Hopfield Networks

Darren Hou, 1330362

Donghui Mai, 1335334

Ziqian Huang, 1237984

AMATH 383, Autumn 2015

Abstract

The central nervous system in humans in a powerful processing tool, and artificial neural networks attempt to use similar circuitry to achieve the same results. The Hopfield neural net, first formulated in 1982, provides a content-addressable technique to restore an incomplete state from previously trained patterns. Here we use Hebbian learning as the training method for later recall. We analyze and prove the stability of state, and use the Lyapunov energy function to justify and prove convergence to a steady state solution. Improvements to the original Hopfield model are suggested involving storage capacity and alternative learning algorithms. By doing so, we use the results with an implemented Hopfield net to test predictions and model changes, demonstrating some practical uses of such a system.

Introduction

The central nervous system in humans is a powerful processing tool. Interconnected nodes called neurons process and transmit electrical and chemical signals via synapses to other cells (Sanger, 1982). These connections drive sensory input and output, relaying messages throughout the body. Together, this system of individual processing elements forms a biological neural network that is adaptive and able to learn and produce meaningful results (Fukushima, 1988).

But although biological networks can be useful, they are also difficult to replicate. Artificial neural networks, or neural nets, attempt to reproduce the processing ability using simplified models, replacing neurons with easier-to-handle nodes (Lippmann, 1987). These neural nets often contain parameters that can be tuned by training algorithms. By doing so, they aim to approximate nonlinear functions of their inputs through the interconnection of basic

variable elements. Sets of neuronal nodes are layered over each other, accepting incoming signals and producing output signals (Schmidhuber, 2014). There may be one or more of these layers, divided into input, hidden, and output, that together solve a specified task.

These neural nets are preferred in certain situations over traditional processing as they allow for the exploration of competing hypotheses simultaneously through parallel computation. Robustness is also a benefit, as a few nonworking nodes in the system can still produce results as a whole. But while low-level tasks such as speech and image visualization can be carried out with little effort by the human brain, while they remain difficult tasks for computers (Lippmann, 1987; Zhou et al., 1988). For such problems with simultaneous constraints, brute force and conventional artificial intelligence techniques have so far yielded few results. However, a neural net which mimics human thought processes may prove to be better suited for these questions. The recent introduction of more-powerful networks, training, and computing has paved the way for the development of the field and has implications in areas such as computer science, neuroscience, physics, and mathematics, among others.

One example of a neural net is the Hopfield network, as specified by John Hopfield in 1982. The system is based on neurobiological concepts, but can be readily implemented by integrated circuits (Widrow et al., 1994). Hopfield originally sought to create a content-addressable memory technique, which retrieves an element previously stored from incomplete information. This is especially useful in situations where there is some expected output, such as in pattern recognition or restoration (Stauffer et al., 2003; Paik and Katsaggelos, 1992; Widrow et al., 1994).

Hopfield neural networks use binary threshold nodes as both inputs and outputs. The state of the nodes are updated during the process according to a specific function in terms of the

original state of the unit j, s_j, the threshold of the unit i, θ_i, and the weight of connection from unit j to unit i, w_{ij}. Updates to the network can be performed synchronously or not, depending on whether one or all nodes are changed at once. Hebbian training, introduced by Donald Hebb in 1949, is used extensively for the Hopfield net to store patterns locally—so that weights are dependent on neurons around the target as well as the target itself—and incrementally—so new patterns can be trained just from information about previously trained patterns.

This paper investigates the mathematical basis behind the Hopfield neural net and its implications for image recognition. We construct an example of such a network and analyze the convergence dynamics. By doing so, we aim to find the basis of attractions and its repercussions, including speed and direction of convergence. This also relates to the network's energy function, which we explore the ramifications of. We also plan to use the constructed Hopfield network to test predictions of our models, which will hopefully shed insight on various modifications we can make to improve them. We discuss the scalability of algorithm and to see how we can improve it by comparing the number of iterations to convergence with the corresponding sizes. Finally, we intend to examine any improvements on the Hopfield network, which has applications in associative memory, image restoration, and solving optimization problems (Hopfield 1982).

