1991

An examination and analysis of the Boltzmann machine, its mean field theory approximation, and learning algorithm

Vincent Clive Phillips
Edith Cowan University

Edith Cowan University

Copyright Warning

An Examination and Analysis of the Boltzmann Machine, its Mean Field Theory Approximation, and Learning Algorithm

By
Vincent Clive Phillips

A Thesis
Submitted to the Faculty of Science and Technology
Edith Cowan University
Perth, Western Australia

In Partial Fulfilment of the Requirements for the Degree
of
Bachelor of Applied Science (Information Science) Honours

December, 1991

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

ABSTRACT

It is currently believed that artificial neural network models may form the basis for intelligent computational devices. The Boltzmann Machine belongs to the class of recursive artificial neural networks and uses a supervised learning algorithm to learn the mapping between input vectors and desired outputs. This study examines the parameters that influence the performance of the Boltzmann Machine learning algorithm. Improving the performance of the algorithm through the use of a *naive mean field theory approximation* is also examined.

The study was initiated to examine the hypothesis that the Boltzmann Machine learning algorithm, when used with the mean field approximation, is an efficient, reliable, and flexible model of machine learning. An empirical analysis of the performance of the algorithm supports this hypothesis.

The performance of the algorithm is investigated by applying it to training the Boltzmann Machine, and its mean field approximation, the exclusive-Or function. Simulation results suggest that the mean field theory approximation learns faster than the Boltzmann Machine, and shows better stability. The size of the network and the learning rate were found to have considerable impact upon the performance of the algorithm, especially in the case of the mean field theory approximation.

A comparison is made with the feed forward back propagation paradigm and it is found that the back propagation network learns the exclusive-Or function eight times faster than the mean field approximation. However, the mean field approximation demonstrated better reliability and stability. Because the mean field approximation is local and asynchronous it has an advantage over back propagation with regard to a parallel implementation.

The mean field approximation is domain independent and structurally flexible. These features make the network suitable for use with a structural adaption algorithm, allowing the network to modify its architecture in response to the external environment.

DECLARATION

"I certify that this thesis does not incorporate, without acknowledgment, any material previously submitted for a degree or diploma in any institution of higher education and that, to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where due reference is made in the text".

Signature:

Date: 13/2/92

ACKNOWLEDGMENT

I wish to acknowledge Dr J.W.L.Millar for his help and guidance, Stephan Bettermann for his constructive criticisms and flying skills, and the support of the Department of Computer Science, Edith Cowan University. Most of all I would like to thank my wife, Bridget, for the help, love, understanding, and patience that she has shown throughout the preparation of this manuscript.

LIST OF FIGURES

Figure 2.1. The synaptic architecture of a Hopfield Network with three neurons. 15

Figure 2.2. Simplified potential energy landscape represented as a two-dimensional surface. 20

Figure 3.1. Activation probabilities as a function of temperature, showing phase transition point for the mean field approximation and the Boltzmann Machine equivalent. 47

Figure 3.2. Layered architecture resulting from removal of hidden-hidden synapses. 53

Figure 4.1. Network architecture for simulations using four neurons. 63

Figure 4.2. Network architecture for simulations using five neurons. 63

Figure 4.3. Network architecture for simulations using six neurons. 63

Figure 5.1. Average number of learning cycles required to learn the exclusive-Or function for the Boltzmann Machine and the mean-field theory approximation using three different sizes of network. 72

Figure 5.2. Average number of learning cycles required to learn exclusive-Or function for the mean field approximation for different learning rates, network sizes, and showing deviation from mean performance. 73

Figure 5.3. Average number of learning cycles required to learn exclusive-Or function for the Boltzmann Machine for different learning rates, network sizes, and showing deviation from mean performance. 74

Figure 5.4. Behaviour of the Boltzmann Machine using five neurons and a fixed learning rate. 75

Figure 5.5. Behaviour of the Boltzmann Machine using five neurons and dynamically reducing the learning rate after discrete time intervals. 76

Figure 5.6. Average number of learning cycles required to learn exclusive-Or function for the Boltzmann Machine using dynamically reducing learning rates, two network sizes, and showing deviation from mean performance. . . 77

LIST OF TABLES

Table 3.1. *Performance of the Boltzmann Machine applied to the encoder problem.* 38

Table 4.1. *Annealing schedule used to learn the exclusive-Or for the Boltzmann Machine and the mean field approximation.* 60

Table 4.2. *Values used for the learning rate (ε) during the simulations.* 61

Table 4.3. *Truth-table for the boolean exclusive-Or function.* 65

Table 5.1. *Number of cycles required by the Boltzmann Machine and mean field approximation using four neurons and single multiple estimated gradient.* 71

Table 5.2. *Performance ratios for the Boltzmann Machine compared to the mean-field theory approximation for different network sizes.* 78

TABLE OF CONTENTS

Abstract ii

Declaration iv

Acknowledgment v

List of Figures vi

List of Tables viii

Section 1: Introduction

1.1 Connectionist Modelling 1

1.1.1 Rationale 1

1.1.2 Artificial Neural Networks 2

1.1.3 Potential 2

1.2 The Boltzmann Machine 3

1.2.1 Significance 3

1.2.2 Limitations of the Network 3

1.2.3 Limitations of the Learning Algorithm 4

1.3 The Approach of the Study 4

1.3.1 Objectives 4

1.3.2 Focus 5

1.3.3 Methodology 5

1.3.4 Suitability 6

1.4 Scope of the Study 6

1.4.1 Specific Problems 6

1.4.2 Limitations 7

1.5 Structure of the Report 8

Section 2: Theoretical Framework

2.1 Artificial Neural Networks 9

2.1.1 Definition of a Neural Network 9

2.1.2 Classifying the Boltzmann Machine 11

2.2 The Boltzmann Machine 12

2.2.1 Origins of the Boltzmann Machine 12

2.2.2 The Hopfield Network 14

2.2.3 Comparisons with Biological Memory 19

2.2.4 Problems With the Hopfield Network 21

2.2.5 Definition of the Boltzmann Machine 22

2.3 Boltzmann Machine Adaption 26

2.3.1 Modelling the External Environment 26

2.3.2 Traversing G-Space 27

2.3.3 The Boltzmann Machine Learning Algorithm 28

2.4 A Mean Field Theory Approximation 29

2.4.1 Approximating the Stochastic State 30

2.4.2 Validity of the Approximation 32

2.4.3 Approximating the Boltzmann Machine Algorithm 32

2.4.4 Advantages of the Approximation 33

Section 3: Review of the Literature

3.1 Introduction 34

3.1.1 Resolving Problems with the Learning Algorithm 35

3.1.2 Major Works Reviewed 36

3.1.3 Measuring Network Performance 36

3.2 Performance of the Boltzmann Machine 37

3.2.1 Encoding and Communicating Information 37

3.2.2 The Relationship Between Two Input Vectors 38

3.3 Reaching Equilibrium Quickly 39

3.3.1 Performance of the Mean Field Approximation 40

3.4 Controlling the Learning Process 41

3.4.1 The Learning Rate 42

3.4.2 The Annealing Schedule 44

3.4.3 Binary and Bipolar Representation 48

3.4.4 Controlling Weight Growth 49

3.4.5 The Network Architecture 51

3.5 Comparisons with Back Propagation 54

Section 4: Experimental Procedures

4.1 Introduction 56

4.2 Experimental Hypotheses 57

4.3 The Simulations 58

4.3.1 Software Implementation 58

4.3.2 External Variables 58

4.3.3 Internal Variables 60

4.4 The Exclusive-Or Function 64

4.4.1 The Exclusive-Or Truth-table 65

4.4.2 Advantages 66

4.4.3 Previous Investigations 67

4.5 Generating Performance Information 67

4.5.1 Data Produced by the Simulations 67

4.5.2 Comparing the Performance of the Networks 69

Section 5: Results of the Simulations

5.1 Introduction 70

5.2 Experiment One - Speed of Learning 71

5.3 Experiment Two - Increasing Redundancy 72

5.4 Experiment Three - Increasing the Learning Rate 73

5.5 Experiment Four - Dynamic Learning Rate 75

5.6 Experiment Five - Back Propagation 76

5.7 Interpretation of the Data 77

5.7.1 Performance of the Learning Algorithm 77

5.7.2 Increasing Network Size 78

5.7.3 Increasing the Learning Rate 79

5.7.4 Dynamically Reducing the Learning Rate 79

5.7.5 Comparisons with Back Propagation 80

5.8 Conclusions 81

Section 6: Summary

6.1 Generalisation of Results 83

6.1.1 Objectives 83

6.1.2 Generalised Conclusions 84

6.1.3 Implications 85

6.2 Limitations of the Study 85

6.3 Future Research Directions 86

6.3.1 Structural Adaption 86

6.3.2 Scalability 86

6.3.3 Parameter Settings 87

6.3.4 Synaptic Modelling 87

6.4 Conclusions 88

Section 7: Bibliography 89

Section 1: Introduction

1.1 Connectionist Modelling

1.1.1 Rationale

The rationale for connectionist modelling is the belief that massively parallel machines will lead to useful, and intelligent, emergent behaviour. Albus declares "that a sensory-interactive, goal-directed motor system is not simply an appendage to the intellect, but is rather the substrate in which intelligence evolved" (cited in Hampson, 1990, p. 11). It is a common thesis that a connectionist model is suitable for such a substrate.

Connectionist models are called *artificial neural networks*. The goal is to emulate the "low level signal processing mechanisms and organization [*sic*] of the brain to try to incorporate these mechanisms into the design of our next generation artifacts [*sic*]" (Lee, 1991, p. 6). Artificial neural networks exhibit many desirable properties of biological systems - parallel computation, graceful degradation, distributed knowledge representation, and the ability to learn (Lee, 1991, p. 4).

1.1.2 Artificial Neural Networks

Artificial neural networks[1] are similar to cellular automata and fuzzy logics (Narendra and Thathachar, 1989, p. 6; Kosko, 1992, p. 7). They contain simple processing elements that model the behaviour of biological neurons. Each processing element communicates with neighbouring elements through a weighted connection. The properties of neural networks emerge from interactions between large numbers of these elements (Hopfield, 1984).

1.1.3 Potential

The ability to adapt to an environment is the most important property of neural networks (Lee, 1991, p. 13). Current learning algorithms are crude and allow only *parameter adaption*, i.e., the architecture of the network is determined prior to learning (Peterson and Hartman, 1989, p. 16; Lee, 1991, p. 14). There is a trend towards the development of learning methods capable of parameter and *structural* adaption.[2] Research in this area has the potential of producing intelligent computing elements that are able to fully adapt to any external environment (Lee, 1991).

