1988

Neural Networks With Asynchronous Control.

Michael Christopher Stinson
Louisiana State University and Agricultural & Mechanical College

Order Number 8904571

Neural networks with asynchronous control

Stinson, Michael Christopher, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1988

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

NEURAL NETWORKS WITH ASYNCHRONOUS CONTROL

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by

Michael C. Stinson

B.S., Central Michigan University, 1971
M.S., Michigan State University, 1975
M.A., Michigan State University, 1977

August 1988

ACKNOWLEDGEMENTS

This work is a reflection of my efforts and knowledge under the guidance of Subhash C. Kak. My dissertation would not have been possible without his assistance and encouragement. I can only acknowledge this, thank him and try to reflect what I have learned from him in my life and work.

The contribution of my colleague Danial G. Foster in the area of phonology was invaluable to my work. His insight and assistance is greatly appreciated.

I would like to acknowledge: Doris Carver, Sitharama Iyengar, Powsiri Klinkachorn, Donald Kraft, and Kwei Tang for graciously serving on my committee and helping to edit and prepare my dissertation.

My fellow graduate students and the computer science staff played an important role in my life while I worked on this dissertation. I feel very fortunate to have met them.

Table of Contents

List of Figures

Abstract

Neural network studies have previously focused on monolithic structures. The brain has a bicameral nature, however, and so it is natural to expect that bicameral structures will perform better. This dissertation offers an approach to the development of such bicameral structures. The companion neural structure takes advantage of the global and subset characteristics of the stored memories. Specifically we propose the use of an asynchronous controller C that implies the following update of a probe vector x by the connection matrix T: $x' = \text{sgn}\ [\ C(x, TX)]$. For a VLSI-implemented neural network the controller block can be easily placed in the feedback loop.

In a network running asynchronously, the updating of the probe generally offers a choice among several components. If the right components are not updated the network may converge to an incorrect stable point. The proposed asynchronous controller together with the basic neural net forms a bicameral network that can be programmed in various ways to exploit global and local characteristics of stored memory. Several methods to do this are proposed. In one of the methods the update choices are based on bit frequencies. In another method handles are appended to the memories to improve retrieval. The new methods have been analyzed and their performance studies it is shown that there is a marked improvement in performance. This is illustrated by means of simulations.

The use of an asynchronous controller allows the implementation of conditional rules that occur frequently in AI applications. It is shown that a neural network that uses conditional rules can solve problems in natural language understanding.

The introduction of the asynchronous controller may be viewed as a first step in the development of truly bicameral structures that may be seen as the next generation of neural computers.

Chapter One

Introduction

The existence of forgetting has never been proved: we only know that some things do not come to our mind when we want them.

Freidrich Nietzsche (1844-1900)

Neural networks have been one of the most exciting areas of research in computer science in the 1980's. They have been investigated for applications to optimization [Hopf85, Hopf86] pattern-recognition [Hopf82, Koho84], semantic information processing [McCl81, Rume82], and vision [Marr82, Fuku82, Farh85]. Nevertheless, neural network models do suffer from several limitations. For example, an associative memory may retrieve spurious states. As optimization networks, they do not yield results that are close to the global optimum. Recent neural network models of associative memory have focused on capacity questions [Abu-85, McEl87, Rume86] . This is due to the fact that, for a given number of memories, the number of spurious states decreases as the size of the neural net is increased. In the area of optimization, several studies [Ande88] have looked at techniques of adaptive optimization; however, this particular area of research is still in its infancy. Many fundamental questions are unresolved.

This work looks primarily at the associative memory application of neural networks. Motivated by the problem of improving the performance of neural networks and minimizing spurious behavior, the work proposes a neural net model that incorporates decision making as one of its characteristics.

Another motivation for the proposed model is the biological reality of the bicameral brain, where there is interaction between two distinct networks that exercise mutual control. Since these distinct networks process information at different levels of abstraction, they cannot be replaced by a single, larger network.

The introduction of external information is made possible by the addition of a controller into the neural nets. Linking the controller to external computational components allows decisions to be made from information that would otherwise be unavailable. However, it does effect the structure and design of the neural net at the physical level. In addition, it effects the way the neural net acts to learn and retrieve information.

While the recent neural net developments in computer science are interesting, they have dealt with relatively simple models. On the other hand, biological neural systems have evolved for millions of years. The fact that animals and humans perform many tasks much better than artificial neural machines should, therefore, not be a surprise. To illustrate certain aspects of the functioning of biological neural networks, consider the case when the smell of oatmeal reminds a person of a particular cold winter morning long ago and mother is cooking breakfast long ago. This is a *recall* using partial information, that is, the smell is significant enough to recall a particular instance or situation from the stored recollections.

Animal neural systems can make mistakes. In *deja vu*, one recalls a previous experience. This experience may or may not have indeed occurred. At other times, we find ourselves with answers that originally seemed correct but later turned out to be wrong.

Animal neural systems have greatly extended our concept of a computation. These changes include new concepts for information storage and retrieval. In the traditional von Neumann machine, information is stored in a single, isolated location. One method of recall is to refer to that specific location. To determine if the data is in the system, each memory location is examined until either a match is found or all locations are disqualified. This is accomplished by a variety of techniques, some of which may speed up the process significantly. Unfortunately, there is a logical limit to the extent such techniques can speed up a search.

Since humans perform the tasks of recall very efficiently, the brain must use another procedure. It is generally accepted that biological memory is associative memory. An associative memory is one that is distributed over the entire structure. In such a memory, partial information serves to index memory. For this reason it is also called content-addressable memory.

As we have seen for human memory with the example of *deja vu*, associative memory can lead to incorrect answers. This may be due to either the probe, partial information that serves to index the memory, being unsuitable, or due to the memory system having been overloaded and driven into a regime of pathological behavior. Although animal neural nets may make mistakes, they perform surprisingly well when compared to artificial neural nets. One possible explanation for the superior performance is the flexibility with which they process information. Humans can remember rules and realize quickly whether or not these rules are applicable to a particular situation. This ability to apply conditional rules in a vast array of situations is attributed in part, to the interplay of the hemispheres of the brain. To date, artificial neural nets

have been unable to use similar conditional rules.

Artificial neural nets were originally conceived by McCulloch and Pitts [McCu43]. Their use as associative memory was shown by D.O. Hebb [Hebb49]. The most popular neural network model is that of Hopfield [Hopf82] . The Hopfield model can be operated in the synchronous and asynchronous modes. The difference between these two modes is the methodology used in updating. This dissertation uses the Hopfield model approach; a choice made partly due to the great popularity of the model and because it often serves as a benchmark for neural net research. The model proposed in this dissertation may be viewed as a variant of the Hopfield model, where a new component is introduced that functions as a controller. The purpose of the controller is to assist in the processing of information by introducing a decision making capability. The controller is used to implement a number of new algorithms that significantly improve the performance. Apart from the motivation to realize a structure similar to the bicameral brain, there is an engineering motivation for the introduction of the controller. Current models do not provide a means to interface the network with an external processor; the controller now permits this interface.

The controller has access to information that is not readily available in a standard neural net. Examples of information that is used by the controller are: the number of pieces of data stored in the system, the size of the memory, the global characteristics of memories such as frequency of particular groupings. Such information may be used to decide on information retrieval strategies; in particular, it would decide which neuron is updated in an asynchronous system. When viewing a neural net as a part of a larger hierarchical, system it may be useful to consider its functions at two distinct

levels: the physical level and the control level. The control level has the ability to add external information to the physical neural net, which permits the operation of the neural net in different modes. Furthermore, the control unit can serve as a gateway for communication with other neural nets or sequential processing systems.

In addition, the introduction of the controller permits organization of the learned memories. This organization is in a manner that leads to a more reliable retrieval system. Such organization of information is not possible with conventional neural nets. It is now possible to approach certain capabilities of biological memory that were hitherto not feasible. Retrieval using correlations is pattern-driven, as opposed to random retrieval in current methods. For example, it is possible to relate memories and to index them, thereby creating an organization that adds a new dimension to retrieval and learning.

Chapter Two provides a background of previous research. The Hopfield model is introduced and a discussion of its weaknesses and strengths is presented. The problems of spurious states and memories that interact, are described.

Chapter Three introduces the asynchronous controller. This discussion includes issues related to hardware implementation, as well as a conceptual framework to analyze the workings of the controller.

Chapter Four develops several methods to utilize the controller-based neural net. Specifically, it deals with techniques to improve the convergence to learned memories. One method uses global probabilistic information, another adds information-dependent handles that define constraints on the memories.

Chapter Five presents an application of the conditional rules. This is illustrated through a current problem in phonology. It is shown that certain problems in speech recognition can now be solved, through the use of this methodology.

Chapter Six presents a summary of this work and suggestions for further research. Several problem areas where conditional rules might be applied to an associative memory are pointed out. Some unresolved questions regarding conditional rules as applied to associative memories are also mentioned.

In summary, while neural nets make it possible to solve problems that are intractable for traditional machines, they suffer from shortcomings of their own. This dissertation proposes a new methodology that makes it possible to overcome some of the shortcomings. It is believed that this new methodology will eventually lead to the development of powerful, bicameral neurocomputers.

Chapter Two

Background

Words do not change their meanings so drastically in the course of centuries, as, in our minds, names do in the course of a year or two.

Marcel Proust (1871-1922)

2.1. Introduction

Neural networks or, more accurately, artificial neural networks, have been examined and implemented in a number of different guises. Work in the area has been done under the headings: connectionist machines [Feld82], parallel distributed processing, [Rume86] and artificial neural nets [Koho84a, Koho84b, Hint81, Fuku82, Kosk84, Gros83, Gros86] . The motivation for this research is partly the dissatisfaction among computer scientists with the functioning of many AI algorithms that are very cumbersome to execute on von Neumann architectures. The history of neural net development goes back to the pioneering works of McCulloch and Pitts [McCu43] and D.O. Hebb [Hebb49] . These early works present an understanding of many of the basic issues regarding neural nets. The Hebbian Rule for learning presented in the work by D.O. Hebb is the ancestor of many modern day neural net connection rules,

> *"When two units are simultaneously excited, increase the strength of the connection between them."*

This rule, although somewhat vague as stated, is the basis for a number of models. The rule presents a common sense approach with great flexibility. The exact manner in which the connection strength is made to reflect positive correlation can be accomplished in different ways.

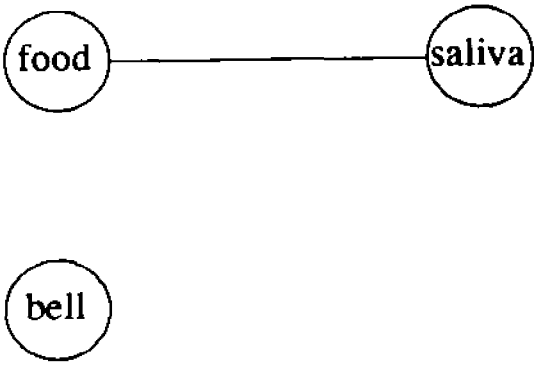


Figure 2.1

Neuron without Linkage

Pavlovian reinforcement is an example of Hebbian learning at work. At the beginning of the experiment there is no correlation between the ringing of a bell and the salivating of the dog. In other words, one may assume the connection(s) between the neuron(s) that represents the bell and the neuron(s) that represents the salivation of the animal is not made. On the other hand, a strong correlation between salivary neurons and food exists, as represented in Figure 2.1. As the experiment proceeds, a positive correlation between the bell and salivation leads to a linkage between the corresponding neurons, as shown by the graph in Figure 2.2. In fact, the linkage is eventually sufficiently strong so as to allow the dog to salivate while only using the bell.

Hebbian learning may also be restated as follows

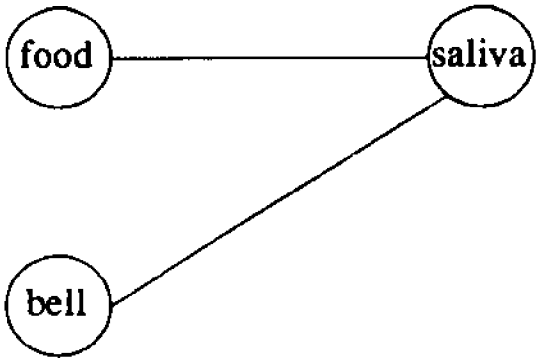


Figure 2.2

Neurons with Linkage

> *"Adjust the strength of the connection (between the two units) in proportion to the product of their simultaneous activation."* [Rume86]

The strength adjusting allows the two components to become more excitatory for positively correlated values and less excitatory for a negatively correlated values. Thus, if one component is strongly related to another in a positive manner, the value will increase in a positive way, and vice versa.

After the works by McCulloch and Pitts and D.O. Hebb, additional contributions were made by Rosenblatt, [Rose59] Widrow and Hoff [Widr60], and also Posch [Posc68] . Unfortunately, computer technology was still in a period of infancy, and, owing to the computationally intensive nature of these algorithms, their ideas lay dormant.

Meanwhile, under the alternate approach of rule based systems, a good deal of research was done regarding techniques to structure knowledge. Structures such as *frames* [Mins75] and *schemata* [Norm76, Rume75] were developed to handle large scale groupings of data [Alle87]. Another structure, known as *scripts* [Scha76], was developed to handle stereotypical causes of action. This structure demonstrated particular advantages in dealing with event sequences [Char85] . In fact, these structures still form the basis for much of the artificial intelligence research being done in this area. They provide interesting insights to the understanding of associative learning in humans, and form the framework for many advances in the area of comprehension.

The field of neural nets experienced a resurgence in the 1980's, with the introduction of a new model by Hopfield [Hopf82, Hopf84]. Hopfield and Tank [Hopf86] showed how this model could be applied to solve an optimization problem. Other important contributions have been made by Rumelhart and McClelland [Rume86], Sejnowski [Sejn86], Grossberg [Gros86], and Kohonen [Koho84a, Koho84b] .

Theoretically, analysis of neural network behavior is extremely difficult. One often uses experiments to validate the implementation of a new algorithm. Nevertheless, recent research has generated a better understanding of the behavior of a neural net. In particular, an improved understanding of the topological space around the stored memories [Hopf82, Hopf84, Hopf86, Rume86, Feld82] has lead to improved algorithms.

Another factor leading to the resurgence of interest in neural nets is the recent advances in VLSI technology. This new technology, permitting the implementation of a great number of processors on a single chip, has made the implementation of

neural nets in hardware possible [Hech86, Mead87, Psal85, Thak85].

One may consider a neural net as a learning and recall instrument. The particular problems and circumstances that arise from using a neural net in this manner are now examined. Specifically, the Hopfield model [Hopf82] is examined extensively, forming a basic structure for the study of these problems.

An associative memory blends information in many independent linked processors to form a single memory. The information is blended in either all or a portion of the memory. The retrieval process separates the desired information from the blended information.

To elaborate on the problem of blending information, consider mixing colors. If blue and yellow are blended, green is obtained. Most artists could look at green and determine the individual amounts of the primary colors, blue and yellow, that would combine to yield green. In other words, the information of the color make-up could be re-generated by some form of analysis. However, this blending may not always allow the information to be re-acquired. For example, if red and blue are mixed to create purple, it is impossible to re-generate the original colors. If the original colors are not known, for some reason, that information is lost. In that instance, enough information was cancelled or lost in the blending of those two colors as to prohibit the re-generation of the initial information.

This analogy may be extended to the problem of how much information is stored. If too much information is blended, it may not always be possible to retrieve the original information from the newly acquired amalgamation. If all the colors of the spectrum are blended together, white is obtained and all individual colors are lost.

Similar types of occurrence can be found during blending of information in an associative memory.

2.2. Neural Net Structure

Artificial neural nets are designed to form a resemblance to animal neural nets. Individual components are defined to resemble the corresponding components in the physiological model. The particular components of concern to neural nets are *nodes* and *synapses*. Additionally, they are combined in a manner that forms a structure called a *topology*, representing the structure and characteristics of the linkage. To understand how these form a neural net, the physiological level of neural nets is examined.

2.2.1. Neurophysiology and Neural Nets

A great deal of basic understanding about the structure of the neuron came about from the work of two men, Ramon y Cajal [Ramo28] and Golgi [Golg83]. These two man created the earliest definition of a network of connected neurons, which are the computational devices. The functioning of the neuron was not actually considered in their anatomical work.

Since that time, research has lead to the development of models that are similar to the neurophysiology of the brain. The biological model is represented as a densely interconnected, dynamic, cellular cluster [Hint81]. This cluster forms a mathematical model of the brain's functions. *Neurons* represent the individual processing nodes that are connected by synapses. The neurons exist in one of two stable states, either firing (active) or not firing (inactive). The *state of the system* is described by the state

of its neurons at any point in time. Neuron nets are similar to cellular automata and Ising spin glass models [Tana80].

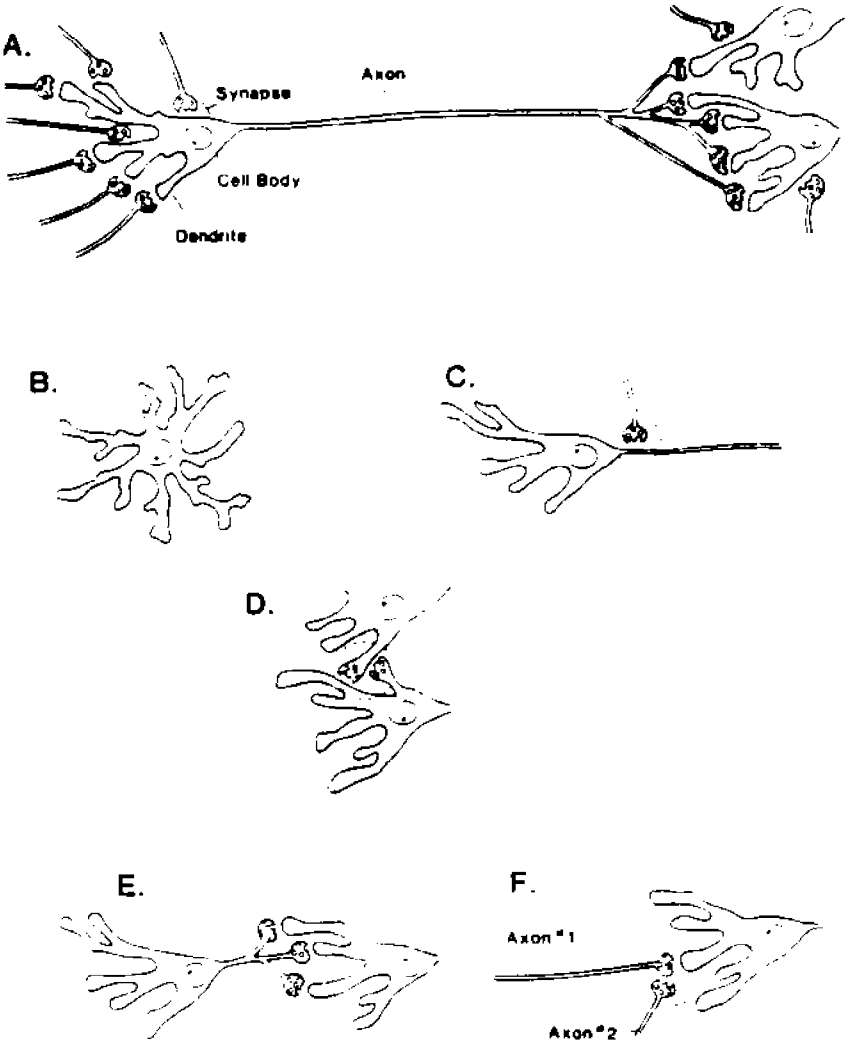


Figure 2.3

Possible Neuron Configurations [Asanuma & Crick 1986]

A neuron with no obvious axon. (Figure 2.3B) The type appears to process information but has no area to receive or transmit information. Axons that form synapses on other axons (Figure 2.3C) Dendrites that form synapses onto other dendrites (Figure 2.3D) Axons that do not produce a "spike", but instead form a graded potential (Figure 2.3E and 2.3F)

2.2.2. The Neuron

The standard neuron is a cell or node around which the neuron is based. The neuron has several dendrites that receive information from other neurons. Additionally, the neuron has a single axon that transmits information from the neuron by outputting a *spike* or *action potential* [Cric86] . The axon branches out to create synapses onto the dendrites or cell body of other neurons. Figure 2.3, from Asanuma and Crick [Cric86], illustrates a number of possible configurations using Figure 2.3A as the standard model of a neuron.

Actual neuron systems are much more complicated than the standard model. Crick and Asanuma [Cric86] discuss the known possible configurations that have been discovered over the years.

2.2.3. Nodes

In a neural net, the basic building block is the *neuron*, also called a *computational node*. The Hebbian rule is applied to a node by noting that each node has a set of inputs from the dendrites. The computational node is then defined as the sum of N weighted inputs passed through a thresholding device T, which normalizes the values depending on the the rules given (see Equation 2.1). The value of a particular node can then be defined by the equation:

$$x_i = \mathbf{f}\left(\sum_{i=1}^{N} t_i\, x_i - \mathbf{T}\right)$$

where t_i represents a set of weights from T and the x_i are previous values for the nodes. The function f represents a binary valued function whose range is either one or minus one, depending on the value of the domain. The system is pictured as shown in

Figure 2.4.

