OREGON STATE UNIVERSITY COMPUTER SCIENCE DEPARTMENT

Dynamics of Neural Nets

Paul Cull
Department of Computer Science
Oregon State University
Corvallis, OR 97331-3902

90-20-2

Dynamics of Neural Nets

Paul Cull

Department of Computer Science

Oregon State University

Corvallis, Oregon 97331

USA

July 25, 1990

1 Introduction

In this short paper, I plan to review the work I have done on neural nets over the course of the last 20 years. As one might reasonably expect the questions being asked and the approaches to solving them have evolved, but there are still fundamental questions which remain unanswered and are pehaps unanswerable with present techniques.

I have used the word "dynamics" in the title to indicate that the major emphasis of the paper will be how the state of a neural net changes in the course of time either autonomously, that is without input, or in response to input. The problems of learning in neural nets will not be directly addressed, although I will argue that many questions about learning can be recast as questions about dynamics.

The paper will be organized in five parts. In this section, Introduction, I will discuss some basic definitions, give some history, and describe my involvement with various issues. In section 2, Dynamics, I will discuss methods for calculating and estimating such quantities as the lengths of state cycles. In section 3, Complexity, I will discuss the computational complexity of answering some questions about neural

nets. In section 4, Chaos, I will discuss the applicability of the concept of chaos to the description of the dynamics of neural nets. In section 5, Analog Models, I will contrast neural models in which the state variables are continuous with the more traditional models in which the state variables are discrete.

A neural net consists of neurons which are connected together so that some neurons give inputs to other neurons and some neurons receive output from other neurons. There are also external inputs and outputs, so that a neuron may receive an input from outside the net as well as from neurons within the net, and so that a neuron may send an output to outside the net as well as to other neurons within the net. The dynamics of the net are specified by the dynamics of the individual neurons and by the topology of the interconnections. In the usual case, neurons are considered to be 2 state devices. A neuron is either "on" or "off". No gradations are allowed. Time is usually assumed to be quantized, so that one can speak of instants of time. The state and output of a neuron at time t+1 will be a function of the inputs to the neuron at time t. Usually the output of a neuron is simply the neuron's state. Since the input lines will carrying one of 2 values, it is usual to identify the values with "true" and "false", and to say that a neuron computes a Boolean function of its inputs. In the following sections, I will assume discrete states and discrete time, but in section 5, I will discuss models which behave in a more continuous manner.

As I mentioned earlier, I have been involved with neural nets for over 20 years. I first became acquainted with them through Rashevsky's(1960) book. There I learned of the work of McCulloch and Pitts(1943). The fact that this work was carried out in Chicago was one of the reasons I decided to do my graduate studies at the University of Chicago. Although neural nets were not a primary area of study at the Committee on Mathematical Biology, I did learn of the work of Landahl and Runge(1945) suggesting the use of matrices in the study of neural nets. I learned more about neural nets and their applications from the Committee on Information Science who studied the design and application of digital computers. Switching theory which is the basis for the logical design of computers, derives from Von Neu-

mann's(1946) adaption of McCulloch and Pitts's neurons. This theory was further developed by Burks and Wright(1953), Kleene(1956), and Rabin and Scott(1959), and is summarized in Minsky's(1967) book. A visit by Caianiello introduced me to his work outlined in Caianiello(1964). A visit by McCulloch introduced me to some ideas of Kauffman(1969). Kauffman joined the faculty at Chicago at about that time. Ricciardi, a student of Caianiello's, supervised my thesis. Subsequently, I have been able to visit Italy to work with Caianiello, Ricciardi, Capocelli, DeLuca, and others. Most of the work that developed out of working with this group of people will be described in section 2. Although I was able to develop methods for solving some problems, other problems seemed to frustrate any reasonable approach. Only when I studied the theory of computational complexity did I understand that many problems about neural nets are in an exact sense "hard". The theory of complexity was developed throughout the 70s and is summarized in Garey and Johnson(1979). I will discuss some hard problems about neural nets in section 3.

Everything has ups and downs from the stock market to natural populations. These booms and busts were usually thought to be occurances where simple mathematical models broke down. The work of Li and Yorke(1975) and May(1976) showed on the contrary that such behaviors were inherent even in the simplest nonlinear models. These bounded but aperiodic or perhaps omniperiodic behaviors were first shown to exist in continuous state discrete time models, and were quickly shown to occur in continuous time continuous state models. The existence and meaning of such chaotic behavior in discrete time discrete state models is more problematic. I will discuss our attempts to measure chaotic behavior in neural nets in section 4. I will also mention some recent discussions about the implications of chaos for models of living systems.

Rashevsky's(1933) two-factor model is a continuous time continuous state neural model which antedates the discrete models. There is a line of research on nets of these model neurons showing that they can model many classes of phenomena. For example, Householder and Landahl(1945) and Rashevsky(1960) include a number of these analog neural net models. A more complete model of single neuron

behavior given by Hodgkin and Huxley(1952) resulted in their winning the Nobel prize. Fitzhugh(1961) and Nagumo et al(1962) have studied simplifications of the Hodgkin-Huxley model. More recently Mead(1989) has suggested that analog neural models may be easy to fabricate in VLSI and may form the basis for a new generation of computers. In section 5, I will discuss the difficulties facing an elucidation of the theory of analog neural nets, and describe our project for designing and building nets of analog neural elements.

2 Dynamics

Each neuron in a neural net will be in one of a finite set at each instant of time, so the state of the whole net may be represented by a vector. The vector will have n components, one component for each neuron in the net. The value of a component will be the state of the corresponding neuron, and the state vector will be a vector over the finite set of states possible for each neuron. There will be a finite set of possible inputs. Often the input is thought of as coming in on several input lines and each line is assumed capable of carrying the same finite set of symbols. Assuming that there are m input lines it is natural to represent the input as an m-dimensional vector over the set of symbols which can be carried on a single line. The input and state vectors are usually subscripted to indicate the time to which they refer. Since time is assumed to be discrete, the dynamics of a neural net may be described by the difference equation:

$$X_{t+1} = F(X_t, Y_t)$$

where X represents state vectors and Y represents input vectors. If there are no inputs or equivalently the inputs are constant, then the net is called an autonomous net and the dynamics are given by the equation:

$$X_{t+1} = F(X_t).$$

With no more structure on the underlying sets, the above equations can still be used to follow the sequence of states the net passes through. More structure on the underlying sets would allow more reasonable ways to answer questions about a net's dynamics. Since neurons are usually assumed to have 2 states, these states can be identified with the Boolean values "true" and "false" and the state vectors can be considered as vectors over a Boolean algebra. If the input lines are also assumed to take on 2 values then functions in the dynamic equations can be considered as Boolean functions and hence these functions can be represented using the three operations of "and", "or", and "not". While these functions have a number of nice properties, "and" and "or" lack inverses which makes some computations difficult.

If the underlying set were a field then traditional methods for analyzing dynamical systems would be available. A reasonable way to introduce field structure is to associate the 2 state values with the integers 0 and 1 and to carry out computation with addition and multiplication both taken mod 2. These two numbers with these two operations form the finite field which is often called GF(2) or Z_2. As we will see later there are certain limitations caused by working in Z_2. To avoid these limitations one would like to work in a larger field, either an infinite field or a field whose size could be increased as the size of the nets under consideration increased. The field Z_2 can be represented within the field of rationals or its dyadic subfield by associating integer 0 with rational 1, associating integer 1 with rational -1, representing addition mod 2 by rational multiplication, and representing multiplication mod 2 by a simple rational polynomial. That is, x*y mod 2 is represented by 1/2 (1 + u + v - uv) where u is the rational representing x and v is the rational representing y.

