Development of an Analog Neural Network Model of Computation

James R. Anderson

TR-87-15

May 1987

DEVELOPMENT OF AN ANALOG NEURAL NETWORK MODEL OF COMPUTATION

James R. Anderson

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

TR-87-15 May 1987

Abstract

Neural networks have been proposed as computational models for difficult problems such as content addressable memory, constraint satisfaction, and optimization. Two types of models have emerged: 1) networks of two-state linear threshold units which are analyzed using methods of statistical physics; and 2) networks of analog devices which are described by a system of coupled non-linear differential equations. A mean field relationship between these two models has been alluded to in the literature, but an explicit treatment of such has not appeared.

This thesis establishes a computational correspondence between these two neural network models. A Markov process analysis of networks of two-state neurons leads to a set of non-linear differential equations similar to those describing analog neural networks. The derived equations are shown to yeild the same steady-state solution as those of the analog neural network models. A portion of the thesis is dedicated to providing an explanation by analogy of the computation performed by neural networks. This analogy is useful in understanding the computational correspondence between the two neural network models.

The development of the analog neural network model as provided in this thesis is important in that it provides a general mechanism for the analysis of networks of two-state neurons. This mechanism provides the basis of the computational correspondence between the two neural network models.

DEVELOPMENT OF AN ANALOG NEURAL NETWORK MODEL OF COMPUTATION

BY

JAMES R. ANDERSON, B.S.E.E.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

May, 1987

In Memory

of my father, Frederick Douglas,
whose physical presence I long for;
and of my son, Richard Alan,
who never arrived to this world,
yet was a part of it just the same.

ACKNOWLEDGEMENTS

Foremost I wish to gratefully acknowledge the support of the Office of Naval Research Graduate Fellowship Program, without which this thesis would never have been written. Only through such a generous program could I have followed the path I have chosen.

To my readers, Dr. Chuan-Lin Wu and Dr. James C. Browne, who made immeasurable accommodations to assist in the completion of my degree, I express heartfelt thanks.

Finally, to Mary and Doug, for lovingly accepting that I am also a student, you are that part of my life that I could never want to change.

... lest I simply go, go, go, do, do, do,
eat, eat, eat, sleep, sleep, sleep,
laugh, laugh, laugh, cry, cry, cry,
go, go, go, ...

Fr. Jacques Weber

James R. Anderson

The University of Texas at Austin
May, 1987

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

Neural networks have been proposed as *computational models* for 'hard problems' that the brain appears to solve easily [20], such as content addressable memory (CAM) [8], constraint satisfaction [7], and optimization [10, 22]. Out of recent work, two types of computational models have emerged:

- Networks of two-state neurons which are analyzed using the methods of statistical physics [8, 7]; and
- Networks of analog devices which are described by systems of non-linear differential equations [9].

A correspondence between the computation results provided by these two models - essentially a result of mean field theory from statistical physics [21] - has been stated or alluded to in the literature [10, 18]. However, there does not not appear to have been an explicit treatment of this correspondence in the literature.

This thesis will establish this correspondence in a general form through a Markov process analysis of the networks of two-state neurons. Analysis of the probabilistic behavior of the two-state neurons in the statistical model leads to a set of deterministic, non-linear differential equations that approximate the neuron state probabilities. These

equations are shown to model a network of non-linear amplifiers similar to the *analog neural networks* presented by Hopfield [9, 10], and they are shown to have the same steady-state solution as Hopfield's equations. Other Markov analyses of networks of two-state neurons have overlooked the correspondence to analog neural network models [18].

Previous identification of the correspondence between the two network models only established that the solution points of the two networks had a one-to-one correspondence[1] [9]. The development presented in this thesis indicates that the solution provided by the analog network is a mean field approximation of the *expected state* of the network of two-state neurons. This provides the basis for interpreting the computation performed by the analog neural network.

1.1 Background

With the development of electronic computing devices, researchers have had a heightened interest in how the human brain performs computation. In particular, since the emergence of digital computers, scientists have searched for corresponding digital computation models for the brain. One of the earliest investigations into *neural network* models was the suggestion by McCulloch and Pitts, in 1943, that neuron activity corresponded to the evaluation of logical propositions [16]. Thus, their investigations were centered around the

---

[1]Hopfield [9] indentified the property of solution point averaging that is introduced by a non-zero *temperature* in the statistical model. In this case, there is a many-to-few correspondence of solution points.

development of a formal logical calculus to describe the operation of networks of a formal two-state neuron model, the linear threshold unit (LTU), shown in Figure 1-1.

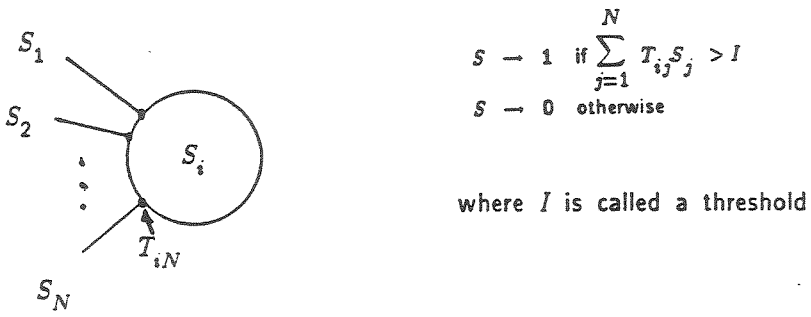


Figure 1-1: The linear threshold unit (LTU) model of a neuron.

The use of a two-state model for neurons culminated in the early 1960's with Rosenblatt's perceptrons [19]. These were a series of machines formulated to perform classification and recognition tasks. The basic perceptron consists of three layers of LTU networks, each layer making connections to the next. At each time step, the neurons in each layer are updated according to inputs received from the previous layer. The connection weights between the neurons in two different layers are selected such that patterns presented at the 'input' layer are classified into specific patterns at the 'output' layer. Rosenblatt presented a training algorithm and a convergence theorem that guaranteed that certain perceptron models could be trained to perform classification tasks. However, an investigation of perceptron properties by Minsky and Papert in 1969 revealed serious limitations [17], and models of this kind soon fell out of interest.

The shortcomings of perceptrons lay essentially in complexity of

analysis. It was shown that *feedback* connections between layers was necessary to perform more complicated classification tasks. But this led to models that were difficult to analyze, and there was no longer a convergence theorem for training the networks.

Recently, Hopfield introduced a new class of LTU networks that are fully interconnected, in which neurons (LTU's) update their state asynchronously [8]. Through an analysis based on an *energy function*, in analogy with the energy of a physical system, Hopfield was able to demonstrate that these networks display *collective computation* properties. The concept of *thermal noise* has been added to these networks, producing stochastic networks of probabilistic LTU's [13]. Using methods from statistical physics, global properties of these networks can be derived from knowledge of only the local neuron behavior [7].