Restrictions

There are several deviations we take in this model from a biological neural network. In the real world, the connections between neuron i and j are directed rather than undirected; w_{ij} is not necessarily equal to w_{ji}. However, Hopfield's initial investigation as outlined in his 1982 paper discovered that if $w_{ij} \neq w_{ji}$, errors in convergence tend to increase while stable minima

are still generated. Therefore we will assume in this paper that the equality holds. Additionally, we assume no self-edges, so that $w_{ii} = 0$.

Asynchronous pseudorandom updating is favored as it guarantees to not favor any node while simulating the firing of neurons themselves. Synchronicity also may lead to oscillation between two states without ever converging (Cheung 1987).

This model deals with binary threshold units and does not accept input other than zero and one. There are other implementations of the Hopfield neural net with thresholds of negative one and one (Easton and Goodyear, 1991; Jeney and Levendovszky, 2000), but they will not be discussed here. There also exist neural networks with more than two threshold units (Haider 2008, Hezave et al., 2012), but they do not pertain to the Hopfield model.

At first glance, this idea shares some similarities with the Perceptron (Rosenblatt 1958; Widrow and Lehr, 1990), but a closer investigation highlights key differences. For instance, neural connections in a Perceptron are linked in only the forward direction; Hopfield nets use back-coupling to achieve its results. Perceptrons also use synchronous neurons, while this model does not. Indeed, this represents a key improvement on earlier attempts at modeling biological brain function.

Mathematical Model

Let a collection of processing devices (nodes) have two possible states as described by McCullough and Pitts: $v_i = 0$ or $v_i = 1$. If a connection between one neuron v_j to another v_i is established, a strength of connection is set as w_{ij}. The pattern is updated repetitively until none of the units' states change. When a pattern no longer changes, the network is in its final stable

state. According to Hopfield, convergence is guaranteed as long as the weight matrix has a zero diagonal, so that the network is assured to converge.

Training

Training a Hopfield network means updating the connection weight between units in a way so that the state of the network will be closer to a desired state. This involves iterating until convergence, so that the energy level of the network decreases over time. A properly trained network will regress to a state that it is trained to "remember," even if incomplete. For example, a network with five units is trained with a desired state (1, 0, 1, 0, 1). If its initial state is (0, 1, 1, 0, 1), after updates, it will converge to the state (1, 0, 1, 0, 1). Therefore, a trained network can be used as a memory system to restore a distorted pattern to a remembered one.

There are various learning rules that can be used to train Hopfield networks. In order for a learning rule to be plausible, there should be some restrictions. One, the rule must be local, meaning each connection weight is updated using the information available to the units on either side of the connection. For instance, when updating w_{ij}, the values of unit i and j, and the values of all the units that are connected to unit i and j will be used to determine the new value of w_{ij}. Two, the rule must be incremental: when a new pattern is used to train, the updated values of connection weight only depend on the old values and the new pattern. The old pattern that has been used to train is not relevant in this case. The Hebbian rule conforms to both of these properties (Hebb 1949).

Hebbian learning attempts to explain associative memory. It describes that when a number of neurons are stimulated, the synaptic strength between them also increase. For the Hopfield network, the Hebbian leanring rule can be used to learn m different n-dimensional

patterns (i.e. stable states) $V^1, \dots, V^n$, where each V^s is in binary vector form. It is implemented by updating the network's connection weight according to the following rule with the initial $w_{ij} = 0$ for all i and j:

$$w_{ij}^k = v_i^k v_j^k, i \neq j$$

$$i, j = 1, 2, \dots, n,$$

$$k = 1, 2, \dots, m$$

where w_{ij}^k represents the weight after the presentation of the pattern V^k, and v_i^k and v_j^k represent the i-th and j-th elements of the pattern V^k. Therefore, the value of w_{ij} after learning all m different patterns is the sum of w_{ij} for each pattern. That is,

$$w_{ij} = \sum_{k=1}^{m} v_i^k v_j^k$$

when $i \neq j$ and $i, j = 1, 2, \dots, n$. In the case of $i = j$, $w_{ij} = 0$.