[1] The term *neural network* is used to refer to connectionist models, references to biological neural systems will be clearly identified as such.

[2] *Structural adaption* is the alteration of the network's architecture in response to the current environment.

1.2 The Boltzmann Machine

The *Boltzmann Machine* belongs to an important class of neural network models. Its dynamic behaviour is "ruled by an energy function which decreases monotonically" (Kamp and Hasler, 1990, p. xi). The resulting behaviour, termed *relaxation*, is used to create content-addressable memories and solve constraint satisfaction problems (Hinton, Sejnowski, and Ackley, 1984, p. 2).

1.2.1 Significance

The Boltzmann Machine is of commercial and theoretical significance (Miller, Walker, and Ryan, 1990, p. 299; Hecht-Nielsen, 1990, p. 195). Practical applications include parsing context-free grammars and image processing systems that reflect "human performance nicely" (Zeidenberg, 1990, p. 187).

1.2.2 Limitations of the Network

The computational cost of simulating the Boltzmann Machine has restricted its use to small problem domains (Hinton, 1990a, p. 21; Hecht-Nielsen, 1990). The development of a *naive mean field theory approximation*[3] of the Boltzmann Machine promises to improve the speed and quality of learning (Peterson and Anderson, 1987).

[3] Referred to as the *mean field approximation*.

1.2.3 Limitations of the Learning Algorithm

The Boltzmann Machine learning algorithm is domain independent (Hinton et al., 1984, p. 6) and controls the parameter adaption of the neural network; it requires domain dependent information to specify the architecture of the network. Structural adaption requires global knowledge (Lee, 1991) - the Boltzmann Machine learning algorithm operates upon local knowledge[4] (Hinton et al., 1984, p. 8) and cannot incorporate structural adaption.

1.3 The Approach of the Study

1.3.1 Objectives

This study resulted from an empirical investigation of the Boltzmann Machine, its mean field approximation, and the Boltzmann Machine learning algorithm.[5] It was initiated to test the hypothesis that the Boltzmann Machine learning algorithm, when used with the mean field theory approximation, is efficient, reliable, and flexible.

Efficient. This is used in reference to the speed of the learning. A comparative measure of learning speed is developed in Section Four.

---

[4] Global knowledge concerns the entire network, local knowledge is knowledge only of a single processing element and its related connections.

[5] *Boltzmann Machine learning algorithm* also refers to the mean field theory approximation learning algorithm.

Reliable. The ability of the algorithm to consistently succeed with little deviation from average learning times is important for the construction of large applications. These properties suggest that the algorithm is stable and predictable.

Flexible. The algorithm should be robust against a number of constraints, including, (a) the rate of parameter adaption, (b) the architecture of the network, and (c) the amount of noise in the network.

1.3.2 Focus

The study focuses upon the effects of changing (a) the size of the network, and (b) the rate of parameter adaption[6] upon the learning ability of the Boltzmann Machine learning algorithm. The algorithm was required to train a neural network the *exclusive-Or* function (see 4.4). The exclusive-Or function is a theoretically significant problem used by Minsky and Papert (1969) to examine the deficiencies of neural networks. This problem is recognised in the literature as an indicator of more general problem solving abilities.

1.3.3 Methodology

Several hypotheses provided direction for a series of simulations of the Boltzmann Machine learning algorithm (see 4.2). Data resulting from these

[6] The rate of learning.

simulations were examined and compared using a simple performance metric. A comparison was also made with the *back propagation* model. Back propagation was selected as a comparative model because, (a) the model is extensively discussed in the literature, and (b) similar comparative studies have been conducted (Peterson and Hartman, 1989).

1.3.4 Suitability

There is an emphasis towards experimental investigation of artificial learning algorithms (Shavlik and Dietterich, 1990, p. 6). The emergent properties of neural networks makes it difficult to investigate their behaviour formally; empirical studies may provide insights into their limitations and suggest future research directions (Kibler and Langley, 1988, p. 5).

1.4 Scope of the Study

1.4.1 Specific Problems

The study sought answers to four questions:

1. What is the performance of the Boltzmann Machine learning algorithm, and how does it compare to the back propagation paradigm?

2. What is the impact of the size of the network and the rate of adaption upon the Boltzmann Machine learning algorithm?

3. What optimisation can be made to the Boltzmann Machine learning algorithm?

4. How flexible is the Boltzmann Machine learning algorithm - can it train networks of dynamic architecture?

1.4.2 Limitations

The study was delimited by:

1. The size of the problem used to test the learning algorithm. The exclusive-Or is a small problem restricting the generality of results.

2. The environment used to *implement* the simulations required that the network models were implemented as *serial* simulations. The Boltzmann Machine is an asynchronous *parallel* machine, it is assumed that serial simulation is valid for such a machine.

The application of the learning algorithm to a number of domains would be a valuable expansion to the study, especially if such an expansion concentrated upon the issue of scaling to large problem domains. This was beyond the scope of the study.

1.5 Structure of the Report

This thesis is presented in three parts:

1. A theoretical formulation and review of important empirical investigations is developed in Sections Two and Three.

2. A description of the experimental methodology, the importance of the exclusive-Or function, and the hypotheses to be tested is developed in Sections Four and Five.

3. A summary of the results and the identification of extensions to the research is contained in Section Six.

Section 2: Theoretical Framework

2.1 Artificial Neural Networks

2.1.1 Definition of a Neural Network

The architecture of a neural network describes a directed graph with the following properties (Müller and Reinhardt, 1990, p. 12):

1. A state variable n_i, associated with each node i.

2. A weight w_{ik}, associated with each link (ik) from node i to k.

3. A bias ϑ_i, associated with each node i.

4. A transfer function $f_i\ [n_k, w_{ik}, \vartheta_i, (k \neq i)]$, associated with each node i, determining the state of the node as a function of (a) its bias, (b) the weights of its incoming links, and (c) the states of the nodes connected to it by these links.

To emphasise the relationship with biological systems each node in the network is called a *neuron*, the links between neurons are called *synapses*, the bias term is called the *activation threshold*, and the transfer function is called the *activation function*.

There are two classes of neuron:

- *Hidden* neurons that cannot *communicate* with the external environment. They receive synapses only from other neurons in the network.
- *Visible* neurons that communicate directly with the external environment. A visible neuron may act as an input device, an output device, or both.

The *external environment* can be any external entity, including another neural network, that communicates with the visible neurons. The task of a supervised learning algorithm is to model the probability distribution of the external environment (Hinton et al., 1984, p. 6). The network's model of the environment is then contained in the weights and location of the synapses.

2.1.2 Classifying the Boltzmann Machine

There are many methods for classifying neural network models, the simplest is to classify each model by its synaptic architecture and method of training. The Boltzmann Machine has the following characteristics:

- Its architecture contains closed synaptic loops and is *recursive* because the output signals of its neurons *feed back* as additional inputs (Kosko, 1992; Kamp and Hasler, 1990). A recursive neural network can be described as a directed cyclic graph.

- Learning occurs through an external supervisor and requires training data consisting of input - output vector pairs. The difference between the desired output and actual output guides the network through a gradient descent of the space of possible synaptic strengths. Supervised learning is equivalent to "descriminant analysis" in statistics (Diederich, 1990, p. 3).

2.2 The Boltzmann Machine

2.2.1 Origins of the Boltzmann Machine

In 1954 B.G.Cragg and N.V.Temperley compared the behaviour of lattices of atoms, or binary alloys (called *Ising models*), to fully connected networks of neurons (Cowan and Sharp, 1988, p. 13). Using this analogy they came to two conclusions (Cowan and Sharp, 1988, p. 14):

1. "Domain patterns that are a ubiquitous feature of ferromagnets, comprising patches of up [+1] or down [-1] spins, should show up in neural nets as patches of excited or quiescent neurons."

2. "Neural domain patterns, once triggered by external stimuli, would be stable against spontaneous random activity and therefore constitute a memory of the stimulus."

These conclusions were followed by the work of Sherrington and Kirkpatrick in 1975, who described a new magnetic material they called a *spin glass*. This material consists of a mixture of "ferromagnetically and antiferromagnetically interacting spins and exhibiting no net magnetism"; its properties include the ability to "store many different disordered spin patterns" (Cowan and Sharp, 1988, p. 14).

The relationship between Ising models, spin glasses, and networks of neuron-like processing elements, led Hopfield (Hopfield, 1982) to describe the *Hopfield* network. The network contains multiple McCulloch-Pitts[7] neurons with random, symmetric synaptic couplings. Hopfield described the use of Hebb's[8] learning law to set the synaptic weights, thus creating locally stable states. This work is regarded as seminal to modern connectionist models and documents two important properties of a recurrent neural network:

1. Such a network has stable states that can always be found by the network when it is started in a random state and allowed to evolve dynamically.

2. Stable states can be created and removed by changing the strengths of the synaptic couplings.

[7] McCulloch-Pitts neurons are named after W.McCulloch and W.Pitts. They are also called *linear threshold units* because they compute the total input from other neurons and activate if this value is greater than their activation threshold (Müller and Reinhardt, 1990, p.13).

[8] Hebb's learning law is named after D.Hebb. Hebb postulated that the strength of a synapse, i.e., its weight, can be adjusted if the level of activity between adjacent neurons changes. This is known as *synaptic plasticity* and is implemented by the reinforcement of synapses between neurons behaving correctly (Müller and Reinhardt, 1990, p.6).

2.2.2 *The Hopfield Network*

The Hopfield network (Hopfield, 1982, 1984) is a fully connected, symmetrically weighted neural network (see Figure 2.1). The network can be described at time t by the state vector $S(t)$, equation (2.1), the synaptic matrix W, equation (2.2), and the activation threshold vector ϑ, equation (2.3).

$$S(t) = \begin{bmatrix} s_1(t) \\ s_2(t) \\ \vdots \\ s_n(t) \end{bmatrix}, \quad s_i(t) \in \{-1, +1\} \tag{2.1}$$

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{bmatrix}, \quad w_{ij} \in \mathbb{R},\ w_{ij} = w_{ji} \ \textit{and} \ w_{ii} = 0 \tag{2.2}$$

$$\vartheta = \begin{bmatrix} \vartheta_1 \\ \vartheta_2 \\ \vdots \\ \vartheta_n \end{bmatrix}, \quad \vartheta_i \in \mathbb{R} \tag{2.3}$$

The Hopfield network contains only visible neurons, each neuron acting as an input *and* output device. During learning the environmental pattern is mapped to the state vector *S(t)* and the network allowed to compute a suitable synaptic matrix. Patterns are recalled by mapping an incomplete, or noisy, version to *S(t)* and the network allowed to dynamically evolve; thus acting as a pattern completion, or pattern correction, device.