2.1 Analysis and Synthesis

Dynamics has an analytical and a synthetic part. In analysis, a dynamic system is given and the task is to find a description of the system's behavior. This description should in some reasonable sense be simpler or more compact than the original representation of the system. For an autonomous system, an analysis might give the number and location of the fixed points, and the periods of any cycles. For a system with inputs, an analysis might give the period of the state cycles as a function of the period of the input, or a characterization of the input sequences which result in a specified output. In synthesis, a behavior is described and the task is to build a system with the desired behavior, or to show that no system in the class of interest can have the desired behavior.

The analysis and synthesis problems for loop-free nets were solved by McCulloch and Pitts(1943).

Theorem 1 *(Analysis Of Loop-Free Nets) In a loop-free net of memoryless neurons the state of the net is determined by the inputs over a fixed finite interval of the past. That is,* $X_{t+D} = G(Y_{t+D-1}, Y_{t+D-2}, ..., Y_t)$.

This theorem states that loop free nets of memoryless neurons have a memory of only finite duration. Notice that initially the state of the net will depend on the state that the net is started in, but for a loop-free net, the initial state will be forgotten after D time steps. Here D is the depth of the net, the length of the longest path from the input of a neuron to the output of any neuron. Nets can have arbitrarily high D, so how useful this theorem is, may depend on how large D is.

McCulloch and Pitts defined two kinds of model neurons. The absolute inhibition neuron, and the additive inhibition neuron. The absolute inhibition neuron has excitatory and inhibitory inputs so that the neuron will be "on" at time t+1 if and only if at time t the sum of its excitatory inputs exceeds the neuron's threshold and no inhibitory inputs are received. The additive inhibition neuron, which is also called a linear threshold neuron, has excitatory and inhibitory inputs so that the neuron will be "on" at time t+1 if and only if at time t the sum of its excitatory inputs minus the sum of its inhibitory inputs exceeds the neuron's threshold. They showed that these neuronal modes were equivalent in the sense that a net of one kind of neurons could be simulated by a net of the other kind of neurons.

In more detail, this simulation means that for each initial state of the first net and any designated subset of the neurons of the first net there is an initial state of the second net and a set of neurons of the second net, and a delay d, so that for every input sequence the state of the specified neurons in the second net at time t+d is identical to the states of the designated set of neurons in the first net at time t.

Using this notion of simulation McCulloch and Pitts showed:

Theorem 2 *(Synthesis of Loop-free Nets) Let G be any function from* $\{0,1\}^{mD}$ *to* $\{0,1\}^n$ *so that the desired net would have as its state equation*

$$X_{t+D} = G(Y_{t+D-1}, Y_{t+D-2}, ..., Y_t)$$

where X has n components and each Y has m components, then it is possible to construct a loop-free net of absolute inhibition neurons or linear threshold neurons so that the constructed net will simulate the desired net.

Note that the constructed net will generally have many more than n neurons, but that if one observes a specified set of n neurons, these neurons will behave as desired. Further they showed that the delay necessary was at most 2.

McCulloch and Pitts also gave an analysis of nets with loops. This analysis was clarified by Kleene (1956), who showed:

Theorem 3 *(Analysis of Nets with Loops) Consider a neural net with a specified initial state and a designated neuron, then the set of input sequences which will cause this neuron to be "on", can be specified by a finite expression using only sets of input sequences with at most one sequence and the operations of union, concatenation, and star.*

This is a strong theorem in that it does not depend on the specific type of neuron used in the net. The only assumptions are that the neurons have a finite number of states, and that the neurons either have a finite memory span or can be simulated by a net of neurons with finite memory span. The dynamic equation for individual neurons may be more complicated than the dynamic equation for McCulloch-Pitts neurons.

Kleene also proved the converse of the above theorem.

Theorem 4 *(Synthesis of Neural Nets) For any set of input sequences which can be specified by a finite expression of input sequences with at most one symbol and the operations of union, concatenation, and star, there is a neural net of McCulloch-Pitts neurons with a designated initial state, a designated neuron, and a delay d, so that when the net is started in the designated state, then the designated neuron will be "on" at time t+d if and only if the input sequence from time 0 to time t is in the set specified by the finite expression.*

There is a nice exposition of Kleene's results in Minsky(1967). One should notice that these results are very strong. They show that nets of the seemingly simple

McCulloch-Pitts neurons can compute everything that can be computed by nets of more complicated neural models as long as the restrictions of discrete time, finite number of states, and finite memory span apply. Minsky also gives a number of neuronal models which are equivalent in computational power to McCulloch-Pitts neurons.

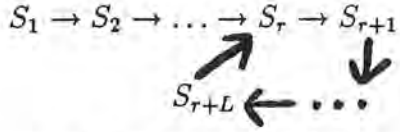
There are at least two ways in which these results are weak, however. The results do not address the possibility of neuronal models which are computationally weaker than the McCulloch-Pitts neurons, and the results do not give direct ways to answer traditional questions about dynamics such as the existence and number of fixed points.

Consider an autonomous neural net. From the point of view of Kleene's theorems this is a net with a constant input. Kleene's result tells us that the behavior of this net can be represented by an expression over a single symbol using the three operators, but even given the n expressions, one for each neuron in the net, it is unclear how one could reasonably use these expressions to find fixed points or cycle lengths.

Consider an autonomous net in which each neuron computes the sum of its inputs; then $X_{t+1} = AX_t$, where A is a matrix. Now finding the fixed points can be easily done by solvng a system of linear equations. It is unclear how Kleene's theorem could help us here.

2.2 Autonomous Nets

As I have mentioned above the usual problem for an autonomous system is to find the fixed points and cycles. For a finite state system this is most of the story. Consider the following diagram:



If we follow the sequence of states starting from the state S_1, then we may pass through some sequences of transient states, but eventually we will end up in

a cycle. Hence finding the number of cycles of each period will describe the long term behavior of such a system, but it will not give information about the short term transient behavior.

If the dynamic equation for a neural net is linear, that is, $X_{t+1} = AX_t$ where A is an nxn matrix, or affine, that is, $X_{t+1} = AX_t + C$, where C is an $nx1$ vector, it is easy to find the fixed points by solving a system of linear equations. Further calculations can determine the number and the periods of the various cycles. Techniques for these calculations were decribed by Elspas (1959).

In my thesis Cull (1970) and subsequent papers, Cull (1971a, 1971b), I sought to give analogous techniques for nonlinear neural nets. Starting from the dynamic equation

$$X_{t+1} = F(X_t)$$

where F is a nonlinear function from $\{0,1\}^n$ to $\{0,1\}^n$, I wanted to apply the techniques of linear algebra to the calculation of the behavior of such a net. The problem was: How to convert a nonlinear system to a linear system? If one thinks of a net in terms of states then the progression from state to state can be captured in a transition matrix. For a net with n neurons, there are 2^n states, so one can define a 2^n x 2^n transition matrix T in which $T_{i,j} = 1$ iff state j goes to state i. Hence one has a linear representation of a net as $Y_{t+1} = TY_t$, in which a state is represented as a 2^nx1 vector which contains a single 1. Notice that most of the vectors of this linear system represent sets of states rather than single states. From this linear transitional representation, it seems reasonable that there should be a way to linearize the nonlinear functions that define the net's dynamical equation. The trick here is to think of a function from $\{0,1\}^n$ to $\{0,1\}$ as as polynomial over GF(2) in n variables. For example, any function from $\{0,1\}^2$ to $\{0,1\}$ can be represented as

$$f(x_1, x_2) = c_0 + c_1 * x_1 + c_2 * x_2 + c_{12} * x_1 * x_2$$

where each $c_i \epsilon \{0,1\}$ and the operations are carried out mod 2. Further by expanding the n-dimensional vector X to the 2^n dimensional vector Z, which has as

its components all the products of the n components of X, the dynamic equation $X_{t+1} = F(X_t)$ may be represented by the linear equation $Z_{t+1} = HZ_t$, where H is the matrix which has as its rows the coefficients of the polynomials which are the products of the polynomials in F.