From the rule above, it is clear that the Hebbian rule is both local and incremental. Each update uses information available to the units adjacent to the particular weight. Also, the updated value for each weight only depends on the old value and the corresponding components of the new pattern.

Recall

Suppose a set of states V^s where $s = 1, 2, \dots, m$ is provided to be stored for later recall, and V is a distorted state to be restored. With the storage matrix W consisting of elements w_{ij} created with the Hebb rule, let the weighted sum S_w of each element i in V, V_i, be defined as

$$S_w = \sum_{i \neq j} w_{ij} V_i^s$$

If S_w is greater than or equal to some threshold θ_i, set $V_i \to 1$. Otherwise, $V_i \to 0$. We will choose $\theta_i = 0$ for the purposes of this paper.

Given that neurons in a human brain fire at varying times, information is to be updated asynchronously in our model. So for a V with n states, we create a random ordering of the states $1, 2, \dots, n$ to update every iteration of recall. For example, given $n = 5$, the nodes of V may be updated in the order $2, 1, 3, 5, 4$. This proceeds until V resolves into a stable configuration for two iterations.

Energy

There is a scalar value associated with each state of the Hopfield network, called energy and denoted E. Energy is the certain amount of cost that is required by the state and paid by the system in order to reach the state. It is a function of state and it is a representation of the activity dynamics of the system. The energy function for the Hopfield network is actually a Lyapunov function, explained by Nasrabadi in 1992, which has the equation:

$$E(V) = -\frac{1}{2}\sum_{i,j} w_{ij} v_i v_j + \sum_i \theta_i v_i$$

E is the energy level at state V where $V = v_1, v_2, \dots, v_n$. Each time the units in the network are randomly selected to update, the energy level of the network E will either decrease or stay the same. It is indicated that after repeated updates, the final stable state to which the network evolves is indeed a local minimum of the Lyapunov function. If a state is a local minimum of the function, it is a stable state of the network. Thus, the Hopfield network is actually an optimization problem. Furthermore, the network tends to start with a higher energy

and then go to a state with lower energy, because after every update the energy E cannot increase.

Solutions and Results

An implementation of the Hopfield net is created and used to analyze certain aspects of the model and produce quantitative data. We analyze the Hebbian learning rule in regard to the stability of the network. We then utilize a provided Lyapunov function to calculate energy and prove convergence of every node in the state, and thus the entire pattern.

Implementation

In order to further examine the properties the Hopfield network, we have developed a Python program that implements its updating and training algorithms mentioned in the previous section. In our program, the network is imprinted with the patterns of the letter "A," "E," "Z," and a stripe pattern. As the two examples shown below, when given a randomly distorted pattern, the program is likely to produce one of the patterns mentioned above.