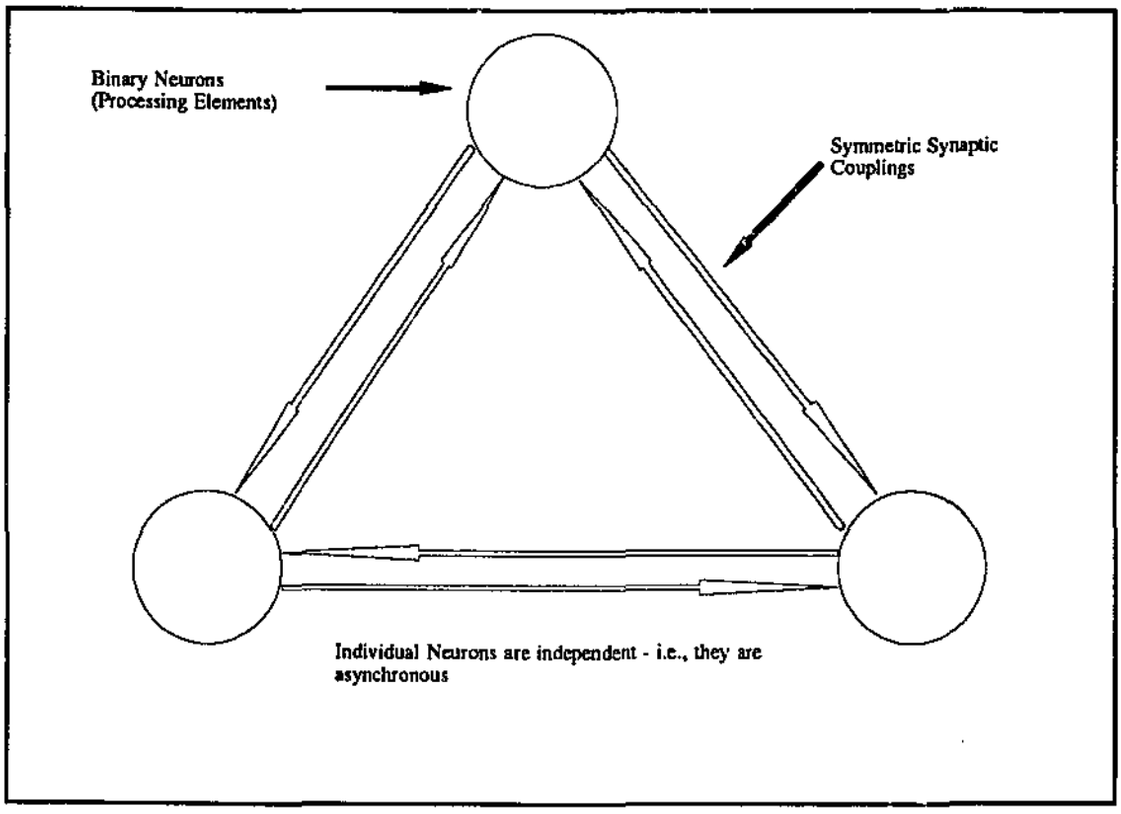

Figure 2.1. The synaptic architecture of a Hopfield Network with three neurons.

Dynamic evolution of network state. During computation the Hopfield network evolves dynamically in discrete time steps.[9] At each time step each neuron calculates the *synaptic potential* that defines its total input, this is shown in equation (2.4) (Kamp and Hasler, 1990; Müller and Reinhardt, 1990).

$$h_i(t) = \sum_{j=1}^{n} w_{ij} s_j(t) \tag{2.4}$$

The activation function, shown in equation (2.5), for each neuron is simply a comparison between the synaptic potential (h_i) and the activation threshold (ϑ_i), i.e., they are linear threshold units; if the activation function evaluates to zero the state of the neuron does not change.

$$s_i(t+1) = sgn\left[h_i(t) - \vartheta_i\right] \equiv sgn\left[\sum_{j=1}^{n} w_{ij} s_j(t) - \vartheta_i\right] \tag{2.5}$$

This is known as the *Heaviside* step function and can be expressed in the form shown in equation (2.6) (Hopfield, 1984, p. 1).

$$\begin{aligned} s_i(t+1) &= \Theta[h_i(t)] \\ &\equiv \left\{ \begin{array}{rl} -1 \ if & \sum_{j=1}^{n} w_{ij} s_j(t) < \vartheta_i \\ s_i(t) \ if & \sum_{j=1}^{n} w_{ij} s_j(t) = \vartheta_i \\ +1 \ if & \sum_{j=1}^{n} w_{ij} s_j(t) > \vartheta_i \end{array} \right\} \end{aligned} \tag{2.6}$$

[9] Discrete evolution simulates the "finite regenerative period of real neurons." (Müller and Reinhardt, 1990, p.12)

The effect of ϑ_i can be simulated by extending the state vector and synaptic matrix to include an additional neuron. This *bias* neuron permanently has a state of -1, although its synaptic connections can be trained (Kamp and Hasler, 1990; Hinton, 1990a). This simplifies the activation function to the form shown in equation (2.7).

$$s_i(t+1) = sgn\left[h_i(t)\right] \equiv sgn\left[\sum_{j=0}^{n} w_{ij}s_j(t)\right] \tag{2.7}$$

The energy landscape. The Hopfield network is symmetric, allowing its dynamic behaviour to be modelled by an *energy*, or *Lyapunov*, function (Müller and Reinhardt, 1990). The energy (E) associated with a particular state vector $S(t)$ is calculated using equation (2.8).

$$E[S] = -\frac{1}{2}\left[\sum_{ij}^{i\neq j} w_{ij}s_i s_j\right] \tag{2.8}$$

E[S] continually decreases with time; a complete proof is offered by Kamp and Hasler (1990, p. 24), and discussed in detail by Müller and Reinhardt (1990, p. 31). The proof begins by considering that the contribution of an individual neuron can be calculated locally, as in equation (2.9).

$$E_i(t) = -s_i(t)\left[\sum_{j \neq i} w_{ij} s_j(t)\right] = -s_i(t)\, h_i(t) \tag{2.9}$$

The activation function, equation (2.7), allows this local energy function to be used to show that *E[S]* will always decline or stabilise, see equation (2.10) (Müller and Reinhardt, 1990, p. 31).

$$\begin{aligned} E_i(t+1) &= -s_i(t+1)\left[\sum_{j \neq i} w_{ij} s_j(t)\right] \\ &= -sgn\left[h_i(t)\right] h_i(t) \\ &= -\left|h_i(t)\right| \\ \Rightarrow &\leq -s_i(t)\, h_i(t) = E_i(t) \\ \therefore\ E_i(t+1) &\leq E_i(t) \end{aligned} \tag{2.10}$$

This continual reduction in the network's energy allows it to find the stable states defined by the synaptic matrix; thus it can act as a content-addressable memory. This *relaxation* process also allows the network to be used as an optimisation device, as used by Hopfield and Tank (1985) to solve the *Travelling Salesman [Salesperson] Problem.*

The Hopfield theorem. The proof offered in equation (2.10) leads to the following theorem (Kamp and Hasler, 1990, p. 24):

> If the synaptic matrix is symmetric with nonnegative [*sic*] diagonal elements, then the asynchronous operation mode of a recursive network is devoid of cycles.

This is an important result that holds for all recursive networks with the appropriate properties, including the Boltzmann Machine. There are interesting parallels between this relaxation behaviour and biological neural systems.

2.2.3 Comparisons with Biological Memory

The operation of human memory can be viewed as a relaxation process, similar to the operation of the Hopfield network (Killeen, 1989). The Lyapunov function, equation (2.8), describes *E*-space[10] - a *landscape* of potential energies with multiple local minima representing equilibrium states (see Figure 2.2). Killeen (1989) describes the operation of biological memory as follows:

[10] *Energy* space.

> The tops of each of the hills are singularities: "immeasurably small energies" applied to an object [biological system, memory] at the top will cause it to roll one way into the basin of one well, or the other way into the basin of a different well. Within a well the system is stable - the object stays in the well unless new energy is added - or the landscape is changed. (p. 4)

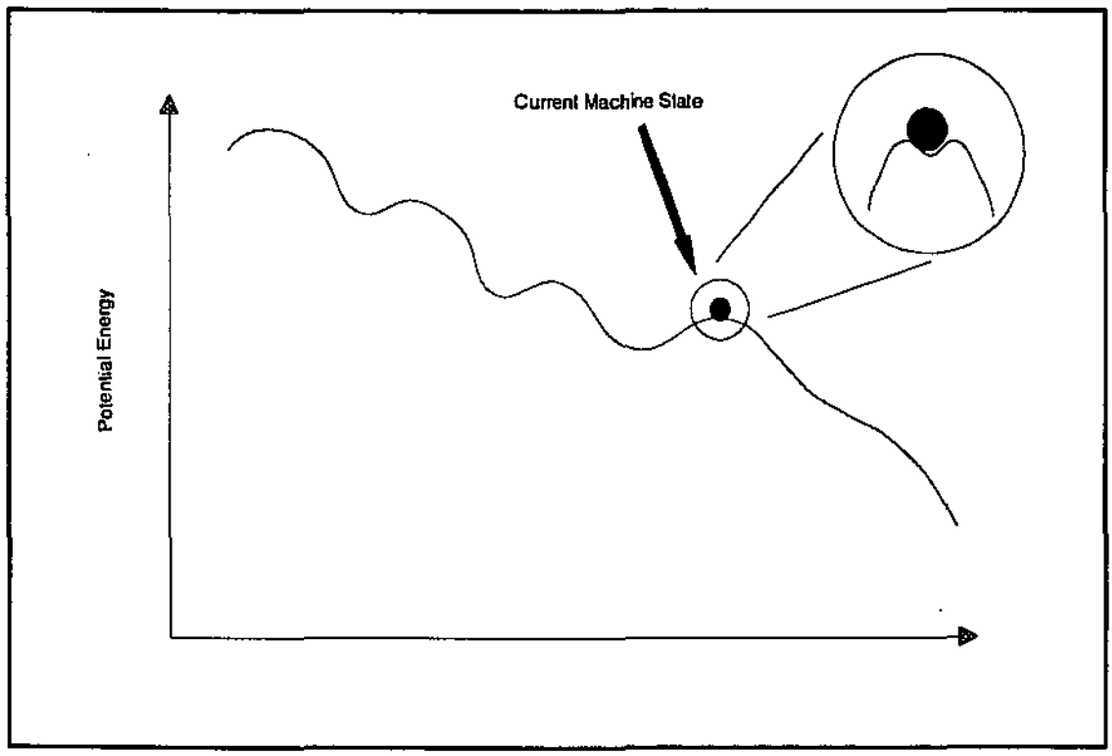

Figure 2.2. Simplified potential energy landscape represented as a two-dimensional surface.