As a small example, if

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_{t+1} = \begin{pmatrix} a_0 + a_1x_1 + a_2x_2 + a_{12}x_1x_2 \\ b_0 + b_1x_1 + b_2x_2 + b_{12}x_1x_2 \end{pmatrix}_t = F\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}_t\right)$$

then

$$Z_{t+1} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_1x_2 \end{pmatrix}_{t+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ a_0 & a_1 & a_2 & a_{12} \\ b_0 & b_1 & b_2 & b_{12} \\ c_0 & c_1 & c_2 & c_{12} \end{bmatrix} \begin{pmatrix} 1 \\ x_1 \\ x_2 \\ x_1x_2 \end{pmatrix}_t = HZ_t$$

where

$$c_0 = a_0b_0$$

$$c_1 = a_1b_0 + a_0b_1 + a_1b_1$$

$$c_2 = a_2b_0 + a_0b_2 + a_2b_2$$

$$c_{12} = (a_0 + a_1 + a_2 + a_{12})b_{12} + (b_0 + b_1 + b_{12})a_{12} + a_2b_1 + a_1b_2$$

.

The equations for the c's come from forming the product of the polynomials which compute the next values for x_1 and x_2, and noticing that $x^2 = x$ in GF(2).

It should be almost immediately clear that eigenvectors of T corresponding to the eigenvalue 1 will give unions of the cycles in the states of the net. Similarly the characteristic polynomial of T can be written so that each factor of the form $\lambda^k + 1$ corresponds to a cycle of length k. It may at first seem surprising that the same statements can be made about the matrix H, but this is true because T and H are similar matrices over GF(2).

In fact, there is a very special matrix P_n which gives the similarity between T and H. This matrix, P_n can be defined recursively by

$$P_0 = [1]$$

$$P_{n+1} = \begin{bmatrix} P_n & P_n \\ 0 & P_n \end{bmatrix}$$

For a net with n elements

$$P_n T = H P_n$$

and since P_n is self-inverse, that is, $P_n * P_n = I$,

$$P_n T P_n = H \text{ and } T = P_n H P_n$$

The special form of P_n depends on the ordering of the components used in the linearization, and with this ordering P_n serves as a fast Fourier transform which is useful in quickly multiplying polynomials over GF(2). This is only a sketch of the results more details are in Cull (1970), Cull (1971a), Cull (1971b), and Cull (1976). The major problem with this linearization is that it fails to distinguish things that are equivalent mod 2. For example,

$$(\lambda + 1)^4 = \lambda^4 + 1 = (\lambda^2 + 1)(\lambda + 1)^2$$

so the characteristic polynomial will not distinguish between 4 cycles of length one, 1 cyle of length four, or 1 cycle of length two and 2 cycles of length one. Further, averaging is problematic because the field does not have elements to express desired fractions. To avoid these problems, we need a field that is larger and behaves like the rationals. Fortunately such a linearization is possible. The trick is to represent 0 mod 2 by 1 rational, and 1 mod 2 by -1 rational, then any function from $\{0,1\}^n$ to $\{0,1\}$ can be represented by a function from $\{1,-1\}^n$ to $\{1,-1\}$.

Theorem 5 *Any function from* $\{1,-1\}^n$ *to* $\{1,-1\}$ *can be represented as a polynomial in n variables with coefficients which are integers divided by* 2^n.

This theorem can be proved by noticing that the coefficients c's are related to the values v's of the function by

$CW_n = V$ where W_n is the matrix defined recursively by

$$W_0 = [1]$$

$$W_{n+1} = \begin{bmatrix} W_n & W_n \\ W_n & -W_n \end{bmatrix}$$

and noting that $W_n W_n = 2^n I$.

With this representation there is also a function matrix, call it G, which gives a linearized dynamic equation. Of course, this function matrix is similar to the transition matrix T because $W_n T = G W_n$ and $T = 1/2^n W_n G W_n$ and $G = 1/2^n W_n T W_n$.

Calculating the characteristic polynomial of G or equivalently T over the rationals will give a polynomial which has a unique factorization of the form

$$\lambda^r(\lambda_1^k - 1)(\lambda_2^k - 1)\ldots(\lambda_m^k - 1)$$

where r is the number of transient states and $k_1, k_2, \ldots, k_m$ are the lengths of the cycles. Much of the work on this linearization comes from DeLuca (1970), Caianello (1973), and Caianello and Grimson (1975).

My major reason for investigating this representation was to use it to calculate the expected lengths of cycles in randomly connected neural nets. The key result needed in this calculation is:

Theorem 6 *The expected number of states in cycles whose lengths divide k is*

$$E(tr(T^k)) = E(tr(G^k))$$

where E is the expectation operator and $tr(M) = \sum_{i=1}^{n} M_{i,i}$ *is the trace of the matrix M.*

Cull (1978) demonstrates that these expected values can actually be calculated when the neural net is random in the sense that each neuron is equally likely to compute any function of n variables. This random situation, which is usually called the totally connected case, can be analyzed by more straightforward combinatoric arguments when it is realized that the situation being studied is the statistics of a random map from a finite set to the same finite set. Such mappings were studied long ago by Rubin and Sitgreaves(1954), and their properties were more recently summarized by Gelfand(1982). The point of Cull(1978) was not to find a new harder

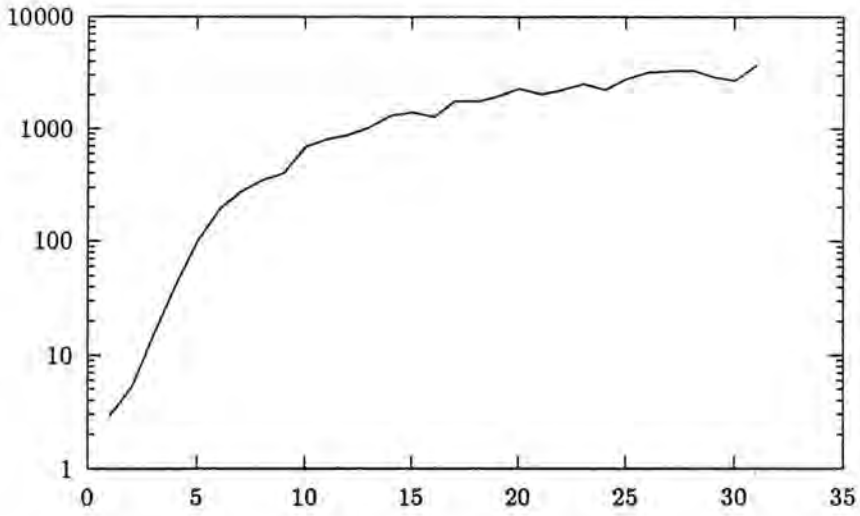


Figure 1: Expected cycle length as a function of connectivity for nets of 31 neurons.

way to compute known results, but rather to develop a technique which could be used in a variety of situations other than the totally connected case. The calculation of known results was only a demonstration of the validity of the technique. With this technique it should be possible to obtain at least asymptotic estimates of the dependence of various statistics on various properties of nets. In particular, Kauffman(1969) and Walker(1984) have carried out simulations to study the dependence of cycle lengths on the number of inputs (connectivity) of each neuron. We have also carried out such computer calulations and an example appears in Figure 1.