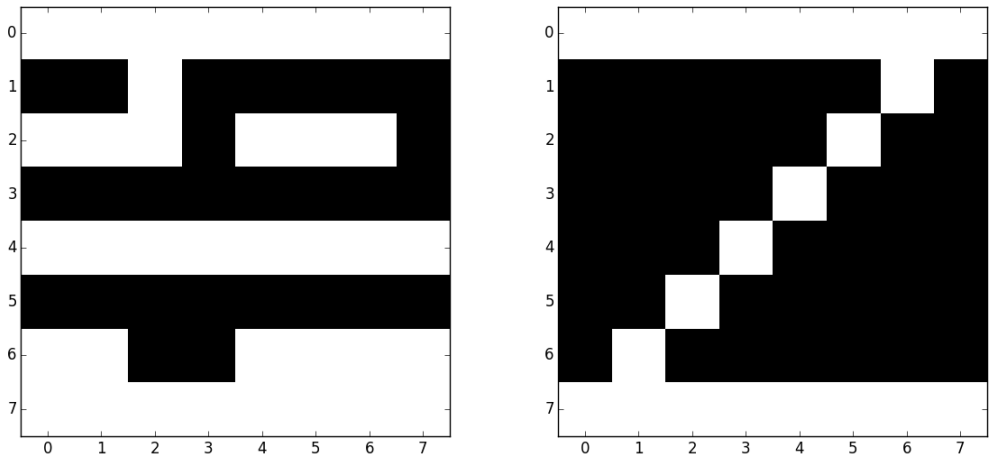

Figure 1: Graphical convergence from distorted (left) to not (right). Intermediate iterations are not shown.

In our program, each time the distorted pattern actually comes from one of the trained patterns. For instance, if given a distortion of "A", it is natural to guess that the final output would also be "A". However, this is not always true. The final pattern can be something else. Since the network is trained with four patterns, the final output will be the one that it first converges to. Sometimes the result may not even be one of the trained patterns. If the network reaches a stable state before reaching any one of the four, it would stay there instead.

The Hopfield network always starts with a higher energy level and then goes downs. In our program, after each update of the network, the energy E is calculated. Given an initial state, with repetitive updates, the energy level of the network decreases monotonically.