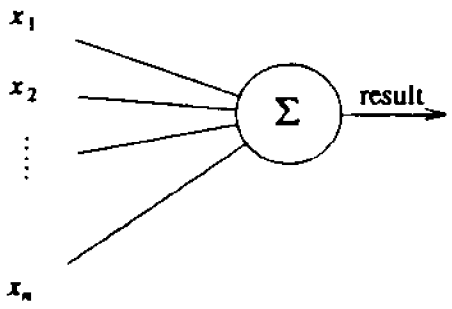


Figure 2.4

A Resultant from a Neuron

2.2.4. Synapses

Signals are transferred from the axons to the dendrites by elements known as *synapses*. The axon contains a synaptic ending that has the ability to release chemicals, called transmitters. A given neuron has only one type of transmitter that it releases through the synapse; however, a number of types of transmitters are known to exist. The transmitters go through the synapse to the membrane of the receiving cell, where the transmission of information is assisted by receptors.

Most of the synapses in the human brain are chemical and not electrical. The number of synapses can vary from a few hundred to tens of thousands per cell. One way to view synapses is by categorizing them into two types [Pete76], *excitatory* and *inhibitory*. Also, synapses can be characterized by the type of transmitter they use.

While this is beyond the scope of this work, it demonstrates that the synapses contain a great deal of the control in a system regarding what is transmitted, and how it is transmitted.

2.2.5. Topology

A *neural net topology* is created by linking a number of nodes together to form a topology. The system is then defined by that topology, *i.e.*, the particular characteristics of the nodes and the learning and retrieval rules that may be present. This work considers systems that are densely connected, that is, all nodes are connectioned to all other nodes. This is not a general requirement of neural nets, nevertheless, it will be used for this work.

2.2.6. Control Characteristics

In addition to the physical level in a neural net, certain other features exist that are critical to its definition. These features are discussed more fully in **Chapter 3** as the control level. The features represent decisions that are made outside of the basic questions regarding with nodes and synapses. The components of the decision level are the answers to questions such as: what type of input to allow in the system (continuous or discrete), whether or not the learning is controlled as it is learned (supervised or unsupervised), and how the timing rules for updating nodes are applied (synchronous or asynchronous).

2.2.6.1. Continuous versus Discrete Input

The difference between continuous and discrete input is that the former can

assume values on real numbers while the later can only assume integer values. This type of input could force a modification to some thresholding techniques, but it appears that the major difference is in the system-theoretical attitude towards neural nets. A neural net as a computational tool can be viewed as either analog or digital, depending on the direction taken. This work is based on a more digital approach.

2.2.6.2. Supervised versus Unsupervised Learning

There are two ways in which neural nets learn, supervised and unsupervised. In supervised learning, the input vector is fixed and a known output is desired. The input vector is entered into the system, acquiring a corresponding output vector. The output vector is then compared to the desired vector to obtain the amount of error in the system. The system may then enter into a training phase to reduce the amount of error in the system. This is accomplished by modifying the system by changing the synaptic weights in the system, and rerunning the system.

In unsupervised learning, the system is assumed to learn the information it is presented. The input vector is applied to the network and the system is "self organizing" so that a consistent output is produced whenever that input vector is presented.

It has been argued that supervised learning is not biologically plausible, because there is no teacher to improve the response and direct training. For this reason, a number of researchers have emphasized unsupervised learning.

2.2.6.3. Synchronous versus Asynchronous Updating

The particular components of the neural nets and their connections have been described, but one relevant aspect has not been discussed. This is the timing

procedure for the activation rule. When does a neuron update its value or informational content?

If, for all elements, the values are determined simultaneously, the system is said to have a *synchronous updating system*. This type of system can be viewed as a discrete difference approximation leading to a continuous, differential equation, whereby all units are updated continuously. The obvious point to this approach is that, in this system, any value that would change, changes immediately. Using this technique, any number of units might change at a given time, the exact number ranging from none to the number of neuron units in the system.

For other models, where units are updated individually, the system is said to have an *asynchronous updating system*. This definition is equivalent to the neurons being updated in a random fashion. This method of updating corresponds more closely to biological neurons, since their connection paths are likely to be of unequal lengths. One particular advantage of asynchronous updating is that the system always reaches a stable point, thus avoiding possible oscillations that can occur in a synchronous system.

Asynchronous timing allows only one neuron to be updated at a time. This allows only one vector component to change from time t to time $t+1$. Which individual component is updated is decided in the following manner:

The component is chosen from the set of n neurons, each neuron having an equal probability of $\frac{1}{n}$ of being chosen.

The neuron is chosen at random from the set of all possible neurons, without

regard to previous updates and any knowledge of the system. Notice that every neuron has the possibility of being updated. Choosing a neuron that does not change does not effect the system in any way and, in this manner, nothing happens until a neuron that can affect the system is chosen.

2.2.7. A Taxonomy of Neural Nets

Neural net may be categorized in different ways. Lippmann [Lipp87] presents a graph (Figure 2.5) illustrating a taxonomy of six systems.

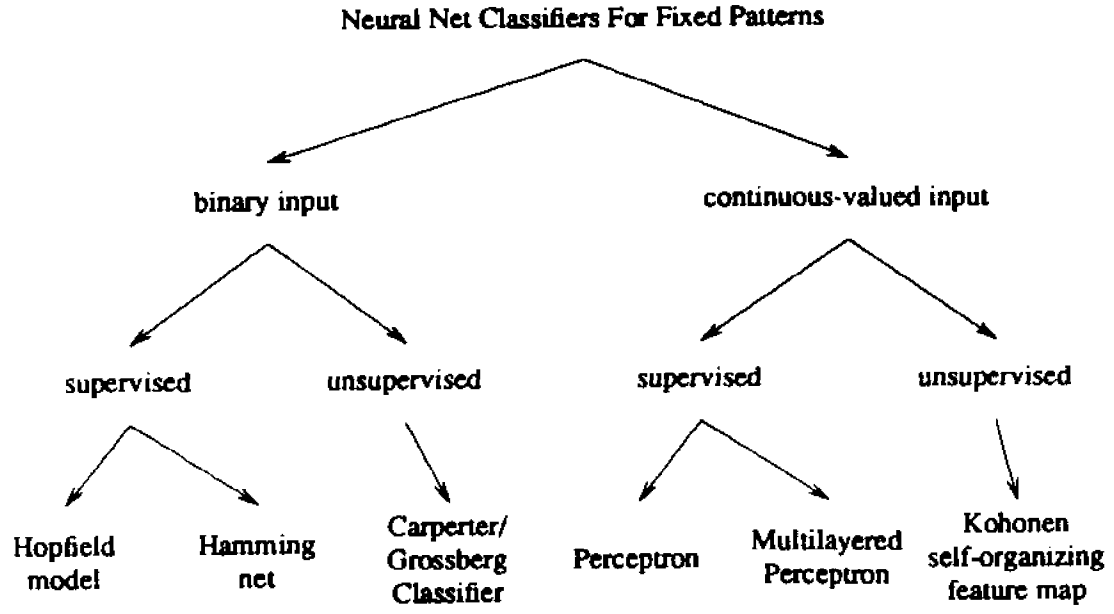


Figure 2.5

A Taxonomy for Neural Nets

This taxonomy is useful in some situations. in this particular taxonomy, the model presented here is considered to be an extension of the Hopfield line. Other methods of classification are based on updating rules, degree of connectivity among

neurons and other various properties.

2.3. The Hopfield Model

Consider the Hopfield model [Hopf82] as a content-addressable associative memory. Each neuron contains one bit of information, since it can only be in one of two states: *on* (+1) or *off* (-1). The *state* of each neuron is defined by the value that each neuron contains. The system is the set of neurons, and the state of the system is the *n*-tuple formed by the *n* neurons in the system.

The neurons are completely connected, that is, neuron *i* is connected by a synapse to neuron *j* for all *i* and *j*. The set of connections form a linear connection matrix T_{ij}, also known as the *transition matrix*. The weights of the matrix (t_{ij}) are symmetric ($t_{ij} = t_{ji}$).

One additional constraint on the system is that the weights of the diagonals in T_{ij} are all zeroes ($t_{ii} = 0$). This implies that a neuron is not connected with itself, and therefore does not affect, in any direct fashion, its own value. Whether or not the neurons are fired is determined using the threshold rule.

$$x_i = \operatorname{sgn}\left\{\sum_{i=1}^{n} T_{ij}x_j\right\} = \begin{cases} +1, & \text{if } \sum T_{ij}x_j \geq 0 \\ -1, & \text{if } \sum T_{ij}x_j < 0 \end{cases} \tag{2.1}$$

The value of the neurons are computed using the threshold decision rule given in Equation 2.1 for a system, where $p = (x_1, x_2, ..., x_n)$ contains the values at each time period. Individual neurons sum the values of all other connected neurons in the system. If the summation equals or exceeds a predetermined threshold, zero in this example, the neuron assumes the value +1. If the summation is less than the

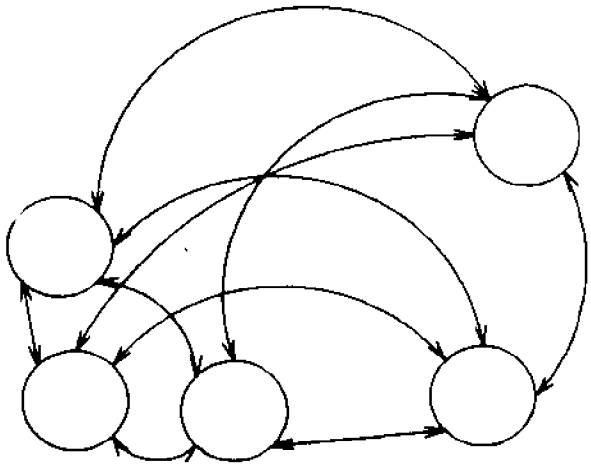


Figure 2.6

A Neural System

prescribed threshold, the neuron assumes a value of -1.

A neural net can be considered as a circuit, the form of the standard circuit is illustrated in Figure 2.7. Notice that although each neuron (x_i) could connect with almost every other neuron, denoted by •, in this model the connection with itself is open, denoted by ⊕. This eliminates feedback to a neuron from itself, as per the definition of this model. Also note, that the system sums all the inputs and thresholds these values, thus determining the sign as either positive or negative.

One important concept for consideration is the timing that the system uses to update each neuron. Assume that a particular neuron, through the summation of the other neurons is to change its value. If other neurons should change at the same time, then the questions might arise whether the summation should be recalculated to consider each change. This is due to the fact that every neuron is dependent on every

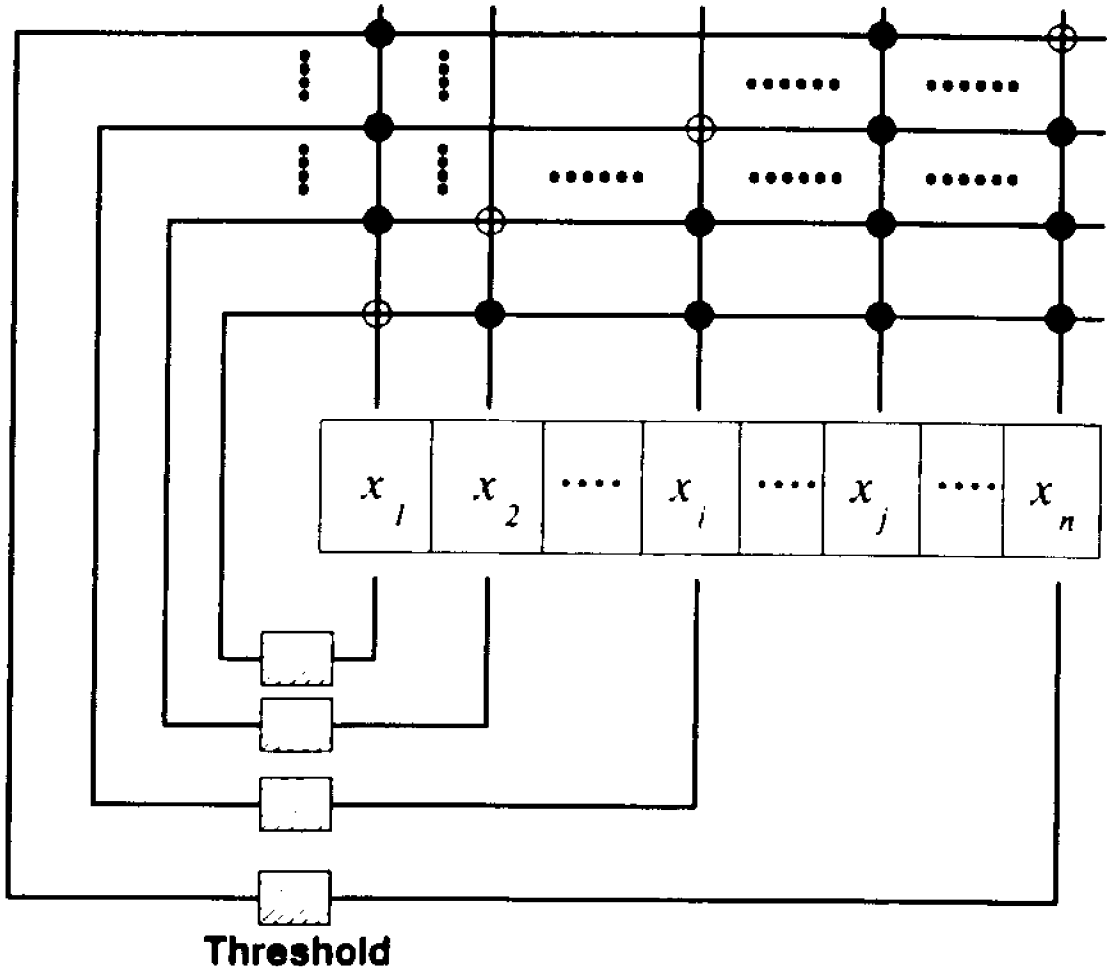


Figure 2.7

A Circuit Diagram of a Neural Net

other neuron. Therefore, the timing used to update the system could quite possibly effect the direction a particular decision takes. This problem is resolved by assuming that the system is synchronized, as to allow only one neuron update per time period.

If information is stored in the memory, the information becomes an *attractor*. The attractor forms a region about it , called an *attraction basin* in which the stable point, the attractor, acts as a magnet to probes in the space formed by the memory. This attraction depends on the closeness of a probe to the information and the strength

of the other attractors.

The memory allows *probes* into the memory that are non-intrusive to the memory. A probe enters the space of the associative memory and is brought to the attractors without changing the space in any manner. In this way, the system effectively shifts the position of the probe to a point closer to the position of an attractor, until eventually, in the asynchronous case, the probe reaches a stable point.

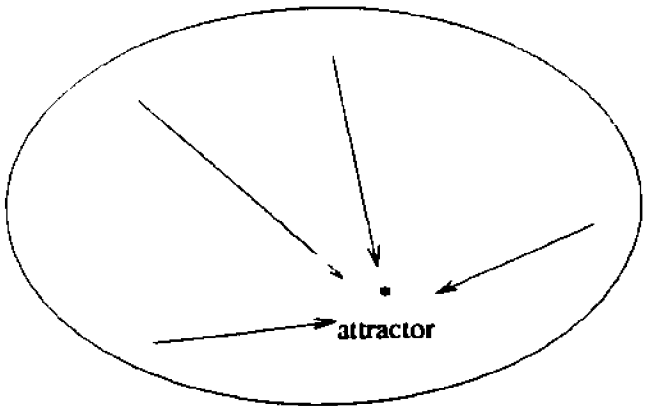


Figure 2.8

An Attractor in an Attraction Basin

In the Hopfield model, learned information from an external source is referred to as *learned memories*. To call all learned memories *stable points* is a misnomer, as other stable points exist in addition to the learned memories, and, as we will see, the learned memories might not even represent stable points. This differentiates between memories that are purposefully stored and those that are developed as characteristics of the system.

We define the learned memory $x_{(k)} = (x_1, x_2, \ldots, x_n)$ to be the k^{th}-memory of *n*-dimensional column vectors. Each vector, as described in Equation 2.1, consists solely

of -1's and +1's. Each memory vector is used to form an n x n matrix using the outer product rule:

$$T^{(k)} = x_{(k)}(x_{(k)})^t - I_n \tag{2.2}$$

In this equation I_n is an n x n identity matrix. Using this technique, the matrix stores the outer product of the vectors with every $x_i\ x_i$ element equal to zero.

The Hopfield transition matrix is then defined as the sum of these outer product matrices:

$$T = \sum_{i=1}^{m} T^{(k)} \tag{2.4}$$

A very important aspect of the Hopfield Model is that once T has been calculated, the individual memories, as well as any additional information regarding them, is lost.

The retrieval process is accomplished using the T matrix. A probe (p) is presented with each position either a +1 or a -1. The probe is then multiplied by T, giving a new probe p′:

$$p\,T = p' \tag{2.5}$$

The vector p′ is then thresholded, using equation 3.2, giving a new position vector. This binary vector is then compared with the original vector p for differences in the values of individual positions. The set of positions that are different is referred to as the *affective set.*

A position is chosen at random from the affective set, updated, and the process is then repeated. This continues until the affective set is null. A null affective set implies that a stable point has been reached.

2.3.1. An Example

The problem is clearly illustrated through an example system that uses the Hopfield Model. This example is referenced throughout the remainder of this work.

There are n=5 neurons and three learned memories:

$$x_{(1)} = (+1\ +1\ +1\ +1\ +1\ )$$
$$x_{(2)} = (-1\ -1\ +1\ +1\ -1\ )$$
$$x_{(3)} = (+1\ -1\ -1\ -1\ -1\ )$$

which yields the following individual transition matrices:

$$T^1 = \begin{bmatrix} 0 & +1 & +1 & +1 & +1 \\ +1 & 0 & +1 & +1 & +1 \\ +1 & +1 & 0 & +1 & +1 \\ +1 & +1 & +1 & 0 & +1 \\ +1 & +1 & +1 & +1 & 0 \end{bmatrix}$$

$$T^2 = \begin{bmatrix} 0 & +1 & -1 & -1 & +1 \\ +1 & 0 & -1 & -1 & +1 \\ -1 & -1 & 0 & +1 & -1 \\ -1 & -1 & +1 & 0 & -1 \\ +1 & +1 & -1 & -1 & 0 \end{bmatrix}$$

$$T^3 = \begin{bmatrix} 0 & -1 & -1 & -1 & -1 \\ -1 & 0 & +1 & +1 & +1 \\ -1 & +1 & 0 & +1 & +1 \\ -1 & +1 & +1 & 0 & +1 \\ -1 & +1 & +1 & +1 & 0 \end{bmatrix}$$

Note that, the indeividual transition matrices are created using the outer product rule for the three individual learned memories. The diagonal elements are then set to zero to represent the lack of feedback from the neuron to itself.

The individual matrices are summed resulting in the transition matrix:

$$T = \begin{bmatrix} 0 & +1 & -1 & -1 & +1 \\ +1 & 0 & +1 & +1 & +3 \\ -1 & +1 & 0 & +3 & +1 \\ -1 & +1 & +3 & 0 & +1 \\ +1 & +3 & +1 & +1 & 0 \end{bmatrix}$$

To test the resulting system, the system is probed with vectors to see the effect on the system from different queries. Let us start by probing the system with a learned memory:

$$x_{(3)} = p = (+1 \; -1 \; -1 \; -1 \; -1)$$

By multiplying this by the transition matrix **T**, p′ is obtained:

$$p' = (+3 \; -2 \; -6 \; -6 \; -4)$$

The values are then thresholded to obtain:

$$p' = (+1 \; -1 \; -1 \; -1 \; -1)$$

Which is the original information vector. This is due to the fact that $x_{(3)}$ has become both an attractor and a stable point for the system. When the probe entered, it simply stayed at that position, thus both defining and exemplifying a stable point. In fact, the goal of neural net processing is to develop a system that, when a memory is learned, it becomes a stable point and attracts probes that are near it.

The system is then probed with another vector that is not a learned memory:

$$p = (+1 \; +1 \; -1 \; +1 \; +1)$$

By multiplying the probe by the transition matrix, we obtain:

$$p' = (0 \; +6 \; +4 \; -2 \; +4)$$

Which will threshold to:

$$p' = (+1 \; +1 \; +1 \; -1 \; +1)$$

The elements of p are then compared to p′, obtaining an affective set containing the values 3 and 4. This is because the values in positions 3 and 4 in p′ are different from those positions in p. The choice of x_3 yields p = (+1 +1 −1 +1 +1), which is a learned memory. The choice of x_4 yields p = (+1 +1 −1 +1 +1), which is not a learned memory; however, this point turns out to be a stable point. That is, the probe (p), expected to stabilize at $x_{(2)}$, in fact stabilized at some other point. If the system considers Hamming distances [Hamm82], then the information vector closest is $x_{(2)}$ and, therefore, one might want the probe to end up there. An observer might be surprised by the resulting vector in the second choice, but does that mean it is an incorrect result? Could it actually mean that the system has achieved a different yet still correct stable point? It is unclear where this other stable point comes from.