I believe that the techniques outlined here and in Cull (1978) will enable one to derive the asymptotic form of curves like the curve in Figure 1, but such calculations will not be trivial. This is an open problem which awaits a researcher with the time and energy to obtain its solution.

3 Complexity

As we have outlined in the previous section, many questions about the dynamics of neural nets may be answered by algorithms whose running time is linear or close to linear in the number of states. These algorithms are reasonable for neural nets whose representation is about the same size as the number of states. While there are nets with such large representations, there are also classes of nets which have smaller representations. For example, a linear net can be represented by a matrix using about n^2 bits while the net has 2^n states. Similarly, linear threshold nets can be represented by about n^2 weights and if these weights have few bits then these linear threshold nets can be represented with about n^3 bits. An obvious question is whether there are algorithms for answering questions which run in time proportional to the size of the representation rather than the number of states. In particular can questions about nets which have $O(n^d)$ representations be settled by algorithms which have running times $O(n^{d+k})$ for some small values of k.

For linear nets, some easy questions are

1. Will a particular neuron ever go "on"?

2. Given a state X is there a state Y so that X is the next state of Y?

3. Are there any transient states?

4. Given two nets are they isomorphic?

Each of these questions can be answered in time at worst $O(n^3)$. Unfortunately the existence of fast algorithms does not seem to hold for all classes of nets with small representations. In particular several problems about neural nets in various representations can be shown to be NP-complete or PSPACE-complete, strongly suggesting that there are no fast algorithms for these problems. For example, the function computed by a neuron can be represented by a Boolean expression which can be satisfied exactly when there is some setting of all the neurons which turn the neuron "on". Cook (1971) showed that this satisfiability problem for Boolean

expressions in conjunctive form is NP-complete. This means that it is easy to show that a neuron turns "on" if one can guess the setting that turns the neuron "on", but there may be no short way to show that the neuron is always "off". This satisfiability problem is as difficult as any problem in NP, meaning that if there were a fast algorithm for satisfiability there would be a fast algorithm for every problem which has a quick verification (short proof). Since many of these problems have been studied for 200 years without the discovery of a fast algorithm, it seems reasonable to assume that no fast algorithms exist for these problems (Garey and Johnson (1979)).

It is tempting to believe that the NP-completeness is a result of the representation being too general and that by restricting the representation NP-completeness would vanish. Often this is the case. As above, restricting to linear neurons makes satisfiability easy. Even if the restriction is to linear threshold neurons, satisfiability is easy. To determine if a linear threshold neuron will ever turn "on" simply determine if the sum of positive weights is greater than the threshold. Thus if all of the neurons with positive effect on the given neuron are "on" and all of the neurons with negative effect are "off", then the neuron will turn "on". But if the sum of the positive weights is less than threshold, then no matter how the settings of the other neurons are arranged the neuron will never turn "on".

The fact that satisfiability is easy for linear threshold neurons might give one some hope that problems about linear threshold nets would be substantially easier than the same problems for general nets, but linear threshold nets are as general as general nets. In particular, any Boolean function can be computed by a two-level linear threshold net. So while one can easily solve satisfiability for a single LT-neuron, satifiability for a net of LT-neurons could be difficult as we will show next.

Theorem 7 *The following questions are NP-complete.*

1. *Is there a setting of the inputs which simultaneously turns on all neurons in a (single-level) LT-neural net?*

2. *Is there a setting of the neurons in an autonomous LT-neural net, so that at the next time step all of the neurons will simultaneously be on?*

Proof: If either of these questions can be answered positively one can guess the required settings and quickly check that the guessed setting has the required property. Hence the problems are in NP. Each question is NP-complete by reduction from 3-SAT. In Garey and Johnson(1979) there is a proof that 3-SAT is NP-complete. An instance of 3-SAT is a Boolean expression in clause form in which each clause contains 3 literals, that is, Boolean variables which appear in complemented or uncomplemented form. The expression is satisfiable iff there is a setting of the variables so that at least one literal in each clause is true.

Given any instance of 3-SAT, construct a net with one neuron for each clause and one input line for each variable, and connect each input line to each neuron corresponding to a clause containing the variable represented by the input line. The connection weights and threshold will be assigned depending on the number of complemented variables. If a clause has no complemented variables, then the corresponding neuron has threshold .5 and three connections each with weight +1. Clearly if at least one input line is on then the net excitation is at least .5 and the neuron fires, while if all the input lines are off the net excitation is -.5 and the neuron does not fire. If the clause has one complemented variable, then the corresponding neuron has threshold -.5, the input line for the complemented variable has weight -1, and the input lines for the two uncomplemented variable each have weight +1. Clearly if the complemented variable is false, the net excitation is at least .5 and the neuron fires. Similarly if at least one of the complemented variables is true, the net excitation is at least .5. Finally if the complemented variable is true and both the uncomplemented variables are false, the net excitation is -.5 and the neuron will not fire. If the clause has two complemented variables then the corresponding neuron has threshold 1.5, the input lines corresponding to the complemented variables each have weight -1, and the input line corresponding to ther uncomplemented variable has weight +2. Clearly the neuron will have net excitation at least .5 unless the complemented variables are both true and the uncomplemented variable is false. So

the neuron will fail to fire exactly in the case that is required. Finally if the clause has three complemented variables then the corresponding neuron has threshold 2.5 and a weight of -1 on each of its three input lines. Clearly this neuron will fire unless all three variables are false, and so the neuron will operate correctly. This construction shows how to construct with very little effort a neural net which has an input setting which will fire all its neurons exactly when the corresponding instance of 3-SAT has a satisfying assignment. We should note that our constructed net has only one level; the connections are from inputs to neurons and there are no interconnections between neurons.

A similar construction could build an autonomous LT-neural net which mimics an instance of 3-SAT. In this construction the neurons would have to represent both the variable and the clauses. This can be managed by letting the number of neurons be the maximum of the number of variables and the number of clauses. Each neuron would compute a function corresponding to a clause. If there are more variables then several neurons would compute the same function, while if there are more clauses each neuron would compute a different function. A subset of the neurons would represent variables and there would be connections from the outputs of neurons representing variables to the inputs of neurons representing clauses which contain those variables. The connection weights and thresholds would be assigned in the same manner as in the previous construction.□

The complete generality of LT-neural nets can also lead to PSPACE-complete problems. A problem is in PSPACE if there is a polynomial p(n) and an algorithm so that every instance of size n of the problem can be solved by the algorithm using at most p(n) bits of memory. A problem, Prob, is PSPACE-complete if every problem in PSPACE can be reduced to Prob using a log space transformation. For example, the equivalence problem for regular expressions has been shown to be PSPACE-complete (Stockmeyer and Meyer (1973), Stockmeyer (1974), Garey and Johnson (1979)).

The generality of LT-nets follows easily from the ability of an LT-net to computer any Boolean function. Minsky (1967) gives a construction in which an n-state, m-

symbol finite state machine can be simulated by an LT-net with about n m neurons. In Minsky's construction the finite state machine is assumed to be deterministic and in the simulating net exactly one of the neurons representing a state-symbol pair is "on" at each time instant. So in this construction while the net has 2^{n*m} potential states, only $n * m$ of these states will be used in the simulation. The unused states of the net can be used to simulate a nondeterministic finite state machine. Nondeterminism here means that instead of a simple next state being determined by the present state and present input, a set of next states is determined. One could view this nondeterminism as giving the machine a choice between possible next states, but it may be more reasonable to simply say that the machine goes to the whole set of next states rather than making a choice. Now Minsky's construction can be used to build a LT-net with nm neurons which simulates a nondeteministic finite state machine with n-states and m-symbols. The limitation that only neuron at a time will be on in the net no longer holds, but at most n neurons at a time will be on. This means that about $m*2^n$ states of the net may be used in the simulation. Hence it is easy to construct an LT-net which can simulate a nondeterministic finite state machine.