As shown in Figure 2.2 there are multiple local minima separated by energy barriers. Hopfield (1982) uses these local minima to store environmental patterns, or *memories*.

2.2.4 *Problems With the Hopfield Network*

The Hopfield network has a very low capacity[11] and suffers from two problems (Hinton and Sejnowski, 1986; Müller and Reinhardt, 1990):

1. The Lyapunov function guarantees that the network will find local, not global, minima. Global minima are stable states that can be used to model the optimal solution to a constraint satisfaction problem (Hinton et al., 1984).

2. The learning algorithm originally suggested by Hopfield is a simple variant of Hebbian learning that cannot train hidden neurons. This means that the network cannot be used to model environments requiring the solution of three or more independent variables. This is a major criticism of connectionist paradigms (Minsky and Papert, 1969; Hinton et al., 1984).

The inability to find global minima is serious due to the occurrence of *metastable*, or *parasitic*, states. These are locally stable states that do not conform to any intended solution state. The presence of metastable states cannot be predicted and they may cause the network to behave incorrectly.

[11]. The Hopfield network can only store $\approx 0.138N$ patterns, where N is the number of neurons in the network; above this limit the capacity of the network falls away dramatically (Müller and Reinhardt, 1990, p.42).

2.2.5 Definition of the Boltzmann Machine

The architecture of the Boltzmann Machine is identical to the Hopfield network (see 2.2.2). Individual neurons can be represented as binary [0,1] or bipolar [-1,+1] elements (see 3.4.3).[12] There are two functional groups of neurons in the Boltzmann Machine:

1. Mandatory visible neurons that act as input / output devices.
2. Optional hidden neurons used to reduce the higher-order constraints present in the problem domain.

The Lyapunov function used to describe the energy of the network is the same as that used in the Hopfield network, see equation (2.8). The activation function, see equation (2.5), is replaced with a form of the *Metropolis* algorithm (Hinton et al., 1984, p. 4). The variable T acts as *temperature* for the system and defines the level of *thermal noise* (randomness) in the network.

$$f(h_i) = \left[\frac{1}{\left(1 + e^{\frac{-h_i(t)}{T}} \right)} \right] \tag{2.11}$$

[12] The Boltzmann Machine uses discrete, i.e., two-valued, states.

The activation function $f(h_i)$ is sigmoidal, being monotonic between the limits shown below:

$$\begin{aligned} &\lim_{h_i \to -\infty} f(h_i) = 0, \textit{ and} \\ &\lim_{h_i \to +\infty} f(h_i) = 1 \end{aligned} \tag{2.12}$$

An important property of the Boltzmann Machine is that the probability of a neuron changing state is independent of its current state. This means that the Boltzmann Machine can move to energy minima from any starting state, and can occasionally move to levels of higher energy. Thermal and stochastic noise supplies the momentum required to escape locally stable states (see 2.2.3).

Reaching thermal equilibrium. If the Boltzmann Machine continues to evolve at a fixed temperature the system will reach a condition called *thermal equilibrium.* Once the system is in equilibrium the probability of two global states is determined by the Boltzmann distribution; the network state depends only upon the relative energies of the available states, see equation (2.13).

$$\frac{P_\alpha}{P_\beta} = \frac{e^{\left(\frac{-E[S]_\alpha}{T}\right)}}{e^{\left(\frac{-E[S]_\beta}{T}\right)}} = e^{\frac{-(E[S]_\alpha - E[S]_\beta)}{T}} \tag{2.13}$$

Using simulated annealing. When a low temperature (T) is used the network is strongly attracted to states of low energy in a deterministic fashion, similar to the Hopfield network. But at low temperatures there may be insufficient thermal noise for the network to escape local minima, and at high temperatures all states become equiprobable. Similar to physical systems, the Boltzmann Machine makes use of *annealing*. This is simulated by starting the network at a high temperature that is reduced, gradually, until the network behaves deterministically. As T approaches zero the network becomes deterministic, behaving like $\Theta(h_i)$:

$$\lim_{T \to 0} f(h_i) = \Theta(h_i) \tag{2.14}$$

Features of the Boltzmann Machine. The Boltzmann Machine captures many properties exhibited by biological neural networks:

- Content addressability/associativity, this is inherently more efficient than traditional address based memories (Kamp and Hasler, 1990).
- Graceful degradation when subjected to localised damage and noisy environments (Hinton et al., 1984).
- The Boltzmann Machine is asynchronous making it suitable for parallel implementation. Asynchronous behaviour is considered a good approximation of biological neural systems (Müller and Reinhardt, 1990).
- The Boltzmann Machine can adapt to the external environment by changing its synaptic matrix, thus learning to compute new functions and making optimal use of its hidden neurons (Hinton et al., 1984).
- The Boltzmann Machine can operate as a pattern classification device, allowing it to generalise when presented with novel environmental patterns.

2.3 Boltzmann Machine Adaption

2.3.1 Modelling the External Environment

The ability to train hidden neurons, and to create an optimal synaptic matrix, is the result of the relationship shown in equation (2.13). The energy of the network is a linear function of the synaptic matrix, leading to a relationship, shown in equation (2.15), between the probabilities of global states and individual synaptic strengths (Hinton et al., 1984, p. 6).

$$\frac{\delta \ln P_\alpha}{\delta w_{ij}} = \frac{1}{T}\left[s_i^\alpha s_j^\beta - p_{ij} \right] \tag{2.15}$$

This relationship makes it "possible to manipulate the . . . probabilities of global states" (Hinton et al., 1984, p. 6), causing them to model the probabilities of environmental states. "The machine's model is just the probability distribution it would produce over the visible units if it were allowed to run freely without any environmental input" (Derthick, 1984, p. 1).

2.3.2 *Traversing G-Space*

The learning algorithm alters the synaptic weights to minimise the difference between the environmental $[P'(V_\alpha)]$ and the internal $[P(V_\alpha)]$ probability distributions, as measured by equation (2.16) (Hinton et al., 1984, p. 7).

$$G = \sum_{\alpha} P(V_\alpha) \ln\left(\frac{P(V_\alpha)}{P'(V_\alpha)}\right) \tag{2.16}$$

This is the information-theoretic measure of the difference between the two states, and is known as the *asymmetric divergence*, the *Kullback* measure, or the *G cost function*. G is a function of the synaptic weights and lies on a "W [number of weights] dimensional surface within the W+1 dimensional space we call *G-space*" (Derthick, 1984, p. 2). Learning occurs by finding the global minimum within *G*-space. Minimising the G measure is similar to minimising the network energy ($E[S]$, see 2.2.3) but one optimisation of G may require many optimisations of the energy function (Derthick, 1984, p. 2).

2.3.3 The Boltzmann Machine Learning Algorithm

When training the Boltzmann Machine the input neurons are *clamped* with the appropriate environmental patterns. The algorithm then proceeds in three phases (Hinton et al., 1984; Peterson, 1991):

1. The desired output pattern is clamped to the output neurons and the network relaxes by updating the unclamped neurons in a series of *learning sweeps*. After a series of sweeps the temperature is lowered according to an annealing schedule. *Co-occurrence* (simultaneous activation) statistics are collected, when the network has reached equilibrium at the final temperature, by running the network for a sampling period (⟨...⟩ represents the average state):

$$\rho_{ij} = \langle s_i s_j \rangle \tag{2.17}$$

2. The output neurons are released, and again the network relaxes by updating the unclamped neurons. When the network reaches equilibrium co-occurrence statistics are again collected:

$$\rho'_{ij} = \langle s_i s_j \rangle \tag{2.18}$$

3. After all training patterns have been applied to the network the synaptic weights are updated according to:

$$\Delta w_{ij} = \varepsilon\left(\rho_{ij} - \rho'_{ij} \right) \tag{2.19}$$

The variable ε is the learning rate of the algorithm and determines the speed of traversal in G-space.

The algorithm is applied repeatedly until changes to the synaptic weights no longer occur, or the network has reached a specified level of performance. A single *learning cycle* consists of presenting all training patterns to the network and allowing the synaptic weights to be adjusted.

2.4 A Mean Field Theory Approximation

The instantaneous state of an individual neuron in the Boltzmann Machine is not important because it is determined through noisy sampling. Of greater relevance is the *mean* activity of the neuron; the neuron's average behaviour during the entire annealing process drives the learning algorithm.

2.4.1 *Approximating the Stochastic State*

Peterson and Anderson (1987), Peterson and Hartman (1989), Peterson (1991), and Hartman (1991) have shown that the "stochastic simulated annealing process in the Boltzmann machine can be replaced by a set of deterministic equations in the so-called mean field theory approximation" (Peterson and Hartman, 1989, p. 1). The approximation can be developed by first considering the average activity of an isolated bipolar neuron whose instantaneous state can be calculated according to equation (2.11) (Müller and Reinhardt, 1990, p. 38).

$$\begin{aligned} \langle s_i \rangle &= (+1) f(h_i) + (-1) f(h_i) \\ &= \left[\frac{1 - e^{-\frac{h_i}{T}}}{1 + e^{-\frac{h_i}{T}}} \right] \\ &= \tanh\left(\frac{h_i}{T} \right) \end{aligned} \tag{2.20}$$

In a network of multiple neurons the activity of an individual neuron is determined by the synaptic potential applied to that neuron - see equation (2.4). The synaptic potential is determined by the instantaneous state of the other neurons in the network - not their average state. Because the function $f(h_i)$ is non-linear it is necessary to apply a *naive mean-field theory approximation*, as shown in equation (2.21) (Müller and Reinhardt, 1990, p. 38; Peterson and Hartman, 1989).

$$\langle f(h_i) \rangle \rightarrow f(\langle h_i \rangle) = f\left(\sum_j^n w_{ij} \langle s_j \rangle \right) \tag{2.21}$$

Using this approximation, the mean activity of a neuron in a network of multiple neurons can be computed using equation (2.22) (Müller and Reinhardt, 1990, p. 39; Peterson and Anderson, 1987).

$$\begin{aligned} \langle s_i \rangle &= \langle f(+h_i) \rangle - \langle f(-h_i) \rangle \\ &\rightarrow \tanh \left(\frac{\sum_j^n w_{ij} \langle s_j \rangle}{T} \right) \\ &= V_i \end{aligned} \tag{2.22}$$

2.4.2 *Validity of the Approximation*

The validity of this approximation depends upon the size of the network and the degree of *connectivity*.[13] For highly connected magnetic systems the approximation works very well as accuracy increases as system size grows (Peterson, 1991; Peterson and Hartman, 1989). Peterson and Anderson (1987) show that the approximation works well using as few as ten neurons with limited connectivity.