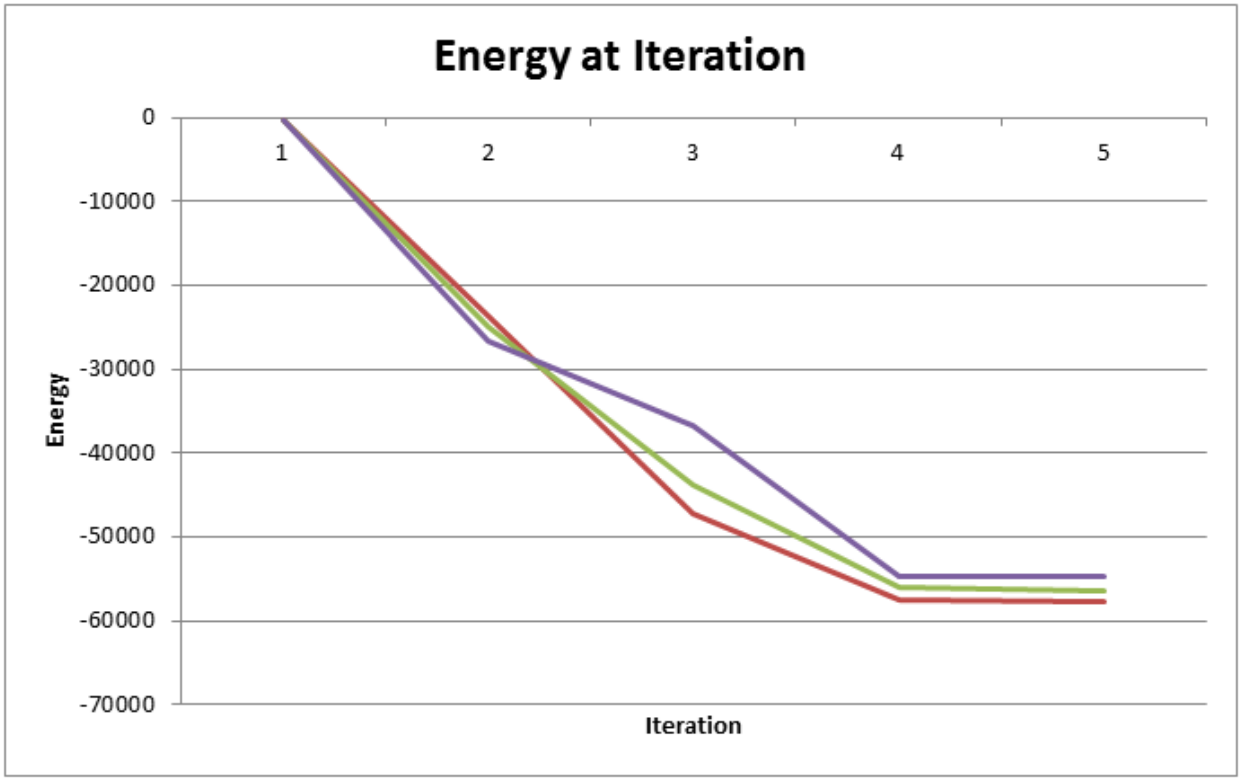

Figure 2: Energy dynamics of the network for convergence of three separate patterns

When the network reaches a stable state, it normally stays there until the state is manually changed. If the change is small, the network will always go back to the stable state. If the change is so large that it passes the threshold, it will instead go to the next stable state.

Stability

The Hebbian rule is applied to each element in the weight matrix of the network. When applying this rule to the whole weight matrix at the same time, it becomes:

$$W_k = (V^k)^T(V^k) - I, k = 1,2, \dots m$$

where I denotes the $n \times n$ identity matrix. Because no unit in the Hopfield network has a connection to itself, the weight matrix has a diagonal comprised of zeros. The subtraction of the identity matrix guarantees that property, as for any binary vector V^k, $V_i^s \times V_i^s = 1$. In the case of m different patterns $V^1, V^2, \dots, V^m$,

$$W = \sum_{k=1}^{m} (V^k)^T(V^k) - I$$

The effectiveness of the Hebbian learning rule can be easily proved. After learning the k-th pattern V^k, the weight matrix is given by:

$$W_k = (V^k)^T(V^k) - I$$

Then the minimum of the energy function E of the Hopfield network with the weight matrix W_k is located at V^k because

$$E(V) = -\frac{1}{2} V W_k V^T = -\frac{1}{2}(V(V^k)^T V^k(V^k)^T - VV^T)$$

and $(V^k)^T(V^k) = n$ for binary vectors. Thus the function

$$E(V) = -\frac{1}{2}\|V(V^k)^T\|^2 - \frac{n}{2}$$

has a local minimum at $V = V^k$, which means that V^k is a stable state of the network.

Energy and Convergence

Using the Lyapunov function described by Nasrabadi and Choo (1992):

$$E = -\frac{1}{2}\sum_{i,j} w_{ij} v_i v_j + \sum_i \theta_i x_i$$

Since $w_{ij} = v_i v_j$,

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

But all $v_{i,j}$ are either 0 or 1, so the maximum of E at each i, j is when $v_i = v_j = 1$.

$$E \leq -\frac{1}{2}\sum_{i,j} 1$$

$$E \leq -\frac{1}{2}(n-1)^2$$

Thus there is a finite energy at each node.

Now let $E(t)$ be the energy at a certain time during recall. When updating neuron v_k,

$$E(t) = -\frac{1}{2}\sum_{i,j} w_{ij} v_i v_j + \sum_i \theta_i v_i$$

$$E(t) = -\frac{1}{2}\left(\sum_{i,j,i\neq k,j\neq k} w_{ij} v_i v_j + \sum_i w_{ik} v_i v_k + \sum_j w_{kj} v_k v_j\right) + \sum_i \theta_i v_i$$

So the energy will be $E(t+1)$ after one iteration. During this time, v_k will be denoted v_k'.

$$E(t+1) = -\frac{1}{2}\left(\sum_{i,j,i\neq k,j\neq k} w_{ij} v_i v_j + \sum_i w_{ik} v_i v_k' + \sum_j w_{kj} v_k' v_j\right) + \sum_i \theta_i v_i$$

Then for the change in energy ΔE,

$$\Delta E = E(t+1) - E(t)$$

$$\Delta E = \frac{1}{2}\left(-\sum_i w_{ik} v_i v_k' - \sum_j w_{kj} v_k' v_j + \sum_i w_{ik} v_i v_k + \sum_j w_{kj} v_k v_j \right) + \theta_k \left(v_k' - v_k \right)$$

Since $w_{ij} = w_{ji}$,

$$\Delta E = \left(\sum_i w_{ik} v_i - \theta_k \right) (v_k - v_k')$$

If the node k does not change, then $(v_k - v_k') = 0$ and $\Delta E = 0$.