As was mentioned above the equivalence problem for regular expressions is PSPACE-complete. This equivalence problem is: Given two regular expressions are the sets of strings represented by the two expressions the same? By a well-known construction (Hopcroft and Ullman(1979)), from a regular expression one can easily construct a nondeterministic finite automation which accepts exactly the set of strings represented by the regular expression. The automaton accepts the string x if when the automaton is started in a specified initial state and x is input to the automaton, the automaton will output YES at the end of the string x. So the equivalence problem for nondeterministic finite state machines is PSPACE-complete. Further since nondeterministic finite state machines can be simulated by LT-nets the equivalence problem for LT-nets in PSPACE-complete. We summarize this discussion in the following theorem.

Theorem 8 *The equivalence problem for LT-neural nets is: Given two LT-neural*

nets, do the nets accept the same set of input strings? This equivalence problem is PSPACE-complete.

Proof (Sketch): By reduction from equivalence of regular expressions, and Minsky's simulation of a finite state machine by an LT-neural net.□

These two examples will serve as examples of arguments classifying problems about neural nets. We have investigated a number of related problems and classified their complexity, but we have not yet published these results.

The following sketch shows some complexity classes and the placement of some problems within these complexity classes.

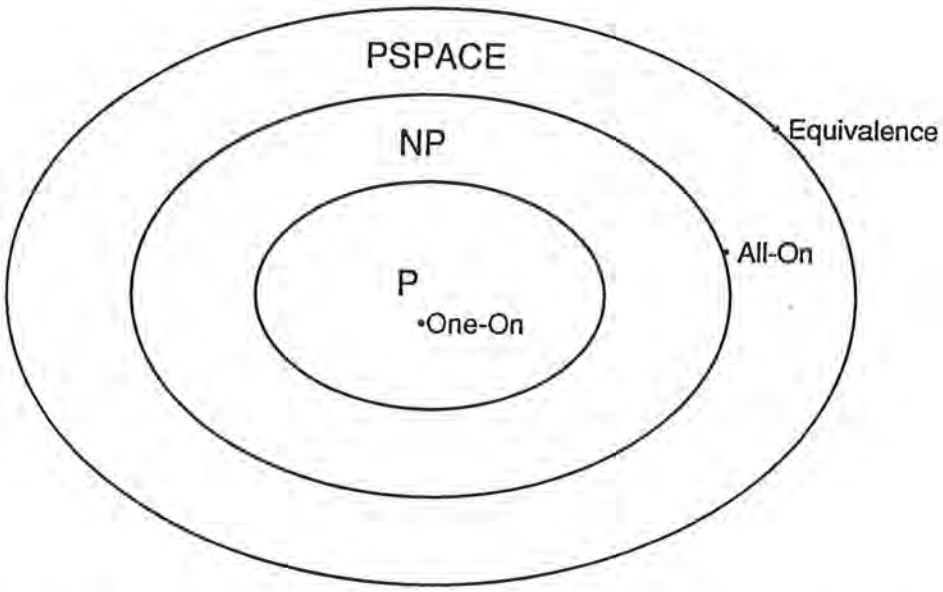


Figure 2: The 3 complexity classes P, NP, and PSPACE, and 3 problems we have discussed. A problem is on the boundary if it is complete for the class, that is, the problem is one of the hardest problems in the class.

4 Chaos

Most introductory texts on dynamics seem to imply that dynamical systems can only have a few kinds of typical behaviors: divergence, convergence to a fixed point, or convergence to a cycle. Of course, different regions of a state space could have different behaviors. Li and York(1975) showed that even simple difference equations could have another typical behavior in which the states in a sequence remained bounded, but the sequence did not converge to any periodic behavior. They introduced the term "chaos" for the situation in which the state space contained this wandering behavior and also contained cycles of every period. Subsequently, the term "chaos" has been used with various definitions by various authors. Chaos now loosely means dynamic behavior which is neither simple convergence, nor simple divergence, nor simple periodic behavior. Two features of most definitions of chaos are:

1. sensitive dependence on initial conditions, so that nearby states will evolve into widely separated states, and

2. non-integral attractors, so that trajectories will be bounded but non-periodic.

In Cull(1988) and Cull, Holloway, and Kaul(1989) we conjectured that chaos might be useful in describing the behavior of certain kinds of random neural nets. Figure 1 in section 2 shows that for low connectivity random nets usually have small cycles, while for high connectivity random nets usually have relatively long cycles with cycle lengths about the square root of the number of states. Hence the low connectivity nets might be described as being periodic, while the high connectivity nets might be described as being random. This leaves random nets with moderate connectivity as candidates for description as chaotic. Let us consider the state diagram of an autonomous neural net. If we start in any state and follow the sequence of states, eventually we will find a cycle of states. Trying to describe at least the cycle behavior in a concise way could be done by giving the lengths of all the cycles. This would be reasonable if there are not too many cycles and their lengths are relatively short.

Since random nets with low connectivity seem to display short cycle lengths, such a description might be reasonable. One might be able to estimate the cycle lengths for a given low connectivity net by choosing a few states at random and following the states until a cycle is detected. In this manner, one could get a reasonable picture of the behavior of the net. For high connectivity nets the cycles will tend to be very long, so following a few states to detect cycles could take a very long time. Thus we suggest that a reasonable description of a high connectivity random net is that the net behaves like a random mapping and it is unreasonable to calculate extra details. For moderate connectivity nets, the cycles may also be rather long. Notice that the curve in Figure 1 rises very quickly. We think that it may be possible to investigate some structure in these moderately connected nets. In particular, if the connections are assigned at random, there is a good possibility that at least for some pairs of nearby states, the trajectories starting at these states will diverge; that is, the two trajectories may go to different cycles, or even if the two trajectories go to the same cycle they may go to the cycle at quite different times or they may join the cycle at widely separated states.

The problem we are left with is to calculate the dimension of an attractor. If we could think of an attractor as being embedded in a Euclidian space then we would expect that if we sat on a typical point of the attractor and looked at the portion of the attractor that is within a radius r of our typical point, we would find that volume of the attractor would grow as r^d, where d is the dimension of the attractor. While finding the volume of the attractor might be difficult, we can approximate this calculation by taking a sample of points on the attractor, and still expect that the number of sample points within radius r of a sample point should grow as r^d . If we let C(r) be the average number of sample points within radius r of a sample point, and plot log C(r) against log r, we expect that this curve will have slope about d. Berge et al(1986) have suggested that the slope only be estimated on the part of the curve that looks like a straight line, and further that this calculation should be carried out on k-tuples of consecutive sample points with C(r) now being the average number of k-tuples within a k-dimensional sphere

around a k-tuple. They further suggest that we should accept the estimate for d when we get essentially the same value for several consecutive values of k. This technique for estimating dimension seems to work quite well for attractors like the Henon attractor in which the state variable is a real number. We used the Henon attractor to check our computer programs. Unfortunately we ran into problems when we attempted to estimate dimensions from neural net data. The most obvious measure of distance for binary state vectors is the Hamming distance, the number of components in which two binary vectors disagree. When we tried to estimate dimension using Hamming distance, the slope did not settle down but rather kept increasing as we increased k. We now believe that it was inappropriate to use Hamming distance with this estimation technique. A different estimation technique should be used with Hamming distance. Since the estimation technique worked for the Henon attractor and in calculations our computer was using a binary vector to represent a real number, we thought of turning the state vector for the neural nets into approximate real numbers by the same conversion; that is, the binary vector $(x_1, x_2, \ldots, x_n)$ could represent the real number $\sum_{i=1}^{n} x_i 2^{-i}$. While we had hope that this technique would work, at least if the ordering of the x's in the vector was random, the results of the calculation did not give a reliable estimate of dimension. Figure 3 shows an example output. If this gave a good estimate of dimension the straight line portions of the curves would be parallel, but in the figure these lines are not parallel.