In the example, the system had a choice of updating two neurons. The choice of which neuron to update first determines whether or not a correct stable point is reached. One choice appears to be better than the other, as it proves to lead to a learned memory. Assuming an incorrect result is obtained, one may ask several questions. If the system is heading in the wrong direction, could we have changed the selection of elements in the process of developing a path through the system in order to achieve a correct result? If a particular element in the system is chosen to be updated, how is this choice made?

2.3.2. Problems with a Learned Memory

Questioning whether or not a particular result is wrong is a fundamental concept that must be addressed before proceeding to other questions. Unfortunately, whether

something is right or wrong is often an arbitrary decision that can be discussed and argued from a variety of angles.

It is possible to visualize systems that depend on a probabilistic response to certain events. An example might be when an electron is shared by the nuclei of atoms. In this example, it is necessary that the electron revolves around one of the nuclei for part of the time and around the other nuclei the rest of the time. The exact location of the electron cannot be determined at a given point in time. We can only speak of it in terms of its probabilities of location.

While we recognize that this is a valid and useful condition, our attitude differs slightly. The approach that we will take to neural nets is that if a probe is put into the system, it should preferably end up, or stabilize, at the nearest information vector. That is, we will consider the nearest information vector to be the best solution to the probe query. However, if the probe stabilizes at any point other than a learned memory, unless otherwise stated, it is assumed to be a wrong answer. We are primarily interested in obtaining the probability of a correct solution.

One may also wish to study what paths lead to the correct result. The best path is the path that optimizes over the probability of obtaining a correct answer; however, in most cases, this is not easily determined.

There are situations that arise with the Hopfield model that insure that errors will be introduced into the system. They are situations in which the information introduces contradictions into the system that lead to incorrect answers to queries.

2.3.3. Complements

In the Hopfield model, if a memory is learned, its complement is generally also learned as a stable point in the system. This is due, in part, to the symmetric transition matrix that is used in the model. The reason that all complements are not stable points in the model is that the model is not quite symmetric. In Equation 2.1, the thresholding is done at 0, but the system sums the neurons and, if they are 0, then the system becomes a +1. If, at 0, the system chose to remain at the value it was prior to updating, then the system would be stable and every learned memory and spurious state would have a complement in the system. However, in most cases, the complement to a learned memory is a stable point in the system and, furthermore, most spurious states will also have a complement as a learned memory.

To demonstrate this particular situation, let us consider the following situation:

$$x_{(1)} = (+1 +1 -1 -1 +1 -1)$$

Gives a transition matrix of:

$$T = \begin{bmatrix} 0 & 1 & -1 & -1 & 1 & -1 \\ 1 & 0 & -1 & -1 & 1 & -1 \\ -1 & -1 & 0 & 1 & -1 & 1 \\ -1 & -1 & 1 & 0 & -1 & 1 \\ 1 & 1 & -1 & -1 & 0 & -1 \\ -1 & -1 & 1 & 1 & -1 & 0 \end{bmatrix}$$

If we test our learned vectors, we obtain:

$$x_{(1)} \times T = (0\ 4\ -4\ -4\ 4\ -4) = (+1 +1 -1 -1 +1 -1) = x_{(1)}$$

However, a further check of every probe in the system gives one additional stable point for the system.

$$\overline{x_{(1)}} \times T = (0 \; -4 \; 4 \; 4 \; -4 \; 4) = (-1 \; -1 \; +1 \; +1 \; -1 \; +1) = \overline{x_{(1)}}$$

We see from this example that the complement of our fundamental matrix is also a stable point in the system. This is as expected in a symmetric system. Although this is expected and accepted in this system, it can be troublesome if the complement is undesired. Additionally, if the complement cannot be recognized as a complement, the result may be disastrous. However, in some instances, a complement might be quite desirable. For instance, a system that simply desires to choose between the learned memory and its complement would benefit from the storage of complements.

2.3.4. Information Too Close Together

As a system learns information, it may happen that this information is too close together to be distinguished. We see this situation in humans occasionally, when people meet twins. Although the twins are not exactly identical, in many instances there are not enough distinguishing features for people to delineate between them. The same type of situation can occur with the Hopfield model. Consider the situation where information is placed into the system with a Hamming distance of one separating them. We will see that, in this case, the information is too close together. Let:

$$x_{(1)} = (+1 \; +1 \; -1 \; -1 \; +1) \text{ and } x_{(2)} = (+1 \; -1 \; -1 \; -1 \; +1)$$

Notice that the Hamming distance for these two vectors is one. We obtain the transition matrix of:

$$T = \begin{bmatrix} 0 & -0 & -2 & -2 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -2 & 0 & 2 & -2 \\ -2 & -2 & 2 & 0 & -2 \\ 2 & 2 & -2 & 0 & 0 \end{bmatrix}$$

Testing our two learned memories we obtain:

$$x_{(1)} \times T = (6\ 6\ -2\ -4\ 6) = (+1\ +1\ -1\ -1\ +1) = x_{(1)}$$

and:

$$x_{(2)} \times T = (6\ 6\ -2\ -4\ 6) = (+1\ +1\ -1\ -1\ +1) = x_{(1)}$$

So $x_{(1)}$ has been learned and is a stable point in the system; however, $x_{(2)}$ was learned, yet, it is *not* a stable point in the system. The information of $x_{(2)}$ is lost to the system.

In fact, the only two stable points in the entire system at this point are $x_{(1)}$, as we have seen, and $x = (-1\ +1\ +1\ +1\ -1)$, which is the complement of $x_{(2)}$. We would have expected to obtain both the information vectors and their complements. In this instance, the problem is that the values are too close together. If the Hopfield model has points of information that are too close together, the system loses the ability to distinguish among the different vectors.

2.3.5. Information Too Far apart

Problems also occur when information is too far apart in a Hopfield model. This type of situation can compared to information that is almost conflicting in its nature. A person might become confused if given directions that are almost completely at odds with one another. The analogy does not hold for information that is completely different, as this situation reinforces the stability of points that are completely opposite. Consider a learned memory in a six neuron system that is a Hamming distance of five from the other piece of information.

If we allow the system to learn both vectors, let us examine the consequences:

$$x_{(1)} = (+\ +\ -\ -\ +\ -) \text{ and } x_{(2)} = (+\ -\ +\ +\ -\ +)$$

Gives a transition matrix of:

$$T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & -2 & 2 & -2 \\ 0 & -2 & 0 & 2 & -2 & 2 \\ 0 & -2 & 2 & 0 & -2 & 2 \\ 0 & 2 & -2 & -2 & 0 & -2 \\ 0 & -2 & 2 & 2 & -2 & 0 \end{bmatrix}$$

If we test the learned vectors we obtain:

$$x_{(1)} \times T = (0\ 8\ -8\ -8\ 8\ -8) = (+1\ +1\ -1\ -1\ +1\ -1) = x_{(1)}$$
$$x_{(2)} \times T = (0\ -8\ 8\ 8\ -8\ 8) = (+1\ -1\ +1\ +1\ -1\ +1) = x_{(2)}$$

The result, at first, might appear quite good, as the two points learned are indeed the stable points. Further investigation reveals that these are the only stable points in the system. As was stated in section 2.3.3, all learned memories have their complement as a stable point in the system. Since these two memories are the only stable points, their complements are not stable in the system. At first, this may not appear to be a major problem; however, it indicates that something unexpected is happening in the system. When the user expects a situation to occur and it does not, the outcome is as bad as something happening that is not expected.

2.3.6. Spurious States

The most troublesome problem with stable points is the formation of *spurious states*. Spurious states are elements that are not either learned memories or complements of learned memories. These elements are created by the system through the interactions of the system during the learning of learned memories.

Spurious states are essentially a result of linear combinations of learned memories. Since the Hopfield model is not symmetric, not all spurious states can be shown to be linear combinations of learned memories [Kant87, Amit87] . In addition, complements may be viewed as special spurious memories; however, in this dissertation, we make a distinction between them. Nevertheless, complements of spurious states are also considered spurious.

Because most spurious states are created by a linear combination of learned memories, the capacity of the associative memory is reduced by a certain degree. Clearly, an increase in the number of learned memories increases the probability of non-orthogonal elements. This generates a larger set of possible combinations.

Consider the following example:

$$x_{(1)} = (+1\ +1\ -1\ -1\ +1\ -1) \text{ and } x_{(2)} = (-1\ -1\ -1\ -1\ +1\ -1)$$

Gives a transition matrix of:

$$T = \begin{bmatrix} 0 & 2 & 0 & 0 & 2 & -1 \\ 2 & 0 & 0 & 0 & 2 & -2 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 & -2 \\ -2 & -2 & 0 & 0 & -2 & 0 \end{bmatrix}$$

If we test every possible probe in the system, we obtain as stable points:

$$x_{(1)} \times T = x_{(1)} \text{ and } x_{(2)} \times T = x_{(2)}$$

Additionally, we find that the complements of learned memories are:

$$\overline{x_{(1)}} \times T = \overline{x_{(1)}} \text{ and } \overline{x_{(2)}} \times T = \overline{x_{(2)}}$$

If we continue by learning another vector:

$$x_{(3)} = (+1 +1 +1 -1 +1 +1)$$

We obtain the transition matrix:

$$T = \begin{bmatrix} 0 & 3 & 1 & -1 & 3 & -1 \\ 3 & 0 & 1 & -1 & 3 & -1 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ -1 & -1 & 1 & 0 & -1 & -1 \\ 3 & 3 & 1 & -1 & 0 & -1 \\ -1 & -1 & 1 & -1 & -1 & 0 \end{bmatrix}$$

We then find that the only stable points in the entire system are a spurious state and its complement, which is also a spurious state. They are:

$$x_{(spur)} = (-1 -1 -1 +1 -1 +1) \text{ and } \overline{x_{(spur)}} = (+1 +1 +1 -1 +1 -1)$$

The particular concern regarding spurious states is that they add a dimension of uncertainty to any stable point encountered by the system. If there are no spurious states, once the system has stabilized it would either be at a learned memory or the complement of the learned memory. The problem becomes less critical under those circumstances. With the introduction of spurious states, it is now unclear whether we can accept any answer with certainty.

2.3.7. Stable Point Not Close to Probe

One reason for the popularity of the Hopfield model is that it can always be shown to converge. Consider the logical implication of a system that always finds a stable solution for every probe.

The implied rule is that if a probe is entered into the system, it should stabilize at a point in the system. The possibility exists that the system does not have the necessary information to answer a particular probe. In the Hopfield model, the system will

still provide a solution for the probe. This may or may not be a desirable feature for the model.

A system that has a large amount of information regarding a particular area, for example, electrical engineering, would not be very good for probes regarding literature; however, the Hopfield model will still respond with an answer.

What appears to be a good quality of the system, always finding a stable point, may or may not be the desired quality when examined as a simple retrieval problem. Consider an eight neuron system with only the following learned memory:

$$x_{(1)} = (+1 \; +1 \; +1 \; -1 \; +1 \; +1 \; -1 \; -1)$$

We obtain a transition matrix of:

$$T = \begin{bmatrix} 0 & 1 & 1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 0 & 1 & -1 & 1 & 1 & -1 & -1 \\ 1 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ -1 & -1 & -1 & 0 & -1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 0 & 1 & -1 & -1 \\ 1 & 1 & 1 & -1 & 1 & 0 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & 0 & 1 \\ -1 & -1 & -1 & 1 & -1 & -1 & 1 & 0 \end{bmatrix}$$

For the probe:

$$p = (-1 \; +1 \; -1 \; +1 \; +1 \; +1 \; -1 \; +1)$$

The system will, with the correct updates, yield $x_{(1)}$ as a solution. The solution has four out of eight positions that are different from the probe, for a 50% difference. This might represent a significantly different concept or even another area of thought, depending upon the meaning of each bit. The problem is that, even though a probe is chosen that is a Hamming distance of four away from the learned value, the probe *must* stabilize at a stable point that is in the system. This is due to the fact that the

probe should always stabilize within the system. This could be considered a logical flaw that is inherent to any system that is forced to stabilize at some point.

2.3.8. Analysis Techniques

For purposes of analysis, we make the following observations. From a vector of length **m** choosing an element at random means that the probability of choosing a particular element is:

$$P(x_i) = \frac{1}{m} \text{ where } m \leq n \text{ and } 1 \leq i \leq n$$

This means that each of the **m** elements has an equal probability of being selected as the element to change.

Since we consider a choice of a particular element as a choice of a path in the neural net, we can then define the probability of a path:

$$P(\text{path}_j) = \prod P(x_i) \text{ for i an element of the path}$$

The probability associated with obtaining a stable point **x** will be found as:

$$P(\mathbf{x}) = \sum_j P(\text{path}_j) \text{ for j an element of the set of paths leading } \mathbf{x}.$$

That is, the probability of a point (x) starting at probe (p) is simply the sum of the probabilities of all the paths that lead to the point.

2.3.9. Convergence of the Hopfield Model

Hopfield [Hopf82, McEl87] shows that for any probe into a system with a symmetric connection matrix **T**, a stable point is reached in the asynchronous model. Hopfield shows this by defining the function E:

$$E = -\sum_{i}\sum_{j} T_{ij}\, x_i\, x_j \tag{2.5}$$

E is a non-increasing function for any change of value in the variable x_j.

Assume ς reprersentes a particular element of probe i is to be updated, then:

$$x_\varsigma = \mathrm{sgn}\sum_{k} T_\varsigma\, k\, x_k$$

Since these represent correlation changes in the associative memory elements, then:

$$\Delta C = C' - C = \sum_{j=1}^{n} T_\varsigma\, j\, (\Delta x_\varsigma)\, x_j + \sum_{i=1}^{n} T_{i\varsigma} x_i (\Delta x_\varsigma) + T_{\varsigma\varsigma}(\Delta x_\varsigma)^2 \tag{2.6}$$

with $\Delta x_\varsigma = x'_\varsigma - x_\varsigma$.

From equation 2.4 and the symmetry of the transition matrix T we note that:

$$\Delta C \geq 2 * \Delta x_\varsigma \left(\sum_{j} T_{\varsigma j}\, x_j \right) \tag{2.7}$$

Furthermore, note that with nonnegative diagonal elements, three possible situations exist:

i. $x'_\varsigma = x_\varsigma$, whereas there is nothing to prove since the system is stable.

ii. $x'_\varsigma > x_\varsigma$ implies $x'_\varsigma = -1$ and $x_\varsigma = +1$, then equation 2.5 yields $E \geq 0$ since $\Delta C \geq 2^* -2\, E > 0$

iii. $x'_\varsigma < x_\varsigma$ implies $x'_\varsigma = +1$ and $x_\varsigma = -1$, then applying equation 2.5 yields $E \geq 0$ and $\Delta C \geq 2^* 2\, ^* E \geq 0$.

Hopfield's proof shows that a nondecreasing function must eventually end up at a stable point. It does not guarantee that the point will be the closest stable point, or a learned memory.

Chapter Three

Neural Networks with Asynchronous Control

When we have understood, we hear in retrospect.

Marcel Proust (1871-1922)

3.1. Introduction

We now introduce a model that improves the convergence behavior of the Hopfield network. The model introduces an asynchronous controller to assist in the updating of particular neurons. The controller also permits the implementation of a number of algorithms that improve the overall performance of the model. The controller is first examined at a conceptual level, then defined at the hardware level.

To understand the controller as a tool, consider the analogy between the Hopfield model and certain aspects of the brain. We note that certain aspects of searching memory that humans utilize are not allowed in the Hopfield model. First, during the recall of information that is partly known, the known aspects are not allowed to change; or are changed very reluctantly. The Hopfield model, having no control over the direction of how an answer is obtained, allows portions of the original information to change and vary in the search for a solution. Second, when we do obtain a solution, we check to see if it is a plausible solution. Thus, the negative of a photograph would be eliminated. The Hopfield model is not equipped to reject any solution. Third, many people use techniques to help them remember information. They add dates or images to the information that improve the remembrance of an idea. In the Hopfield model, adding information could be implemented; however, once again, the added

information can easily change in the retrieval process and would therefore not provide any significant assistance to the process. Indeed, merely tagging a date to information might be counterproductive if the correct tag were not used.

The Hopfield model needs specific improvements to assist retrieval in some instances. Features that might improve the system are an ability to:

fix on particular parts of information,

backtrack away from answers that are wrong, and

improve learning with appended information.

The addition of these capabilities is desirable if the system is to maintain its physical structure. A controller should not effect the basic design of the physical system.

3.2. Physical versus Control Level

Separating the physical and control levels of the neural net is necessary for accomplishing improved performance. This unbundling separates the neural net system into two levels: a physical (neuron/synapse) level and the informational (control) level. It is demonstrated that the increased flexibility of the model with a controller allows the system to use information previously unavailable to the decision process. This will allows the use of additional external techniques to complement the learning and retrieval processes in neural nets.

Critical problems to this approach are the definition of model and the standardizing and implementing the control structure for the net. The control structure must be sufficiently flexible as to utilize additional information, yet specific enough to drive the neural net system.

Neural nets are described by the neurons, their individual threshold for firing and the synapses that connect this system of neurons. These are the physical level components of a neural net, linked together and often thought of as a complete neural net. In addition, this definition usually includes the number of neurons in the neural net. However, there are a number of additional aspects of a neural net system that must be considered. These aspects are fundamentally different from the physical characteristics of a neural net.

It is clear that humans perform a considerable amount of information preprocessing before storing this information. In other words, a great deal of information is disregarded or rearranged before the actual processing begins. When a neural system has information presented to it for learning, how the system reacts to the information is not only a physical reaction of the system, but is also due to an implied intervention by its control structure. For example, a system might choose to ignore information. On the other hand, a system may accept parts of information, or add tags to the incoming information to facilitate storage. It may be necessary to increase the size (number of neurons) of a neural network to add this tag information.

How the system reacts to requests for stored information is important. For instance, the update method can be utilized to determine if the information is available. The type of update system, synchronous or asynchronous, may determine whether a system stabilizes to an answer. When a search has been concluded, the success of this search must be determined. Even in an apparently successful retrieval, the possibility of error remains. These problems must be dealt with at the control level, not at a physical level.

To reiterate, we see that it is reasonable to view a neural net at various levels, to be denoted as the physical level and the decision or control level. The physical is characterized by the types of neurons, their thresholds and the method for reaching the threshold, the possibility of blocking the firing, and the type and number of links between the neurons. The control level is characterized by the reaction to values of the neurons, the actions taken when information is presented to be learned, and the actions to a probe.

At the control level, the actions are subject to rules or decisions based on the conditions present. These effect the flow of information onto, within, or out of the neural net. The decisions are referred to as the control process, which is discussed extensively in the work. Figure 3.1 illustrates the separation of a neural net into the two parts, a physical core and an outer control layer.

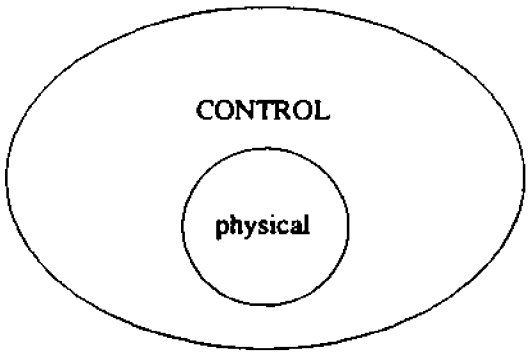


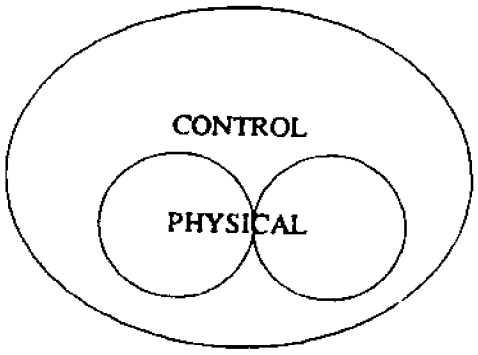
Figure 3.1

Separation of the Control and the Physical

3.3. Merging of Systems

Traditional sequential algorithms can assist in decision processes that govern the decision structure at the control level. There is a great deal of information that can be categorized into either types or a group structure, which can improve convergence techniques. To date, no method has been developed to accomplish these in a Hopfield neural net system. An example of this is information clustering, which is utilized in the areas of information retrieval and pattern recognition [Tou74] . No methodology currently exists to integrate this into a Hopfield neural net environment. Other models do accomplish clustering [Gros86, Lipp87] ; however, these are not the same basic model used in this dissertation. The neural net has no way to *know* that the information is grouped and, therefore, this information is not used. Additionally, similar information can lead to an increased probability of spurious states. Also, the number of learned memories is not known by the neural net. While, in fact, the system may be degraded by a abundance of stored information, the system has no way to realize this, and the system has no way to change its actions even if it did realize this fact.

Linking neural nets to other neural nets has also received some attention. Rumelhart and McClelland [Rume86] have listed a number of types of neural net connections. These connections, however, are neuron to neuron type links. The unbundling of the control structure allows the linking to be accomplished through the control segment and, therefore, is different in nature from the previous types of connections. In fact, the neural nets might be sufficiently different in their nature such that a connection at the neuron/synapse level would have little or no meaning.



