Hopfield Networks

Joshua Hsueh, John Kim

May 2020

1 Introduction

Hopfield networks are a type of recurrent neural network that uses associative memory. The idea of Hopfield networks has been around since the 1970s and is still useful for pattern recognition. In this lecture we refer to two color inputs, white and black, but Hopfield networks can work with any colors it just has to be only two.

2 Hopfield Network Overview

Given a corrupt or fragmented black and white image, the Hopfield network will reconstruct the input into one of its previously learned patterns. To put it into perspective, say you were listening to Post Malone's album, Hollywood's Bleeding, which reminds you of the time you went to his concert last summer. This is associative memory which Hopfield networks are trying to replicate.

2.1 Structure

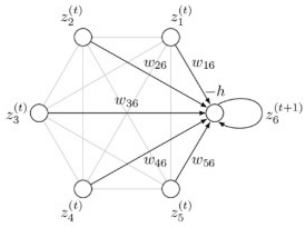

Figure 1: Hopfield Network Structure

All nodes are interconnected. Each node is an input for all other nodes. All other nodes also serve as inputs for that node. Since it uses the outputs of these nodes as inputs for other nodes, Hopfield networks are a form of recurrent neural networks. Conventionally, these nodes take on values of either -1 or 1 (binary).

The connection, weight, between any two nodes for a certain pattern is just the value of one node times the value of the other. This connection is symmetrical, meaning that for nodes i and j, $W_{ij} = W_{ji}$. A node is never connected to itself which is denoted as $W_{ii} = 0$. A fully drawn out weight matrix for one pattern should look something like this:

$$V = \begin{bmatrix} +1 \\ -1 \\ -1 \\ +1 \end{bmatrix} \quad W = \begin{bmatrix} 0 & -1 & -1 & +1 \\ -1 & 0 & +1 & -1 \\ -1 & +1 & 0 & -1 \\ +1 & -1 & -1 & 0 \end{bmatrix}$$

Figure 2: Example Weight Matrix

But let's say that we need our Hopfield network to be able to recognize multiple patterns. We would have multiple of those weight matrices above, but how would it learn all of them?

2.2 Learning

A Hopfield network learns by updating the weight matrix according to various input patterns. The new weight matrix equals to the old weight matrix added to a new input weight matrix with a learning rate applied to it. In order to maintain $W_{ii} = 0$, we subtract the identity matrix from the new weight matrix.

$$\mathrm{W}^{\mathrm{new}} = \mathrm{W}^{\mathrm{old}} + \mathrm{\mathcal{E}}(\mathrm{p}^{*}\mathrm{p}^{\mathrm{T}}\text{-}I)$$

Figure 3: Equation to Learn New Patterns

This learning method is called a Hebbian Learning rule, which is inspired by our current understanding of biological neural functions. Because we never update the neural network with a known correct output, Hopfield Networks are unsupervised neural networks (there is no teacher for the network). Let's understand why the Hebbian Learning Equation works. Consider a Hopfield Neural Network that has learned three patterns. Two of the patterns learned has a black pixel at position [1][1], while the last pattern has a white pixel at [1][1]. If we were to test and try to reproduce one of those patterns there is a high likely hood that the pixel at [1][1] should be black. This rule can be easily understood as, "Neurons that fire together, wire together."

2.3 Testing

Testing begins with a distorted image for which we want to recall a learned pattern for. To begin, we will flatten (transform into a 1D array) the image into an initial array, -1 for white pixels and 1 for black pixels. Then we select

a random neuron to fire first. This just means that the neuron value will be updated according to the equation found below.

$$x_i^{new} = \sigma\left(\sum_{i \neq j} w_{ij} x_j^{old} - \theta_j\right)$$

Figure 4: Equation to Learn New Patterns

We repeat this process until a learned pattern is fully recovered. Let's take a look at a demonstration:

First, we begin by inputting images for the Hopfield network to learn. So for example, an input matrix for an image could look something like this after being flattened: [1, -1, -1... , 1]. 1 would represent pixels that are black and -1 would represent white pixels. If we were to feed that matrix into our Hopfield network, every node i in our network would take on the value of weight-matrix[i]. N_1 would be 1, N_2 would be -1, N_3 would be -1, and so on. To update, we just use the equation in Figure 4 to produce the new updated value for the randomly selected node.

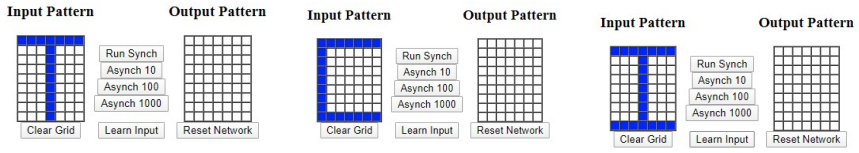

Figure 5: We train three patterns for the letters: "T", "C", "I"

Next, we have a new image we wish to produce a pattern for. It looks like a distorted 'I'. As we fire 100 neurons at a time (Asynch 100), the distorted image slowly transforms into the pattern 'I' we trained.

