# Understanding Hopfield Neural Networks

Jeremy Lu, Jonah Wu

## 1. Abstract

This paper provides an overview of the neural network first popularized in the deep learning community by J.J. Hopfield in 1982. Built on the idea of content-addressable (or "associative") memory systems along with neuron behavior, and used for a variety of purposes such as pattern recognization, optimization, and categorization, the Hopefield network is an important foundation for recurrent neural networks and machine learning in general. We take an in-depth exploration at the core mathematics behind this network, the biological foundations of this model, and discuss both its strengths and shortcomings.

## 2. Problem Description

When humans first encounter new information, we keep this in our short term memory; and then by strengthening these memories through consolidation, we can place them in our long term memory, where they become known as "declarative" memory (Suzuki 2005). The human brain also possesses the ability to learn and remember the relationships of various unrelated memories by forming connections, known as "associative" memory (Suzuki 2005).

Now, suppose we had the item "AMATH 383: Introduction to Continuous Mathematical Modeling" stored in our memory. Based on the idea of associative memory, we would be able to recall this entire term based on only partial information, such as "AMATH 383", or "Introduction to Mathematical Modeling". Furthermore, humans also have the capability of correcting errors during memory retrieval. Given "AMETH 383", we should, ideally, also be able to come up with the original item (Hopfield 1982).

Based on this fundamental idea in memory recall, Hopfield sought to find a way to mathematically and computationally model this behavior of the human brain and its neurons. He does this successfully using an asynchronous method, by modeling neurons with a complete, directed graph and expressing memories as points in an $n$-dimensional space (Hopfield 1982).

## 3. Variables

- $V_i$: represents a single neuron. Is binary, with 1 representing firing and -1 not firing (Hopfield used 0 in his original paper)
- $\mathbf{V} = [V_1, V_2, \dots V_n]^T$: represents the entire neural network, in vector form.
- $T_{ij}$: represents the strength or weight of a connection between neurons $V_i$ and $V_j$. This is determined with Hebbian Learning.
- $\mathbf{T} = \begin{bmatrix} 0 & T_{12} & T_{13} & \cdots & T_{1n} \\ T_{21} & 0 & T_{23} & \cdots & T_{2n} \\ T_{31} & T_{32} & 0 & \cdots & T_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ T_{n1} & T_{n2} & T_{n3} & \cdots & 0 \end{bmatrix}$: the weights put together in matrix form. Note that $T_{ii} = 0$ and that this matrix is symmetric.
- $U_i$: The fixed threshold determining neuron state. Unless otherwise stated, $U_i = 0$.
- $\mathbf{X_i} = [v_1, \dots, v_n]^T$: a new training input pattern where $v_i$ is the state of neuron $i$ of the new pattern where $v_i$ is either 1 or -1.

  $\mathbf{E}$ : Energy of the Hopfield Network, based on the state of the neurons in the network.

## 4. Mathematical Model

### *4.1. Modeling Neuron Behavior*

Drawing from our understanding of biology, Hopfield noted that neurons in our brain were either active or not, and thus they could be represented as a binary variable, as described in Section 3 (Hopfield 1982). He also noted that neurons fired only when the action potential reached a certain threshold (Hopfield 1982). Thus $U_i$ is used to represent this sort of action potential. If a neuron possesses has an input signal greater than 0, it will become active and have a state of 1. If the input signal is less than 0, the neuron is inactive and has a state of -1. This field is calculated, for a neuron $V_j$ as

$$\sum_{i \neq j}^{n} T_{ij} \cdot V_j \tag{1}$$

(Hopfield 1982)

The input-output model relationship is shown in this graph:

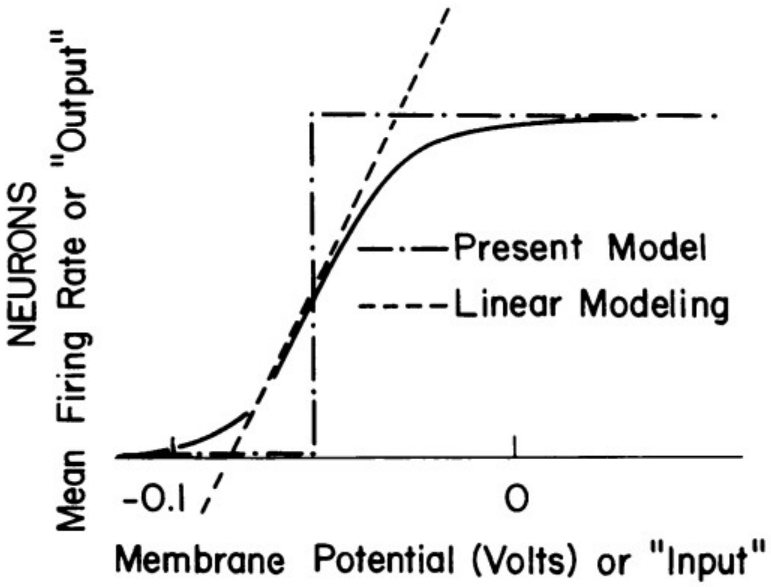


Figure 1: Firing Rate based on biological acting potential for a neuron is shown in the solid line. The "Present Model" line shows the Hopfield Network's simplification using the input signal equation (Hopfield 1982)

*4.2. Hebbian Learning*

Another key biological aspect of the Hopfield Networks is its use of Hebbian Learning (Hopfield 1982). The core concept, which was first proposed in 1949, is that neurons which are associated with each other will have stronger connections (Hebb 1949). Essentially, "neurons which fire together, wire together" (Hebb 1949). A common example of this phenomenon which is easily observed is muscle memory, where human cognitive performance is refined both in speed and accuracy through repeated trials and activation of the same group of neurons. Hopfield uses Hebbian learning to assign weights between neurons, with $T_{ij}$, the strength of connection between a neuron $V_i$ and $V_j$ calculated as:

$$T_{ij} = \sum_k v_i^k v_j^k \tag{2}$$

(Based on Hopfield 1982)

Given $k$ memories that are stored in the Hopfield Network, the connection between neurons $V_i$ and $V_j$ will be the sum of the product of each of these neurons in each pattern. This means that when $v_i^k = v_j^k$ for a pattern $X_k$, the contribution to $T_{ij}$ will be 1, and when the neurons are not the same, the contribution will be $-1$. This follows the idea of Hebbian learning, in that neurons which are in the same state (meaning they "fire together") have stronger connections.

The major advantage of this method of assigning weights, which is essentially "training", is advantageous in that we require only one instance to learn a memory or weight. This behavior is similar to how humans learn, which is often from one-shot experiences. In fact, the CA3, which is located in the brain's hippocampus, has "been shown behaviorally that [it] supports spatial rapid one-trial learning, learning of arbitrary associations where space is a component, pattern completion, spatial short-term memory, and sequence learning by associations formed between successive items" (Cursuridis & Wennekers, 2006). Meanwhile, many modern neural nets require multiple instances of data in training.

### *4.3. Neural Network*

Now that we have defined both the neuron behavior and Hebbian-based weight assignments, we can explain how exactly the Hopfield Network functions. The key point Hopfield made in his original paper is that this method is meant to be asynchronous (Hopfield 1982). The behavior of a neuron is based on the input signal, which is dependent other neuron states; if we update every neuron in the network at the same time, the model is prone to oscillating. This is also not realistic, as neurons in our brain work randomly, not all at the same time (Hopfield 1982).

Given an input memory or pattern, called $X_{in}$, we select a random neuron to update, called $V_i^{in}$. Then, we have:

$$V_i^{in} = \begin{cases} 1 & \text{if } \sum_{j \neq i} T_{ij} V_j \geq U_i \\ -1 & \text{otherwise} \end{cases} \tag{3}$$

(Based on Hopfield 1982)

where $U_i = 0$ unless otherwise stated. This process continues, with the random, asynchronous update of neurons, until $V_{in}$ reaches a learned memory/pattern, which will be a stable point. At this point, it can be considered the output memory which the Hopfield Network has identified and recovered.

This model is referred to as a recurrent neural network. The first key characteristic of this model architecture is that each neuron is connected to every other neuron. Next, each neuron updates based on the state of the other neurons, which reinforces the aforementioned Hebbian learning concept, as neurons will be influenced more by the neurons which it has strong connections to. This means that given a very noisy or partially broken input, the Hopfield Network is still able to recover the correct learned memory (Hopfield 1982).

### *4.4. Energy Function*

Referring to the foundations of the Ising model, Hopfield also defined an Energy (or Lyapunov) function, which describes the current state of the neural network (Hopfield 1982). This equation is:

$$E = -\frac{1}{2} \sum \sum_{i \neq j} T_{ij} V_i V_j \tag{4}$$

(Hopfield 1982)

which is the product of the value of two neurons along with the strength of the connection between them. As the neural network updates an input, this function is monotonically decreasing. If we let $\mathbf{E}_{new}$ be the new energy of the network after an asynchronous evaluation of a random neuron, and $\mathbf{E}_{old}$ be the energy of our network before this change, we have:

$$\mathbf{\Delta E} = \mathbf{E}_{new} - \mathbf{E}_{old} = -(V_i^{new} - V_i^{old}) \sum_{i \neq j} T_{ij} V_j = -\Delta V_i \sum_{i \neq j} T_{ij} V_j \tag{5}$$

(Based on Hopfield 1982)

This gives us 3 cases for $\Delta V_j$. The case that the neuron $V_j$ does not change is trivial, as $\Delta V_j = 0$, meaning $\Delta \mathbf{E} = 0$ as well.