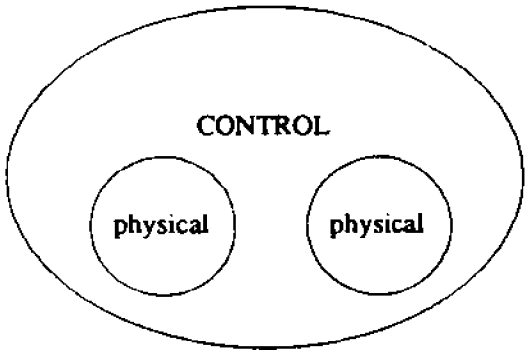


Figure 3.2a Figure 3.2b

Tightly and Loosely Coupled System

Figure 3.2a illustrates systems that are linked at the physical level, whereas Figure 3.2b pictures systems that are linked only through a control structure. A neural net linked at the physical level is referred to as a *tightly coupled* system, while a system linked only through a controller is referred to as a *loosely coupled* system.

As pointed out in Rumelhart and McClelland [Rume86] , tightly coupled systems are no more powerful than a single system. It is shown that a loosely coupled system can achieve computational powers that are not available with a single neural net.

Consider the advantages of a loosely connected system, as compared to a tightly connected system. Perhaps the most important advantage is that, in all cases, the method and direction of obtaining a solution in the neural net is not linked entirely to the physical type of the neural net. This frees the decision process to include information from other sources that improve the performance of the system. In other words, the physical system is separated from the control level decision process, and therefore

the decisions can be made from information derived from independent sources.

The separation allows the addition of new components that specialize in different aspects of the decision making process. The resulting specialized processes do not impinge on the existing physical system.

Another point to note is that, since the controller allows the unbundling of decisions from the system, the basic properties are significantly more understandable. This allows optimizing techniques to be studied and quantified individually, concentrating solely on the cost and complexity of each. In addition, the unbundling allows the improvements to be separated for possible distribution to other processors.

One may now use different types of neural nets in conjunction with one another. This allows the interaction of neural nets, each of which is optimal for processing a certain feature of the overall problem.

3.4. A Biological Interpretation

Synapses as controllers have been discussed in **Chapter 2**. It appears that, in neurological systems, the actual control of where and how neurons are connected is accomplished through the synapses. As has been pointed out, the actual manner in which this is accomplished is not yet known to researchers; however, the fact that some of the system control processes reside in the synapses is known.

3.5. Strong Attractors

What is the specific gain by the use of a controller that is not obtained by merging two neural nets? The current model of the neural net excludes one particular,

arguably necessary, requirement that neuron values can be clamped to constants. Solutions are then forced or strongly attracted to a particular value. This is not determined by the topology of the neural net alone.

Definition:

A stable point that attracts a probe by forces other than those of the physical neural net system is referred to as a **strong attractor**.

If the values of certain neurons are known, why should they not be selected to change as the system updates? Since a neuron value reflects the cumulative effect of the correlations with all other neurons, one would expect that, if the probe is not *too far* from the learned state, the effect of those neurons which are in error will be overwhelmed by the much larger set of neurons that are correct. By allowing a known, correct value of a neuron to change, one would thus let the probe move away from the stable point in the attraction basin, thus increasing the probability of obtaining a wrong response.

Such clamping of neuron values has not been considered earlier by other researchers. For example, in [McEl87] , all neurons are allowed to change, resulting in performance that can be unsatisfactory.

3.6. A Hopfield Model with a Controller

The model presented is based on the Hopfield model [Hopf82], which has been described in **Chapter 2**. The Hopfield model is an asynchronous model with binary inputs and Hebbian learning, it may be viewed as a supervised system. The new model maintains the characteristics of a binary input and supervised learning. The

system changes the asynchronous updating to a system that simulates asynchronous updating. The method for simulating the updating will be explained shortly. The memory will have only one neuron updated at a time. This, by itself, does not fit the definition of an asynchronous model, yet it reflects, in part, the characteristics of an asynchronous model in that only one neuron is updated on every pass.

Since the principal difference resides in the updating procedure, a brief discussion is in order. The asynchronous Hopfield model requires a random selection from all the neurons with equal probabilities to update the memory. This means that any neuron could be used to update the system. This is determined by comparing Equation 2.1 at pass k with Equation 2.1 at pass $k+1$. The method presented allows the selection to be chosen from a set of neurons determined by the controller. This set represents a subset of the original set of neurons; however, certain decisions may impose constraints on the choices.

3.6.1. The Affective Set

It was mentioned in section 2.3 that when testing the system for a learned memory in the synchronous model, the system obtains a set of neurons that will change at time $t+1$ on each pass. By definition, in the Hopfield model, the only possible neurons that could change the system would be one of the same neurons that are obtained in the synchronous model [McEl87] . The set of neurons that would not change in the synchronous model cannot change the system in the asynchronous model either. Since the set whose neurons can change the system is the same, we will refer to this set as the *affective set* (AS) for the asynchronous model. The set (AS) is

used when examining the possible changes of the neuron values in the synchronous model.

In the model proposed, the probability of choosing a particular neuron is decided not by randomness in the updating process, as in the Hopfield model, but is chosen from *AS* by the controller, which imposes a choice on the system. This is significant not only as a concept, but also in terms of increasing the probability of obtaining a proper solution.

The model is an augmentation of the Hopfield model, and includes the Hopfield model as a subset. If it is assumed that the probabilities of all the elements of *AS* are equal, the system will obtain the same solution patterns as in the Hopfield model. Additionally, the concept of an affective set is just as applicable for the Hopfield model as the new model. As we have pointed out, the elements of *AS* are the same for the Hopfield model and the proposed model.

3.6.2. Probabilities of the Affective Set

Assume that there exist known probabilities as to whether an element in *AS* should be changed to another value in the search of a correct answer. Were this possible, the affective set would have assigned to each element some probability of each neuron being a particular value. Each neuron associated with the updating of a probe vector would have a probability associated with it. The associated probabilities will be called the *affective probability* for each neuron. The affective probability is the probability given for updating the neuron when each neuron is considered prior to either a pass with a probe vector or multiplication with a transition matrix. The neu-

rons that have no affect on the system are also given probabilities as to their eventual values. These probabilities reflect the probability of the neuron value being a predetermined value, i.e., probability of $(x_1 = 1) = 0.5$. The probabilities of the affective set may in fact be zero.

The affective probabilities are put in vector form, in what is known as a *positional probability vector*, represented by Ω. From this the following representation is obtained.

$$\Omega = (\omega_1, \omega_2, \cdots, \omega_n)$$

Where each of the ω values represents the probability that the position is +1.

This results in a vector of individual probabilities for all the neurons of the system. Assume a system has five neurons, then an example might find the probabilities that the neurons should be +1 to be as follows:

$$\Omega = (0.5, 1.0, 0.0, 1.0, 0.5)$$

For example, if neurons 1 and 3 were determined to be in the affective set, we could rewrite Ω as

$$\Omega = (0.5, -, 0.0, -, -)$$

Notice that the probability of an element that is not an element of the affective set is simply no longer considered. This does not, however, preclude the possibility of using the original probabilities. It does allow us to differentiate between the affective probabilities and those of all neurons.

This raises new questions. The system must now have a control structure that implements and determines the probabilities of the element choices. Furthermore, the use of this control structure creates questions of where and how the changes will be

made. A probabilistic update system comes with its own problems. For instance, if the update probability is precisely zero, then does the system come to a halt, or can it consider other choices?

If an element is labeled as having zero probability, it should be blocked as a path to a stable point. Even if it is the only neuron in the affective set, it should not be chosen. If the path to a stable point must go through that element, then all paths to that stable point are blocked, and no stable point should be found for that path. This not only changes the idea of stabilization, but also the concept of an affective set.

3.6.2.1. The Reduced Affective Set

The introduction of probabilities has produced the capacity to eliminate particular neurons as choices for bringing change to the system. This will also change the affective set *AS*. To accomplish this, we offer the concept of the Reduced Affective Set.

The Reduced Affective Set (AS_k'), for any pass k, is the affective set *AS* for pass k if one or more of the neurons have been eliminated from *AS* as a position to choose for possible updates. For our purposes, the prime signifies that the set has had a value rejected for some reason.

Assume every learned memory the system is presented with has +1 as the value of its first position. Assume also that a probe for the system has +1 as a value in the first position. Clearly, it does not make sense to stabilize at a position in the neural net that has a value -1 in the first position, since we know the first element must be +1. Therefore, it also does not make sense to allow the probe to change the value in

its first position during intermediate passes. Assume, during pass k, that the affective set is AS_k = (1, 8, 12). It would make sense to reduce that set to eliminate the 1 as a position, AS_k' = (8, 12).

Another situation that can arise when using *AS* is that it may be desirable to eliminate choices from *AS* on a temporary basis or until no other choice exists. For this purpose, the system will not remove such elements but **mark** them. *Marked elements* are effectively removed, unless there are no other choices or the system decides that they should no longer be marked.

The elements of the affective set and the reduced affective set must be obtained and stored somewhere. The results are then imposed on the system. The resulting system is run with the new information.

3.7. Implementation of The Controller

Any implementation of the controller must include certain capabilities to create the features mentioned. The capabilities necessary are:

i. *take as input all values before processing at the neurons* ,

ii. *synchronize the incoming values*,

iii. *compare the input with the neuron values*,

iv. *add new lines (neurons) to the system*,

v. *memory to backtrack to previous decisions*, and

vi. *output value(s) to the neurons*.

The characteristics are of two basic types, hardware characteristics (i, ii, and vi)

and non-hardware characteristics (iii, iv, and v). The controller is considered to have the same basic breakdown into these components.

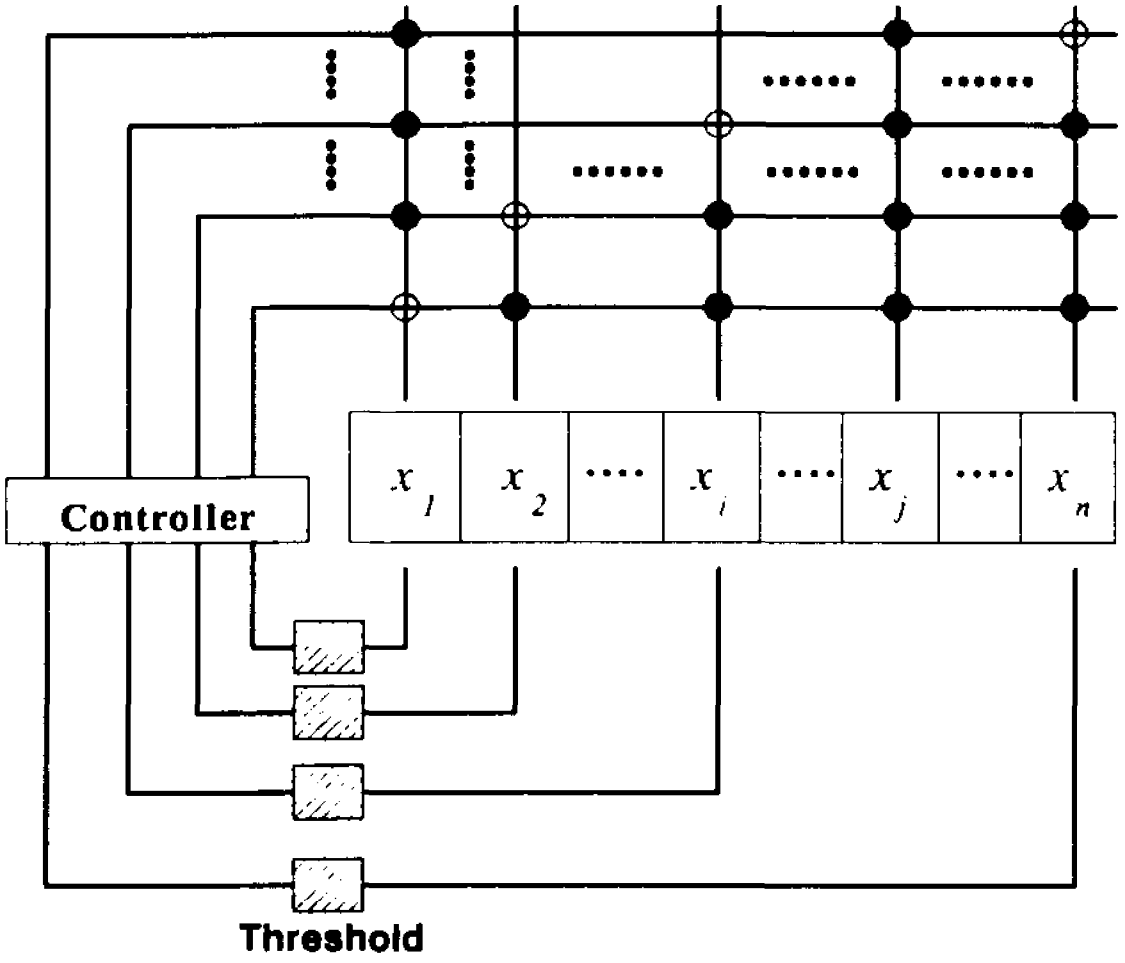


Figure 3.3
Neural Net with Control Before Threshold

The placement of the controller is an important issue, and is pictured as part of the circuit in Figure 3.3 and Figure 3.4. The pictures illustrate that two distinct possibilities exist for placing the controller in a circuit for the neural net. The controller could be placed after the summation of the neurons and could be placed before thresholding, as in Figure 3.3, or it could be placed after the values have been thresholded, as in Figure 3.4.

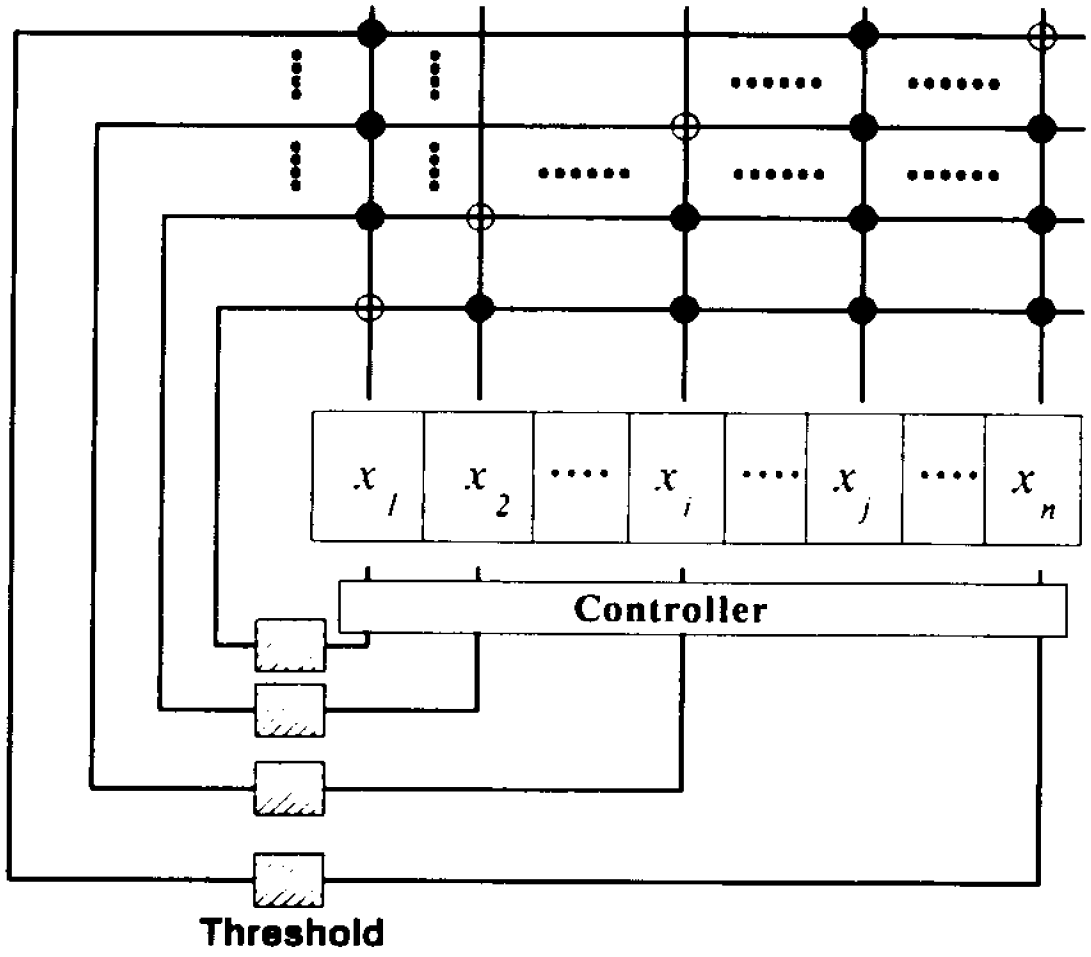


Figure 3.4
Neural Net with Control After Threshold

The choice of placing the controller after thresholding in the new model is made for the following reasons. If the controller is placed prior to thresholding, there is an indication as to the strength of the decision to change the value of the neuron. This might appear as a valuable tool, but the additional problem of thresholding in the controller made us decide to place the controller after thresholding. To control the thresholding as part of the controller would have changed a basic characteristic of the system.

3.7.1. Hardware Needs

The actual physical implementation that directly connects to the synapses is called an Update Assist Vector (UAV). The important characteristics of the UAV are:

receive values from the line,

synchronize the system, and

place values on the line.

To accomplish these with the UAV, the following hardware solution is offered. Each synapse has a line that samples the value on the synapse (line **a** in Figure 3.6). The UAV has a line that inhibits the processing of information on the synapse (line **b** in Figure 3.6), and the controller has an inverter for each synapse to allow its value to be changed at the desire of the controller (line **c** in Figure 3.6). Figure 3.5 pictures the hardware at one synapse. The UAV is the set of all synapse controls combined into a single element.

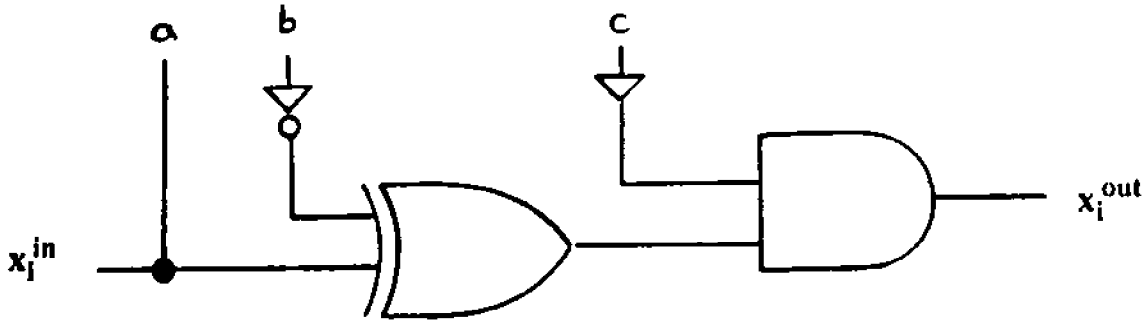


Figure 3.5
An Element of the UAV

3.7.2. Non-hardware Needs

The controller intercedes in the hardware of the system for the purpose of imposing the results of external algorithms onto the system. The particular non-hardware systems presented in this chapter are separated into three levels:

1. *Response Level* - interacts with external entities,
2. *Execution Level* - controls the systems movement as to synchronization and input of neuronal choices, and
3. *Decision Level* - a two-tiered level that consists of the particular decision to add a neuron or update a neuron and the analysis of that choice.

These levels are not independent in their nature and interact very closely with both each other and the hardware. Listing them separately allows each one to be considered in regard to its individual processes. The non-hardware level is a triad, with each particular segment integral to every other part. The non-hardware segments are pictured in Figure 3.7.

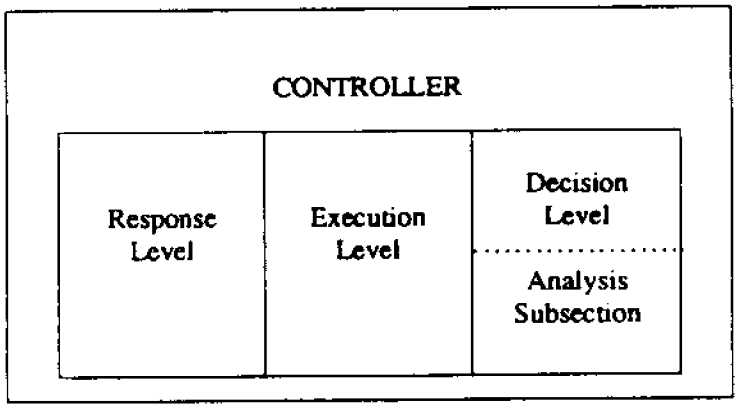


Figure 3.6
The Non-hardware Segments of the Controller

3.7.2.1. Response Level

The response level of the controller interacts with any situations that are external to system. It is assumed that the response level determines if the system is in either a learning mode or a query mode. Additionally, the system needs be able to respond to both too much information being entered into the system and a probe stabilizing at an unacceptable solution.

Consider the system in the query mode. The Controller gives two responses to the user from a probe: **solution** and **no solution**. If the response from the controller is **solution**, then the user of the system simply notes the value of the output lines of the neural net for the stable point corresponding to the probe. A response of **no solution** to a query means that no valid stable point was obtained from the particular query. In this situation any response from the system can be ignored.