2.4.3 *Approximating the Boltzmann Machine Algorithm*

By averaging the co-occurrence statistics collected at equilibrium through the use of the mean field approximation there is no need to let the network stochastically find equilibrium. The learning algorithm then takes the form:

1. Environmental patterns are clamped and the network is annealed to its final temperature when mean activity is given by:

$$\rho_{ij} = V_i V_j \tag{2.23}$$

[13] The greatest number of synapses connected to any individual neuron in the network.

2. The output neurons are unclamped, and the network again annealed to its final temperature:

$$\rho'_{ij} = V_i V_j \tag{2.24}$$

3. Synaptic weight updating occurs as for the Boltzmann Machine learning algorithm - see equation (2.19).

2.4.4 Advantages of the Approximation

The mean field approximation provides two advantages over the original Boltzmann Machine:

1. During annealing the Boltzmann Machine requires many learning sweeps to bring the network to thermal equilibrium at each temperature step. The mean field approximation only requires one sweep at each temperature to generate an average behaviour.

2. At the final temperature co-occurrence statistics must be collected. To reduce the noise in the sample many update sweeps are necessary. The mean field approximation requires only a single sweep at the final temperature to generate this information.

This implies that the mean field approximation will learn, and operate, faster than the original Boltzmann Machine.

Section 3: Review of the Literature

3.1 Introduction

Two factors have limited investigation into the performance of the Boltzmann Machine (Hinton, 1990a, p. 21):

1. The time to reach thermal equilibrium is proportional to the size of the network. This means that it becomes costly to investigate large networks, i.e., *real-world* applications (Hartman, 1991).

2. Estimating the gradient of *G*-space is difficult, and introduces complications.

The first problem is inherent to the Boltzmann Machine and is solved by the mean field approximation (see 2.4). Controlling the traversal of *G*-space is complicated by several related problems (Hinton, 1990a, p. 21):

- If thermal equilibrium is not reached a systematic error is introduced into the gradient, eventually causing the learning algorithm to fail.

- If insufficient samples are taken at equilibrium the estimated gradient will be noisy and inaccurate.
- There is a tendency for hidden neurons to *suicide*. This takes two forms, (a) the neurons remain inactive and contribute nothing to performance, or (b) the neurons become dominant and active despite the input vector (Derthick, 1984, p. 24).

3.1.1 Resolving Problems with the Learning Algorithm

These problems can be solved empirically for small networks in simple problem domains. For larger domains these problems remain difficult "because it is very easy to violate the assumptions on which the mathematical results are based" (Hinton and Sejnowski, 1986, p. 17). The mean field approximation solves certain problems, however, the problem of suicidal behaviour and the difficulty of estimating the gradient remains.

3.1.2 Major Works Reviewed

This review concentrates upon the work of Hinton et al. (1984); Derthick (1984); Hinton and Sejnowski (1986); Peterson and Anderson (1987); Peterson and Anderson (1988); Peterson and Hartman (1989); Peterson (1991); and Hartman (1991). The review discusses various solutions to the above problems, and presents performance information where relevant.

3.1.3 Measuring Network Performance

Kibler and Langley (1988) suggest that a suitable performance measure for supervised learning algorithms "is the percentage of correctly classified instances" (p. 1). For the purposes of this study this measure of performance is expanded, consisting of two related items, (a) classification ability - the percentage of input patterns correctly classified, and (b) the number of operations required to reach this level of performance. The latter is important when comparing performance between different network models.

3.2 Performance of the Boltzmann Machine

Hinton et al. (1984), and Hinton and Sejnowski (1986) document two experiments with the Boltzmann Machine. The Boltzmann Machine proved to be very slow and, although learning to solve the problems, often failed to reliably classify the input patterns.

3.2.1 Encoding and Communicating Information

The *encoder* problem (Hinton et al., 1984) involves communicating binary patterns between two groups of visible neurons through an intervening hidden layer. No direct synaptic links exist between the two visible layers, thus the hidden layer acts as an information bottleneck. The architecture used for the encoder problem is denoted by the form V_1-H-V_2. A 4-2-4 encoder network has four input neurons, communicating their states to four output neurons through a layer of two hidden neurons.

The performance of the encoder experiments conducted by Hinton et al. (1984) is shown in Table 3.1. Of note is the number of learning cycles needed to reach the required level of classification ability.[14] This experiment was repeated by Peterson and Anderson (1987; see 3.3.1).

Table 3.1. *Performance of the Boltzmann Machine applied to the encoder problem.*

Architecture	Number of Trials	Successful Trials[a]	Cycles to Learn[b]	
			Mean	Maximum
4-2-4	250	100%	110	1810
4-3-4	200	100%	270	1090
8-3-8	20	40%	1570	?
40-10-40	1 (?)	?	800-850	?

Note. Not all information was provided in source - indicated by ?.
[a] Number of experiments that learnt to communicate through hidden layer.
[b] A cycle is application of algorithm for all input patterns.

3.2.2 The Relationship Between Two Input Vectors

The *shifter* problem (Hinton et al., 1984) requires the network to identify the *shift* applied to a binary pattern. Training data includes two input patterns, the first is the original pattern, the second being the shifted version. Each pattern contains eight bits and the shifts allowed are left shift, right shift, and no shift. Three output

[14] This level is not defined.

neurons allow the network to *recognise* the shift status.

The experiment conducted by Hinton et al. (1984) showed that the Boltzmann Machine learnt this task extremely slowly, requiring 9,000 learning cycles to learn the correct mapping. After learning, the network did not classify patterns well. "If the number of on units in V_1 [original pattern] is 1, 2, 3, 4, 5, 6, 7 the percentage of correctly recognized [*sic*] shifts is 50%, 71%, 81%, 86%, 89%, 82%, 66% respectively" (Hinton et al., 1984, p. 21). This level of performance could not be improved by extending the learning time. Hinton et al. (1984, p. 23) speculate that the long learning time is due to the network having to learn a small subset of the 2^{19} possible input patterns.

3.3 Reaching Equilibrium Quickly

The Boltzmann Machine is "computationally expensive since correlations [co-occurrences] of stochastically fluctuating quantities $\langle s_i s_k \rangle$ have to be measured" (Müller and Reinhardt, 1990, p. 124). These quantities are measured after extensive annealing, when the network has reached thermal equilibrium. Müller and Reinhardt (1990, p. 124) suggest two solutions to this problem:

1. Removing the need for the annealing stage by running the network at zero temperature. The network is then deterministic, like the Hopfield network, offering a learning algorithm for hidden neurons as its only advantage. The network would then suffer the same problems as the Hopfield network (see 2.2.4).

2. Constructing a mean field approximation, thus capturing the thermodynamic properties - and advantages - of the Boltzmann Machine. Peterson and Anderson (1987) report substantial improvements in learning speed over the Boltzmann Machine when applied to a variety of problems.

3.3.1 Performance of the Mean Field Approximation

Peterson and Anderson (1987) detail a series of three experiments comparing the performance of the Boltzmann Machine to its mean field approximation:

1. Learning to compute the binary exclusive-Or function, see 4.4.3 for discussion of these results.

2. The encoder problem, using a 4-2-4 and 4-3-4 network (see 3.2.1).

3. Detecting the symmetry of an input vector.

The encoder problem. Peterson and Anderson (1987, p. 17) observe that the mean field approximation provides a factor 3 improvement in learning speed over the Boltzmann Machine when performance is measured as the percentage of the input space completely learnt. For the 4-3-4 network this improvement is reduced to 2.5. See Table 3.1 for the performance of the Boltzmann Machine.

Detecting symmetry. The networks were trained to detect the *symmetry* of a six bit binary pattern. After learning continued for 500 learning cycles the mean field approximation showed considerable advantages over the Boltzmann Machine. Most significant was the number of successful experiments conducted; 90% of the simulations using the mean field approximation were successful in learning the input space, only 20% of the Boltzmann Machine simulations succeeded.

3.4 Controlling the Learning Process

There are few guides to setting the parameters that guide the Boltzmann Machine learning algorithm. The report by Derthick (1984) is an exception, presenting some important analytical results. In the discussion below it is assumed that the approaches are valid for the Boltzmann Machine and mean field approximation.

3.4.1 The Learning Rate

The learning rate controls the rate of change in the synaptic couplings, thus it dictates the rate of movement through *G*-space. There are two approaches to setting the learning rate:

1. *Manhattan Updating* - using discrete weight adjustments in the direction of the slope (Peterson and Hartman, 1989). The direction of movement is the sign of the difference between the two co-occurrence samples, shown in equation (3.1).

$$\Delta w_{ij} = \kappa\left[sgn\left(p_{ij} - p'_{ij}\right) \right] \tag{3.1}$$

2. Movement proportional to the difference between the co-occurrence samples. This is simply the learning rule shown in equation (2.19)

Manhattan updating. Manhattan updating is used by Hinton et al. (1984) and Hinton and Sejnowski (1986) in the encoder problem discussed above (see 3.2.1). This is not a steepest descent technique (Derthick, 1984, p. 6) but does offer significant advantages. Derthick (1984, p. 18) suggests that Manhattan updating leads "to wider searching when the gradient is small and there is nothing obvious to do". However, this may lead the network a long way from the origin thus causing unbounded growth of the synaptic weights. The only way of preventing this occurring is to set very small values for κ, thus reducing the speed of learning.