Very few, if any, biologists believe that biological neurons behave as two-state devices. In response to this observation, Hopfield introduced a network model based on analog devices which are thought to provide a more accurate model of biological neurons [9]. While this type of model is not unique to Hopfield[2], they have received much attention because Hopfield identified a correspondence between properties of these networks and the collective properties of his earlier networks. This correspondence provides an insight into the computation performed by the analog networks, something that was not apparent in previous work.

---

[2]The most notable examples of analog network models are *competitive learning* models; for example, von der Malsberg [23] and Grossberg [5].

This thesis develops a computational model for analog neural networks. Derivation of the model from the analysis of LTU networks provides a basis for the mechanisms of the correspondence between the two neural network models.

1.2 Reader's Guide

Chapter 2 provides the operational details of Hopfield's LTU networks. These are extended to consider the stochastic networks produced by application of *simulated annealing* [13] to Hopfield networks, such as in the Boltzmann Machine [7]. Effort is given to describe by analogy the computation being performed by these models. This discussion provides insight into the underlying mechanisms of computation with neural networks. This insight is particularly useful in understanding the computation performed by the analog neural networks defined in Chapter 3.

Chapter 3 presents a Markov process analysis of Hopfield and stochastic two-state neuron networks. The results of this analysis are used to define a probability measure on the state of each neuron in the network. This measure is used to define an analog neural network similar to Hopfield's analog networks [9]. Both forms of analog networks are shown to yield the same steady-state solution.

Chapter 4 contains a discussion of the results of Chapter 3. The computation model developed in Chapter 3 is interpreted in terms of the computation performed by networks of two-state neurons. Areas for further investigation and extension are identified.

A very active area of investigation with neural networks is concerned with their ability to learn [7, 20, 1]. These aspects will not be covered here.

Chapter 2

Networks of Two-State Neurons

The networks described in this chapter are based on the use of a linear threshold unit (LTU), which was presented in Figure 1.1, as a rough approximation of biological neurons. Under this approximation, the state of a neuron is interpreted as either 'firing' or 'not firing'. The use of such an approximation inherently implies an interpretation of a neural assembly, such as the brain, as a digital machine. This particular interpretation is convenient as it leads to network models that resemble magnetic-spin (spin-glass) models from statistical physics [4]. This similarity allows the use of results from statistical physics for the analysis of the computation performed by such neural networks.

This chapter presents the details of Hopfield's LTU network [8] and an interpretation by analogy of the computation performed by this model. The model is then extended to stochastic networks of probabilistic LTU's. These networks are the basis for optimization and constraint satisfaction applications of neural networks [10, 7].

2.1 Computation with Hopfield Networks

Hopfield showed that a completely connected network of LTU's can perform *collective computation* through the minimization of a global measure on the state of the network [8]. Consider a network of N LTU neurons. Each neuron defines a coordinate s_i of the system. The value of each of the coordinates $s_1, s_2, \cdots, s_N$ is defined by the state of the corresponding neuron. We can now define a *state vector* for the network

$$S = [s_1, s_2, \cdots, s_N] \tag{2.1}$$

where

$$s_i = \begin{cases} 1 & \text{if } i^{\text{th}} \text{ neuron is 'firing'} \\ 0 & \text{otherwise} \end{cases}, \quad i=1,2,\ldots,N \ .$$

The collection of all possible values for the state vector S defines the state space of the network. Thus the instantaneous configuration of the 'neuron firing' pattern represents a point in the state space of the system.

Hopfield was interested in the use of neural networks as a content-addressable memory (CAM). In such a memory system, it is desired to retrieve a stored item that is closest in some sense to the given input. Hopfield suggested that a system whose evolution in state space is described by a phase-flow with several distinct attractors could be used as a CAM. A two-dimensional example of a phase-flow is shown in Figure 2-1.

By starting at any point in the state space and following the

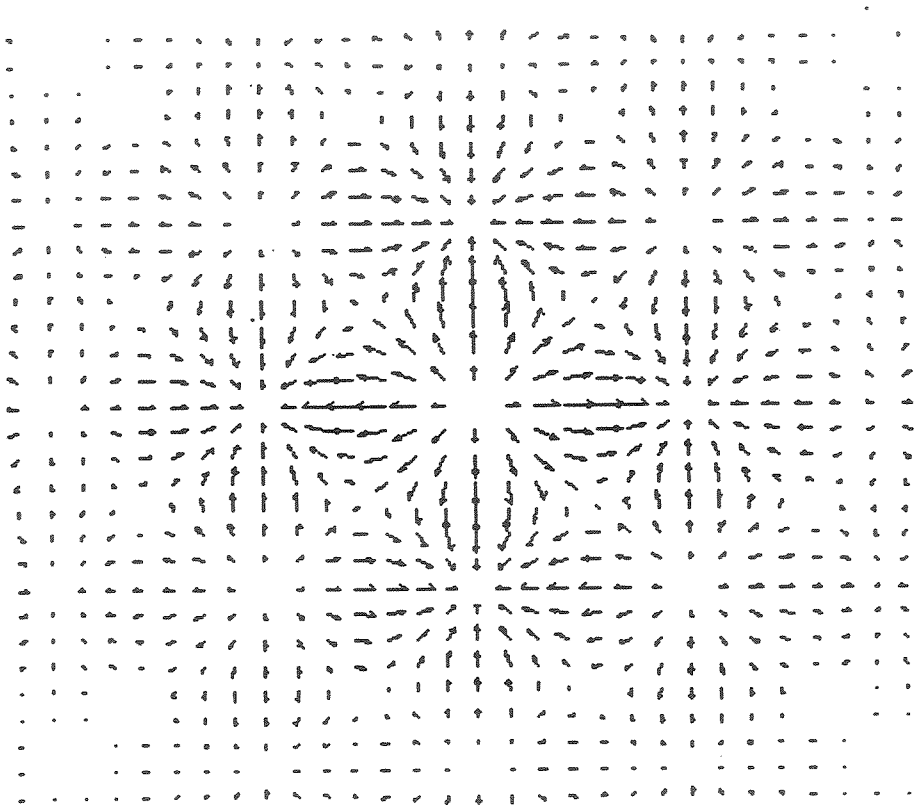


Figure 2-1: Example of a phase-flow in two dimensions.

phase-flow lines, the system reaches a stable point. The phase-flow can be considered as having been generated by a state space 'landscape' of hills and valleys, such that the system state always flows toward the landscape valleys. A two-dimensional state space landscape corresponding to the phase-flow of Figure 2-1 is shown in Figure 2-2. This landscape defines a measure on the system state, which we will call the system *energy*. Therefore, a system governed by a phase-flow with distinct attractors will evolve such that the system energy evolves to a local energy minimum.