This mapping is from the probe to the binary output, in the following manner:

$$\text{C(probe)} \rightarrow \begin{cases} \textit{solution} \text{ if } AS_k \text{ is empty for some k} \\ \text{no solution if } AS_1' \textit{ is empty} \end{cases} \qquad 3.1$$

where the values of AS and AS_1' are determined as follows:

AS_k is the affective set for any level that is obtained by the Controller from the synapses after an update, and

AS_1' is the reduced affective set for level one.

The actual method for reducing the affective set is discussed under backtracking in section 3.7.2.3. This can be summarized follows: if the set of neurons allowed for

update is empty and the system is not stable, the system cannot achieve a satisfactory answer.

To actually implement the correspondence with an external source, the following items should be implemented by the response level. Two fields (neurons) indicate whether a solution has been determined. The introduction of a probe at level 0 activates the UAV, setting the values to -1. In the event that a stable point is determined ($AS_k = \varnothing$ for $k > 1$), then both fields are set to +1. In the event that no stable point can be determined ($AS_0 = \varnothing$), the system would set values to -1 and +1. In this manner, when a probe enters the system, it can be determined whether a solution has been obtained by probing the tag neurons.

3.7.2.2. The Execution Level

The controller must be able to achieve physical updating of the elements of the system. This controller will accomplish this by controlling the UAV, and will realize the methods for backtracking through retrieval, will recognize component values and will retain memory.

The execution level runs two segments that interact as needed. The hardware segment runs the UAV and physically controls the synapses. The non-hardware segment requires the system to execute different moves depending upon particular properties that have been ascertained. Additionally, the execution level is required to store information in the event of backtracking. To run the UAV, the execution level uses the following UAV controller to execute passes from a probe to a stable point.

3.7.2.2.1. The UAV Control

The UAV control runs the system at different levels for each pass of the system through memory. A query entering the system is presumed to be at level 0, and the system runs a tree structure of k levels.

level 0

When a probe enters the system in a query phase, the probe is considered at level 0. If any additional information (neuron values) is needed, it is appended to the query probe at this time. The probe is then placed into the physical neural net as a probe. As in the Hopfield model, this is accomplished by multiplying the probe by the transition matrix. From this information a new vector is acquired. The system is said to enter level 1 at this time.

level 1

The UAV collects the output from the thresholded synapses (using Equation 2.1), and these values are compared, element by element, with the probe at level 0. From this comparison, the system obtains the affective set AS_1 by listing any positions that have changed. If AS_1 is empty, the system is at a stable point and the controller responds with **solution**.

If AS_1 is not empty, the **decision segment** of the controller is invoked. The decision segment will be discussed at length, but for the present only its effects are mentioned. The decision segment may choose to eliminate a neuron (or multiple neurons) from AS_1, if so AS_1 -> AS_1'. If AS_1' is empty, the controller

responds with **no solution**. The decision segment may alternately downgrade a neuron by tagging it as a choice of last resort. If either AS_1 or AS_1' is not empty, the decision segment chooses a neuron for update and the system is updated, moving to level 2.

level k

For any level k (k>0), AS_k is obtained by comparing with AS_{k-1} as in level 1. That is, if AS_k is empty, the **response level** gives an answer of **solution**. If any neurons are not allowed, as in level 1, we obtain AS_k'. The main difference between level k and level 1 is what is accomplished when AS_k' is empty. If AS_k' is empty, the system *backtracks* to level k-1. The choice of neurons in the affective set (or reduced affective set) that was used to get to level k-1 is then eliminated. The system then starts from level k-1 using AS_{k-1}'.

Notice that, in the case of the UAV controller, the system is defined recursively. Each level k has two possibilities. It can, if AS_k is empty, stop the system with a solution. The only other possibility for the system is to call itself at another level, either k+1 or k-1. The k-1 level is invoked if there are no available choices and the system must backtrack a level to try to obtain a choice. The k+1 level is invoked if there is a viable level. The decision regarding which choice to make has not as yet been considered.

An additional point that should be emphasized is that to backtrack and start at a previous point, the system state for the previous point must have been stored. This

could be implemented by a stack that holds the state of the system at every level it has visited. If any backtracking occurs, the system pops the stack to the previous state, any action necessary is taken and the system continues.

3.7.2.3. The Technique of Neuron Value Recognition

It is important that the system be able to recognize and consider any neuronal value. Since the system has the ability to set the value of any neuron, the system can poll the value of the synapse using the UAV and determine that particular value. This may not appear significant: however, in every case of improving the convergence, it is important that the system has the ability to set the values. Additionally, the controller must recognize the data and have the ability to update it.

3.7.2.4. The Technique of Backtracking

The problem can be visualized as one of managing a sequence of choices in such a way as to find a set of choices that leads to a solution. For example, in Section 2.3.1, we have two choices for the next update to be accomplished, $x_{(3)} \rightarrow -1$ or $x_{(4)} \rightarrow 1$. Although $x_{(4)} \rightarrow -1$ is a valid choice in the Hopfield model, it turns out to be a poor choice as the convergence is to a spurious state.

One approach to updating is the method of *backtracking*. We first describe it in a general form.

Backtracking can be applied to any computation with these properties:

1. There exists both a starting point and a goal.

2. The goal may be reached by starting at the starting point and following some path consisting of defined operations separated by nodes. At each node, some

arbitrary choice among a finite set must be made. An operation leading from one node to another can either succeed or fail. We say the computation blocks **if an operation fails.**

3. Depending on the sequence of choices made, the computation will either reach its goal or block on some operation. If the computation blocks, we must back up one node and try another set of choices.

A backtracking method will examine every choice and continue this technique until either the goal is obtained, or all the possible choices have been tried and have failed. Unfortunately, the computation may continue for quite a long time. In fact, it is important to know whether or not the number of operations is bounded. In the new model the system has a finite number of paths, and, therefore, will terminate in a finite number of steps.

There may be more than one path to an acceptable answer. However, the path taken depends on the order in which the choices, associated with nodes are tried. The system allows the controller to determine these choices and in no way allows the backtracking segment to involve itself in the decision process for that operation.

The backtracking method itself must be capable of backtracking in a neural net setting. To accomplish this it must have the following three abilities:

1. A *forward move* that allows the neural system to go forward in the search for a stable value. The system will use the choice from the controller which is one of the elements of the affective set. This operation continues until the system encounters one of the following situations:

a) a path is blocked

b) the system reaches a stable point

c) a halt is reached

2. A *backtrack move* due to a blocked path. To accomplish a backtrack from a blocked path to an unblocked path, the execution level needs to have the last set of neuron values, the previous affective set, and the choice for update that was previously used for the previous level. For level k this is the state of the system at level k-1 with an elimination of a choice in the affective set.

3. A *halt* is obtained as there are no more levels to backtrack from (i.e., at level 1) and there are no more choices available.

3.7.2.4.1. Application of Backtracking to the Controller

Assume the system has as a probe the string $(x_1, x_2, \cdots, x_n)$ of finite length. Assume that any required path information is available. To accomplish this at any point, place on the stack (1) the value of the probe (2) the affective set that is available (note that: any previous choices have been deleted) and (3) the choice for the next step.

A *forward move* in the system is a decision of a particular neuron to update, combined with the updating of the probe. This results in the move:

$x_1, x_2, \cdots, x_j, \cdots, x_n \rightarrow x_1, x_2, \cdots, x_j^{inv}, \cdots, x_n$ where $1 \leq j \leq n$

This new vector is then multiplied by the transition matrix T, to form a new vec-

tor p'. The new vector is compared to the old vector element by element for changes at particular positions in the new vector, forming the affective set at level k (AS_k). If every element is found to be unacceptable in AS_k, then that level from that path is considered blocked.

A *backtrack move* is implemented when a path is blocked. Upon returning to level k-1, the particular element that created the blocked state (j in this instance) is now removed from the affective set at level k-1, and changed back in the probe.

$x_1, x_2, \cdots, x_j^{inv}, \cdots, x_n \rightarrow x_1, x_2, \cdots, x_j, \cdots, x_n$ where $1 \leq j \leq n$

Using this technique, the path to the blocked state is itself effectively blocked, at least through that path.

A *halt* happens whenever an element is removed from the system and the following occurs. Since j has been removed from the affective set, that level may be empty. If that level is empty, the system will attempt to backtrack, thus created the possibility of trying to backtrack from level 1. There is no affective set for level 0, so not backtracking can occur to that level.

3.7.3. The Technique of Fixed Points

The Technique of Fixed Points exploits the controller's capability to fix neuron values. In the Hopfield model, every element in *AS* has equal probability of updating the system. This is true even in cases where particular neurons have known values and are found in *AS*. This is not reasonable, as it does not make sense to allow the

system to move from correct values to incorrect values during traversal of a path to a stable point.

The Technique of Fixed Points recognizes this fact and prohibits updating of neurons that are known to be correct. This is accomplished by removing from AS_k any element that the system has determined to be fixed. For example, if an update produced an AS of (x_1, x_4, x_8) and x_4 is fixed, then the reduced affective set AS' would consist of (x_1, x_8). This blocks the controller from choosing x_4 as the neuron to update. This is a technique that prohibits the system from ever updating that position. Therefore, once the value of a point is fixed, it can not be altered.

3.7.4. The Decision Level

The decision level of the controller is the portion of the controller that determines which neuron to update. The decision is accomplished by utilizing an analysis sub-level of learned memories and rules. To augment the decision process, an algorithm, or possibly a number of algorithms, must be chosen. There are two types of algorithms that are available to the user in the decision segment. The algorithms are those that assist in learning and retrieval, and those that assist in retrieval only.

Algorithms that assist in the learning process will have access to the information on the synapses as the information is being learned. This gives the system the ability to count, analyze and append information. Appending information is accomplished by adding neurons of particular values. This type of algorithm should obviously be chosen by the system before any information has been learned.

Decisions that assist in retrieval only have as their information all previous infor-

mation obtained during learning and information, but do not input any information during the learning process. Analysis is completed with that information only, and a neuron is chosen to update the system.

3.7.4.1. Analysis Level

This level is comprised of the different analysis methods, the results of which are used to determine a choice for updating neurons in the retrieval process. It utilizes the **execution level** to add neurons if so desired, and implements fixed elements to assist in its duties.

This segment may be considered as analogous to the programming of a traditional computer. If the programmer determines that a summation technique is sufficient, then that technique would be used; in other cases, a complex arrangement of hashing and comparison may be necessary. **Chapter 4** will explore some of these issues.

Chapter Four

Algorithms Using the Asynchronous Controller

Some men's memory is like a box, where a man should mingle his jewels with his old shoes

Earl of Halifax

4.1. Introduction

This chapter discusses various methodologies for utilizing the controller. These algorithms are applied at the decision level of the controller. The addition of the controller to a neural net might be compared to the addition of a steering mechanism to a car. While the steering mechanism provides the car's operator with the capability to maneuver the vehicle, it also leaves open the possibility of steering into a tree, if not used correctly. From this analogy, we can see that any steering mechanism is only as good as its guidance control. In this chapter, algorithms are presented that utilize the controller to improve the decision process in the system as the system converges to a stable point.

Five unique approaches that assist in the decision process are presented. The methods are: The Oracle, The Hidden Bit Method, The Method of Handles, the Fixed Point Method, and the Method of Acquired Information. The Oracle is a probabilistic assistance technique. The Hidden Bit Method adds a one bit field to improve recognition of complements. The Method of Handles adds a set of orthogonal tag bits to recognize indiviual memories. The Fixed Point Method is an *a priori* data manipulation technique to assist in acquiring a known element, and the Method of Acquired

Information searches for partial information to lead to additional information.

In addition to the methods of convergence, a method is introduced that allows the system the option of not converging. While it is desirable to converge in most instances, situations do exist where avoiding convergence may be more reasonable. We have seen an example of this in section 2.3.7, where the information is simply not close enough to the probe to be a valuable match. The algorithm is referred to as the Naive solution.

Finally, in this chapter a section is dedicated to a discussion of the cost of determining stable points. An example is given to clarify the cost of different choices in the system.

If the probability of a particular value were known at a particular location in a probe vector, it might be able to assist in the decision process. If it were known that position k has a 90% probability of being a +1, the system might not wish to change that position to a -1 unless it had no other option, or it might decide to change it to a +1 for the same reason. One reason for this decision is that the probability of the stable point's complement having a -1 in that position is 90%. This would increase the probability of the system stabilizing at a complement vector. An algorithm that would simply be the tally of neurons and provide this information to the decision process could be invoked before a choice is made to update a particular neuron. This algorithm is referred as the **Oracle**.

As was shown earlier, almost every learned memory creates two stable points, the actual learned memory and its complement. This fact can be used to avoid complements. To assist in avoiding complements, it is observed that if every learned

memory started with +1, only learned memories and spurious states would begin with +1. The Hopfield model does not allow a position to be fixed in value; however, the addition of a controller permits this capability. The controller can tag every information vector with an additional neuron that contains +1. With this, any element without +1 in that position can immediately be eliminated from consideration as a learned memory. This will be called the **Hidden Bit Method**, as a bit is placed onto the information at learning time by the system.

If one bit can assist the system, the question arises as to whether a set of bits might improve performance. In particular, the addition of orthogonal values as additional system information might give better performance during retrieval. This approach, called the **Method of Handles**, is of great significance as it is the first step towards indexing fundamental memories, and, therefore, testing a system for stable points without knowing their values.

An additional method that might be utilized is the idea of fixing the value of neurons prior to the retrieval process. If a system has determined that a particular grouping of information contains a particular pattern, and the probe used indicates that pattern is desired, then the system should fix that pattern. This is different from the Method of Handles in that the information is not appended to existing information, but determined to be part of the body of the information prior to commencement of retrieval. This is be referred to as the **Fixed Point Method**.

The final approach is accomplished during the actual retrieval process. When searching for a solution in a particular context, pieces of information that are discovered point to other information, often with some probability attached. In a

neural net system with a controller, there is no reason that, if a certain pattern is discovered, it could not lead directly to other patterns. In this manner, the speed and direction of the search are improved for acquiring a particular solutions.

In a search process, a pointer or clue can possibly be misleading. This is due to the nature of discovered information. The process introduces clues, which are pointers to additional information. Clues have a positive correlation with additional material, but are not necessarily a strong correlation for all known information. This method is similar to fixing information at the beginning of the retrieval process, but it is accomplished during the actual execution of the neural net. This method is referred to as the **Method of Acquired Information**.

To illustrate how this last methodology might work, consider a query about an animal with a beak. If it is determined that the probe has gotten sufficiently close to a predefined set, *i.e.*, bird, the information may force the additional constraint of the animal having wings upon the system. This could be accomplished by requiring an additional set of points in the probe to be fixed. The probe is then extended as if it had discerned the new constraint.

Each of these methods is designed to improve the possibility of obtaining a stable point in the system that corresponds to a learned memory. The reason a particular learned memory is desirable may vary, but the method is sufficiently flexible as to sustain this variability.

4.2. The Oracle

The Oracle examines global properties, particularly the numbers of elements

that assist in the update procedure. The summation of elements is accomplished with a simple sequential procedure on the learned memories. This demonstrates how neural networks can be augmented using sequential processing techniques to improve their capability for obtaining a solution. This technique might alternately utilize neural nets for the procedure.

4.2.1. Definitions and Fundamental Conditions

The system of neural nets is presented with fundamental memories as vectors of information:

$$x_{(k)} = (x_1 , x_2 , \ldots x_n)$$

where **k** represents the **k**th element of **m** fundamental memories and **n** is the number of neurons or positions in each vector. Using the Hopfield model, we know that the neurons (x_i) of the vectors for $i \in [1 .. n]$ are either +1 or -1. As with the Hopfield model, we assume that the probability of either in any position is fifty percent:

$$\text{Prob}(x_i = 1) = 0.5 \quad \forall\, i \le m \qquad 4.1$$

In addition to the above requirement, it is important that all individual elements, as learned, are assumed independent. That is, whether or not a particular position is a particular value has no effect on the probability of any other primary element in the vector. This can be written as:

$$\text{Prob}(x_i = a \mid x_j = b) = \text{Prob}(x_i = a) \quad \forall\, 1 \le i, j \le m \text{ and } a, b \in (1,-1) \qquad 4.2$$

In addition, the model assumes that the distribution of information points given in the memory are themselves a random distribution of points. That is, that the individual information vectors to be learned are not dependent on previous vectors.

By applying the above concepts, we can state that the probability of a particular element being a +1 can be given by the equation:

$$\text{Prob}(x_i = 1) = \frac{\sum_{j=1}^{m} x_j}{2 \times m} + 0.5 \qquad 4.3$$

The value for the probability that x_i equals -1 can be obtained as 1 - P (x_i = 1).

However, in practice, it will not be necessary to know the probability of a particular value, only whether it is most likely the correct value for that position. To accomplish this, the **Oracle** will use a probability vector. This vector is obtained by using the probabilitites (Equation 4.3) from each position.

$$\Omega = (\text{Prob}(x_1 = 1), \text{Prob}(x_2 = 1), \cdots \text{Prob}(x_m = 1)) \qquad 4.4$$

This vector contains values from 0 to 1, each dependent on the number of positive or negative elements at each neuron. This vector is used to give a general description of the system's state for a particular neuron.

4.2.2. Methodology

The method uses the technique of creating the individual matrix $\mathbf{T}^i$, an n by n matrix, from the outer product of the m fundamental memories ($x_{(k)}$). These matrices are then summed to form the transition matrix **T**.

The transition matrix is used to obtain the stable point in the following manner. A probe vector (p) is then multiplied by the transition matrix as:

$$p\,T = p' = (x_1 , x_2 , \ldots x_n)$$

These components form the new vector p′, where the individual components of p are compared with p′ for changes in the sign of individual neurons. The set of all changes

is the *affective set*. If there are no changes, then the point is stable. However, if there are changes, the changes in sign indicate the components that are possible candidates for change in the probe vector. With the Hopfield model, the technique chooses a neuron *at random*, changes the sign of that neuron, and restarts the process. This is where the **decision segment** of the controller consults the Oracle.

The Oracle offers a method to probabilistically improve the process of choosing a neuron to change once an affective set has been determined. The basic theory is that an improvement in the accuracy of the sign of any particular neuron will improve the probability of obtaining the correct stable point for the entire vector.

If the recommended change is assumed to be from -1 to +1, then technique gives a recommendation value ranging from 0 to +1. The range of values is considered to vary from *not recommended* to *strongly recommended*, based on the range 0 to +1. The set of recommendations is then searched for its largest value. Thus, the neuron most likely to lead to a correct value is chosen. If however, the recommended change is from +1 to -1, then $1 - P(x_i=1)$ is used for the recommendation value.

The change of sign is noted by its position, and is now considered as a set of suggestions to be analyzed. The Oracle vector is considered for each position and each probability is determined.

The values of the Oracle act as likelihood operators for each element. Since we have assumed the independence of each element, this technique simply improves the likelihood of getting that particular element chosen correctly.

The number of information vectors (m) is the same as the number of elements in

position k. Obtaining a *best* choice for the k^{th} position can be accomplished by looking at the likelihood of either value, 0 or 1.

An additional area of concern is the probabilistic inference that is made by the Oracle. The fact that a decision is made using probabilistic information leads to some potential concerns that should be addressed.

First, the Oracle does require a few mathematical assumptions that effect the decisions and outlook of the system. It is assumed, as with the Hopfield model, that the probability of either binary value (+1 or -1) for a neuron is 0.5. This assumption permits the observance of a 90% rate of +1 or -1 to something out of the ordinary. This also allows the system to consider a particular value to be of equal importance; therefore making the value of a significantly higher rate a more desirable choice. If the value itself were important, then the opposite value might even become more significant.

In fact, the Oracle determines a statistical hypothesis and tests against that hypothesis. In this instance, the hypothesis might be that the neuron value is a +1. This type of decision basis for a problem creates two types of errors. These are the same as the types of errors that occur in statistical hypothesis testing, *Type I errors* and *Type II errors*. In Type I errors, a hypothesis is accepted that is not valid. In the Oracle, the decision to choose a neuron for update direction is made in a direction that may not be valid. In Type II errors, a valid hypothesis is rejected due to insufficient data. This could occur in the Oracle if a path that should be taken is not taken due to insufficient data. The value that is determined as a minimal value for a decision is referred to as the δ value.

4.2.3. The Algorithm

This is a retrieval algorithm to improve the probability of convergence to a learned memory.

Step One: Determine a minimal value for the decision δ

Step Two: Determine a Positional Probability Vector Ω

Develop a vector Ω of length **n** using Equation 4.3 such that each position contain the probability ω_i of that neuron being the value one (+1). Note, the value of the position being minus (-1) will be $1 - \omega_i$.

Step Two: Obtain the Affective Set

Apply the probe to neural net and collect the affective set.

Step Three: Select the element to update

From the affective set, choose the largest value out of ω_i ,for a recommendation to a +1 or $1 - \omega_i$, for a recommendation of -1. (Assume ω_i, for clarity). If $\omega_i - \delta \geq 0$, then update neuron i. If $\omega_i - \delta < 0$, then randomly select the next neuron from *AS*, updating the corresponding neuron.

Step Four: Repeat Until Complete

Repeat steps two and three until an affective set is nul.

The actual job of the Oracle is to keep track of the probability that a neuronal position is a particular value. When the system is run during retrieval, if there are large values then they can be exploited. That is, to find a neuronal probability significantly different from 0.5 as to merit using it as a selection.

