \tag{27}$$

Which just like 22 is positive when $x_i^{s'} = 1$ and negative when $x_i^{s'} = 0$, thereby ensuring that $\mathbf{x}^{s'}$ remains a fixed point under the new scheme.

References

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8):2554–2558, April 1982. ISSN 0027-8424. URL http://view.ncbi.nlm.nih.gov/pubmed/6953413].

Hopfield, J. J. Hopfield network. *Scholarpedia*, 2(5):1977, 2007. doi: 10.4249/scholarpedia.1977. revision #91363.

Latham, P. Hopfield networks, 2018. URL http://www.gatsby.ucl.ac.uk/~pel/tn/notes/hopfield.pdf. [Online; accessed April 27, 2020].

Rosen, B. Hopfield networks. URL http://web.cs.ucla.edu/~rosen/161/notes/hopfield.html. [Online; accessed April 27, 2020].

Vidyasagar, M. *Nonlinear Systems Analysis (2nd Ed.)*. Prentice-Hall, Inc., USA, 1993. ISBN 0136234631.