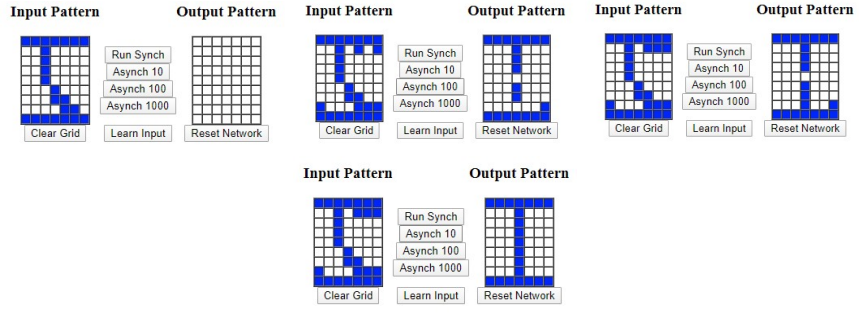

Figure 6: Firing 100 neurons at a time until a pattern is recovered

3 Energy

Energy is a scalar value representing the state of a Hopfield network. States are where the network is at during the process of reproducing a learned image. In Figure 10, these are a few examples of the different states the network goes through before reaching the final locally minimum state. After testing, the energy of a Hopfield network will converge to a local minimum. Below is a visual representation of energy for different states of a network. As you can see, we will start at some point on the graph with a new input and each update will attract it to a local minimum that we return as the "recovered pattern".

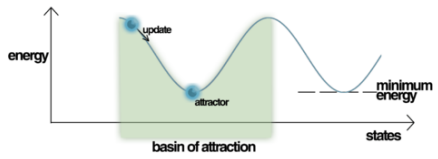

Figure 7: Energy graph with labeled parts

For every Hopfield network, we have a total energy defined by:

$$E = \frac{-1}{2}\sum_{j=1}^{n}\sum_{l=1}^{n} w_{jl} x_j x_l + \sum_{j=1}^{n} \theta_j x_j$$

Figure 8: Energy equation

Because we want to converge to a value that we reproduce as a recovered pattern, the energy of the network always decreases after a neuron fires. We can prove this by looking at the differences in energy before and after. To find the difference in energy between two states, we use:

$$E^{new} - E^{old} = -\left(x_i^{new} - x_i^{old}\right)\left(\sum_{j=1}^{n} w_{ij} x_j - \theta_i\right)$$

Figure 9: Difference of Energy Equation

Every time a neuron fires, our network energy decreases. Let's prove this. For a randomly chosen neuron, this is the net energy change. From this equation, only x_i can change because we chose one neuron to update while all other factors remain the same. We have two conditions to satisfy for this proof.

First, if x_i^{new} ¿ x_i^{old}, the first term is positive and implies that the sum of the weighted inputs was greater than the threshold, making the second term positive as well. The negative sign at the front negates this product to have a decrease in energy.

Secondly, if x_i^{new} ¡ x_i^{old}, the first term is negative. This implies that the sum of the weighted inputs was less than the threshold, making the second term negative as well. With the negative sign at the front, we again have a negative value to show a decrease in energy.

As a result, the energy of a network always decreases after a neuron is fired. Therefore in another sense, a pattern will be reached when the energy is at it's lowest possible amount. Going back to our demonstration, we can see that our energy decreases from the initial energy level until a pattern is recovered.

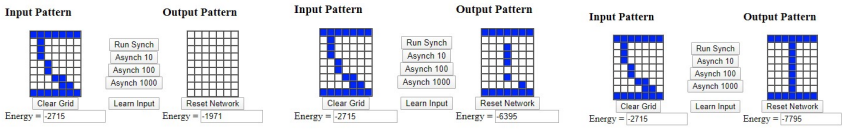

Figure 10: Energy decreases after every fire

4 Applications/Limitations

Because of their capabilities in pattern recognition, Hopfield networks can be applied to many fields in need of image recognition. For instance, Hopfield networks are currently being used to enhance X-ray images and restore medical images. These networks also provide a model for learning how human memory functions. These networks are pretty obsolete because of its limitations, but serve as a good knowledge foundation for other reputable networks such as Deep Belief Networks and Boltzmann Machines.

But like many other neural networks, Hopfield networks have their faults. Like we learned previously, feed forward neural networks can reach local minimum which hinders the neural networks accuracy. Something similar can be found in the form of spurious states in Hopfield networks. When decreasing in energy, we can reach local minima which produce different patterns than expected. This can include patterns that we previously trained or even totally new patterns.

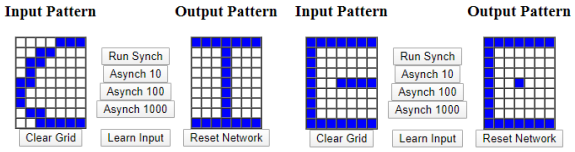

Figure 11: Examples of Wrong Outputs

Here is a graphical representation of why spurious states occur in the energy graphs

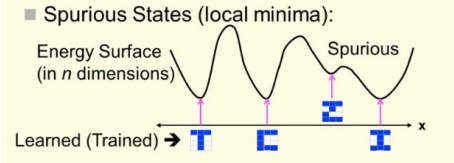

Figure 12: Graph of Spurious States

5 Conclusion

Hopfield networks are a form of recurrent neural networks that use unsupervised learning and associative memory to reconstruct a pattern for a given input. However, these networks could produce a wrong pattern because they converge to local minima. This lecture should have given you an insight into the functions for Hopfield networks.

6 Acknowledgements

https://www.researchgate.net/figure/Schematics-of-a-Hopfield-network-update-step-a-Classical-Hopfield-network-update-Here$_{f}ig2_{3}$21417576

$http://web.cs.ucla.edu/\ rosen/161/notes/Hopfield.html?fbclid = IwAR3_{D}-LBrZXO9Bd_{I}0dhZMX3mECzW1pja6JCnC4xlJAhVIsIHIObIHmB2p0$

$https://www.youtube.com/watch?v = gfPUWwBkXZY$

$http://faculty.etsu.edu/knisleyj/neural/neuralnet2.htm$

$https://towardsdatascience.com/Hopfield-networks-are-useless-heres-why-you-should-learn-them-f0930ebeadcd$