The question of chaos in neural nets has become more important recently with the work of Freeman(1988) and Goldberger(1989) who suggest that real biological systems behave chaotically when they are operating in the normal range. They suggest that real systems only show periodic behavior when something abnormal is happening to an organism. Questions about chaos and what it means and how to measure it in quantum systems which have discrete states are being studied by theoreticians in physics and chemistry. Since chaos was originally defined for continuous systems, one may need a definition of continuity for discrete systems before chaos can be defined and measured. We leave chaos with the conviction that

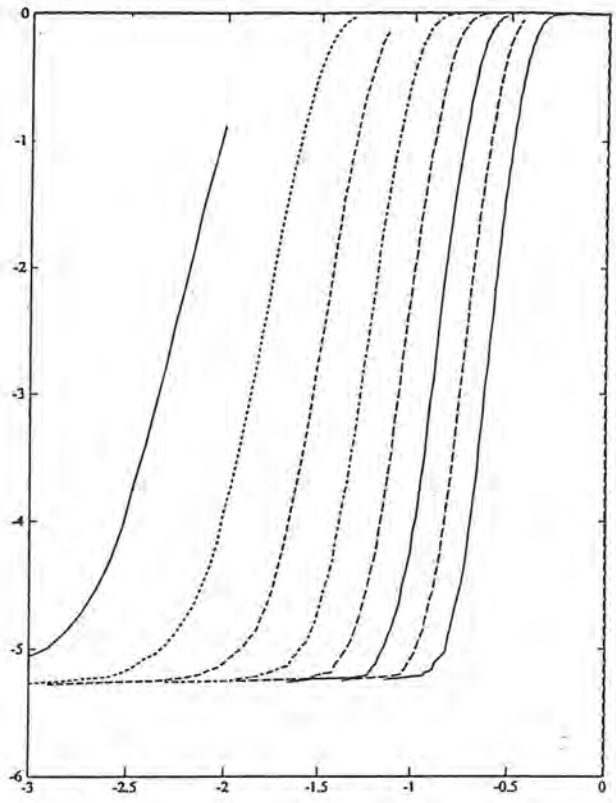


Figure 3: Typical plot of log C(r) vs. log r in normalized units. For a low dimensional attractor the curves should be parallel.

this important aspect of dynamics needs to be integrated into the theory of discrete neural nets, and with the reminder that chaos has been measured in continuous neural nets which are the topic of the next section.

5 Analog Models

Up to this point I have discussed discrete time finite state neurons, but in this section I will consider continuous models which were the subject of my first publication, Cull(1967). Continuous models have a long history from the work of Rashevsky(1933) and Hill(1936) through the Nobel prize winning work of Hodgkin and Huxley(1952) to the analog devices of Mead(1989). The work of Hodgkin and Huxley concentrated on a single neuron and produced a model with parameters that corresponded closely with measurable or potentially measurable physical and chemical properties of the neuron. Their model is rather complicated and suffers from numerical instability in that digital simulations of the model often produce behaviors which do not occur in the continuous model. To avoid those problems of too many parameters and numerical instability, most studies of neural nets use models for a single neuron which are less complicated than the Hodgkin- Huxley model. In particular, Rashevsky and co-workers showed how fairly small nets of model neurons could be used to model a wide variety of phenomena. This work is summarized in Rashevsky(1960) and Householder and Landahl(1945). This work is important since it shows that many phenomena particularly in the area of perception can be modeled with simple neural nets of simple model neurons. These models have two state components ("excitatory" and "inhibitory"), obey a linear differential equation, and have a nonlinear threshold function on the output. Many of these models can be represented in the form in Figure 4.

A reasonable conjecture is that a net of Rashevsky neurons can simulate the behavior of any net of reasonable continuous model neurons. In particular, this conjecture suggests that when creating models using continuous neurons, it might be more reasonable to use fairly simple model neurons like Rashevsky neurons rather than the more complicated but more exact Hodgkin-Huxley model. The difficulty with this conjecture is that it is only a conjecture and it has never been proven.

I want to point out a major difference between the theory of discrete time, finite state neural nets, and the theory of continuous time, continuous state neural nets.

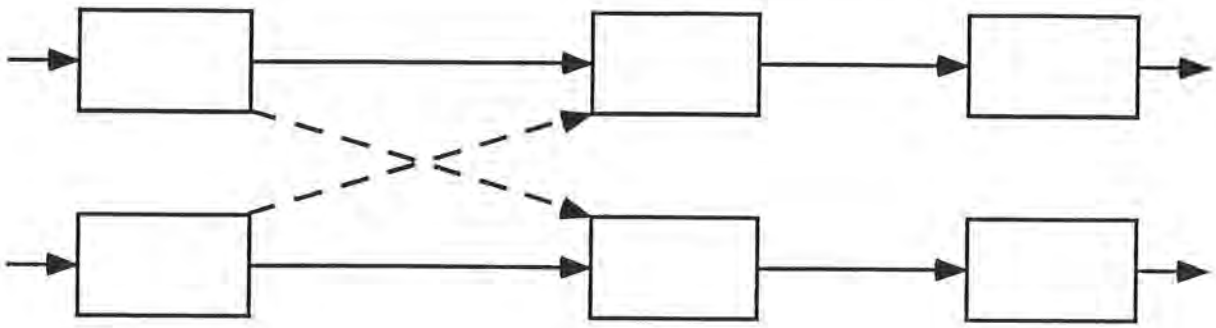


Figure 4: This cross-coupled connection can exhibit an amazing variety of behaviors depending on the values chosen for the various parameters.

For the discrete time, finite state nets, the Kleene theorems, given in section 2 of this paper, exactly characterize the class of sequences recognized by such nets. Since these theorems show that McCulloch-Pitts neurons suffice to simulate anything which can be done by discrete time, finite state model neurons, one can show that a model also suffices by showing how a net of these model neurons can simulate a McCulloch-Pitts neuron. Hence, a Kleene-like theorem for continuous neurons would settle or help to settle the above conjecture about Rashevsky neurons.

What difficulties are there is proving a continuous analog to Kleene's theorem? I think that the principal difficulties are: 1) finding a continuous analog to the discrete sequence, and 2) dealing with memory within neurons rather than nets. If we look back at Kleene's theorem, we see that it involves very simple sets of sequences, that is, sets with at most one sequence with length at most one. These simple sets are enough because each neuron (without a loop) can only remember for one time instant. Continuous neurons, on the other hand, can use their potential infinity of states to remember for arbitrary amounts of time. Of course, most continuous

models include some sort of, usually exponential, decay after a finite time span. If this decay is rapid enough then it might not be unreasonable to view even a continuous neuron as having only a finite extent memory. One is then tempted to consider functions which are constant for intervals of time corresponding to the memory time of neurons, as the continuous analogs of input sequences. Unfortunately functions which are not constant seem to be reasonable as inputs for continuous neural nets. For example, a linearly increasing function might be used as an input for a continuous net which does contrast enhancement or discrimination. While some variability seems necessary, it seems unreasonable to allow variation which makes the function nonintegerable. Discrete time enforces a sequence of time intervals; it is unclear how or if one should try to impose time intervals on input functions for continuous nets.