If $\Delta V_j > 0$, this means that both $\Delta V_j$ and $\sum_{i \neq j} T_{ij} V_j$ are positive, resulting in a negative $\Delta \mathbf{E}$, due to the negative sign at the beginning of our energy function.

If $\Delta V_j < 0$, this means that $\sum_{i \neq j} T_{ij} V_j$ must be negative as well, since this is what must occur for a neuron to go from activated not no activated. Thus we have $\Delta \mathbf{E}$ is negative as well, proving that this function is monotonically decreasing.

This implication is important because it means that state changes will occur in our neural network until a local minima is reached. Because of this property of the Hopfield network, we can be guaranteed to reach a stored memory, which are local minima, based on any input. However, in some rare cases, this will not work, as we demonstrate in Section 5.

## 5. Discussion

### *5.1. Storage Capacity*

A major question Hopfield also wanted to address was finding the maximum number of stored memories $X_k$, given $N$ nodes, with minimum error recall (Hopfield 1982). This makes sense intuitively because the more memories we have, the harder it is for us to make both accurate and relevant predictions. For example, we can consider the extreme case where we have $N$ memories and $N$ neurons; this means that our network is useless, as every possible configuration of neurons is a stable point (Raj 2019). This is because each memory $X_k$ has an orthogonal memory which is also a stable point, so we have $2N$ stable points and also a total $2N$ configurations of our neural network. Therefore the model is trivial as any input will return itself as an output.

Using Monte Carlo methods, Hopfield found that this amount of memories was $0.15N$ (Hopfield 1982). However, further calculations have computed that if the number of our

memories is less than $0.14N$, we will have a less than 0.4% probability that our stored patterns (Raj 2019). To explore how we get this number, let us store $K$ memories in our neural network of size $N$ such that they are expressed in vector form:

$$X_k = [X_1, X_2, \ldots, X_k], k = 1, 2, \ldots, K.$$

In order for the Hopfield Network to recall each memory, we want each of our memories to be stable points; for this this to happen, we must have, for any memory $X_p$, the product of its field at the neuron and the value of the neuron must be positive. We can express this as:

$$V_i^p \sum_j T_{ij} V_j^p > 0, \forall i, \tag{6}$$

where

$$T_{ij} = \frac{1}{N} \sum_k V_i^k V_j^k.$$

Substituting the weights back into equation (5), we have:

$$V_i^p \frac{1}{N} \sum_j \sum_k V_i^k V_j^k V_j^p > 0, \forall i. \tag{7}$$

We can break up this summation and remove the $p$-th pattern's contribution to the overall weights and get:

$$V_i^p \frac{1}{N} \sum_j V_i^p V_j^p V_j^p + V_i^p \frac{1}{N} \sum_j \sum_{k \neq p} V_i^k V_j^k V_j^p > 0, \forall i. \tag{8}$$

The first term will always be 1, because $V_j^p V_j^p = V_i^p V_i^p = 1$ (remember $V$ only takes on values of 1 or $-1$), and the expression becomes the average of $N$ instances of 1.
The second term, which is referred to as the "cross-talk" term, must then satisfy:

$$V_i^p \frac{1}{N} \sum_j \sum_{k \neq p} V_i^k V_j^k V_j^p > -1, \forall i. \tag{9}$$

By negating this inequality, we come up with a condition that we cannot store $K$ memories in a Hopfield network of $N$ neurons. Expressing each cross-talk term as $C_i$, we then have:

$$C_i^p = \frac{1}{N} \sum_j \sum_{k \neq p} V_i^p V_i^k V_j^k V_j^p < -1. \tag{10}$$

As $N$ and $K$ become large, the distribution of the condition $C_i^p < -1$ becomes Gaussian, with a mean of 0 and variance of $K/N$. Thus given that $C\ N(0, K/N)$, we have that for $K/N < 0.14$, the probability of having no negative cross-talk terms is $P(C < -1|\mu = 0, \sigma^2 = K/N) < 0.004$ (Source of calculations and equations/inequalities: Raj 2019).