4.2.4. Example

A small system is illustrated that exemplifies the utilization of the method, even though the system does not have a great number of learned vectors. The vectors that the system has learned are:

$$x_{(1)} = (\ 1, -1,\ 1, -1,\ 1,\ 1, -1\ )$$
$$x_{(2)} = (\ -1, -1,\ 1, -1, -1,\ 1,\ 1\ )$$
$$x_{(3)} = (\ 1,\ 1,\ 1, -1,\ 1, -1,\ 1\ )$$

The δ value is chosen as 1.0 in this instance, as there is not a great deal of data. These learned memories have a probability vector (Ω) of:

$$\Omega = (\ 0.67, 0.33, 1.0, 0.0, 0.67, 0.67, 0.67\ )$$

The probabilities represent the probabilities that the values, if arbitrarily determined, would be +1. This is an arbitrary choice and could have just as easily represent the probability of being -1.

From this set of learned memories we obtain the transition matrix:

$$T = \begin{bmatrix} 0 & 1 & 1 & -1 & 3 & -1 & -1 \\ 1 & 0 & -1 & 1 & 1 & -3 & 1 \\ 1 & -1 & 0 & -3 & 1 & 1 & 1 \\ -1 & 1 & -3 & 0 & -1 & -1 & -1 \\ 3 & 1 & 1 & -1 & 0 & -1 & -1 \\ -1 & -3 & 1 & -1 & -1 & 0 & -1 \\ -1 & 1 & 1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

The following probe is applied to the learned system:

$$p = (-1 -1 +1 +1 -1 -1 +1)$$

From this, we obtain the new probe:

$$p' = (-1 +1 -1 -1 -1 +1 +1)$$

Resulting in the affective set:

$$AS = (2, 3, 4, 6)$$

From this set, the Oracle determines the probability of each individual neuron being changed to a new value.

The probability that **position 2** should be a +1 is **0.33**.

The probability that **position 3** should be a -1 is **0.0**.

The probability that **position 4** should be a -1 is **1.0**.

The probability that **position 6** should be a +1 is **0.67**.

From this set of choices, the Oracle concludes that the proper choice for change is position 4 from +1 to -1. This is done, and a new set of choices is given as:

$$p = (-1 +1 +1 -1 -1 +1 +1)$$

Yielding a new affective set:

$$AS = (2, 6)$$

The probability of each neuron is given as $\omega_2 = \text{prob}(x_2 = 1) = 0.33$, and the $\omega_6 = \text{prob}(x_6 = 1) = 0.67$. Since neither value is greater than the predesignated δ value of 1.00, the system randomizes among the two to determine a value.

4.2.5. Proof of Correctness

Two important questions need to be addressed when probabilistic updating is

used:

Does the system converge to a fixed point?

Does the use of probabilistic updating improve the system?

Consider probabilistic updating, which incidently has no backtracking, in termes of the choice of paths it takes. When a path is chosen, it will invariably be a path that is a possible path for the Hopfield model. This leads to the conclusion that it must exhibit the properties of the Hopfield model for that path.

Theorem: Probabilistic updating of a probe using the asynchronous model converges to a fixed point.

Proof: Consider the decision to choose a particular element for updating. This element is also an element of the affective set (*AS*) of the Hopfield model. Therefore, each position updated is equivalent to an update decision in the Hopfield model. Since the Hopfield model always converges (section 2.3.9), then it converges for any particular decision to update elements that form any path. Therefore, the path chosen by the probabilistic updating method will converge to a fixed point.

To answer the second question, it is imperative that an understanding exists as to what is meant by an improvement to the system. The type of improvement that is considered is an improvement to the probability of stabilizing at a learned memory. Every system has approximately 50% of its stable points created by complements of learned memories or complements of spurious states.

Theorem: Probabilistic updating can improve the probability of a probe stabilizing at a learned memory.

Proof: There are three cases to consider.

Case 1: The probability (P) of a probe p_i of obtaining a learned memory is zero (P = 0).

If P=0, then no path can improve the probability of obtaining a learned memory.

Case 2: The probability of a probe p obtaining a learned memory is one (P = 1).

If P=1, then any path will lead to a learned memory and no choice of path can improve the probability.

Case 3: The probability of a probe obtaining a learned memory is grater than zero and less than one (0 < P < 1).

Assume that a probe must go to level k to achieve stability. Since the probability of a +1 or -1 is assumed to be 50%, the probability of obtaining a learned memory versus a complement memory is 50%. This is obtained by summing the probability of each element for each path. The probability of each path is $\prod_{k} 0.5$. These are summed over all paths and approximately 50% is achieved.

Consider the following two cases for x_i:

a. For all x_i, $\omega_i \leq \delta$.

Since no element has a value great enough to implement the system the values of the probabilities do not change.

b. For some x_i, $\omega_i > \delta$.

Since the system uses the update method, the probability of the value being a learned memory is δ, which is an increase of δ - 0.5. Therefore, the probability of obtaining a learned memory at some level is increased by δ - 0.5. This increase is translated to the increase δ - 0.5 + $\prod_{k=1}$ 0.5.

4.3. The Hidden Bit Method

The Hidden Bit method is designed to eliminate the convergence to the set of complementary stable points. We have seen that one problem with the Hopfield model is that, when any fundamental memory is learned by the system, its complement becomes a stable point in that system (section 2.3.3). This is due to the definition of the system using +1 and -1. The Hidden Bit method uses the probabilistic characteristics of the new model, and backtracking to disallow the stabilization at a complementary point.

4.3.1. Definitions and Fundamental Conditions

We have seen, in the Hopfield model, the systems are chosen to have **n** neurons (x_i). Each neuron is an element of the memory system:

$$x_{(k)} = (x_1, x_2, \ldots, x_n)$$

The control has the ability to activate additional neurons to assist in the learning process. We will utilize an additional neuron in a position that is referred to as a *hidden bit position*. This is normally located in the first position, for clarity.

4.3.2. The Methodology

The system is an **n+1** neuron system. As each fundamental memory is learned by the system, a tag bit of +1 is placed on the end of the vector hidden from the user. Each augmented vector that is learned will now look as follows:

$$x_{(k)}^{+} = (+1, x_1, x_2, \ldots, x_n)$$

Therefore, all memories that are intended to be in the system will have an additional +1 that has been apppended. To obtain a solution to a query, the system will require a +1 in the first position.

As the system goes from a learning mode, to a retrieval mode the system sets the method of fixed point retrieval to require a +1 in the first position. This is accomplished by the hardware using the method of Neuron Value Recognition (see section 3.7.3). This combination defines a situation with a strong attracter in the first position and therefore guarantees that this position will either be a +1 or no solution will be found for that particular probe.

4.3.3. Analysis

Appending a tag bit accomplishs two specific points that improve the retrieval process. By appending +1 to the information, it is guaranteed that every fundamental memory is defined at least a Hamming distance of two from every fundamental memory complement. Clearly, this alone does not accomplish much that the Hopfield model does not already possess as a capability. The significant aspect of the Hidden Bit Method is that it gives a position that has a known value.

The retrieval process is accomplished in the following manner. A probe that enters the system immediately has a tag field with a one in it placed on the end. The probability of allowing the tag field to be chosen for updating is set to zero. Using this technique, the tag field can never change from a +1 to a -1. The system is therefore never allowed to stabilize at a complementary stable point. The system cannot ever give as its answer to the probe anything other than fundamental memories or spurious states with a +1 in the first position.

This method guarantees that the probability of a complementary stable point is zero. It does not guarantee that spurious states will not be obtained. For spurious data, their complements will also be recognized.

4.3.4. The Algorithm

This is The Hidden Bit algorithm for improving the learning and retrieval processes of the model.

Learning

Step One (L): Append a positive one (+1) to any information that is learned in

the hidden bit position.

Retrieval

Step One (R): Set the Control to the Fixed Value of +1 in the hidden bit position.

The value for the fixed point is set in the control structure.

Step Two (R): Obtain the Affective Set

Using the usual method of obtaining updates, obtain the set of values to change.

Step Three (R): Disallow Bad Elements

In the set of neurons that might change, remove any element that would change the values of fixed element. If there are no element left in the set, then backtrack to a previous level. If there are no other iterations, stop.

Step Four (R): Repeat Until Done

Repeat steps Two (R) and Three (R) until the set is null.

It should be pointed out that there is no guarantee that the system will determine a stable point associated with a particular probe. In the case of a probe having no path to a stable point, the probe could then use a technique such as simulated annealing

[Kirk] to enhance the possibility of finding a path. However, this is of no concern to this technique.

4.3.5. Example

The following example demonstrates how a system could add a tag field of +1 onto the left-most position for n = 7 neurons. Assume the fundamental memories are:

$$x_{(1)} = (-1,\ \ 1, -1, -1,\ \ 1,\ \ 1,\ \ 1)$$
$$x_{(2)} = (\ \ 1,\ \ 1,\ \ 1, -1,\ \ 1, -1,\ \ 1)$$
$$x_{(3)} = (-1, -1, -1, -1, -1,\ \ 1,\ \ 1)$$

From these the following transition matrix is formed:

$$T = \begin{bmatrix} 0 & 1 & 3 & 1 & 1 & -3 & -1 \\ 1 & 0 & 1 & -1 & 3 & -1 & 1 \\ 3 & 1 & 0 & 1 & 1 & -3 & -1 \\ 1 & -1 & 1 & 0 & -1 & -1 & 1 \\ 1 & 3 & 1 & 1 & 0 & -1 & 1 \\ -3 & -1 & -3 & -1 & -1 & 0 & 1 \\ -1 & 1 & -1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Consider the possibilities when the probe:

$$p = (1, 1, 1, 1, 1, -1, 1)$$

is entered into the system. This probe is a Hamming distance of 1 from the third learned memory ($x_{(3)}$). This is attributed to a difference in position seven (7) as the probe has a +1 in that position and the learned memory has a -1. Additionally, the distances between the probe and the other learned memories are six (6) and three (3). It would be assumed that this probe would therefore stabilize at this point. The affective set that is obtained is (4,7):

$$AS = (\ 4, 7\ )$$

If the seventh position is chosen and updated, the desired result is obtained; however, if the forth position is chosen and updated, then another stable point is obtained. This stable point is a complement of one of the other learned memories.

To apply the new technique, we tag each memory with an additional neuron that is fixed at +1 as the memory is learned. The learned memories are then:

$$x_{(1)} = (\ 1\ -1\ \ 1\ -1\ \ 1\ \ 1\ \ 1\ \ 1\ )$$
$$x_{(2)} = (\ 1\ \ 1\ \ 1\ \ 1\ -1\ \ 1\ -1\ \ 1\ )$$
$$x_{(3)} = (\ 1\ -1\ -1\ -1\ -1\ -1\ \ 1\ \ 1\ )$$

With the corresponding transition matrix:

$$T = \begin{bmatrix} 0 & -1 & 1 & -1 & -3 & 1 & 1 & 3 \\ -1 & 0 & 1 & 3 & 1 & 1 & -3 & -1 \\ 1 & 1 & 0 & 1 & -1 & 3 & -1 & 1 \\ -1 & 3 & 1 & 0 & 1 & 1 & -3 & -1 \\ -3 & 1 & -1 & 1 & 0 & -1 & -1 & 1 \\ 1 & 1 & 3 & 1 & -1 & 0 & -1 & 1 \\ 1 & -3 & -1 & -3 & -1 & -1 & 0 & 1 \\ 3 & -1 & 1 & -1 & -3 & 1 & 1 & 0 \end{bmatrix}$$

With the new system, apply the original probe to a slightly enlarged system:

$$p = (\ 1, 1, 1, 1, 1, 1, -1, 1\ )$$

The affective set obtained is:

$$AS = (\ 1, 5, 8\ )$$

This is probably not too surprising as the forth position has now shifted to the fifth position and the seventh position has now shifted to the eighth position. Therefore,

these two are the same update selections as previously. The fact that the first position is in the affective set means that the system is attracted to a point that is not allowed. The affective set for level 1 becomes:

$$AS_1' = (5, 8)$$

A choice is now obtained from this reduced affective set (AS'). If the eighth position is updated, the desired result and correct stable point is reached. If, however, the fifth position is chosen, the system offers as an affective set:

$$AS_2 = (1)$$

Since position 1 is not allowed, the affective set is reduced by removing 1 as a choice and it becomes:

$$AS_2' = \varnothing$$

Since this set is empty, the system backtracks to level 1 and eliminates the selection the fifth position, resulting in:

$$AS_1' = (8)$$

The system now stabilizes at the correct memory.

4.3.6. Proof of Correctness

The purpose of this method is to eliminate a solution for a probe that is a complement vector to a learned memory.

Theorem: The Hidden Bit Method does not allow the system to stabilize at a complement vector to a learned memory

Proof: For every learned memory $x_{(i)}$ an additional element of +1 is added to a hidden bit position on the vector. Therefore, the complement to any

vector will have the complement to that value (-1) in the hidden bit position. Since the system removes the position from the affective set, that position can never be changed to a -1. Therefore, no complement can ever be a stable point in the controlled system.

4.3.7. Sample Results

To obtain insight into the results that might be obtainable using this method, sample results of various systems are included in table 4.1, noting their global properties. In this instance, it is assumed that each attraction basin has the same size for each stable point.

Table 4.1 illustrates that using the Hidden Bit Method usually improves the overall probability of obtaining a fundamental memory. Unfortunately, in some cases adding a bit actually creates extra spurious states and the system deteriorates. The particular effect of adding a single bit may well decrease as the size of the system increases. Notice that adding a similar bit is approximately 14% of the information vector. Since it is known that spurious states are created from linear combinations of states, it seems reasonable that the addition of 1% or 2% as tags would not influence the creation of spurious states as significantly.

The system showed remarkable improvement in the probability of obtaining learned memories. The decrease in improvement from smaller to larger is at present unaccounted. It may well have to do with the inability of the smaller system to hold the same number of information vectors as larger systems. Therefore, an assistance

A Sample Table of Results for One Added Bit Using 7-bit Information							
Hopfield Model				New Method			
f	*c*	*s*	% *Correct*	*f*	c^+	*s*	% *Correct*
1	1	0	50	1	0	0	100
1	1	0	50	0	0	2	0
3	2	0	60	2	0	0	100
2	1	0	66	1	0	1	50
2	2	1	40	3	0	0	100
3	3	0	50	3	0	4	43
3	3	0	50	3	0	0	100
2	2	0	50	2	0	0	100
2	0	0	100	2	0	0	100
4	2	0	67	4	0	0	100
1	1	0	50	3	0	0	100
1	1	2	25	1	0	0	100

where:

f	potential fundamental stable points
c	potential complement stable points
s	potential spurious stable points
c^+	$c = 0$, no stable point complements allowed

Table 4.1

technique might be more valuable to a system that is having problems.

A Sample Table of Observed Improvement for Certain Values of N			
N	Hopfield % Correct	New Method % Correct	Observed Improvement
7	51.4	86.4	35.0
8	49.8	89.5	39.7
9	48.6	88.6	40.0
10	46.4	68.0	21.6
11	44.7	78.8	32.1
12	38.2	61.6	23.4
13	39.1	62.2	23.1
14	38.7	61.7	23.0

Table 4.2

4.4. Method of Handles

The Method of Handles is offered as a technique to append data to information prior to learning, in an effort to improve the ability of the system to retrieve it. The previous method used a single appended bit to assist in the retrieval of valid results. This method extends that concept to appending multiple bits, referred to as *handles*.

4.4.1. Definitions and Fundamental Concepts

Handles exist in two varieties, *group handles* and *orthogonal handles*. The purpose of the group handles is to tag a particular piece of information as belonging to a group. This is accomplished by appending the same handle to any element in that group. If, during the retrieval process, that particular handle appears, then the system may assume that this indicates that particular group is desired. A potential problem is that of encouraging linear combinations in the memory, thus causing spurious states.

The second type of handle is the orthogonal handle. These handles are specifically chosen as to be unique for each piece of information. If these handles are orthogonal, the linear combination of the handles would be easily recognizable and therefore rejected.

The remainder of this section deals exclusively with orthogonal handles. The group handle can be treated as if it were part of the information, and not appended by the system. It may therefore be considered in the section on the Fixed Point Method.

4.4.2. Methodology

The system will only append orthogonal vectors to discourage the creation of linear combinations of information vectors, thus avoiding the formation of spurious stable points. It is instructional to digress on the matter of orthogonality.

If two vectors are orthogonal, their inner product is zero. The linear combination of any set of the orthogonal vectors cannot create one of the original vectors. It has been mentioned that one type of spurious state is a direct result of linear combinations of learned memories. This approach eliminates the possibility of obtaining this type of spurious state as a result of retrieval.

4.4.3. Algorithm

This is a learning and retrieval algorithm to assist in convergence to specific learned memories.

Learning

Step One (L): Develop the Handles

Select the desired size of the handle for the information. Develop v orthogonal vectors of length **k** ($v \leq k$).

Step Two (L): Add the Handles

Concatenate the handles to the information during the learning process.

Retrieval

Step One (R): Set the Control to the Fixed Values

In the control structure, the values for the handles are initialized to their known values. Thus, the system sets them as noncandidates in an affective set.

Step Two (R): Obtain the Affective Set

Probe the system and obtain an affective set from the UAV.

Step Three (R): Disallow Undesirable Elements

From the set of neurons that might change, remove any that are disallowed by step one. If there are no element left in the set, then backtrack to a previous iteration. If there are no other iterations, stop.

Step Four (R): Repeat Until Done

Repeat steps Two (R) and Three (R) until the set is null.

The importance of the separation of the learning segment from the retrieval segment in this algorithm should be noted, due to the potential complexity of the learning algorithm.

4.4.4. Example

To illustrate this technique, consider three fundamental memories. For the three fundamental memories, three orthogonal handles are chosen to be added to the system. In the example illustrated, the three fundamental memories happen to correspond to a simple set of 1, 2 and 3. The only real requirement is that they are orthogonal. This particular system is compared to a system with the same learned memories, including the handles, to demonstrate the difference in the retrieval techniques.

Consider the fundamental memories to be:

$$x_{(1)} = (-1,\ 1, -1,\ 1,\ 1,\ 1,\ 1)$$
$$x_{(2)} = (\ 1,\ 1, -1, -1,\ 1, -1,\ 1)$$
$$x_{(3)} = (-1, -1, -1, -1, -1,\ 1,\ 1)$$

and the handles are:

$$h_{(1)} = (-1, -1,\ 1)$$
$$h_{(2)} = (-1,\ 1, -1)$$
$$h_{(3)} = (\ 1, -1, -1)$$

By appending the handles to the fundamental memories, we obtain the set of fundamental memories as:

$$h_{(1)}x_{(1)} = (-1, -1, \ 1, -1, \ 1, -1, \ 1, \ 1, \ 1, \ 1)$$
$$h_{(2)}x_{(2)} = (-1, \ 1, -1, \ 1, \ 1, -1, -1, \ 1, -1, \ 1)$$
$$h_{(3)}x_{(3)} = (\ 1, -1, -1, -1, -1, -1, -1, -1, \ 1, \ 1)$$

and the transition matrix is determined to be:

$$T = \begin{bmatrix} 0 & -1 & -1 & -1 & -3 & 1 & -1 & -3 & 1 & -1 \\ -1 & 0 & -1 & 3 & 1 & 1 & -1 & 1 & -3 & -1 \\ -1 & -1 & 0 & -1 & 1 & 1 & 3 & 1 & 1 & -1 \\ -1 & 3 & -1 & 0 & 1 & 1 & -1 & 1 & -3 & -1 \\ -3 & 1 & 1 & 1 & 0 & -1 & 1 & 3 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & 0 & 1 & -1 & -1 & -3 \\ -1 & -1 & 3 & -1 & 1 & 1 & 0 & 1 & 1 & -1 \\ -3 & 1 & 1 & 1 & 3 & -1 & 1 & 0 & -1 & 1 \\ 1 & -3 & 1 & -3 & -1 & -1 & 1 & -1 & 0 & 1 \\ -1 & -1 & -3 & -1 & 1 & -3 & -1 & 1 & 1 & 0 \end{bmatrix}$$

The system is now examined by applying the probe:

$$p = (-1, -1, 1, -1, -1, 1, 1, 1, 1, 1)$$

It is desirable for this system to converge to the fundamental memory $x_{(1)}$. Another way to look at it, is that it is imperative that it converge to the probe labeled with a (-1, -1, 1). At level one the affective set is:

$$AS_1 = (1, 5, 6, 8, 10)$$

However, it is known that the first position must remain a -1, so the affective set is reduced accordingly:

$$AS_1' = (5, 6, 8, 10)$$

If the Oracle is now consulted, the following probabilities are found for the four positions:

$$\Omega = (0.67, 1.0, 0.33, 0.0)$$

The Oracle would certainly decide to update position six (6). From this decision a new affective set is determined, $AS_2 = (\ 5\ )$. The stable point is directly obtained by updating this value.

4.4.5. Proof of Correctness

The Method of Handles is designed to allow a system to refine its choices. This improves the probability of retrieving a correct element by improving the learning of the system. The improved learning is utilized to obtain an improved probability of converging on a learned memory.

Theorem: The Method of Handles can improve the probability of obtaining a learned memory.