If each of the energy minima is considered to be a stored piece of information, the information can be retrieved by starting the system

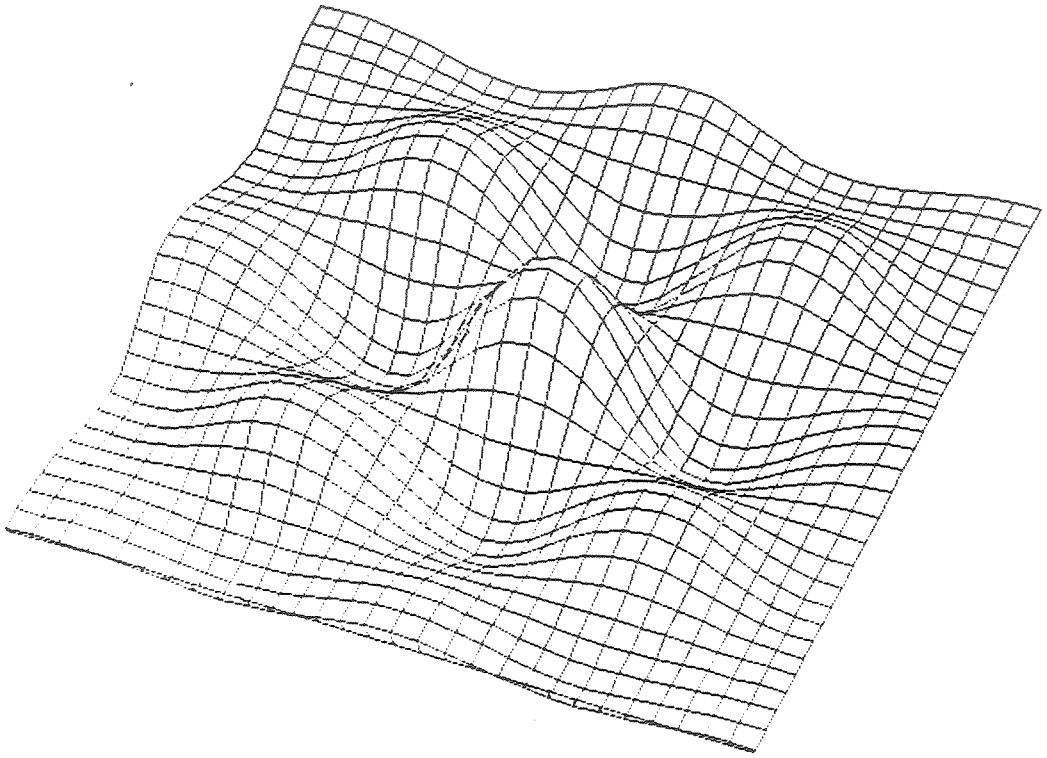


Figure 2-2: Energy landscape that produced phase-flow shown in Figure 2-1.

at some point near the minima. This is the essence of computation with LTU neural networks. Information, or knowledge, is stored in the form of minima in the network energy function. In other words, the network state defined by an energy minima corresponds to a stored item. Similarly, the initial network state corresponds to an 'input' to the network. Information is retrieved by allowing the initial network state to evolve to the nearest state corresponding to stored information. This forms the basis of the use of neural networks for CAM.

2.2 Details of the Hopfield Model

Now consider the network of N LTU neurons. Each neuron i has a connection to neuron j with a *synaptic strength* of T_{ij} , and $T_{ij}=T_{ji}$. In general, each neuron has a threshold I_i . Each neuron modifies its state according to the LTU rule

$$s_i = \begin{cases} 1 & \text{if } \sum_{j=1}^{N} T_{ij}s_j - I_i > 0 \\ 0 & \text{otherwise} \end{cases} . \tag{2.2}$$

Hopfield gives the energy function for the network as

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} T_{ij}s_i s_j + \sum_{i=1}^{N} I_i s_i .$$

The change in energy due to a change in the state of neuron k is given by the partial derivative

$$\begin{aligned} \Delta E_k = \frac{\partial E}{\partial s_k} &= -\frac{1}{2}\sum_{i=1}^{N} T_{ik}s_i\Delta s_k - \frac{1}{2}\sum_{j=1}^{N} T_{kj}s_j\Delta s_k + I_k\Delta s_k \\ &= -\Delta s_k\Big[\sum_{j=1}^{N} T_{kj}s_j - I_k\Big] \end{aligned}$$

where we utilize the fact that $T_{ij}=T_{ji}$ and Δs_k denotes the state change make by neuron k. Because we only want state changes that cause the energy to decrease, we require that $\Delta E_k \le 0$, which yields

$$\Delta s_k = \begin{cases} \ge 0 & \text{if } \sum_{j=1}^{N} T_{ij}s_j - I_i > 0 \\ \le 0 & \text{otherwise} \end{cases} .$$

This is simply the LTU update rule from equation (2.2). Thus, if neurons change their state asynchronously (one at a time) the network will evolve to an energy minimum. Typically, neurons are selected to change state randomly, although by the above result, the network will settle to a minimum for any selection scheme.

For the given energy function, the energy of a given state is a function of the T_{ij} . Therefore, information is stored in this network through proper selection of the T_{ij} . Given a number of states S^{α} to be stored, Hopfield gives the following Hebbian [6] learning rule for selection of the T_{ij}:

$$T_{ij} = \sum_{\alpha} (2s_i^{\alpha}-1)(2s_j^{\alpha}-1) \qquad \text{with} \quad T_{ii} = 0 \ .$$

This rule produces minima in the energy function for each of the stored patterns S^{α}, up to some storage capacity limit. Thus the network will perform a CAM function. There are many interesting issues related to storage and retrieval in Hopfield network [8, 18] that are beyond the scope of this analysis.

2.3 Optimization with Neural Networks

A different sort of problem from CAM's is that of optimization, or constraint satisfaction. In this problem, constraints are stored in the T_{ij} such that energy minima correspond to network configurations that satisfy the constraints. There are typically many such energy minima. The goal is to have the network find the state that optimally satisfies the constraints. This is equivalent to finding the global, rather than local, minimum of the network energy function.

In terms of the energy landscape analogy, the goal is to find the lowest valley in the state space. The problem with the Hopfield networks described in Section 2.2 is that they get stuck in local minima; they only perform gradient descent until they get to the bottom of the nearest valley. One way to try to find the global minimum would be to start the Hopfield Net at different random starting points in state space and observe which of the resulting solutions has the lowest energy. However, this does not prove to be a practical or reliable method (we may be dealing with systems that have many, many stable states).

A method of solving problems of this sort is through *simulated annealing* [13]. The name of this technique comes from an analogy with the physical process of annealing. This is a process used to form crystalline structures in solids, the crystal being a low energy state. The process consists of melting the material at a high temperature and allowing it to cool very slowly. The initially high temperature of the system has the effect of giving the molecules enough mobility (energy) so that they realign into an ordered arrangement (order having the effect of lowering the energy).

By an analogy with the energy landscape, the heat introduced by annealing gives the system enough thermal energy to occasionally make uphill jumps in the energy landscape. This would be as if the state of the system were represented by a marble rolling around the energy landscape. With annealing, the marble is occasionally given additional energy in the form of *thermal noise*, thus allowing the marble to jump over a hill into a possibly lower valley. If the size of the thermally induced jumps are slowly decreased, the marble will tend to stay in the deeper valleys. This is the essence of annealing.