If the node k changes from $0 \to 1$,

$$\Delta E = \left(\sum_i w_{ik} v_i - \theta_k \right) (0 - 1)$$

which is negative since $w_{ik} v_i > \theta_k$ (so $\sum_i w_{ik} v_i - \theta_k$ is positive) for the node to be attracted to 1, under the definition of a Hopfield network. Conversely, if the node k changes from $1 \to 0$,

$$\Delta E = \left(\sum_i w_{ik} v_i - \theta_k \right) (1 - 0)$$

which is also negative as $w_{ik} v_i < \theta_k$ (so $\sum_i w_{ik} v_i - \theta_k$ is negative). Thus $\Delta E \leq 0$.

A steady state exists when $\Delta E = 0$. This occurs when the system undergoes no changes after an iteration, which is the definition of convergence. This is reflected in our model as the number of possible states is finite, so the network must eventually reach a state at which the energy can no longer decrease. Thus steady states exist at local minimums.

This agrees with our observation in the implementation, as we see the energy of the network strictly decrease across the iterations.

Improvements

Although the Hopfield network has been shown to converge quickly, with an upper bound of $O(\log(\log(n)))$ (Komlós and Paturi, 1988), its ability to store states for later recall could use improvement. Here we discuss several problems with storage in the Hopfield network and possible improvements.

Storage Capacity

Suppose many patterns are needed to be stored in the weight matrix, possibly more than it can hold. It would be reasonable to assume some sort of upper bound on the network's storage capacity. MacKay demonstrated in his 2003 paper that randomly choosing weights and setting them to zero, still results in stable states though fewer of the patterns are recovered correctly to the desired states. Thus the network can still work as associative memory even when damaged. However, not all of the distorted patterns can be recovered to the originals, and storing too many results in associative failing since it is easy for the pattern to flow into spurious stable states. Hopfield (1984) suggests there exists a maximum stored number of the patterns N_{max}, with I neurons, is $0.15I$. It was showed later by Amit el al. (1985) that

$$N_{max} = \frac{I}{4\ ln\ I + \ 2ln(1/\varepsilon)}$$

$N_{max} = 0.138I$ is considered to be the critical value of the number of patterns stored to the number of neurons (Wei and Yu, 2000). If N/I ratio is large, the process might end up a different result from the desired pattern. If N/I ratio is small, there should be a stable state very close to each desired pattern. The discontinuity between the two behaviors occurs at the critical value of N, 0.138I.

In order for the associative memory to be useful, it should be able to correct at least one flipped bit. MacKay gives an objective function based on the error function in the training of single neuron as a binary classifier:

$$G = -\sum_i \sum_n t_i \ln y_i + (1 - t_i) ln(1 - y_i)$$

where $\begin{cases} t_i = 1 \text{ when } x_i = 1 \\ t_i = 0 \text{ when } x_i = -1 \end{cases}$. Activation $a_i = \sum w_{ij} x_j$, and output will be the sigmoid function of a_i, $y_i = \frac{1}{1+exp(-a_i),}$. When minimizing this objective function, the learning algorithm does a better than the Hebbian learning rule, storing more patterns to recover a desired stable state (MacKay, 2003).