Proof: The probability of obtaining a correct solution for a system is the sum of the probabilities of obtaining a correct solution for each path. Therefore, it is sufficient to improve the probability of obtaining a correct solution for each path. The probability of obtaining a correct solution for a path is the product of the probability of obtaining a correct solution at each level. The probability for obtaining a correct solution at any level j can be determined as the number of elements at level j that return a correct solution, divided by the number of elements at level j. The Method of Handles eliminates elements at any level j that directly lead to an undesirable answer. Therefore, the probability of obtaining a correct solution at level j is greater with the Method of

Handles. Hence, the probability is improved for that path.

4.5. Method of Fixed Point Retrieval

Fixed Point Retrieval is so named because it assumes a portion of the information is fixed prior to retrieval. Although it is considered to be strictly a retrieval algorithm, it is used as a rear-end addition to group handles. The system fixes particular portions of the primary probe and these elements are not allowed to vary.

4.5.1. The Methodology for Fixed Point Retrieval

We will assume know information exists that, when encoded, should not change throughout the retrieval process. This means that the only acceptable solutions to a probe with a fixed portion of information are those that only contain this fixed portion.

4.5.2. Algorithm

The method of fixed point retrieval is a learning and retrieval algorithm to improve the probability of converging to specific learned memories. This is an algorithm to improve the learning and retrieval process of the Hopfield model by fixing portions of the information.

Learning

Step One (L): From the information given, determine if there are fixed patterns.

Retrieval

Step One (R): Fix a Portion of the Probe Vector

At level 0, a portion of the probe vector is determined to be fixed. This is accomplished by setting the UAV control to the respective values.

Step Two (R): Obtain the Affective Set

Using the usual method of obtaining updates, obtain the set of values to change form the UAV control.

Step Three (R): Disallow Undesired Elements

In the set of neurons that might change, remove any that would change the values of fixed elements. If there are no element left in the set, then back-track to a previous iteration. If there are no other iterations, stop.

Step Four (R): Repeat Until Done

Repeat steps Two (R) and Three (R) until the set is null.

Notice that the retrieval method in the Fixed Point Method is identical to the Method of Handles. In both cases, the result of one portion of the solution is known prior to stabilization at a point.

4.6. The Method of Acquired Information

The Method of Acquired Information differs from the Fixed Point Retrieval Method in two ways. First, the elements of the probe are fixed during updating of the probe; second it is based on the creation of *clues*. This method assumes that the system is searching for information that contains multiple keys. If a portion of a probe resembles a key, then that portion of the vector becomes a fixed point. This could initiate a probe with the original data for the probe replaced with the new information, or

continue with the probe at that level with the remaining elements as they were when the resemblance was discovered.

4.6.1. Definitions and Fundamental Concepts

The system determines a path, or choice of neuron updates, based on the discovery of certain patterns of neurons. These patterns are either in the probe as it enters the system or become part of the probe through the update process. The patterns that the probe is checked against are called *keys*, and the patterns that are discovered in the probe are called *clues*. The keys that are matched to the set of recommended neurons are update choices; these are called the *suggestion base*.

4.6.2. Methodology

The controller contains a set of keys prior to the actual retrieval process. The keys are linked to the update of one or several recommended neurons. In the search process, the controller checks at every level for a match to any of the keys. If a match is discovered, the affective set is examined for a match to the suggestion base. If one or more matches are present, then only those matches are used to make a decision during the update process. If there is no match, then the process updates as a random process.

4.6.3. Algorithm

This is an algorithm to improve the learning and retrieval process of the Hopfield model.

Learning in the Controller

Step One (L): Develop the Set of Keys

The set of keys is placed in the controller for comparison to the probe as it is updated by the system.

Retrieval

Step One (R): Input the Probe and Obtain the Affective Set

Step Two (R): Compare the Set of Keys for matches using the Affective Set as Possible Updates

For each key, compare the probe to the key to see if any possible match exists. This comparsion is done using combinations of elements of *AS* and the probe.

Step Three (R): Determine an Update Element

If an element from *AS* could create a key, use that element to update the probe. If there is no choice that will accomplish this task, randomize over the set of elements to obtain a choice.

Step Four (R): Repeat Until Done

Repeat steps Two (R) and Three (R) until the set is null.

Again, the separation of the control instructions and the system update should be noted.

4.6.4. Example

In the following system, assume the learned memories are:

$$x_{(1)} = (-1, -1, 1, 1, -1, -1, 1, 1, -1)$$
$$x_{(2)} = (1, 1, 1, 1, 1, -1, 1, 1, -1)$$
$$x_{(3)} = (1, -1, -1, -1, -1, -1, 1, 1, -1)$$

In addition, the controller has acquired the following key:

$$k_{(1)} = (1, 1, 1, -, -, -, -, -, -, -)$$

This means that, if a situation arises such that updating a particular neuron would result in the formation of the key, then this updating should be performed. If, in this instance, two out of the three key neurons were +1 and the choosing of a third neuron would form the key, then this set should be chosen.

Then, for the probe:

$$p = (1, -1, 1, -1, 1, -1, 1, -1, -1)$$

The affective set and the probabilities are obtained as:

$$AS = (2 \quad 3 \quad 4 \quad 5 \quad 8)$$
$$\Omega = (0.33 \; 0.33 \; 0.67 \; 0.67 \; 1.00)$$

Clearly, a correct choice at this point is to update the eighth position, as it has a very high probability of being a +1 in a fundamental memory:

$$AS = (2 \quad 4 \quad 5)$$
$$\Omega = (0.33 \; 0.67 \; 0.67)$$

At this point, the key might be noted and the second position chosen over the others that have a higher probability of obtaining a learned memory, but do not lead to a key component. In doing so, the affective set and the probabilities are then obtained as:

$$AS = (\ 4\ )$$
$$\Omega = (\ 0.67\ )$$

Which leads to the fundamental memory:

$$p = (\ 1,\ 1,\ 1,\ 1,\ 1, -1,\ 1,\ 1, -1\ )$$

An important point to note is that by choosing either 4 or 5 to update prior to choosing 2, this system will not stabilize at the same fundamental memory.

4.7. Naive Solution

Another potential problem in neural nets is that of a naive solution to a query. A naive system might have too little, or the wrong kind, of information to truly be utilized as a method to recall particular data. Since the asynchronous method always gives a solution, some restrictions might be arguably added. In a naive system, we may assume that any query, may be a significant distance in the neural net from any point the neural net has learned. If this is the case, it may be useful to add an additional constraint to the backtracking method. This constraint would allow a termination of a search, or minimally, a path that is not sufficiently similar to a particular stable point.

4.7.1. Definitions and Fundamental Conditions

It is assumed that information should be within a prescribed distance of stable point s, called the *realm of similarity* of s. A probe that is not in a realm of knowledge

of **s** is considered too far away from **s** to have **s** as its solution. However, there is no way to know, prior to the probe, where it will be in the system. To determine its decision, the method can stop a probe that has gone through too many levels in search of a stable point. Since the updating technique always converges, if the probe has searched too many steps, then it was originally too far from **s**. Stopping the processing along a path is considered the same as a blocked state.

Assume that a neural net has **n** elements. This creates a space of 2^n points. A boundary value of θ is assumed, such that if the level $k > \theta$, then the probe is not in the realm of similarity of **s**. We assume that the query is not in the realm of knowledge of the stable point if it is not stable after at level k. This eliminates a very long search path to a stable point.

4.7.2. Algorithm for a Naive System

This algorithm is implemented to eliminate search paths that are too long. That is, the method eliminates solutions that are simply too far away from a stable point to be considered as a match for the probe.

Step One: Determine a Boundary Value

> Determine a distance θ, such that if such that if $\exists$ **p** such that $| \mathbf{p} - \mathbf{s} | > \theta$, then information contained in **p** is dissimilar enough to **s** as to ignore it as a solution. That is, it is outside the realm of similarity of **s**.

Step Two: Apply the retrieval method for level k, $k \geq 0$.

Step Three: If k > θ, then set $ES_k = \varnothing$ otherwise repeat step two until done.

4.7.3. Analysis

It has been shown in section 2.3.7 that a situation can occur that leads to a solution that is not logically close enough to be satisfying as a solution. The system has the ability to define a maximal difference (θ) such that it can differentiate between values that are sufficiently close and those that are not.

In the example in section 2.3.7, the probe was a Hamming distance of four from the solution. If the maximal value θ had been set at 30%, then not more than 2 neuronal values would have been allowed to change. In this situation, there would have been no solution.

The problem of no solution can be dealt with as indicated in section 3.7.2.1 by the technique of processing tag bits. The processing tag bits are set if the system terminates for all paths.

Consider how a response might turn out to be naive. Assume a child has only learned a few groupings, such as bird, fish, dog and person. Then, if a child sees a robin, he correctly categorizes it to be a bird. An artificial neural net should work the same way. If the child sees a whale swimming in the ocean, the response would be a fish. An adult might be amused at the lack of knowledge and correct the child. After learning that the whale breaths air, has hair, hand-like skeletal structures for fins, and also has mammary glands, we classify it as a mammal. This was made possible by the additional learned memories have been added to the system. If the system has not learned very many of the whale's characteristics, it might be best not to even attempt

to classify the whale.

This type of situation brings about the introduction of a technique to respond to naive systems. A system can now be compared to other systems in regards to the overall system accuracy, where one system has not obtained a great deal of knowledge and one that has been trained extensively.

4.8. Cost of Decisions

The techniques introduced in this work have implicitely assumed that the retrieval probability for all learned memories is equal. This may not be true for many applications.

Neural nets, when used as an associative memory or as a pattern-recognition machine, may be viewed as a process for deciding between two hypotheses. For example, a neural net that stores images may be used to determine whether or not an image that is presented to it, is in its memory. The two hypotheses that are now associated with the experiment are:

(i) Image recognized

(ii) Image not recognized

The probe, if it corresponds to a stored image, may be viewed as a stored pattern that is corrupted by noise [Shan48]. Thus, we are not dealing with a unique learned pattern, but rather with a family of patterns.

Consider the example of a neural net on a ship, being used to detect submarines based on the information picked up by its hydrophone sensors. Consider further that it is designed to operate in a time of hostility. If the network fails to recognize an

enemy vessel, the ship might be lost. On the other hand, if the neural net announces an enemy vessel, when in reality there is none, the ship may take evasive maneuvers. Each of these responses, in addition to the correct responses of the neural net, has an associated cost. However, these costs are not identical. Therefore, some decisions are more costly to ignore than others.

If the *a priori* probability of the hypotheses are known and the costs of each response can be computed, then the Bayesian criterion can be used to minimize the average cost. If the cost values as well as the *a priori* probabilities of the hypotheses are unknown, then one needs to use the Neymann-Pearson criterion. The criteria may now be incorporated in the asynchronous controller.

Chapter Five

An Applications to Phonology

Do what the left half of my brain is thinking, not what the right half is saying.

Anonymous

5.1. Introduction

In the endeavor to create computers that can deal with natural language, scientists have dealt separately with the production and perception of speech. NETtalk is the neural net approach to speech developed by Sejnowski and Rosenberg [Sejn86]. It uses a supervised learning algorithm that trains on a chosen body of text. The neural network automatically embodies the phonological regularities that are used for pronunciation.

In one approach to speech perception, a neural net transforms acoustic information into letters [Koho88] and thereby obtains the words. While this approach has achieved some success, it suffers from some inherent weaknesses. It is known acoustic and phonological data do not always form a one to one transition, and, therefore a direct translation is not always possible. Furthermore, as such a system must be trained to a particular voice to be able to respond to individual styles of speech.

A neural net with an asynchronous controller creates a bicameral structure that allows conditional rules for speech perception. This application to phonology will now be described.

5.2. The Phonological Model

In phonology, we consider at least three levels of representation: (a) systematic phonemic, (b) systematic phonetic, and (c) physical phonetic [Fost85] . A fourth level between a and b, called (a′) classical phonemic, is not actually a true level of representation [Chom64], but is well known in the field.

The levels a, a′, and b are at the psychological/mental level [Fost85] and are defined in terms of distinctive features (DF's), each segment being designated by a DF matrix. These levels are related by phonological rules as stated in terms of DF's [Fost85] . Level c is defined at the physical phonetic level and is described in terms of acoustic information (perception) and articulatory movements (production). We are most concerned here with the acoustic information. It is frequently emphasized [Park77, Repp81, Fost85] that there is no isomorphic mapping (one to one) from levels a, a′, and b to speech production or to the properties of sound at level c. Instead, perception of phonological segments (mental entities) depends on acoustic cues, which may not be uniquely or invariantly associated with that segment (see [Buck87] regarding problems this non-isomorphism has caused for linguists working in other languages). Thus, there are two relevant components of phonology that must be modeled: mental/psychological (systematic and classical phonemics and systematic phonetics) and physical (physical phonetics). Relations among the psychological levels may be mapped isomorphically, but those between the physical and psychological levels cannot be.

5.2.1. Example

Consider the distinction between pairs of voiced and unvoiced stops: /b, p/, /d, t/, /g, k/ (slashes indicate that the segments are phonemes). Each voiced/unvoiced pair has the same DF matrix except for the DF [±voice]: /b/ and /p/ have the same distinctive feature matrix except that /b/ is [+voice] and /p/ is [-voice]. Likewise, /d, t/and /g, k/ are distinguished by the same feature, the first member of each pair being [+voice] and the second being [-voice]. It is important to keep in mind that these DF's are mental distinctions having no necessary connection to any aspect of the physical speech event, either the acoustic signal or the articulatory movements [Park77, Fost85] . That is, the perceptual distinction [±voice] is not tied in any necessary way to the presence or absence of vocal cord vibration.

At the systematic phonetic level, which is a second *mental* level, we can predict more precisely the phonological character of a stop segment. For instance, when /p/ occurs in initial position in a stressed syllable immediately before the vowel, it is always realized as aspirated p : [p^h]. (Square brackets indicate that the relevant level is the systematic phonetic level, or, in other words, that the segment is an allophone, or variant of the phoneme). An unaspirated *p* in that position will be judged odd by a native speaker of English. The relation between /p/ and [p^h] may be expressed by the following rule [Fost85] :

$$\begin{bmatrix} -\text{continuant} \\ -\text{voice} \end{bmatrix} \rightarrow [+\text{aspirated}] \quad /\#\_\_V'$$

(A voiceless stop is aspirated in syllable initial position before a stressed vowel). Remember that the systematic phonetic level is a mental level. [p^h], then, is cued by certain physical acoustic signals, for instance, a delay in voice onset time (VOT), the

start of vocal cord vibration. However, delayed VOT is not invariably associated with [p^h]. While the initial *p* in *poker* is [p^h] at the systematic phonetic level, the initial *p* in *potato* is not aspirated, in spite of the fact that both words have a delay in VOT. *Potato* has stress on the second syllable, so it is the first *t* that fits the aspiration rule and is, consequently, aspirated. The *p* does not fit the rule and so is not aspirated. A delay in VOT, then, is not isomorphic with the feature [± aspiration].

5.3. The Scope of the Problem

If the Hopfield model is used, the information must be placed in a vector and the system must learn that vector. This could be done in three ways. If all the information were to be placed on one vector, the system could search for the combination of levels on one vector. It has, however, already been mentioned that we cannot place all of the acoustic and phonological information in one vector, as the relationship is not one to one. One might consider placing the information on multiple vectors of a similar nature. However, the information is intrinsically linearly dependent and, therefore, is prone to create spurious states. Additionally, an overload would occur if too many information vectors are learned.

A better solution is to have the neural net learn only the acoustic cues. This is already successfully done in some models [Koho88]. The phonetic probe would then seek a solution in the neural net containing the cues. To augment the update decision structure, a set of rules may be placed in a controller to be used, if needed. The Hopfield model may be used for learning the acoustic cues, but it has no structure to contain the rule set. Without this, the system has no ability to control the variations in

the mappings from the physical to the mental. On the other hand an asynchronous control structure can accomplish this task.

To clarify, consider the implications of vowel length (or, more precisely, the duration of periodic noise associated with the perception of vowels). It is generally agreed that the [±voice] distinction in word-final stop segments can be made even when the part of the acoustic stream that corresponds to the perceived final stop is not clearly identifiable in isolation as either voiced or unvoiced [Dene55, Raph72, Wals83, Wals84, Wals87] . When ending a word, both /d/ ([+voice]) and /t/ ([-voice]) may correspond to the same physical stimuli (see [Wals87]). In such cases, whether the listener perceives /d/ or /t/ is governed by the preceding vowel. There exists a controversy whether the critical cue is vowel length (VL) or manner of vowel termination (VT; i.e., the presence or absence of a falling F1 (first formant) in the vowel [Wals84]). Walsh and Parker [Wals83] claim the following: If the preceding vowel is of less than a critical duration (CD_1), then the following stop will be perceived as [-voice]:

$$VL < CD_1 \rightarrow [-voice]$$

If the preceding vowel is greater than a critical duration (CD_2) (where $CD_2 > CD_1$) then the following stop will be perceived as [+voice]:

$$VL > CD_2 \rightarrow [+voice]$$

Between those CD's, length is not an accurate cue. Therefore, vowel length cannot be the primary cue to final consonant voicing. Walsh and Parker claim that the primary cue to [±voice] is the manner of VT and that "vowel length as cue to [±voice] in post-vocalic stops in English is (at best) redundant and (at worst) misleading" [Wals83] .

They further claim that since VT is a cue even in very long or very short vowels, it should be considered the primary cue.

Walsh and Parker [Wals87] present results which indicate that in simulated speech subjects strongly prefer a [-voice] interpretation of the final stop regardless of the rate of decline of the F1 transition when the VL is extremely short (100 ms). At the greatest VL's they test, their results do not indicate a strong preference for a [+voice] interpretation regardless of rate of decline of F1 transition. Instead, a flat F1 transition with a VL of 250 milliseconds (ms) results in near random interpretations. However, 250 ms is less than CD_2 according to the results presented by Raphael [Raph72] . His results also show a high percentage of final consonants judged [-voice] when the preceding VL is 250 ms. However, when the preceding VL is slightly more than 300 ms, judgments were consistently [+voice]. We will assume, then, that the formulation presented in the Walsh and Parker's earlier work [Wals83] is essentially correct with regard to the role of VL as a cue.

The use of a bicameral neural net relieves us of the necessity of determining which cue is primary in all cases. We can proceed on the assumption that no single cue can be considered primary in all instances. What is relevant here is that there are at least two features of the preceding vowel that can affect interpretation of the following consonant. Furthermore, they may affect interpretation differently. And, finally, each of them may be the primary cue under different circumstances. We wish to show that neural nets with bicameral natures due to the addition of asynchronous controllers can handle multiple (and possibly misleading) cues quite naturally while it would be difficult or impossible to do so using standard neural net models.

5.4. The Neural Net Model

Assume that the asynchronous control of a neural net has learned the phonological rules for our example. This means that it contains a stable points at [-voice] for a VL less than CD_1 and [+voice] for VL greater than CD_2. In addition there are other cues for [±voice], most importantly manner of VT, if the vowel length falls between CD_1 and CD_2. The physical correlates of the [+voice] or a [-voice] interpretation to which a speaker is normally exposed may be considered the *prototypical cue*. Thus each prototypical cue would create one stable point. Physical stimuli not fitting the prototype exactly must be resolved to one of the available stable points.

It is known that each of these VL's, CD_1 and CD_2, fixes the attraction basin, but if the probe is in the intermediate range then the system must not fix on the corresponding [±voice] on the basis of VL alone.

Using the asynchronous controller the system can consider particular conditions of the probe and, fix portions of the probe and still obtain a stable point. If the portion of information vector corresponding to VL is greater than the upper bound (CD_2) the system should fix the portion of the probe (and solution) to the attractor for CD_2 and [+voice]. Conversely, if the portion of the information vector corresponding to VL is less than the lower bound (CD_1) the system should fix the portion of the probe (and solution) to the attractor for CD_1 and [-voice]. However, if VL is between CD_1 and CD_2, the system should not fix any portion of the probe on the basis of vowel length information, though VL may still act as an attractor.

This can be interpreted as saying that as an attractor VL is at times a *strong*

attractor and at other times not. Depending on its value, VL can serve to direct that the portion of the probe to a particular solution. For other values, the same portion of the probe is not fixed and the probe is allowed to stabilize freely in the neural net. Without the asynchronous controller, the Hopfield model cannot handle the conditional rules necessary for dealing with the variable importance of VL as an attractor.

5.4.1. Example

Digitalization of acoustic information is commonplace. To simplify the digitalization it is assumed that the VL information is contained in the first four bits of an information vector and the VT information is contained in the last four bits. All -1's will represent a very short vowel, and +1's will represent a very long vowel. All +1's will represent a very rapid F1 decline for vowel termination, and all -1's will represent very slow F1 decline. In addition assume that CD_1 = (-1, -1, -1, +1) and CD_2 = (-1, +1, +1, +1). If VL is $\leq CD_1$ the system should stabilize at [-voice], and if VL is $\geq CD_2$, the system should stabilize at [+voice]. These will be combined to form prototypes by placing the vowel length (VL) before the vowel termination (VT) to form VL/VT.

For this example the following learned memories are prototypes.