Kleene's theorem involves operators as well as sets of sequences. These operators — union, concatenation, and star — correspond to the three basic ways neurons can be connected. Union corresponds to parallel connection. Concatenation corresponds to serial connection. Star corresponds to neurons connected in a loop. One would then expect the analog of these three operators to appear in the continuous analog of Kleene's theorem. Of course, one expects that these operators have to be suitably generalized to deal with continuous functions rather than sequences.

So far in this section we have discussed continuous state, continuous time neuronal models, but recently some researchers have been using continuous state, discrete time neurons (Rumelhart et al (1986)). The essential feature of these models is the graded output. Instead of the output of a neuron being 0 or 1 as in the finite case or in the continuous case with a threshold nonlinearity, these models have an output in the interval [0,1]. The step function threshold nonlinearity is replaced by a continuous S-shaped function. The graded output of these models is used in deciding how to modify connections between neurons during the learning phase when these neuronal nets are being "taught" their desired behavior. After learning it seems possible to replace the S-shaped output with the step nonlinearity without affecting the behavior of the net. As in the continuous state, continuous time case,

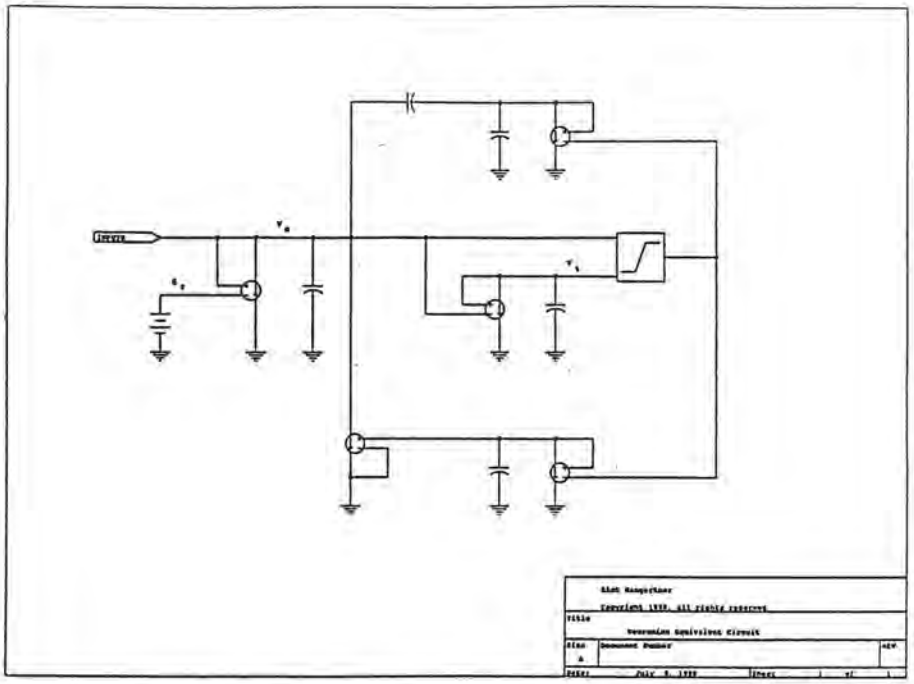


Figure 5: Circuit for a neuromime suitable for VLSI implementation.

no analog of Kleene's theorem has been proved for continuous state, discrete time nets. Discrete time might allow one to make a theorem in terms of input sequences, but these nets are often dealt with in an asynchronous manner so that some neurons have their states changed before the new states of other neurons are calculated. This asynchrony will make it difficult to prove a Kleene-like theorem.

The nets mentioned in the last paragraph arise because their creators have to simulate their nets on digital computers. From the definitional point of view, these nets are continuous state, continuous time; present technology forces the simulations to be discrete time. Changes in the technology seem possible. In particular, the work of Mead(1989) suggests that it is possible to not only design continuous state, continuous time nets, but also to fabricate silicon chips based on these designs and to build analog neural net computers. Following this line of reasoning one of my students, Hangartner(1990), has designed a chip suitable for VLSI implementation. Figure 5 gives a circuit diagram of the basic neuromime used in his design. A chip based on this design is being fabricated this summer and Hangartner expects to start experimentation on his chip this fall.

I have already mentioned two reasons why people are studying continuous state, continuous time nets. One reason is that relatively small nets seem to be able to capture some interesting behavior. Another reason is that better learning algorithms might be possible for such nets. Yet a third reason is the hope by some, for example Aleksander(1989), that analog nets can circumvent NP-completeness. In section 3, NP-completeness was defined and it was pointed out that many people believe that any digital computer program for such problems will have exponential or at least superpolynomial running time in the size of the problem. If one considers nets whose size can grow as a function of the input size, then this digital computer conjecture becomes that the product of the size times the depth of a discrete time, finite state net which solves an NP-complete problem must be superpolynomial. The advocates of analog nets argue that their nets will have an infinity of states so that they may be able to solve NP-complete problems quickly without violating the digital computer conjecture. Opponents of analog nets point out that one is only able to measure inputs and outputs to some fixed number of digits of accuracy, so that one should be able to simulate analog nets digitally, and hence that analog nets will be no more powerful computationally than digital nets. While from a theoretical point of view, I agree with the opponents of analog nets, from a practical point of view I believe that it might be possible for analog nets to quickly compute at least locally optimal solutions to hard problems. My practical belief comes from the ability of soap bubbles to quickly find locally optimal solutions to the Steiner tree problem, which is an NP-complete problem.

The next few years should be an exciting time for analog nets. We can look forward to hardware implementations which may demonstrate the practical usefulness of analog nets. We can also look forward to a firm theoretical basis for the abilities and limitations of analog nets.

6 Conclusion

To keep this paper within bounds I have ignored several aspects of neural nets. From my own work I have left out work on control sequences for neural nets and related problems on sequential machines and digraphs, but I will cite some references to this work (Cull (1974, 1975, 1976, 1977, 1978, 1980, 1982)).

I have also ignored learning for a few reasons. First, I have not directly published any work on learning. Second, following an argument of Minsky and Papert (1969) learning is a special case of nets with inputs. Their argument is that because of there are only a finite number of functions which can be computed by a net, and because weights between neurons are only going to be represented to finite accuracy, weights can be stored and modified in an extra neural net, and by slightly modifying the original net and adding the extra net, this augmented neural net will mimic a net with learning without a separate learning process, so that, learning will only be a special case of a net with inputs. Third, since the seminal work of Valiant (1984), and the subsequent COLT(Conference on Learning Theory) meetings, the theory of learning has blossomed, and would require a book length treatise to include the known results.

I have concentrated on topics in which I have worked and on topics which I think are important. The examples of complexity are not meant to encourage more such examples, but rather to warn that almost all of the things one would like to do with neural nets face the problem of computational intractability. The other sections suggest some possible ways to avoid intractability and suggest, I hope, some reasonable open problems.

Even if calculating the dynamics of individual nets is hard, it may not be hard to compute the expected or average behavior of random nets. I hope the techniques I described in section 2 may be used to calculate expected behavior for various classes of neural nets.