From statistical physics, it is known that the result of annealing will be to put the system into states of lowest energy. The probability of being in a particular state is proportional to a function of the system energy at that state, E_i , and the system temperature, T . This probability is given by the *Boltzmann factor* [14]

$$P\{S=i\} \;\alpha\; e^{-E_i/T} \; .$$

Thus the *relative* probability of finding the system in state i with respect to the probability of finding it in state k is given by

$$\frac{P\{S=i\}}{P\{S=k\}} = \frac{e^{-E_i/T}}{e^{-E_k/T}} = e^{(-E_i+E_k)/T} = e^{-\Delta E_{ik}/T}$$

where ΔE_{ik} denotes the energy difference in going from state k to state i. This is the Boltzmann distribution, which is shown in Figure 2-3. Notice, that as T is increased, the relative probability of being in any state approaches 1 . At high temperatures, all states are equally likely, while at low temperatures, states with low energy become more likely than high energy states.

2.4 Stochastic Networks of Probabilistic LTU's

Simulated annealing has been applied to Hopfield networks, resulting in stochastic network models [18], the best known example being the Boltzmann Machine [7]. Use of simulated annealing gives rise to a probabilistic update rule for the LTU's. The form of the new update rule is [7]

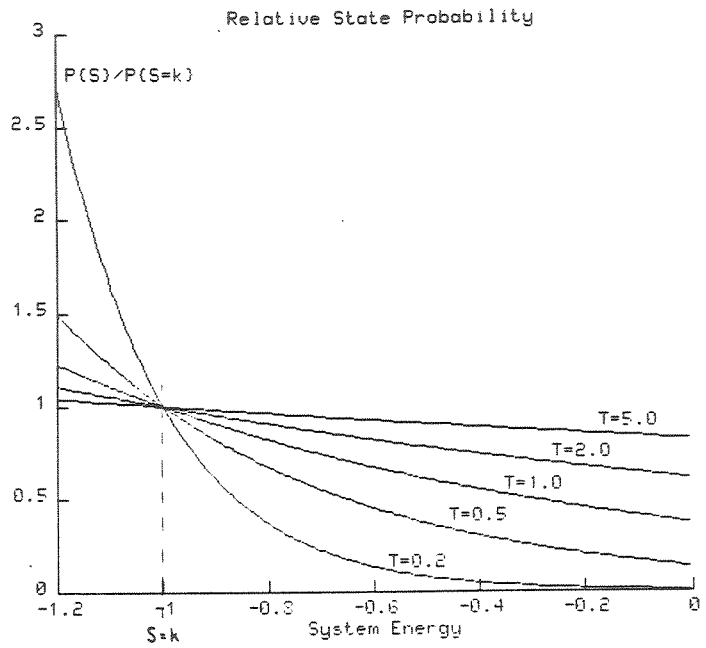


Figure 2-3: Relative state probabilities given by the Boltzmann distribution.

$$p_i = P\{s_i \rightarrow 1\} = \frac{1}{1+e^{-\Delta E_i^0/T}} \qquad (2.3)$$

$$= \frac{1}{2}\left[1+\tanh(\Delta E_i^0/2T)\right]$$

$$= G\left(\Delta E_i^0, T\right)$$

where

$$\Delta E_k^0 = E(s_k{=}0) - E(s_k{=}1) = \sum_{j=1}^{N} T_{ij} s_j - I_i \ .$$

This function is shown in Figure 2.4. Operation of the network under this rule has the effect that the final states produced by the network

will conform to the Boltzmann distribution. States with the lowest global energy will be produced with a higher probability than states with higher energies.

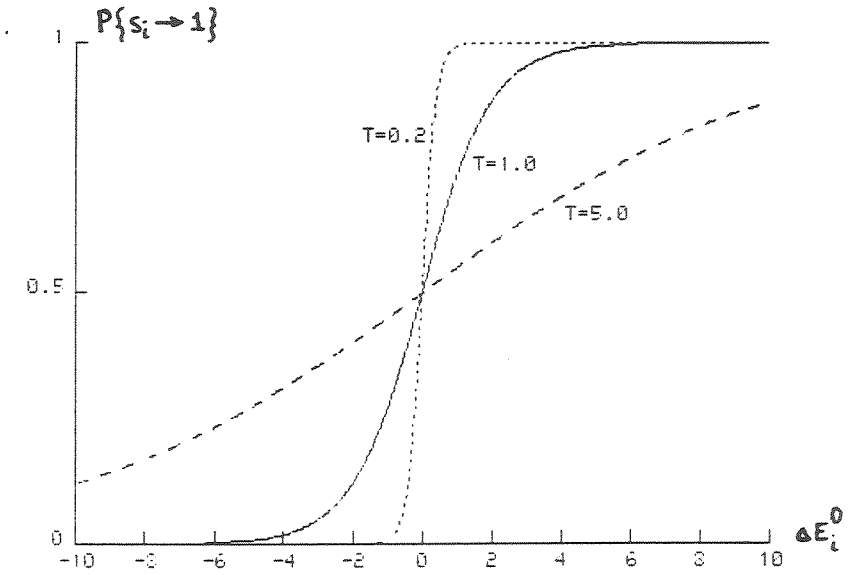


Figure 2-4: Stochastic LTU update function, $G(\Delta E_i^0, T)$.

Notice that for T=0, this update rule degenerates to the normal LTU rule for Hopfield networks. Therefore, the Hopfield network corresponds to the special case T=0 of the stochastic network, and both networks can with modeled by the stochastic network model.

Chapter 3

Derivation of Analog Neural Network Models

The neural network models of Chapter 2 have been shown to be capable of performing useful CAM and optimization computations [10, 22, 7]. However, there are two motivations for the analysis presented in this chapter:

1. Simulation of the networks described in Chapter 2 are very time consuming, particularly those of the stochastic networks; and,

2. To establish an a posteriori derivation of analog neural network models, after Hopfield's analog networks [9].

In a sense the LTU networks of Chapter 2 are suboptimal in terms of a space-time cost. By their definition, the models are multiple-instruction, multiple-data (MIMD) computations. It is required that neurons change state asynchronously, one at a time. Thus, only $\frac{1}{N}$ of the computation space is begin used. For parallel computation, it is desirable to find a single-instruction, multiple-data (SIMD) computation, such that all of the computation space is used. Furthermore, the stochastic networks require a large number of updates, due to their probabilistic nature, to assure that an equilibrium solution is acheived.