/t/ is cued when VL/VT = (−1, −1, −1, +1, +1, −1, −1, +1)
/d/ is cued when VL/VT = (−1, +1, +1, +1, +1, +1, −1, +1)

These formulations may be considered prototypical cues: the physical correlates closest to what a particular speaker finds in speech to which he is normally exposed (or perhaps that to which he was exposed during the critical period of language

acquisition). Notice that /t/ is cued by a fairly short VL and medium VT, whereas /d/ is cued by a fairly long VL and a fairly rapid VT. The neural net model would contain the following transition matrix.

$$T = \begin{bmatrix} 0 & 0 & 0 & -2 & -2 & 0 & 2 & -2 \\ 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 \\ -2 & 0 & 0 & 0 & 2 & 0 & -2 & 2 \\ -2 & 0 & 0 & 2 & 0 & 0 & -2 & 2 \\ 0 & 2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & -2 & -2 & 0 & 0 & -2 \\ -2 & 0 & 0 & 2 & 2 & 0 & -2 & 0 \end{bmatrix}$$

Consider two different probes to observe how the system works to stabilize either one. Let a probe $p = (-1, -1, +1, +1, +1, -1, -1, -1)$. This probe has a medium VL and a slow VT. Notice that by applying $p \times T = p'$ the affective set is obtained as:

$$\begin{array}{rcccccccc} p = (& -1 & -1 & +1 & +1 & +1 & -1 & -1 & -1 &) \\ p' = (& -1 & +1 & -1 & +1 & +1 & +1 & -1 & +1 &) \\ (& - & x & x & - & - & x & - & x &) \\ (& 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 &) \end{array}$$

The x's mark where p′ differs from p and those positions mark the affective set, in this instance (2,3,6,8). Position 2 was randomly selected, thus changing the second position from -1 to +1. By applying the new probe (p′) to the transition matrix T in the same manner ($p' \times T = p''$) the choices of 2,3 and 6 were obtained. Using position 6, the system updated to the stable point, which cues a /d/. The probe could have updated to cue a /t/ if the choice of neurons for each level had been 6-2-8 respectively. In other words, in this case, the stimulus did not definitively favor either a /d/ or a /t/

interpretation. This is similar to the result in the phonological experiments, as neither VT or VL gave determinant clues.

Now, consider the probe $p = (-1, -1, -1, -1, +1, +1, -1, -1)$. Notice that the VL is very short and that the system therefore should stabilize at a /t/ interpretation. By multiplying the probe with the transition matrix, the affective set obtained is (1,2,3,4,6,7,8). Unfortunately, if we update randomly (as the Hopfield model requires us to do) the very short VL is easily lost. For example, updating position 2 we obtain a choice from the next pass of (1,3,4,7,8). The system will stabilize at /d/ if in the following passes we chose 3,4 and 8 respectively.

Using the rules in the controller, positions 1-4 should only be updated as a last resort because the controllers note that VL < CD_1. Therefore, the set of choices is narrowed to (6,7,8). Using the Oracle method the system has determined that 6 and 8 have a very high probability of leading to a learned memory (1.0 in this case). Randomly choosing from these two, the system updates 8 (6 works equally well). The new probe $p' = (-1, -1, -1, -1, +1, +1, -1, +1)$ gives a choice of position 4 to update. By updating the fourth position, the system stabilizes, and additionally it stabilizes at the desired place $p'' = (-1, -1, -1, +1, +1, +1, -1, +1)$, which corresponds with the prototype cue for /t/. This is necessary to meet with the results of the phonological tests.

5.5. Conclusions

The use of a bicameral neural net model allows us to handle the essential non-isomorphism between the physical acoustic signal and the phonological segments per-

ceived by humans. The asynchronous model can handle not only multiple cues and contradictory ones, but also its asynchronous controller allows the use of conditional rules. This latter capability allows us to treat either VL or manner of VT as the primary cue to the [±voice] distinction in word-final stops depending on the circumstances. The ability to make use of various bits of acoustic information for a single interpretation, to stabilize to an interpretation regardless of contradictory cues, and to vary the importance of cues, makes these bicameral neural nets valuable for the solution to problems in phonology caused by the non-isomorphic relationship between the acoustic signal and the perceived phonological segments. These abilities also suggest directions to proceed in the pursuit of speech-recognition machines.

Chapter Six

Conclusions

In the beginner's mind there are many possibilities, but in the expert's there are a few.

Shunryu Suzuki 1905-1971

6.1. A Neural Network with Asynchronous Control

This dissertation presents a new approach to improve the performance of neural networks used as associative memory. This research was motivated by the fact that the cortex has hemispheric asymmetry, and the two halves of the brain appear to specialize in sequential and parallel tasks, respectively. The asynchronous controller proposed in the dissertation performs global or spatial analysis similar to the spatial analysis by the right hemisphere. The introduction of the controller in a neural network defines a powerful bicameral framework. The development of this controller is the most important contribution of this dissertation.

Several algorithms for running the asynchronous controller are presented. They include: naive Solutions, hidden bit method, the oracle method, the method of handles, fixed point method, and the method of acquired information.

Naive Solutions is a method that disallows a neural network to stabilize at a solution that is too dissimilar to the probe. The method is based on defining a realm of similarity for a learned memory.

The Hidden Bit Method is a technique that alters the learned memories slightly to improve the overall retrieval. This technique eliminates the possibility of the sys-

tem stabilizing at a point that represents the complement of a learned memory.

The Oracle Method uses a probabilistic approach to obtaining learned memories as stable points. This method analyses the learned memories and searches for statistical or structural markers in the data that may be exploited for the purpose of retrieval. The approach centers on particular probabilities of neurons that are obtained as update choices.

The Method of Handles allows the system to add orthogonal pieces of data during the learning process. The addition of this data improves the performance of obtaining those particular elements in the retrieval process. This method is particularly good at identifying and eliminating spurious states.

The Fixed Point Method exploits known entities in the data. It uses information about the data itself that is acquired from analysis of the learned memories such as clustering.

The Method of Acquired Information analyses the probes as they are updated in the system, searching each one for possible clues to assist in the update process.

The new bicameral model can solve many problems that previously could not be solved using conventional neural nets. These include problems that are based on conditional rules, as in phonology.

6.2. Directions for Future Research

This dissertation should be considered as a first step in the development of truly bicameral neural structures. Before we might expect to approach an understanding of the capabilities of the bicameral brain and be able to design more powerful bicameral

computers, further work is needed in several directions. Some of them are listed below:

i. Neural networks with specialized internal structure such as limited diameter and neighbor correlation,

ii. Neural networks with nonclassical synaptic transmission,

iii. Neural networks with hierarchical control,

v. Neural network analysis using indexed information, and

vi. Topological problems of neural networks with fixed elements.

Specialized Structures

Biological neurons are much more complex than the current models in neural computing. Furthermore biological neural assemblies have considerable structure as well as bounded connectivity. A controller may help to implement some of these conditions of limited diameter and neighbor correlation that appear to be a part of animal neural nets.

For example, the mammalian neocortex consists of two sheets of neurons, each sheet consisting of four to six layers. The neuronal cells are stratified horizontally and dendrite and axonal branchings are stratified vertically. It appears that such a grouping is a significant component of the neural net structure. The structure of the neural net may well affect the utility for certain circumstances. Hubel and Wiesel [Hube62, Hube74] show that for the primary visual cortex of cats and monkeys, the cortical columns are highly correlated to function.

This proximity of similarly oriented neurons points to some control structure that directs the learning process. The learning process developing in specific regions must again be controlled in some manner for retrieval.

Non-classical Synaptic Transmission

The synaptic transmission through peptides [Cric86] is of a nonclassical nature as peptides modulate the synaptic functions rather than activate them. The effect of peptides may be sustained for seconds or even minutes and their effect may be caused at points far removed from where they were created.

Nonclassical synaptic transmissions may lead to a model that more closely resembles the apparently chaotic workings of the human brain. It has been argued [Kak88] that the mind appears to go through chaotic states and that these states can be simulated with nonclassical synaptic functions. It seems evident that any system that truly resembles the mind will have some chaotic elements to its functioning and therefore will contain nonlinear synaptic transmissions.

A controller could both implement and control the amount of chaos in a neuronal system. A controlled chaotic behavior would define a random search that might improve recall.

Hierarchical Control

Since control adds significant improvement to the processing of information, then one may expect that additional controls would improve this processing even further. The utilization of conditional rules in large scale systems may well be feasible only with hierarchical control. To accomplish this one would determine a realm

of application for a rule set using some correlation method. Then a conditional rule set for that subsystem might be applied.

Indexing

A scheme for indexing information in neural networks has been introduced in this work. The ability to index learned memories in a neural net may lead to the ability to test autonomous systems. The question of what an autonomous system has learned and in what order could be answered if indexing were available. The use of indexing for different applications needs to be investigated. This would require the framework for the design of neural network data bases that deal with indexed information.

Topology

The topology of attraction basin of memories that have fixed elements needs to be considered in greater detail. It may be possible to develop a calculus that considers the movement of probes throughout the network, under the constraints of fixed elements. This particular area needs to be examined because fixed elements are not an inert component of the system, yet still influence the flow and direction of probes. Therefore, the interplay of components could potentially cause conflicts in the system that are not readily apparent.

In addition to these areas, the bicameral model of the dissertation will apply to such problems that involve a conditional rule base. This suggests that the model would be useful in the design of speech recognition systems.

Bibliography

[Abu-85] Abu-Mostafa, Yaser S. and St. Jacques, Jeannie-Marie, "Information Capacity of the Hopfield Model," *IEEE Transactions on Information Theory*, vol. IT-31, no. 4, July 1985.

[Alle87] Allen, J., *Natural Language Understanding*, Benjamin/Cummings Publishing Co.,Inc., 1987.

[Amit87] Amit, D.J., Gutfreund, H., and Sompolinsky, H., "Information storage in neural networks with low levels of activity," *Physical Review A*, vol. 35, no. 5, pp. 2293-2303, March 1987.

[Ande88] Anderson, J.A. and Rodenfeld, E., *Neurocomputing*, The MIT Press, Cambridge, Mass, 1988.

[Buck87] Buckingham, H.W. and Yule, G., "Phonemic false evaluation: theoretical and clinical aspects," *Clinical Linguistics and Phonetics*, vol. 1, pp. 113-125, 1987.

[Char85] Charniak, E. and McDermott, D., *Introduction to Artificial Intelligence*, Addison Wesley Publishing Co., 1985.

[Chom64] Chomsky, N, *Current issues in linguistic theory*, Mouton, The Hague, 1964.

[Cric86] Crick, F. and Asanuma, C., "Certain Aspects of the Anatomy and Physiology of the Cerebral Cortex," in *Parallel Distributed Processing Volume 2: psychological and Biological Models* , ed. McClelland, J.L., pp. 333-371, The MIT Press, 1986.

[Dene55] Denes, P., "Effect of duration on the perception of voicing," *Journal of the Acoustical Society of America*, vol. 27, pp. 761-764, 1955.

[Farh85] Farhat, N.H., Psaltis, D., Prata, A., and Paek, E., "Optical implementation of the Hopfield model," *Applied Optics*, vol. 24, pp. 1469-1475, 1985.

[Feld82] Feldman, J.A. and Ballard, D.H., "Connectionist Models and Their Properties," *Cognitive Science*, vol. 6, pp. 205-254, 1982.

[Fost85] Foster, D., Riley, K., and Parker, F., "Some Problems in the Clinical Application of Phonological Theory," *Journal of Speech and Hearing Disorders*, pp. 294-297, August 1985.

[Fuku82] Fukushima, K., Miyake, S., and Ito, T., "Neocognition: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Transactions on SMC*, vol. 13, no. 5, pp. 826-834, 1982.

[Golg83] Golgi, C., *Il Sistema Nervosa Centrale*, Villardi, Milano, 1883.

[Gros83] Grossberg, S., "Some Nonlinear Networks Capable of Learning a Spatial Pattern of arbitrary Complexity," *Proceedings of the National Academy of Sciences*, vol. 9, pp. 368-826, 1983.

[Gros86] Grossberg, S., *The Adaptive Brain I:Cognition, Learning, Reinforcement, and Rhythm, and The Adaptive Brain II:Vision, Speech, Language and Motor Control*, Elsevier/North-Holland, Amsterdam Holland, 1986.

[Hamm82] Hamming, R.W., *Coding and Information Theory*, Prentice-Hall Inc., Englewodd Cliffs, New Jersey, 1982.

[Hebb49] Hebb, D.O., *The Organization of Behavior*, Wiley, New York, New York, 1949.

[Hech86] Hecht-Neilson, *American Institute of Physics*, 1986.

[Hint81] Hinton, G.E. and Anderson, J.A. eds., *Parallel Models of Associative Memory*, Erlbaum, Hillsdale, NJ, 1981.

[Hopf82] Hopfield, J.J., "Neural networks and physical systems with emergent collective computational abilities," *Proceeding of the National Academy of Science, USA*, vol. 79, pp. 2554-2558, 1982.

[Hopf84] Hopfield,J.J., "Neurons with Graded pesponse have Collective Computational Properties Like Those of Two-State Neurons," *Proceedings of National Academy of Science, USA*, vol. 81, pp. 3088-3092, May 1984.

[Hopf85] Hopfield, J.J. and Tank, D.W., "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, vol. 52, pp. 141-152, 1985.

[Hopf86a] Hopfield, J.J. and Tank, D.W., "Simple "Neural" Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit," *IEEE Trans. on Ckts. and Sys.*, vol. CAS-33, no. 5, May 1986.

[Hopf86b] Hopfield, J.J. and Tank, D.W., "Computing with Neural Circuits: A Model," *Science*, vol. 233, pp. 625-633, August 1986.

[Hube62] Hubel, D.N. and Wiesel, T.N., "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *Journal of Physiology*, vol. 160, pp. 106-154, 1962.

[Hube74] Hubel, D.N. and Wiesel, T.N., "Sequence regularity and geometry of orientation column in the monkey striate cortex," *Journal of Comparative Neurology*, vol. 158, pp. 267-294, 1974.

[Kak88] Kak, S.C., "Chaotic States of Mind - Neural Networks with Nonclassical Synaptic Function," *Unpublished manuscript*, April 1988.

[Kant87] Kanter, I and Sompolinsky, "Associative recall of memory without errors," *Physical Review A*, vol. 35, no. 1, pp. 380-392, January 1987.

[Kirk] Kirkpatrick, C.D. Jr. and Vecchi, M.P., "Optimization by simulated annealing," *Science*, vol. 220, pp. 671-680.

[Koho84a] Kohonen, T., Masisara, K., and Saramaki, T., "Phonotopic Maps-Insightful Representaion of Phonological Features for Speech Representaion," *Proceedings of IEEE 7th International Conference on Pattern Recognition*, Montreal, Canada, 1984.

[Koho84b] Kohonen, T., *Self-organization and Associative Memory*, Springer-Verlag, Berlin, 1984.

[Koho88] Kohonen, T., "The "Neural" Phonetic Typewriter," *Computer*, vol. 21, no. 3, pp. 11-24, March 1988.

[Kosk84] Kosko, B., *Bidirectional Associative Memories*, Springer Verlag, New York, New York, 1984.

[Lipp87] Lippmann, R.P., "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4-22, April 1987.

[Marr82] Marr, D., *Vision*, pp. 19-38, 54-61, San Francisco, 1982.

[McCl81] McClelland, J.L. and Rumelhart, D.E., "An interactive activation model of context effects in letter perception: Part 1. An account of basic findings.," *Psychological Review*, vol. 88, pp. 375-407, 1981.

[McCu43] McCulloch, W.S. and Pitts, W., "A Logical Calculus of the Ideas Imminant in Nervous Activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943.

[McEl87] McEliece, R.J., Posner, E.C., Rodemich, E.R., and Venkatesh, S.S., "The Capacity of the Hopfield Associative Memory," *IEEE Transactions on Information Theory*, vol. IT-33, no. 4, July 1987.

[Mead87] Mead, C., *Analog VLSI and Neural Systems*, Addison-Wesley, Reading, Mass., 1987.

[Mins75] Minsky, M., *A Framework for Representing Knowledge in The Psychology of Computer Vision P.H.Winston (Ed.)*, pp. 211- 277, McGraw-Hill, New York, New York, 1975.

[Norm76] Norman, D.A. and Bobrow, D.G., *On the Role of Active Memory Processes in Perception in The Structure of Human Memory C.N. Cofer (Ed.)*, pp. 114-132, Freeman, San Francisco, 1976.

[Park77] Parker, F., "Distinctive features and acoustic cues," *Journal of the Acoustical Society of America*, vol. 62, pp. 1051-1054, 1977.

[Pete76] Peters, A., Palay, S.L., and Webster, H., *The fine structure of the nervous system*, W. B. Saunders, Philadelphia, 1976.

[Posc68] Posch, T.E., *Models of the Generation and processing of Signals by Nerve Cells:A Categorically Indexed Abridged Bibliography*, USCEE Report 290, August 1968.

[Psal85] Psaltis, D. and Farhat, N., "Optical Information processing Based on an Associative Memory Model of Neural Nets with Thresholding and Feedback," *Optics*, 1985.

[Ramo28] Ramon y Cajal, S.R., "Degeneration and Regeneration of the Nervous System (R.M. May,Trans.)," *Oxford University Press*, Oxford, 1928.

[Raph72] Raphael, L.J., "Preceding vowel duration as a cue to the perception of voicing in word-final consonants in American English," *Journal of the Acoustical Society of America* , vol. 51, pp. 1296-1303, 1972.

[Repp81] Repp, B.H., "On levels of description in speech research," *Journal of the Acoustical Society of America*, vol. 69, pp. 1462-1464, 1981.

[Rose59] Rosenblatt, R., *Principles of Neurodynamics*, Spartan books, New York, New York, 1959.

[Rume75] Rumelhart, D.E., *Notes on a Schemata for Stories in Representation and Understanding D.A. Brobow and A. Collins (Eds.)*, pp. 211-236, Academic Press, NewYork, New York, 1975.

[Rume82] Rumelhart, D.E. and McClelland, J.L., "An interactive activation model of context effects in letter perception: Part 2. The contextual enhancement effect and some tests and extensions of the model," *Psychological Review*, vol. 89, pp. 60-94, 1982.

[Rume86] Rumelhart, D.E., McClelland, J.L., and PDP Research Group, *Parallel Distributed Processing Explorations in the Microstructure of Cognition*, 1 Foundation, MIT Press, Cambridge Massachusetts, 1986.

[Scha76] Schank, R.C., *The Role of Memory in Language Processing in Perception in The Structure of Human Memory C.N. Cofer (Ed.)*, pp. 187-247, Freeman, San Francisco, 1976.

[Sejn86] Sejnowski, T. and Rosenberg, C.R., "NETtalk: A Parallel Network That Learns to Read Aloud," *Johns Hopkins University Technical Report JHU/EECS-86/01*, 1986.

[Shan48] Shannon, C.E., "A Mathematical Theory and Communication," *Bell system Technical Journal*, vol. 27, 1948.

[Tana80] Tanaka, F and Edwards, S.F., "Analytical theory of the ground state properties of a spin glass: I. Ising spin glass," *Phys. F. Metal Phys*, vol. 10, pp. 2769-2778, 1980.

[Thak85] Thakoor, A.P., Moopen, A., Lambe, J., and Khanna, S., "Electronic Hardware Implementation of Neural Networks," *Applied Optics*, vol. 26, no. 23, 1985.

[Tou74] Tou, J.T. and Gonzalez, R.C., *Pattern Recognition Principles*, Addison-Wesley Publishing Co., Reading, Mass., 1974.

[Wals83] Walsh, T. and Parker, F., "Vowel length and vowel transition: cues to [±voice] in post-vocalic stops," *Journal of Phonetics*, vol. 11, pp. 407-412, 1983.

[Wals84] Walsh, T. and Parker, F., "A review of the vocalic cues to [±voice] in post-vocalic stops in English," *Journal of Phonetics*, vol. 12, pp. 207-218, 1984.

[Wals87] Walsh, T., Parker, F., and Miller, C.J., "The contribution of rate of F1 decline to the perception of [+/- voice]," *Journal of Phonetics*, vol. 15, pp. 101-103, 1987.

[Widr60] Widrow, B. and Hoff, M.E., "Adaptive Switching Circuits," *1960 IRE WESCON Conv. Record*, vol. 4, pp. 96-104, August 1960.

VITA

Michael C. Stinson started at Louisiana State University in August 1984. His research is neural networks augmented with control structures. His other research interests include programming languages, data security, and parallel processing. He is an IEEE Expert on the Pascal language and served on the IEEE/ANSI Joint Pascal Standards Committee from 1984-1988.

Before enrolling at L.S.U. he served as an assistant professor of computer science at Radford University in Radford, Virginia. He earned a Master of Arts in Mathematics from Michigan State University in 1977 and a Master of Science in Probability and Statistics from Michigan State University in 1975. His Bachelor of Science in Mathematics was earned from Central Michigan University in 1971.

After graduation he will be joining the faculty in the Department of Computer Science at Central Michigan University, Mt. Pleasant, Michigan.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Michael C. Stinson

Major Field: Computer Science

Title of Dissertation: Neural Networks with Asynchronous Control

Approved:

Subhash Kak
Major Professor and Chairman

F. Glen Van Buy
Dean of the Graduate School

EXAMINING COMMITTEE:

Larry Hinkle

David A. Gustafson

Harris L. Carver

S. Sitharama Iyengar

K. Tang

Date of Examination:

July 6, 1988