> It [Manhattan updating] can make significant progress on dimensions where G changes gently without taking very large divergent steps on dimensions where G falls rapidly and then rises again. There is no suitable value for the ε . . . in such cases. Any value large enough to allow progress along the gently sloping floor of a ravine will cause divergent oscillations up and down the steep sides of the ravine. (Hinton et al., 1984, p. 9)

This situation may occur regularly, but describing G-space is a computationally intensive task, especial' when the network is large. This implies that controlling the learning rate requires explicit domain knowledge discovered through experiment and parameter adjustment. Peterson (1991, p. 11) concludes that Manhattan updating is useful when many training patterns are presented to the network before adjusting the weights. Deciding how many examples are "many", and what values for κ are suitable, reduces the worth of this heuristic.

Steepest descent with ε. The use of ε is a steepest descent technique and uses not only information about the direction of the slope in *G*-space, but also the magnitude of the slope. As indicated above, this technique may not be suitable for all *G*-spaces and, as with Manhattan updating, the magnitude of the change must still be chosen. Derthick (1984, p. 2) shows that the conservative estimate of the size of ε shown in equation (3.2) will always result in descent; twice this distance can be moved without ascending.

$$\varepsilon \approx 2\left(\frac{|\nabla G|}{N_{weights}} \right) \tag{3.2}$$

Estimating the gradient of G-Space. The slope of G-space can be estimated using the relationship shown in equation (3.3) (Derthick, 1984, p. 27). Unfortunately, the estimated gradient resulting from equation (3.2) and (3.3) decreases in proportion to network size, implying slower learning for large networks.

$$| \nabla G | \leq \sqrt{N_{weights}} \tag{3.3}$$

3.4.2 The Annealing Schedule

The annealing schedule must provide (a) sufficient time for the network to reach thermal equilibrium, and (b) enough thermal noise for the network to escape local minima. Xu and Oja (1990, p. 1) suggest three problems that are often faced when implementing large networks:

1. The time spent at each temperature step is inadequate for the network to reach thermal equilibrium. Müller and Reinhardt (1990, p. 106) suggest that an infinite amount of time is required to preserve equilibrium. However, the learning algorithm is relatively insensitive to noise and can accommodate fluctuations in equilibrium.

2. The speed of annealing is too fast.

3. The final temperature used to gather co-occurrence samples is not low enough to guarantee global equilibrium, as required by equation (2.13). The learning algorithm is not adversely affected by the first two problems if the final temperature is *reasonable* and the network can reach equilibrium before collecting co-occurrence statistics (Peterson and Hartman, 1989, p. 5).

Determining the speed of annealing. Peterson and Hartman (1989, p. 5) suggest two methods for determining the annealing schedule:

1. A fixed, geometrically determined schedule. Each temperature is a percentage of the preceding value.

2. Calculating the local energy for a given synapse and setting the initial temperature to this value; the final temperature step is one third of this value. The schedule is calculated every learning cycle.

Both methods provide satisfactory performance (Peterson and Hartman, 1989) and provide faster learning than the theoretical optimum. Wassermann (1989) suggests that the rate of temperature reduction must be in proportion to the reciprocal logarithm of the current time step. This would result in greatly extended learning times.

Determining a suitable final temperature. As shown in Figure 3.1 the activation functions become deterministic when the temperature approaches zero - see equation (2.14). Campbell, Sherrington, and Wong (1989, p. 9) conclude that there are no stable states above a temperature of one, below one there are *always* minima and stable states.

For the mean field approximation the *critical temperature*, or temperature at which the network effectively ceases to be deterministic, occurs at approximately T = 0.46 (Müller and Reinhardt, 1990, p. 43). Empirical investigations (Müller and Reinhardt, 1990; Peterson and Anderson, 1987) have determined that the final temperature should be slightly above this value. Figure 3.1 indicates that the critical temperature for the Boltzmann Machine is *lower* (approximately T = 0.23) than that of the mean field approximation.

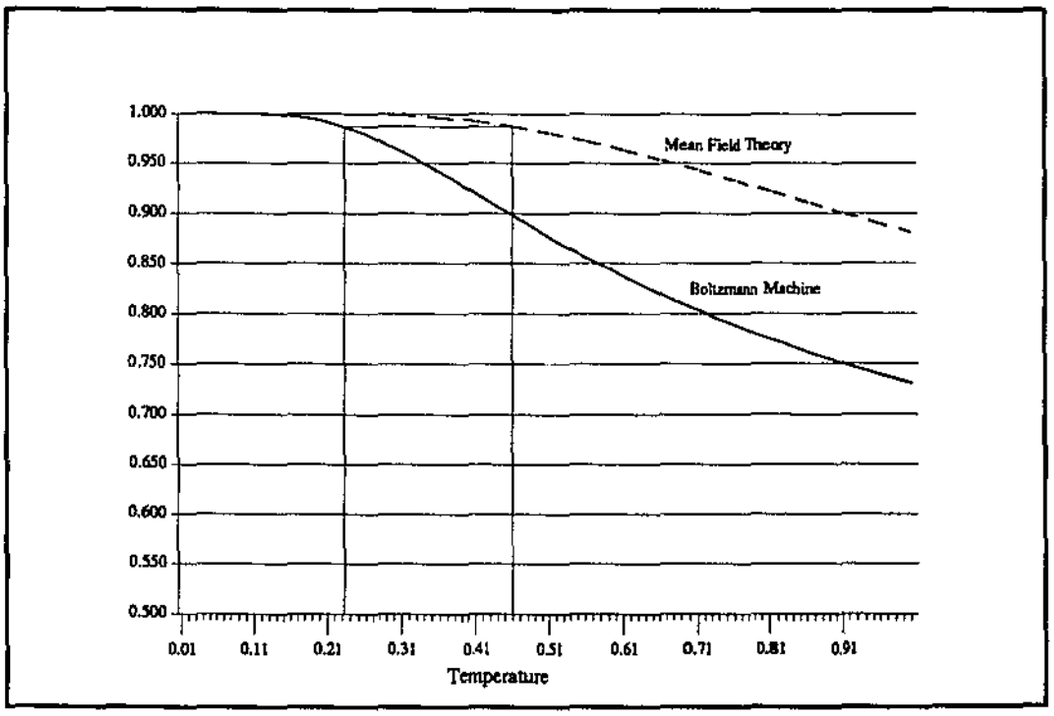

Figure 3.1. Activation probabilities as a function of temperature, showing phase transition point for the mean field approximation and the Boltzmann Machine equivalent.

3.4.3 Binary and Bipolar Representation

Each neuron can take output states in [0,1] or [-1,1], where the Boltzmann Machine uses discrete states and the mean field approximation uses continuous states. Hopfield and Tank (1985) criticise the use of discrete states because it "ignores the very important use that can made of analog variables to represent probabilities, expectation values, or the superposition of many possibilities" (p. 11).

The literature reviewed generally disregards the issue of representation. Peterson and Anderson (1987), and Peterson and Hartman (1989) provide the only genuine discussion of the effects of representation choice.

Advantages of bipolar representation. The use of bipolar representation means that *correlation* statistics, not co-occurrence statistics, drive the learning algorithm - see equation (2.17). Instead of reinforcing synapses between neurons that are simultaneously active, correlation measurements provide reinforcement between neurons with identical states. This doubles the learning rate ε, achieving faster learning (Peterson and Anderson, 1987; Peterson and Hartman, 1989). Whether this leads to oscillatory behaviour is unknown (see 3.4.1).

Advantages of binary representation. Peterson and Anderson (1987) suggest that the use of a bipolar representation is detrimental to the network's generalisation properties. This results from a lack of reinforcement when the neuron states are undecided ($s_i(t) = 0$), using binary representation undecided states are approximately 0.5, resulting in some learning. There are, however, no definitive studies on the use of these two representations.

3.4.4 Controlling Weight Growth

There is no boundary condition in the learning algorithm preventing the creation of large positive or negative weights, eventually leading to the suicidal behaviour discussed by Derthick (Derthick, 1984; Hinton et al., 1984; see 3.1 and 3.4.1). Energy landscapes containing these weights have large barriers that may prevent the network reaching equilibrium, thus causing the learning algorithm to fail.

Decaying weights towards zero. Hinton et al. (1984, p. 22) claims that continually decaying all weights towards zero using a small ($= 0.0005$) constant will prevent unbounded weight growth. This ensures that all synapses not contributing to network performance tend towards having a zero weight. A possible disadvantage of this method is the bounding of the search of G-space to a small area around the origin - where there are only shallow minima (Derthick, 1984).

Synaptic clipping. Müller and Reinhardt (1990, p. 98) discuss synaptic clipping, which is used extensively for Hopfield networks. The most extreme form of clipping is to identify synaptic connections only by their sign, using a fixed and absolute magnitude. A more reasonable approach is to use a "bounding weight function" (Müller and Reinhardt, 1990, p. 99) to control the magnitude of synaptic connections. When used with an associative memory there is a *blurring* effect as the number of stored patterns increases. This is in contrast to the dramatic capacity loss exhibited by the Hopfield network (Müller and Reinhardt, 1990, p. 99). Experiments using synaptic clipping and bounding have not been conducted with either the Boltzmann Machine or the mean field approximation.

Characteristics of networks requiring weight control. Derthick (1984, p. 24) suggests that networks containing more hidden than visible neurons will tend to create large weights. Hartman (1991) supports this conclusion, drawing evidence from investigations into memory systems requiring large numbers of hidden neurons. This behaviour arises because most of the synaptic interaction is between hidden neurons, not between visible and hidden neurons. Peterson and Anderson (1987), Peterson and Hartman (1989), and Hartman (1991) make extensive use of Manhattan updating (see 3.4.1) to control weight growth during their investigations.

3.4.5 The Network Architecture

The issue of network architecture receives surprisingly little coverage in the literature reviewed. Lee (1991) discusses the importance of architectural decisions:

> Because an artificial neural network can only change the interconnection [synaptic] weights, and its structure has to remain fixed, the network designer faces the difficult task of figuring out the optimum structure of the network, thus placing a very tight limitation on its adaptability. (p. 2)

There are two architectural decisions to be made:

(1) The number, and function, of input, output, and hidden neurons.

(2) The connectivity of the neurons.

Each decision affects the generality, the knowledge representation, and the biological plausibility of the model. Unfortunately, these decisions remain domain dependent, forcing the network designer to encode global *a priori* knowledge, thus biasing the solution method.

Determining network size. The size of the network is crucial to the performance of the learning algorithm. Unfortunately there is little guidance in the literature, yet it "is important to have a minimal architecture for a given problem" (Peterson and Hartman, 1989, p. 16).

By not using the smallest possible architecture two problems arise:

1. As the proportion of hidden neurons to visible neurons rises there is an increased tendency for suicidal or dominant behaviour (see 3.4.3).