By deriving an analog neural network model, it is found that computations can be performed in only a few model time constants [10]. This does not have immediate implications for computational speed, as simulation of the network requires solving a coupled system of non-linear differential equations. Numerical techniques for this computation are still computation intensive. However, two benefits are realized:

1. An understanding of analog neural network computation in terms of the analogies presented in Chapter 2; and

2. A computational model to motivate and guide development of implementation technologies.

3.1 Markov Process Model of LTU Networks

Markov processes provide useful models for analyzing the states of stochastic systems. We defined the state $S=[s_1,s_2, \cdots ,s_N]$ of a LTU network in equation (2.1). Also recall that the Hopfield network was found to be the special case $T=0$ of the stochastic LTU network. A discrete time system can be modeled as a Markov process if the transitions from one state to the next satisfy the Markovian assumption [11]

$$P\{S(n+1)=j \mid S(n)=i,S(n-1)=k,\ldots,S(0)=m\}$$

$$= P\{S(n+1)=j \mid S(n)=i\}$$

This property states that the probability that the system will enter state j on the next transition depends *only* on the *current state*. Any information regarding the past states of the system does not affect the next state probabilities. We can immediately see that the LTU networks described in Chapter 2 satisfy this property.

For a network of N two-state neurons, there are 2^N values for the state vector S. Because the temperature T is usually varied slowly with respect to neuron updates, we can treat the neuron update probabilities $p_i(n)$ as a function only of the network state. Under these conditions, we can model the network as a time-invariant Markov process. We define the state probability vector

$$\Pi(n) = [\pi_1(n) \;\; \pi_2(n) \cdots \pi_{(2^N)}(n)] \;\; ,$$

where the state probabilities

$$\pi_\alpha(n) = P\{S(n)=\alpha\} \;\; , \;\; \alpha=1,2,\ldots,2^N$$

satisfy the condition

$$\sum_{\alpha=1}^{2^N} \pi_\alpha(n) = 1 \; .$$

Then the state probability evolution equation is given by

$$\Pi(n+1) = \Pi(n)P \;\; , \;\; n=0,1, \ldots \tag{3.1}$$

where P is the state probability transition matrix

$$P = [p_{\alpha\beta}]$$

with elements $p_{\alpha\beta}$, the state transition probabilities, given by

$$p_{\alpha\beta} = P\{S(n+1)=\beta \mid S(n)=\alpha\} \;\; ; \;\; \alpha,\beta=1,2,\ldots,2^N \; .$$

The definition of the network requires that the neurons change state one at a time. Therefore, a single step change in the *system* state, $S(n) \rightarrow S(n+1)$, is due to a change in only one *neuron* state. We can relate the *state transition* probabilities $p_{\alpha\beta}$ to the *neuron update* probabilities by

$$p_{\alpha\beta} = \begin{cases} \frac{1}{N} p_k & \text{if } \Delta s_k = 1 \\ \frac{1}{N}(1-p_k) & \text{otherwise} \end{cases}$$

where $\frac{1}{N}$ is the probability that neuron k changes state, p_k is the k^{th} neuron update function given by equation (2.3), and $\Delta s_k = \pm 1$ is the change made by neuron k in taking the system from $S=\alpha$ to $S=\beta$. If the network states $S=\alpha$ and $S=\beta$ differ in more than one neuron state, $p_{\alpha\beta} = 0$.

The quantity $\pi_i(n)$ gives us the probability that the i^{th} system state is occupied at time n. It has been shown that for stochastic LTU networks, the limiting state probabilities, $\lim_{n \to \infty} \Pi(n)$, are given by the Boltzmann distribution [18]. This is the same result as was given in the discussion on stochastic networks in section 2.3. Therefore, we can use the state probability evolution equations (3.1) to find the limiting state probabilities of the network. As $T \to 0$, we know that the network will settle into a stable state corresponding to an energy minimum. This state will be identified by the limiting state probability vector. However, we have not gained much because there are 2^N state probability equations to be solved for a network of N neurons.

Another analysis of the network can be based on modeling each neuron as a Markov process, and applying the concepts we have just discussed to these Markov processes. Now we have N two-state Markov processes, as shown in Figure 3-1. However, these processes are not time-invariant. At each time step, the neuron transition probabilities $p_i(n)$ are a function of the network energy, which is a function of the current states of the N two-state processes. The *neuron* state probabilities for the i^{th} neuron are given by

$$\Pi^i(n) = [\pi_0^i(n) \quad \pi_1^i(n)] \quad , \quad i=1,2,...,N$$

where

$$\pi_0^i(n) = P\{s_i(n)=0\}$$
$$\pi_1^i(n) = P\{s_i(n)=1\}$$

and $s_i(n)$ is the state of the i^{th} neuron at time n. The *neuron* state probability evolution equation is given by

$$\Pi^i(n+1) = \Pi^i(n)P^i(n) \quad , \quad n=0,1,2,... \quad , \quad i=1,2,...N$$

where $P^i(n)$ is the time-varying *neuron update* matrix for the i^{th} neuron

$$p^i(n) = \begin{bmatrix} p_{00}^i(n) & p_{01}^i(n) \\ p_{10}^i(n) & p_{11}^i(n) \end{bmatrix} = \begin{bmatrix} (1-p_i(n)) & p_i(n) \\ (1-p_i(n)) & p_i(n) \end{bmatrix}$$

Now we have only $2N$ state probability equations to be solved. These equations describe only the statistics of the N neuron Markov processes. But each neuron transition probability $p_i(n)$ is a function of the N neuron states $s_j(n)$. The i^{th} Markov process cannot know the current state of the other processes because only the statistics are given. So these equations cannot be solved directly. However, they can be solved after making an approximation for the neuron states $s_i(n)$, and hence the system state $S(n)$.

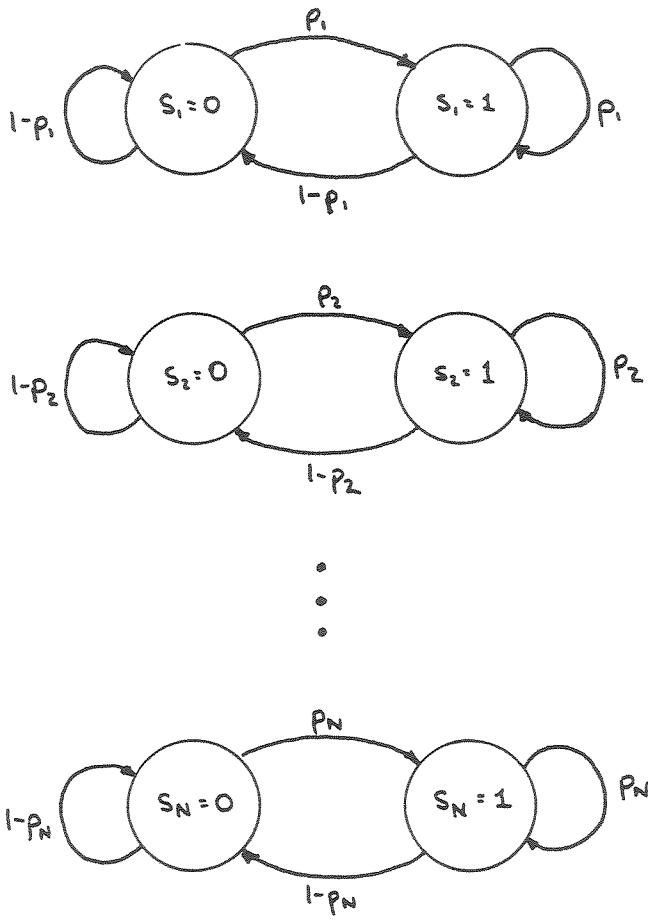


Figure 3-1: Resulting vector Markov process when each neuron is modeled as a two-state Markov process.