The limit of $0.14N$ only holds for random memories; it is possible to store more than this amount of memories and have a valid network if we choose the memories more carefully (Raj 2019). However, these cases are not general enough to calculate and place a limit on. Additionally, while this limit ensures a maximum recall error of 0.4% , this is not necessarily ideal in real life. Having every 4 out of 1000 predictions be wrong can be very bad in real life. Overall, capacity is a primary reason why the Hopfield-based Networks have been taken over by newer neural network architectures, although there have been many studies tweaking the Hopfield Network to try and and improve capacity (Ramsauer 2020).

### *5.2. Spurious States*

Hopfield Networks rely on the Energy function to help the neural network reach stable minima, which are, during the majority of the time, going to be where we have memories located (the $X_a, X_b$, etc...). However, we also have other local minima (which are also stable) besides the stored memories, called spurious states (Raj 2019). Thus on some runs of the Hopfield Network, there may be a convergence at a system which is not a learned memory.

This problem arises as a result of the weights determined via Hebbian learning. Besides memories, their complements, or orthogonal memories (when the on neurons are off and vice versa) are also minima, along with any linear combination of an odd number of memories, such as $(X_a + X_b + X_c)$ (Edalat 2015). A graph of an example Energy Contour illustrates such a phenomenon occurring:

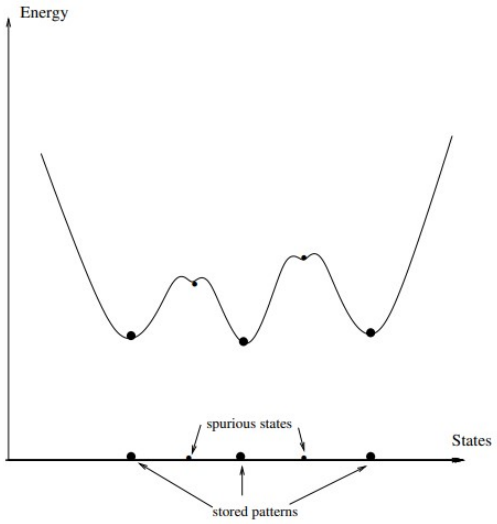


Figure 2: Presence of Spurious States in the Hopfield Network (Edalat 2015)

In order to remedy such an issue, we want to adjust our weights, $W$, in such a way such that we can minimize the energy $E$ for the various stored/trained memories $X_k$, and maximize $E$ for all other $X$ in the Energy Contour (Raj 2019). Thus rather than use the Hebbian principles to create our weights, we have to solve the problem of optimizing $W$. Gradient descent is the preferred method to address this, although this problem can also be easily avoided by choosing more modern network architectures, such as convolutional neural networks, at least in the case of pattern identification.

## 6. Conclusions

Overall, when describing neurons with simple properties, we can form a network with little structure (Hopfield 1982). By connecting the neurons in a complete, directed graph, and modeling them with biological principles such as acting potential and Hebbian learning, Hopfield was able to successfully model memory retrieval based on incomplete and/or corrupted data. The Hopfield Network can be applied in a variety of ways, such as image recognition, error correction, and much more (Hopfield 1982). Today, researchers are continually building upon and tweaking Hopfield networks, addressing many of its original problems. For example, a recent paper titled "Hopfield Networks is All You Need" features a modern Hopfield Network that can "can store exponentially (with the dimension) many patterns, converges with one update, and has exponentially small retrieval errors" (Ramsauer

2020). This network has proved to be influential as it was one of the first computational models based on neural behavior, and applied to a variety of processes.

## 7. References


Edalat, A (2015). *Hopfield Networksp [Slides]* Retrieved from Imperial London College, Dynamical Systems and Deep Learning Class. `http://www.doc.ic.ac.uk/~ae/papers/Hopfield-networks-15.pdf`.

Hebb, D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory.* New York: John Wiley & Sons.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8), 2554-2558. doi:10.1073/pnas.79.8.2554.

Raj, B (2019). N*eural Networks, Hopfield Nets and Auto Associators [Slides]* Retrieved from Carnegie Mellon University, Class 11-785, Deep Learning. `https://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Fall.2019/www.f19/document/lecture/lec17.hopfield.pdf`.

Ramsauer, H (2020). Hopfield Networks is All You Need. In *Submitted to International Conference on Learning Representations*. `https://arxiv.org/abs/2008.02217`.

Suzuki, W. A. (2005, February). Associative Learning and the Hippocampus. *Pyschological Science Agenda*. doi:10.1037/e400222005-005.