|  |
| --- |
|  |
| Hopfield Networks |
| EECS 6980: Neural Networks: Theory and RF/Microwave |

|  |
| --- |
| Brett Snyder  4/3/2012 |

# Introduction

Neural networks were designed on analogy with the human brain. However, the brain’s memory works by association or in a content-addressable fashion. For example, we can recall a complete sensory experience, including sounds and scenes, after hearing only a short piece of music, or recognize a familiar face in an unfamiliar environment almost instantaneously (Negnevitsky, 2005). The Hopfield neural network is an artificial neural network which is able to store and retrieve patterns in a fashion similar to the human brain. Further, the Hopfield network is capable of recovering full patterns when only partial information is supplied.

A Hopfield network is a sub-type of recurrent neural network (RNN) with symmetric weights, binary threshold units and random updating (Reil). A RNN has feedback connections between its outputs and inputs which provides a sense of time and memory of previous state(s) leading to the aforementioned associative behavior. A Hopfield Network is specifically a fully connected RNN, in which each neuron is connected to every other neuron (no self-connections). The units in the Hopfield network are binary threshold units—either “on” or “off” (Rojas, 1996). The Hopfield network uses either bipolar(-1, +1) or binary (0,+1) logic levels; yet, we will adhere to the bipolar convention.



Figure 1. Hopfield Neural Network (n-neurons)

Hopfield networks are asynchronous networks in which each neuron computes its excitation and changes its state at random times independent from all other neurons according to the sign of its total excitation. As network training progresses the energy function of the Hopfield network is monotonically decreased and trained states become attractors (Rojas, 1996). Whenever a state is stored in the Hopfield network its complement state is also stored.

Hopfield networks are guaranteed to converge to a local minimum and hence stable state(s) (local minimum in the energy function), but are not guaranteed to converge to a stored pattern (Trivedi). Thus, the goal of training the Hopfield network is to lower the energy of states that the network is supposed to remember. This will allow the network to function as a content addressable memory system that will converge to the stored patterns even if it is only given part of the state (Rojas, 1996).

# Activation Algorithm

The Hopfield network uses binary threshold neurons with the sign activation function as its computing element. (Negnevitsky, 2005)

# C:\Users\Brett\Dropbox\EECS 6980-002 NN\sign activation.png

Figure 2. Sign Activation Function

# Hebbian Learning

If we want to “store” *m* unique states or patterns in the network we need to find the proper weights for all of the network connections. The simplest way to train the Hopfield network is to use what is known as Hebbian learning or “one-shot” learning. Hebbian learning is carried out by iteratively processing each of the *m*, n-dimensional states and adding their individual contribution to the weight matrix (initially set to the zero matrix) as in the following equation:

where

Due to the restriction that it is apparent that the weight matrix diagonal must be a zero diagonal.

We can then rewrite the weight update equation as a matrix equation as follows:

where

Sometimes Hebbian learning is not adequate to find a weight matrix for which all m of the states will be stable states even though such a weight matrix may exist. This is especially true when the state vectors to store lie near each other. In this case other learning rules may be used such as variants of perceptron learning (Rojas, 1996).

# Perceptron Learning

A state to be memorized by the network will only be a stable state if after being stored in the network, the network global state remains unchanged. This occurs if for every neuron the excitation has the same sign as the current state, which yields the following n inequalities for a n-neuron Hopfield network with thresholds θi:

The factor sign(xi) is used in each inequality to obtain equations which all utilize the less than sign. Only the n(n-1)/2 non-zero entries of the weight matrix as well as the n thresholds show up in the inequalities. Let **v** denote a vector of dimension n + n(n-1)/2 whose components are the non-diagonal entries wij of the weight matrix **W** (with i<j so that each weight only appears once) and the n thresholds with negative sign. The vector **v** is then given by:

The state vector **v** is transformed into n auxiliary vectors **z1**, **z2**, … , **zn** of dimension n + n(n-1)/2 given by the expression:

)

Now the previous inequalities can be rewritten in the following form:

This last set of inequalities shows that the problem is solved by computing a linear separation of the vectors **z1**, **z2**, … , **zn**.

In the case where M state vectors **x1**, **x2**, … , **xm** are given to store in the network, we use the above transformations for each one. There is then *nm* different vectors which must be linearly separated. If they vectors are in fact linearly separable, perceptron learning will find the solution to the problem. The perceptron learning algorithm updates only the weights of the edges attached to a single unit and its threshold on each iteration (i.e. if **z1 . v** has an incorrect sign, then only the weights w12, w13, …, w1n and the threshold θi are updated). During training all units are set to the desired stable states. If the sign of a unit’s excitation is incorrect for the desired state, then the weights and threshold of this respective perceptron are corrected (Rojas, 1996).

# Storage Capacity

The number of patterns that can be stored in a Hopfield model is determined by the total number of neurons in the network. It was shown by Hertz, Krogh, and Palmer that the retrieval of stored states will be effective if the number of states, M, follows the following relationship: where n is the number of neurons in the network. For example, 1000 neurons are required to successfully store 138 patterns (Mehrota, Mohan, & Ranka, 1996).

Realistically, more patterns can be stored while still retaining reasonable recall accuracy. Yet, it must be noted that the practical applications of Hopfield networks are severely limited by its poor storage capacity. If a network is overloaded with states, it will not converge to clearly defined attractors (Mehrota, Mohan, & Ranka, 1996).

# Complexity of Learning

Due to the equivalence of Hopfield networks and perceptrons, every learning algorithm for perceptrons can be transformed into a learning algorithm for Hopfield networks. The learning problem for Hopfield networks can then be solved in polynomial time, since learning algorithms for perceptrons exist where the complexity grows polynomially with the number of states and their dimension (Rojas, 1996).

# Popular Applications

* Recalling or reconstructing corrupted patterns
* Error correction
* Large-scale computational intelligence systems
* Handwriting recognition
* Traveling salesman problem

# Examples

# References

*Hopfield network.* (2012, March 28). Retrieved March 29, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Hopfield\_network

Heaton, J. (2007, December 23). *The Hopfield Neural Network.* Retrieved March 30, 2012, from Heaton Research: http://www.heatonresearch.com/articles/2/page5.html

(n.d.). Hopfield and recurrent networks. In http://www4.rgu.ac.uk/files/chapter7-hopfield.pdf. Robert Gordon University.

Mehrota, K., Mohan, C., & Ranka, S. (1996). *Elements of Artificial Neural Networks (Complex Adaptive Systems).* A Bradford Book.

Negnevitsky, P. E. (2005). Artificial neural networks.

*Neural Network to Solve Traveling Salesman Problem.* (n.d.). Retrieved March 30, 2012, from Slideshare: http://www.slideshare.net/samisounda/neural-network-2009851

Reil, T. (n.d.). *Artificial Neural Networks.* Retrieved March 27, 2012, from web.cecs.pdx.edu/~mperkows/.../2005/L005.Neural\_Networks.ppt

Rojas, R. (1996). Neural Networks. Berlin: Springer-Verlag.

Smistad, E. (2010, February 27). *Hopfield network.* Retrieved March 28, 2012, from The Big Blob: http://www.thebigblob.com/hopfield-network/

Trivedi, U. (n.d.). *Hopfield Networks.* Retrieved March 27, 2012, from home.iitk.ac.in/~mfelixor/Files/Hopfield%20Networks.pptx