3.2 Solution of the State Probability Equations

Before making the neuron state approximation, it is convenient to transform the network to a continuous-time process. It would be possible to make this transformation after solving for the state probabilities with an Euler difference approximation. But for reasons of generality and convenience, we will make the transform now.

In the discrete-time LTU networks, there is no significance to a time delay between neuron updates. The only requirement is that updates be made one at a time, and that the results of this update are immediately available for the next update evaluation. Any other timing characteristic has no effect on the computation. But in order to transform to a continuous-time Markov process, an assumption must be made regarding the time between neuron updates. We require that the time between updates for a given neuron be exponentially distributed with mean λ [12]. In general, it is possible to have a different λ for the two different neuron states, but we will assume they are equal. With this assumption, the state probability equations for the continuous-time i^{th} neuron Markov process are given by [12]

$$\frac{d}{dt}\Pi^i(t) = \Pi^i(t)\,\Lambda^i\,[P^i(t)-I]$$

where

$$\Lambda^i = \begin{bmatrix} \lambda^i & 0 \\ 0 & \lambda^i \end{bmatrix}, \quad I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Now we will derive an approximate solution for the neuron state probability equations. Since we do not have information regarding the current neuron states $s_i(t)$, we will approximate the $s_i(t)$ by their expected values $<s_i(t)>$, which we will denote as $V_i(t)$:

$$V_i(t) = \langle s_i(t) \rangle = \sum_{n=0}^{1} n \pi_n^i(t) = \pi_1^i(t).$$

We now evaluate the time derivative of $V_i(t)$:

$$\frac{d}{dt} V_i(t) = \frac{d}{dt} \pi_1{}^i(t) = \pi_0^i(t)\, \lambda^i\, p_{01}^i + \pi_1^i(t)\, \lambda^i\, (p_{11}^i - 1) \ .$$

But $\pi_0^i = (1 - \pi_1^i)$, and $p_{01}^i(t) = p_{11}^i(t) = p_i(t)$, so

$$\begin{aligned} \frac{d}{dt} V_i(t) &= \lambda^i\, p_i(t) - \lambda^i\, \pi_1^i(t)\, p_i(t) + \lambda^i\, \pi_1^i(t)\, p_i(t) - \lambda^i\, \pi_1^i(t) \\ &= -\lambda^i\, \pi_1^i(t) + \lambda^i\, p_i(t) \\ &= -\lambda^i\, V_i(t) + \lambda^i\, p_i(t) \ . \end{aligned}$$

Setting $\lambda^i = \lambda$ and substituting the update function from equation (2.3) for $p_i(t)$ yields

$$\begin{aligned} \frac{d}{dt} V_i(t) &= -\lambda V_i(t) + \lambda G\, (\Delta E_i^0\, ,\, T\,) \\ &= -\lambda V_i(t) + \lambda G\, \Big(\sum_{j=1}^{N} T_{ij} s_j(t) - I_i\, ,\, T\, \Big) \end{aligned}$$

Finally, we make the approximation $s_i(t) \approx \langle s_i(t) \rangle = V_i(t)$, which gives us the following set of coupled differential equations

$$\frac{d}{dt} V_i(t) = -\lambda V_i(t) + \lambda G\Big(\sum_{j=1}^{N} T_{ij} V_j(t) - I_i\, ,\, T\, \Big) \quad , \quad i = 1,2,\ldots N \ . \tag{3.2}$$

The approximation $s_i \approx \langle s_i(t) \rangle$ is effectively a mean field method [21]. This technique from statistical physics allows replacement of variables, such as $s_i(t)$, with their expected value when taking the expected value of some other variable.

3.3 Analog Neural Network Model

The coupled set of equations (3.2) can be realized by a network of nonlinear amplifiers. The general form of the circuit for this analog neural network is shown in Figure 3-2. These should be compared with the network given by Hopfield shown in Figure 3-3.

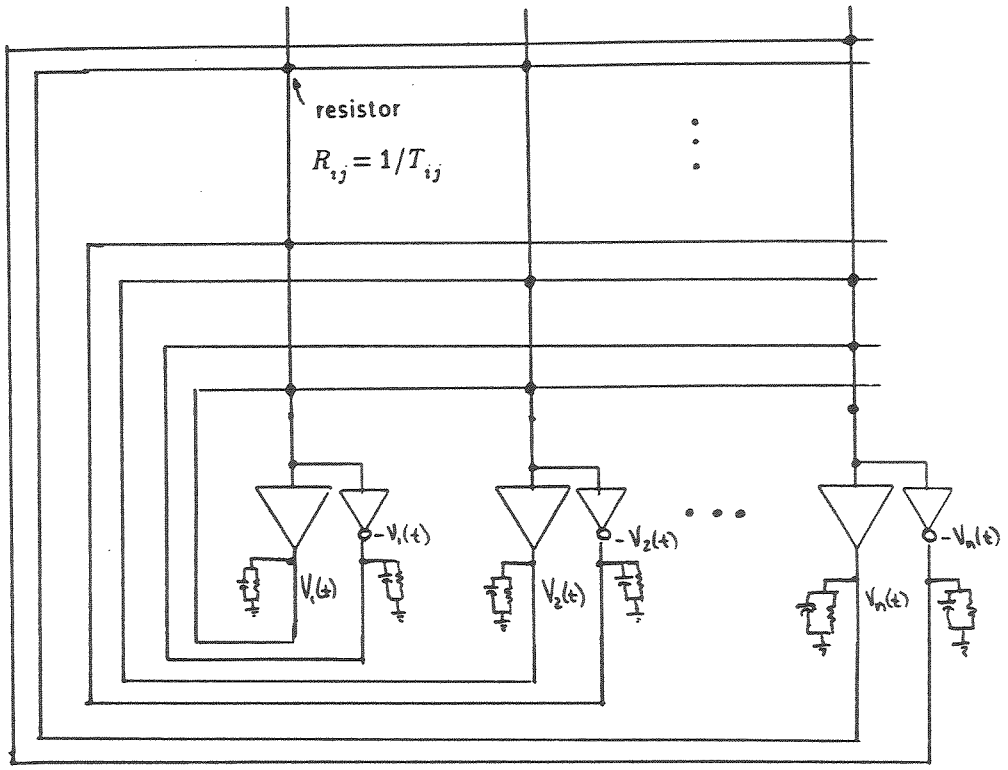
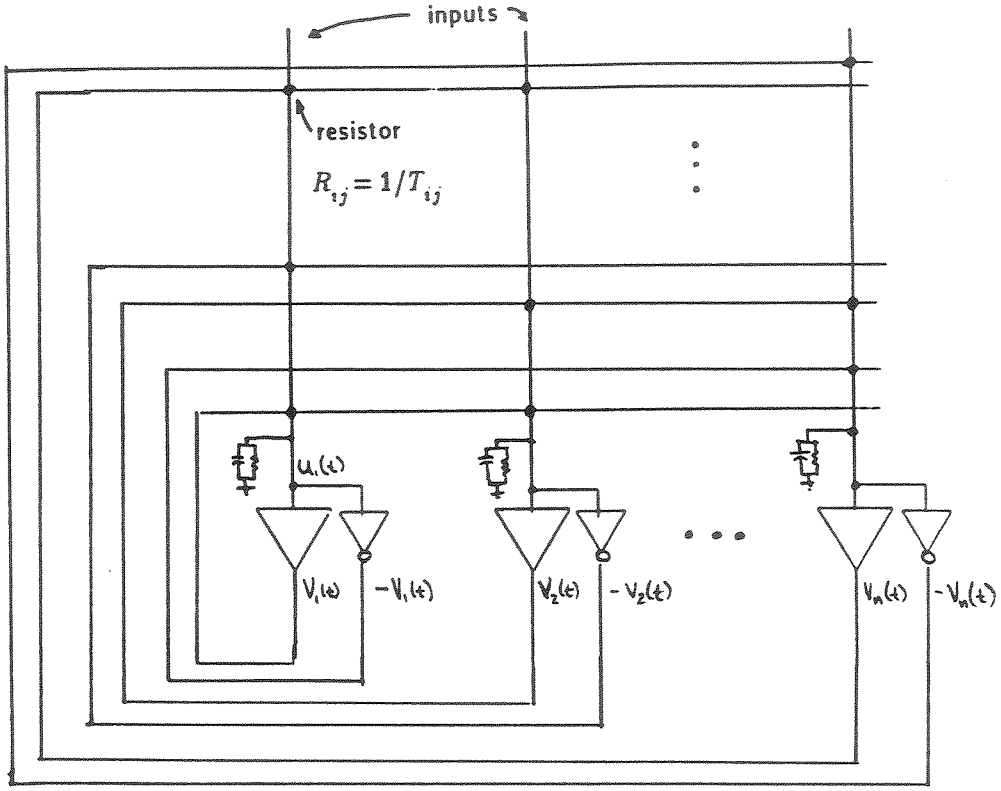