Optimal Solution of Convergence

Although there have been various investigations on combinatorial optimization problems using Hopfield networks and theoretical works to analyze the network dynamics, often the objective is only to make the network converge to a feasible solution rather than to an optimal one (MacKay, 2003; Storkey, 1999). As of yet, there has not been any defined relationship between the values of the network coefficient and the quality of the feasible solution the network converges to. Matsuda's 1998 paper investigates the "optimal" Hopfield network with a linear cost function. For this to happen, the cost function is

$$\min\left(\sum_i \sum_j c_{ij} v_{ij}\right)$$

where c_{ij} is defined as the unit cost and v_{ij} as a two dimensional variable matrix. As a result, a solution with a minimal cost function is defined as an optimal or nearly-optimal solution.

Storkey Learning Rule

The Storkey learning rule was first introduced by Storkey and Valabregue in 1997 and expanded on in later years. A Hopfield network trained with the Storkey rule tends to have larger capacity than the one trained with the Hebbian rule. The rule is in the following general form with initial $w_{ij}^0 = 0$ for all i and j:

$$w_{ij}^k = w_{ij}^{k-1} + \frac{1}{n}\left(v_i^k v_j^k + v_i^k h_{ji}^k + v_j^k h_{ij}^k\right)$$

where

$$h_{ij}^k = \sum_{l=1, l\neq i,j}^{n} w_{il}^{k-1} v_l^k$$

is a form of local field at unit i, under the conditions $i \neq j$, $i, j = 1, 2, \ldots, n$, and $k = 1, 2, \ldots, m$.

The Storkey learning rule is local, since each weight is updated by taking into account only adjacent units. It is also incremental, since the equation for w_{ij}^k is recursive. The information of the old pattern is not required for learning a new pattern. This rule serves as an improvement over the Hebbian as it retains all the functionalities of the Hebbian, but has a greater capacity as well.

Each learning rule has an absolute capacity defined as the following: Suppose $v(n)$ random patterns are store in the network with n units. If in the limit of $n \to \infty$, the probability of all the patterns are stable is 1, and $v(n)$ is the largest asymptotic relationship for which this to occur, then $v(n)$ is the absolute capacity of the network. According to Storkey, this new learning rule has a higher absolute capacity of $\frac{n}{\sqrt{2 \ln n}}$ compared to the Hebbian at $\frac{n}{2 \ln n}$, meaning the network can store more patterns and requires fewer iterations to converge to a local minimum. Compared to the Hebbian rule, the improvement of the capacity of the Hopfield network is

significant. In his 1997 paper, Storkey shows that at the networks of millions of units, the absolute capacity of the new rule is nearly five times as that of the Hebbian rule, as well as having faster convergence.

Furthermore, the Storkey rule is proved to have greater resilience to correlations. When given time-series correlated patterns, the Storkey rule is nearly unaffected by the correlations while the Hebbian rule capacity decays quickly as the correlations increase (Storkey, 1997).

Conclusion

This paper has investigated the mathematical basis behind the Hopfield neural net and its implications for image recognition. We used the mathematical model developed by Hopfield and analyzed certain properties, including energy, stability, and convergence. We implemented one instance of the net, using Hebbian learning, to generate visual patterns and demonstrate recall. We then suggested several improvements, including optimization of storage capacity, and an alternative learning rule.

Despite this and previous investigations, the Hopfield model remains an area to be explored. The Hebb and Storkey learning rules are by no means the only ones, and further optimizations exist. Possible modifications include the changing of thresholds and synchronicity in updating, both of which remain beyond the scope of this paper. The Hopfield net is but one example of an artificial neural network that imitates certain aspects of biological brains. Although imperfect, the model provides a foundation for investigation into natural processes and time-efficient computation.

Appendix

The implementation of the Hopfield network used in this paper can be found at https://github.com/houdarren/hopfield.

References

Amit, D. J., H. Gutfreund, and H. Sompolinsky. "Statistical Mechanics of Neural Networks near Saturation." *Annals of Physics* (1985): 30-67.

Easton, M. G., and C. C. Goodyear. "A CELP Codebook and Search Technique Using a Hopfield Net." *International Conference on Acoustics, Speech, and Signal Processing* 1 (1991): 685-88.