2. Learning times are proportional to the network size, networks larger than necessary waste computational resources.

Judging the optimal size network is a matter of experience and empirical evidence.

Determining network connectivity. The encoder and shifter problems described by Hinton et al. (1984) make use of specialist connection architectures. The same architectures are used to produce satisfactory performance by Peterson and Anderson (1987); Peterson and Hartman (1989); Peterson (1991); and Hartman (1991). The networks contain no connections between hidden neurons or between input and output neurons. This is in contradiction to the definition of the Boltzmann Machine as a fully connected network (see 2.2.5), although it is an important point that the learning algorithm is flexible enough to train various architectures.

Hartman (1991, p. 8) proves empirically that the presence of connections between hidden neurons reduces the pattern storage capacity of a mean field approximation network used as an associative memory. This is similar to the behaviour described by Derthick (1984; see 3.4.4).

The lack of connections between hidden neurons leads to a *layering* effect with hidden neurons distinguished from visible neurons. This type of architecture, shown in Figure 3.2, leads to hidden neurons acting as *grandmother* neurons, each specific to a particular input pattern. The idea of a grandmother neuron is based upon a biological theory that is probably incorrect (Müller and Reinhardt, 1990, p. 56). Network knowledge is no longer distributed, and performance will degrade when grandmother neurons are damaged.

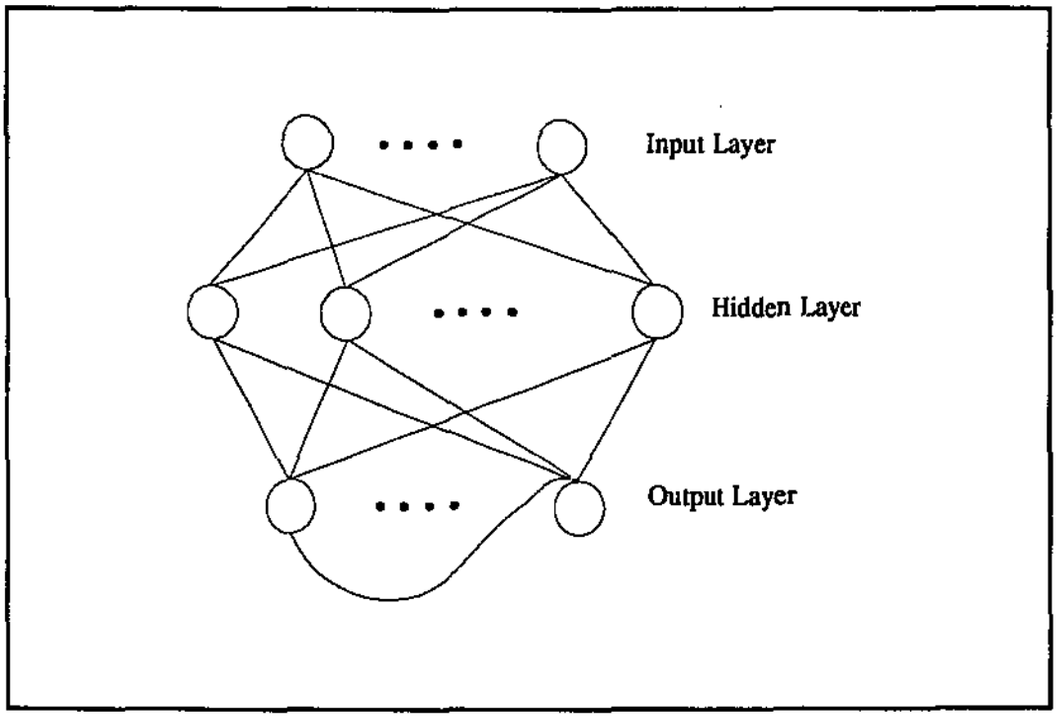

Figure 3.2. Layered architecture resulting from removal of hidden-hidden synapses.

Synaptic dilution and exhaustion. Synaptic dilution has been used with Hopfield networks. It is done by randomly eliminating a percentage of the synaptic connections (Müller and Reinhardt, 1990). Unlike the purposeful removal discussed above this *degrades* the performance of the network, although only marginally, and it is biologically plausible (Müller and Reinhardt, 1990, p. 100). *Death by exhaustion* occurs when synaptic connections are removed if they fall below a certain threshold level. Müller and Reinhardt (1990, p. 100) report that, when used with synaptic clipping, network performance is enhanced. Neither of these methods have been investigated for the Boltzmann Machine learning algorithm.

3.5 Comparisons with Back Propagation

The Boltzmann Machine cannot compete for efficiency with the back propagation network. The mean field approximation is competitive and, although slower in serial simulations, has tremendous advantages as an asynchronous machine.

Peterson and Hartman (1989), and Peterson (1991), document two comparative studies:

(1) Mirror symmetry - the networks were required to find the axis of symmetry of a line in a *NxN* input matrix.

(2) Statistical pattern recognition - the networks were required to classify patterns generated by two overlapping Gaussian functions in eight dimensions.

Their conclusions were as follows (Peterson and Hartman, 1989, p. 9):

- Generalisation capabilities of both paradigms were similar although the mean field approximation performed better using bipolar representation. The mean field approximation was less sensitive to the representation choice than the back propagation network.
- The mean field approximation learns in fewer learning cycles than back propagation; with Manhattan updating the learning times were similar.
- For the statistical pattern recognition problem both networks classified well, approaching the theoretical maximum in approximately the same number of learning cycles.

Peterson (1991) summarises the comparison as follows (p. 27):

> For serial simulations MFT [mean field approximation] takes a factor 2-5 longer time than BP [back propagation] to learn. The real gain is in real-time applications when custom-made hardware is required.

Section 4: Experimental Procedures

4.1 Introduction

The performance of the Boltzmann Machine learning algorithm was investigated by applying it to learning the exclusive-Or function. Although a small problem, it exhibits higher-order constraints requiring the use of hidden neurons, and it has been extensively studied in the literature.

The learning algorithm was used to train a Boltzmann Machine and its mean field approximation. An implementation[15] of a feed-forward, back propagation network was also applied to learning the task, allowing for a comparison between the three models, and between the two learning paradigms.

[15] The implementation used was PERBOOL, a software simulation provided by Müller and Reinhardt (1991). Details of the simulation can be found in this reference.

4.2 Experimental Hypotheses

Direction for the testing process was provided by the following hypotheses:

1. The Boltzmann Machine learns *slower*, i.e., requires more learning cycles, than its mean field approximation to learn an input function.

2. Increasing the number of hidden neurons beyond the minimum required to solve the problem, i.e., introducing redundancy, will reduce the number of learning cycles required to learn the exclusive-Or function.

3. Increasing the learning rate beyond the value suggested by Derthick (1984; see 3.4.1) will *reduce* the number of learning cycles required to learn the exclusive-Or; it also will lead to *instability* during learning as shown by the deviation from average learning times and the dynamic behaviour of the network.

4. Dynamically reducing the learning rate will lead to *smoother* learning for the Boltzmann Machine, reducing the influence of thermal and stochastic noise when the network has learnt to solve the exclusive-Or function.

5. A serial implementation of the mean field approximation compares well with the feed-forward back propagation network - in terms of learning quality and the amount of processing power required to learn the input space.

4.3 The Simulations

4.3.1 Software Implementation

Two software machines were constructed to produce the performance information required to examine the hypotheses listed above (see 4.2). One machine simulated the mean field approximation, the other the Boltzmann Machine. Both implementations supported a fully connected recurrent network allowing either discrete or continuous neurons, and simulating simple asynchronous updating.[16]

4.3.2 External Variables

The external environment influenced the network simulations through, (a) the method that the training data was presented to the network, and (b) the random number generator required to support a simulated network.[17] There are two methods of presenting training information to the network:[18]

1. One-shot training - updating of the synaptic weights occurs after presentation of each training pair. This does not affect the ability of the network to learn but reduces storage capacity (Hinton et al., 1984, p. 24).

[16] The implementations, written in C, use serial updating to ensure that all neurons are updated at least once each learning sweep. This is not required by the learning algorithm.

[17] Several implementations of the Boltzmann Machine learning algorithm are hardware based, using electrical noise to provide the random number generation.

[18] Noisy training, discussed in Hinton et al. (1984, p.9), is another method of presentation - designed to prevent synaptic weights growing exponentially. As it introduces additional noise it has a negative influence upon learning performance.

2. Averaged learning - updating of the synaptic weights occurs after all training patterns have been presented to the network. The change in synaptic weights is then based upon the influence of all training patterns. This is used in the software implementations.

Generating random numbers. Random number generation is required by both simulations and has a serious influence upon performance. The Boltzmann Machine requires random number generation to provide values against which the activation probabilities provided by equation (2.11) are compared. The mean field approximation contains no thermal noise and requires stimulation of the synaptic weights to begin searching *G*-space. To control the influence of the random number generator ten random number sequences were selected and used for all experiments.[19] For the mean field approximation the initial weights were set to very small values - a range of -10^{-2} - $+10^{-2}$ was used.

[19] The random number generator used is more properly called a *pseudo*-random number generator. To produce a sequence of numbers the generator requires a *seed* value; ten seeds were used throughout the experiments.

4.3.3 *Internal Variables*

The annealing schedule. The annealing schedule shown in Table 4.1 was used for all experiments. This schedule is similar to the one used by Peterson and Anderson (1987, p. 13), however, the Boltzmann Machine is taken to a lower final temperature to account for the critical temperature shown in Figure 3.1.

Table 4.1. *Annealing schedule used to learn the exclusive-Or for the Boltzmann Machine and the mean field approximation.*

Boltzmann Machine		Mean-field Theory Approximation	
Temperature	Number of Updates	Temperature	Number of Updates
30	1	30	1
25	2	25	1
20	4	20	1
15	8	15	1
10	8	10	1
5	8	5	1
1	16	1	1
0.4	20[a]	0.5	1

[a] Network was assumed to be in thermal equilibrium and all sweeps were used to collect co-occurrence information.

Altering the learning rates **(ε).** The initial learning rate was estimated using Derthick's equation (Derthick, 1984, p. 2; see 3.4.1). A range of multipliers, shown in Table 4.2, were applied to this rate for use with the simulations.