Figure 3-2: Analog neural network circuit for equations (3.2).

It should be emphasized that the solutions provided by the mean field method yield steady-state solutions. We can solve for the



The equations that model this circuit are

$$du_i(t)/dt = -u_i(t)/\tau + \sum_{j=1}^{N} T_{ij} V_j(t) + I_i$$

with $V_j(t) = G(u_j(t))$.

and τ is an RC time constant determined by the parameters of the circuit. I_i is an external input to neuron i, and the neuron has a zero threshold, Θ.

The external input has the effect of lowering the threshold of the neuron. Here we redefine the threshold to be $I' = \Theta - I_i$. We will use the following equations when referring to Hopfield's network. usually with $\tau = 1$:

$$du_i(t)/dt = -u_i(t)/\tau + \sum_{j=1}^{N} T_{ij} V_j(t) - I_i' \ .$$

Figure 3-3: Neural network circuit given by Hopfield [10]. The equations used for comparison are derived from those given by Hopfield.

steady-state solution to equations (3.2) by setting $dV_i(t)/dt = 0$, which yields, with the temperature T fixed,

$$V_i(t) = G\Big(\sum_{j=1}^{N} T_{ij}V_j(t) - I_i\Big).$$

With τ=1, Hopfield's equations [10] (see Figure 3-3) are given by

$$\frac{d}{dt}u_i(t) = -u_i(t) + \sum_{j=1}^{N} T_{ij}V_i(t) - I_i$$

where $V_i = G(u_i)$. At steady-state,

$$u_i(t) = \sum_{j=1}^{N} T_{ij}V_j(t) - I_i \ .$$

Applying $V_i = G(u_i)$ gives

$$V_i(t) = G\Big(\sum_{j=1}^{N} T_{ij}V_j(t) - I_i\Big) \ .$$

Therefore, both analog neural networks have the same steady-state solution; only the dynamics of the networks differ.

It is interesting to note that Hopfield's equations [10], as given in Figure 3-3, can be derived directly from the discrete time Markov model by solving for a different quantity:

Let

$$u_i(n) = \Delta E_i^0(n) = \sum_{j=1}^{N} T_{ij}s_j(n) - I_i \ ;$$

then, using equation (2.3)

$$V_i(n) \;=\; G(\,u_i(n)\,) \;=\; P\{s_i(n+1)=1\}\;;$$

using an Euler approximation,

$$\begin{aligned}
\frac{d}{dt}u_i(n\tau) \;&\approx\; \frac{u_i(n\tau+\tau)-u_i(n\tau)}{\tau} \\
&\approx\; \frac{1}{\tau}\Big[\sum_{j=1}^{N} T_{ij}s_j(n\tau+\tau) - I_i\,\Big] \;-\; \frac{1}{\tau}\Big[\sum_{j=1}^{N} T_{ij}s_j(n\tau) - I_i\,\Big] \\
&\approx\; -\frac{1}{\tau}u_i(n\tau) \;+\; \frac{1}{\tau}\Big[\sum_{j=1}^{N} T_{ij}s_j(n\tau+\tau) - I_i\,\Big]\;;
\end{aligned}$$

making the same mean field approximation,

$$\begin{aligned}
s_j(n\tau+\tau) \;&\approx\; <s_j(n\tau+\tau)> \\
&\approx\; P\{s_j(n\tau+\tau)=1\} \;=\; V_j(n\tau)\;,
\end{aligned}$$

yields

$$\frac{d}{dt}u_i(n\tau) \;=\; -\frac{1}{\tau}u_i(n\tau) \;+\; \frac{1}{\tau}\Big[\sum_{j=1}^{N} T_{ij}V_j(n\tau) - I_i\,\Big]\;;$$

with $\tau=1$ and $n=t$, we get

$$\frac{d}{dt}u_i(t) \;=\; -u_i(t) \;+\; \sum_{j=1}^{N} T_{ij}V_j(t) \;-\; I_i$$

which are the same as Hopfield's equations given in Figure 3-3.

Chapter 4

Discussion

4.1 Interpretation of the Computation Model

In Chapter 3, we derived a set of equations that model an analog circuit of amplifiers and interconnection resistors. This circuit is similar to a conventional *analog computer*. We will now interpret the computation that this circuit performs.

Recall that equations (3.2) were derived from a description of a LTU neural network. Typically, this LTU network is defined to perform some sort of computation such as a CAM or optimization. The network *computes* by evolving to a steady-state configuration from a given initial configuration. The final configuration represents the solution to the computation.