Fukushima, Kunihiko. "Neocognitron: A Hierarchical Neural Network Capable of Visual Pattern Recognition." *Neural Networks* 1, no. 2 (1988): 119-30.

Hebb, Donald O. *The Organization of Behavior; a Neuropsychological Theory*. New York: Wiley, 1949.

Hezave, Ali Zeinolabedini, Mostafa Lashkarbolooki, and Sona Raeissi. "Using Artificial Neural Network to Predict the Ternary Electrical Conductivity of Ionic Liquid Systems." *Fluid Phase Equilibria* 314 (2012): 128-33.

Hopfield, John J. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities." *Proceedings of the National Academy of Sciences* 79, no. 8 (1982): 2554-558.

Jeney, G., and J. Levendovszky. "Stochastic Hopfield Network for Multi-user Detection." *European Conf. Wireless Technology* 147, no. 1 (2000): 150-53.

Komlós, János, and Ramamohan Paturi. "Convergence Results in an Associative Memory Model." *Neural Networks* 1 (1988): 239-50.

Lippmann, Richard P. "An Introduction to Computing with Neural Nets." *IEEE ASSP Magazine* 4, no. 2 (1987): 4-22.

MacKay, D. J. C. *Information Theory, Inference, and Learning Algorithms*. Cambridge

University Press, 2003.

Matsuda, S. "'Optimal' Hopfield Network for Combinatorial Optimization with Linear Cost Function." *IEEE Transactions on Neural Networks* 9, no. 6 (1998): 1319-330.

Nasrabadi, N.M., and C.Y. Choo. "Hopfield Network for Stereo Vision Correspondence." *IEEE Trans. Neural Netw. IEEE Transactions on Neural Networks* 3, no. 1 (1992): 5-13.

Paik, Joon K., and Aggelos K. Katsaggelos. "Image Restoration Using a Modified Hopfield Network." *IEEE Trans. on Image Processing* 1, no. 1 (1992): 49-63.

Rojas, Raúl. *Neural Networks: A Systematic Introduction*. Berlin: Springer, 1996.

Rosenblatt, Frank. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain." *Psychological Review* 65, no. 6 (1958): 386-408.

Sanger, Terence D. "Optimal Unsupervised Learning in a Single-layer Linear Feedforward Neural Network." *Neural Networks* 2, no. 6 (1989): 459-73.

Schmidhuber, Jürgen. "Deep Learning in Neural Networks: An Overview." *Neural Networks* 61 (2014): 85-117.

Stauffer, D., A. Aharony, L. Da Fontoura Costa, and J. Adler. "Efficient Hopfield Pattern Recognition on a Scale-free Neural Network." *The European Physical Journal B - Condensed Matter* 32, no. 3 (2003): 395-99.

Storkey, Amos J., and R. Valabregue. "Hopfield Learning Rule with High Capacity Storage of Time-correlated Patterns." *Electronics Letters*, 1997, 1803-804.

Storkey, Amos J., and R. Valabregue. "The Basins of Attraction of a New Hopfield Learning Rule." *Neural Networks* 12, no. 6 (1999): 869-76.

Wei, Gang, and Zheyuan Yu. "Storage Capacity of Letter Recognition in Hopfield Networks." *Faculty of Computer Science, Dalhousie University*, 2000.

Widrow, Bernard, and M. A. Lehr. "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation." *Proceedings of the IEEE* 78, no. 9 (1990): 1415-442.

Widrow, Bernard, David E. Rumelhart, and Michael A. Lehr. "Neural Networks: Applications in Industry, Business and Science." *Communications of the ACM* 37, no. 3 (1994): 93-105.

Zhou, Y-T, R. Chellappa, A. Vaid, and B. K. Jenkins. "Image Restoration Using a Neural Network." *IEEE Transactions on Acoustics, Speech and Signal Processing* 36, no. 7 (1988): 1141-151.