The ideas from chaos theory might also give a way to describe neural nets without becoming computationally bogged down. Unfortunately there are still some

open questions about the proper definition of chaos for finite state systems. If the right definitions can be found, then it may be possible to use the techniques of section 2 to calculate the expected behavior of random nets in the sense of being able to predict that chaos is or is not expected and in the sense of estimating some parameters like dimension of attractors and Lyapunov exponents.

For analog nets the theory of chaos will be directly applicable and we can expect measurements on both simulated and constructed nets within the next few years. On the other hand, characterization theorems for analog nets are needed. Otherwise we will continue to see many different models with no clear way to compare their abilities and limitations.

In the end, I find working on and thinking about neural nets is both fun and exciting. I will continue working on them. Whether neural nets will serve to solve the problems about thinking addressed by Penrose (1989) is beside the point. What is important is that we can have a good set of open problems, and a set of techniques which may allow us to solve some of these problems.

References

[1] I. Aleksander. Why neural computing? a personal view. In I. Aleksander, editor, *Neural Computing Architectures*, pages 1–7. MIT Press, Cambridge, MA., 1989.

[2] P. Berge et al. *Order within Chaos.* John Wiley & Sons, New York, 1987.

[3] A. W. Burks and J. B. Wright. Theory of logical nets. *Proc IRE*, 41:1357–1365, 1953.

[4] E. R. Caianiello. Outline of a theory of thought processes and thinking machines. *Journal of Theoretical Biology*, 2:204–235, 1961.

[5] E. R. Caianiello. Some remarks on the tensorial linearization of general and linearly separable boolean functions. *Kybernetik*, 12:90–93, 1973.

[6] E. R. Caianiello and E. Grimson. Synthesis of boolean nets and time behavior of a general mathematical neuron. *Biological Cybernetics*, 18:111–117, 1975.

[7] E. R. Caianiello, A. De Luca, and L. M. Ricciardi. Neural networks and reverberations. *Kybernetik*, 4:10–18, 1967.

[8] S. A. Cook. The complexity of theorem-proving procedures. *Proc 3rd ACM STOC*, pages 151–158, 1971.

[9] P. Cull. General two factor models. *Bulletin of Mathematical Biophysics*, 29:405, 1967.

[10] P. Cull. Linear analysis of switching nets. *Kybernetik*, 9:31–39, 1971.

[11] P. Cull. Linearization and control of switching nets. *Proc 4th Hawaii International Conference on System Sciences*, pages 537–539, 1971.

[12] P. Cull. Control of switching nets. *Biological Cybernetics*, 19:137–145, 1974.

[13] P. Cull. Touring sequential machines. *Proc 8th Hawaii International Conference on System Sciences*, pages 140–141, 1975.

[14] P. Cull. Touring machines and digraphs. *Proc Northwest 76 ACM/CIPS Pacific Regional Symposium*, pages 204–208, 1976.

[15] P. Cull. Short tours of sequential machines. In J. Traub, editor, *Algorithms and Complexity*, page 447. Academic Press, New York, 1977.

[16] P. Cull. A matrix algebra for neural nets. In G. Klir, editor, *Applied General Systems Research*, pages 563–573. Plenum, New York, 1978.

[17] P. Cull. Randomly connected neural nets. *AAAS 144th National Meeting Abstracts*, page 194, 1978.

[18] P. Cull. Tours of graphs, digraphs and sequential machines. Technical Report 78–20–1, Department of Computer Science, Oregon State University, 1978.

[19] P. Cull. Tours of graphs, digraphs and sequential machines. *IEEE Transactions on Computers*, 29:50–54, 1980.

[20] P. Cull. Hamiltonian circuits in additive machines. Technical Report 82–20–2, Department of Computer Science, Oregon State University, 1982.

[21] P. Cull. Dynamics of random neural nets. *Proceedings of the Pacific Division, AAAS*, 7:26, 1988.

[22] P. Cull, J. Holloway, and L. Kaul. Cycles, chaos, and randomness in random neural nets. *AAAS Annual Meeting Abstracts*, page 179, 1989.

[23] A. DeLuca. On some representations of boolean functions. application to the theory of switching elelments nets. *Kybernetik*, 9:1–10, 1971.

[24] B. Elspas. The theory of autonomous linear sequential networks. *IRE Transactions*, CT-6:45–60, 1959.

[25] R. FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, 1:445–466, 1961.

[26] W. Freeman. If chaos in brains is for real, what is it for? *Proceedings of the Pacific Division, AAAS*, 7:28–29, 1988.

[27] M. R. Garey and D. S. Johnson. *Computers and Intractability.* W. H. Freeman, New York, 1979.

[28] A. E. Gelfand. A behavioral summary for completely random nets. *Bulletin of Mathematical Biology*, 44:309–320, 1982.

[29] A. L. Goldberger. Fractals, chaos and sudden cardiac death. *AAAS Annual Meeting Abstracts*, page 4, 1989.

[30] R. Hangartner. An analog neuromine suitable for vlsa implementation. Technical Report 90–20–1, Department of Computer Science, Oregon State University, 1990.

[31] A. V. Hill. Excitation and accommodation in nerve. *Proc. Roy. Soc. London, Series B*, 119:305, 1936.

[32] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117:500–544, 1952.

[33] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison–Wesley, Reading, Massachusetts, 1974.

[34] A. S. Householder and H. D. Landahl. *Mathematical biophysics of the central nervous system.* Principia Press, Bloomington, Indiana, 1945.

[35] S. A. Kauffman. Metabolic stability and epigenesis in randomly connected gentic nets. *Journal Theoretical Biology*, 22:437–467, 1969.

[36] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, Princeton, New Jersey, 1956.

[37] H. D. Landahl and R. Runge. Outline of a matrix calculus for neural nets. *Bulletin of Mathematical Biophysics*, 8:75–81, 1946.

[38] T.Y. Li and J. A. Yorke. Perioid three implies chaos. *American Mathematical Monthly*, 82:985–992, 1975.

[39] R. M. May. Simple mathematical models with very complicated dynamics. *Nature*, 261:459–467, 1976.

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

[41] C. A. Mead. *Analog VLSI and neural systems.* Addison–Wesley, Reading, Massachusetts, 1989.

[42] M. Minsky. *Computation: Finite and Infinite Machines.* Prentice–Hall, Englewood Cliffs, New Jersey, 1967.

[43] M. Minsky and S. Papert. *Perceptrons.* MIT Press, Cambridge, Massachusetts, 1969.

[44] J. S. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proc IRE*, 50:2060–2070, 1962.

[45] R. Penrose. *The Emperor's New Mind.* Oxford University Press, Oxford, 1989.

[46] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 1959.

[47] N. Rashevsky. Outline of a physico–mathematical theory of excitation and inhibition. *Protoplasma*, 20:42, 1933.

[48] N. Rashevsky. *Mathematical Biophysics.* Dover, New York, 1960.

[49] H. Rubin and R. Sitgreaves. Probability distributions related to random transformations on a finite set. Technical Report 19A, Applied Math. and Stat. Lab., Stanford University, 1954.

[50] D. E. Rumelhart et al. *Parallel distributed processing.* MIT Press, Cambridge, Massachusetts, 1986.

[51] L. J. Stockmeyer. *The complexity of decision problems in automata theory and logic.* Doctoral thesis, Department of Electrical Engineering, MIT, Cambridge, Massachusetts, 1974.

[52] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. *Proc 5th ACM STOC*, pages 1–9, 1973.

[53] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

[54] J. Von Neumann. First draft of a report on the EDVAC. Technical report, Moore School of Electrical Engineering, University of Pennsylvania, 1945.

[55] C. C. Walker. Beyond the binary case in random nets. *Bulletin of Mathematical Biology*, 46:845–857, 1984.