Likewise, for the analog neural network defined by equations (3.2), the solution to its computation is represented in the final, or steady-state, values of the variables $V_i(t)$. These variables represent a *probability measure* on the neurons in the underlying LTU network from which equations (3.2) were derived. Each variable $V_i(t)$ represents the probability that the i^{th} neuron in the LTU network is 'firing'.

Thus, if the LTU network has some final stable configuration, $S = [1,0,1,1,...,0]$, we can say that each neuron has some probability, 0.0 or 1.0, of being in the 'firing' state. The final state $V(t) = [V_i(t),...,V_N(t)]$, will represent these probabilities under a *mean-field* approximation.

Consider a CAM computation where the initial state is exactly between two (or more) stored states. In terms of the energy landscape analogy, the marble starts at the peak of a hill. Because the Hopfield LTU network makes a neuron update randomly, the first neuron state change may move the system towards any one of the surrounding stable states. On different simulations of the computation, different stable states make occur. The marble may fall into any of the surrounding valleys, and on different trials will fall into different valleys. In this case, the solution given by the analog network will indicate the average state of each neuron, averaged over the possible final states of the LTU network. A similar situation arises for the stochastic network at a non-zero temperature. In this case, there is a non-zero probability that each neuron will 'fire', due to *thermal noise.* Thus, the probability that the i^{th} LTU neuron is 'firing' will be represented by a corresponding non-zero value for $V_i(t)$.

4.2 Extensions and Future Work

The key issue regarding this computation model is the accuracy of the mean-field approximation. Its application in statistical physics typically relies on a large numbers of particles to take an average over. There are examples in which the mean-field theory fails to accurately model system properties [2]. Cited examples of computation with analog

neural networks are all based on networks *a priori* designed for the computation [10, 22], so they cannot be used as verification of the mean field approximation of an underlying LTU network computation. Further analysis is required to determine the conditions under which the approximation is valid. A possible extension of the approximation method used here is approximation through Markov random fields [3, 21].

The analog neural networks presented by Hopfield have previously been identified as being mean-field approximations to the corresponding LTU networks [10]. However, a derivation and general form of the approximation has not appeared in the literature. One of the advantages provided by the general derivation given in Chapter 3 is its possible extension to approximating the statistics of systems of units with more than two states. It is felt that the computational model developed can be extended, at least under certain conditions, to model a very general system of continuously-valued units. In such a system, each unit (neuron) is described not by a two-state transition rule, but by a *probability density function* (pdf). It is expected that such an extension would lead to a system of *stochastic differential equations* for $V_i(t)$, in place of the deterministic differential equations (3.2). There is a growing body of theory related to stochastic non-linear systems in the areas of control and signal estimation [15] from which the study of such computational models could draw from.

References

[1] C. W. Anderson.
Learning and Problem Solving with Multilayer Connectionist Systems.
PhD thesis, University of Massachusetts, 1986.

[2] K. Binder (editor).
Monte Carlo Methods in Statistical Physics.
Springer-Verlag, Berlin, 1979.

[3] S. Geman and D. Geman.
Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images.
IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6:721-741, 1984.

[4] R. J. Glauber.
Time-dependent statistics of the Ising model.
Journal of Mathematical Physics 4:294-307, 1963.

[5] M. A. Cohen and S. Grossberg.
Absolute stability of global pattern formation and parallel memory storage by competitive neural networks.
IEEE Transactions on Systems, Man, and Cybernetics SMC-13:815-826, 1983.

[6] D. O. Hebb.
The Organization of Behaviour.
Wiley, New York, 1949.

[7] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley.
Boltzmann machines: Constraint satisfaction networks that learn.
Technical Report CMU-CS-84-119, Carnegie-Mellon University, Pittsburgh, PA, 1984.

[8] J. J. Hopfield.
Neural networks and physical systems with emergent collective computational abilities.
Proceedings of the National Academy of Sciences, USA 79:2554-2558, 1982.

[9] J. J. Hopfield.
Neurons with graded response have collective computational properties like those of two-state neurons.
Proceedings of the National Academy of Sciences, USA 81:3088-3092, 1984.

[10] J. J. Hopfield and D. W. Tank.
"Neural" computation of decisions in optimization problems.
Biological Cybernetics 52:141-152, 1985.

[11] R. A. Howard.
Dynamic Probabilistic Systems.
Wiley, New York, 1971.

[12] R. A. Howard.
Dynamic Probabilistic Systems.
Wiley, New York, 1971.

[13] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi.
Optimization by simulated annealing.
Science 220:671-680, 1983.

[14] C. Kittel and H. Kroemer.
Thermal Physics.
Freeman, San Francisco, 1980.

[15] V. Krishnan.
Nonlinear Filtering and Smoothing: An Introduction to Martingales, Stochastic Intregrals and Estimation.
Wiley, New York, 1984.

[16] W. S. McCulloch and W. Pitts.
A logical calculus of the ideas immanent in nervous activity.
Bulletin of Mathematical Biophysics 5:115-133, 1943.

[17] M. Minsky and S. Papert.
Perceptrons: An Introduction to Computational Geometry.
MIT Press, Cambridge, MA, 1969.

[18] P. Peretto.
Collective properties of neural networks: A statistical physics approach.
Biological Cybernetics 50:51-62, 1984.

[19] F. Rosenblatt.
Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.
Spartan Books, Washington, DC, 1961.

[20] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group.
Parallel Distributed Processing: Explorations in the Microstructure of Cognition.
MIT Press, Cambridge, MA, 1986.

[21] A. J. F. Siegert.
From the mean field approximation to the method of random fields.
Statistical Mechanics at the Turn of the Decade.
Marcel Dekker, Inc., New York, 1971, pages 145-174.

[22] D. W. Tank and J. J. Hopfield.
Simple "neural" optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit.
IEEE Transactions on Circuits and Systems CAS-33:533-541, 1986.

[23] C. von der Malsberg.
Self-organization of orientation sensitive cells in the striate cortex.
Kybernetik 14:85-100, 1973.

VITA

James R. Anderson was born in Ft. Campbell, Kentucky, on November 22, 1960, the son of Mary Pittman Anderson and Lt. Frederick Douglas Anderson. After his father's leave of the U.S. Army, his family eventually settled in Franklin, Tennessee. Upon graduation from Franklin High School in 1978, he entered Tennessee Technological University in Cookeville, Tennessee. While in attendance at Tennessee Tech he was active in the Student Chapter of the Institute of Electrical and Electronic Engineers (IEEE) and Eta Kappa Nu. He was awarded the degree of Bachelor of Science in Electrical Engineering, Summa Cum Laude, in May, 1983. Upon graduation he was awarded with graduate fellowships from the National Science Foundation, Tau Beta Pi, and the Office of Naval Research, the latter of which he accepted. He began his graduate studies at Tennessee Tech, and then entered The Graduate School of Engineering of the University of Texas at Austin in September, 1984.

Permanent Address: James R. Anderson
1411 Elm Brook Drive
Austin, Texas 78758

This thesis was typed by the author.