Table 4.2. *Values used for the learning rate (ε) during the simulations.*

		Multiple of Derthick Estimate		
Network Size	No.Synapses	1x	2x	3x
Four Neurons	7	0.76	1.51	2.27
Five Neurons	12	0.58	1.15	1.73
Six Neurons	18	0.47	0.97	1.46

Representation of neuron states. Preliminary experiments indicated that the use of bipolar states produced local minima that caused extreme problems during learning.[20] This behaviour was not observed when a binary representation was used. Although it may negatively influence learning times the binary representation was used for the simulations (see 3.4.3).

[20] This problem is apparent in the failure rate of the back-propagation simulation, which was restricted to using a bipolar representation.

The range of synaptic weights. For a small problem, such as learning the exclusive-Or function, there is no need for mechanisms to prevent the growth of large weights (see 3.4.4). The synaptic matrix was simulated as an *array* of 80 bit floating point numbers, giving the weights a large range of possible values.[21]

Network size and connectivity. The minimum network required to solve the exclusive-Or function contains one hidden neuron, as shown in Figure 4.1. Networks containing two and three hidden neurons were also used to discover if redundancy improved learning performance, shown in Figure 4.2 and Figure 4.3. Note that the network architectures are fully connected, having connectivities of three, four, and five respectively.[22]

[21] The C data type `long double` was used, providing 10 bytes for representation.

[22] The bias neuron is not counted in regard to network connectivity, but it is used to estimate the optimal learning rate.

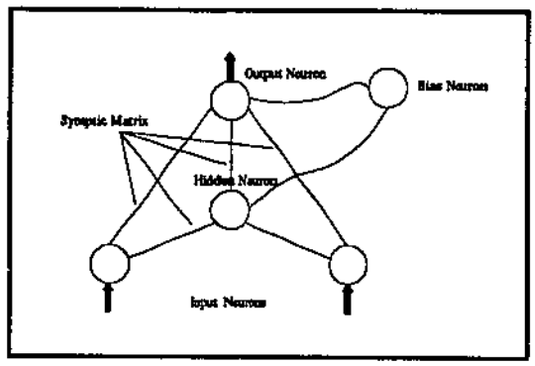

Figure 4.1. Network architecture for simulations using four neurons.

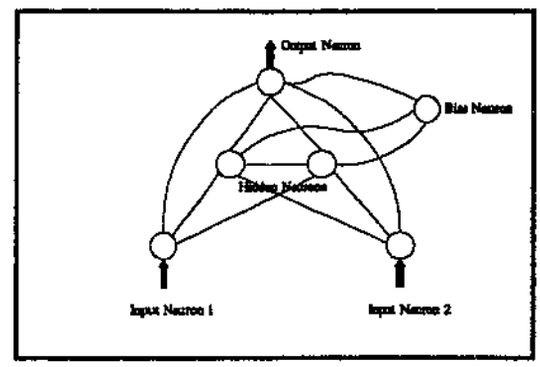

Figure 4.2. Network architecture for simulations using five neurons.

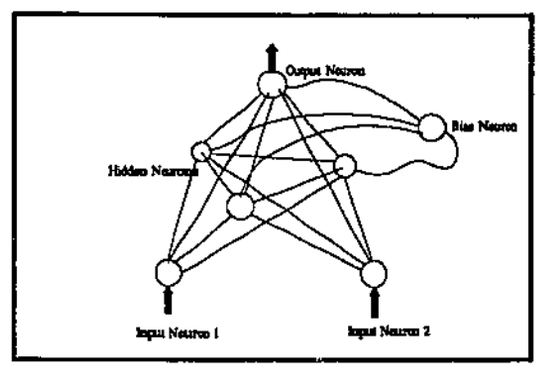

Figure 4.3. Network architecture for simulations using six neurons.

4.4 The Exclusive-Or Function

The exclusive-Or function is a special case of the *parity* function.[23] Feed-forward networks, like back propagation, suffer two problems when solving parity functions (Minsky and Papert, 1969):

1. The smallest number of synaptic connections to any neuron in the network, i.e., its connectivity, must be at least equal to the number of external inputs (p. 56). This implies that the connectivity required to solve problems in large domains is essentially unbounded. As biological neural networks have up to 10^5 synapses leading into a single neuron (Schwartz, 1988, p. 3) this problem may be unavoidable.

2. Synaptic weights grow exponentially with the size of the input set (Minsky and Papert, 1969, p. 153). As the synaptic weights encode the knowledge of the network a large problem domain would require exponential storage and accuracy. This is in contrast to biological systems that use average activation rates to transmit information, relying upon one or two significant figures of accuracy (Sejnowski, 1989, p. 1).

These limitations are important when considering the experimental hypotheses and the data produced by the simulations. A complete examination of these issues was beyond the scope of this study.

---

[23] The parity function is formally defined as follows (Minsky and Papert, 1969, p.56):
$\psi_{PARITY}(X) = \lceil |X| \text{ is an odd number} \rceil$
It is a binary function that returns a true value when the number of active inputs is odd.

4.4.1 The Exclusive-Or Truth-table

The exclusive-Or function can be characterised by the *truth-table* shown in Table 4.3. The function outputs an active value only when the number of active inputs is odd, i.e., it is the simplest example of the parity function.

Table 4.3. *Truth-table for the boolean exclusive-Or function.*

$Input_1$[a]	$Input_2$	$Input_1 \veebar Input_2$
1	1	0
1	0	1
0	1	1
0	0	0

[a] 1 ≡ active, 0 ≡ inactive

The connectivity limitation of feed-forward networks, as formulated by Minsky and Papert (1969), requires that at least one neuron in the network accesses the truth value of *all* inputs[24]. This requirement is avoided by introducing hidden neurons, resulting in a reduction in the difficulty of the problem (Hinton et al., 1984, p. 27):

[24] This is in contradiction to neural networks using only *local* information.

> One can view the set of states of the visible units on which the machine is trained as a single, very high-order, disjunctive constraint. To perform search efficiently, the machine must reduce this constraint to a large set of first and second-order constraints, and to do this it must typically use extra "hidden" units that are not mentioned in the task specification.

Learning the exclusive-Or function is *hard* because the truth table does not reveal how these hidden neurons should be used.

4.4.2 Advantages

The exclusive-Or function is interesting because the ability to learn it suggests an ability to learn the more general parity function. The exclusive-Or function is especially useful for exploring the properties of a learning algorithm because:

- The algorithm can be trained upon the entire truth table.
- The required network is small enough for an empirical analysis of the behaviour of the learning algorithm.
- Few external variables can affect the algorithm and they are easily controlled.

The advantages provided by the exclusive-Or function are unique for a problem of this size. The ability to solve the exclusive-Or is important when examining the efficiency of a connectionist learning algorithm.

4.4.3 Previous Investigations

Peterson and Anderson (1987) describe the use of the Boltzmann Machine learning algorithm to learn the exclusive-Or function. The experiments used a layered architecture (see 3.4.5) with four neurons in the hidden layer.[25] The experiments indicate that the mean field approximation learns "asymptotically better" than the Boltzmann Machine (p. 14). The mean field approximation learnt at least 10-15% faster than the Boltzmann Machine. Similar results for the Boltzmann Machine are reported by Müller and Reinhardt (1990, p. 124).

[25] This is a clear example of the use of grandmother neurons - each hidden neuron detects a single set of input values.

4.5 Generating Performance Information

4.5.1 Data Produced by the Simulations

Testing the success of the algorithm. The simulations were given 250 learning cycles to learn the exclusive-Or function. After each learning cycle the network was presented with the entire input set and allowed to generate its response. The network was considered successful if it was able to correctly identify all four input patterns. Due to the stochastic nature of the Boltzmann Machine this test was repeated 30 times every learning cycle, whereas the mean field approximation was only tested 20 times. The number of errors made in each test was recorded.

Observing the movement of the network. The magnitude of the individual synaptic changes made after each learning cycle was recorded. This data indicates the general movement of the network through *G*-space. Synaptic change data is more informative during learning than observations of movement through *E*-space, as used by Tsang and Bellard (1990; see 2.2.3).[26]

Data from the back-propagation simulations. As little direct control was available with the back propagation implementation, only the number of learning

[26] Changes to the synaptic weights depend upon the learning rate and the co-occurrence information, as shown in equation (2.19). The data used by Tsang and Bellard (1990) is generated by equation (2.8), which depends upon the magnitude of w_{ij}; as learning continues this value naturally becomes larger (more negative).

cycles required to learn the problem was observed. A learning rate (η) of 0.05, a momentum factor (α) of 0.9, and a steepness parameter (β) of 1.0, were found to be the most suitable values for controlling the learning (Rumelhart, Hinton, and Williams, 1984).

4.5.2 Comparing the Performance of the Networks

A measure of the learning speed of a connectionist model is the number of synaptic connections updated (τ) before the problem is learnt. This is approximately equal to the number of operations required (Peterson, 1991, p. 13). For the networks used in the experiments the calculations are shown in equation (4.1).

$$\begin{aligned} \tau_{MFT/BM} &= 2N_L n_t \left(2n_h + n_h n_o + n_i n_h + n_i n_o + n_o - 1 \right) \\ \tau_{BP} &= 2N_L \left[n_h + n_o + n_h \left(n_o + n_i \right) \right] \end{aligned} \tag{4.1}$$

Where: n_t = number of temperature steps used for annealing

n_i, n_h, n_o = number of input, hidden, and output neurons respectively

N_L = number of learning cycles required to learn problem

This value can be used to calculate R, the performance ratio between two different network models (Peterson, 1991, p. 13). The value of τ_{BM} should be multiplied by the number of update sweeps made at each temperature step, see Table 4.1.

Section 5: Results of the Simulations

5.1 Introduction

Five experiments generated the data to test the hypotheses detailed in the previous section, see 4.2:

1. The Boltzmann Machine and the mean field approximation were simulated using four neurons and a learning rate calculated as described by Derthick (1984; see 4.3.3).

2. Experiment one was repeated using one, two, and three hidden neurons (see 4.3.3).

3. Experiment two was repeated for three multiples of the original learning rate.

4. The Boltzmann Machine simulations were repeated; every ten learning cycles the learning rate was reduced by 10%.

5. The back propagation network was simulated 30 times using two hidden neurons and the parameter values previously described (see 4.5).

5.2 Experiment One - Speed of Learning

Table 5.1. *Number of cycles required by the Boltzmann Machine and mean field approximation using four neurons and single multiple estimated gradient.*

Network	Average Learning Time[a]	Deviation	Min.	Max.	Number of Failures
Boltzmann	96.8	46.1	57	205	2
Mean Field	59.2	2.9	55	63	0

[a] The Boltzmann Machine was considered to have learnt the problem when it reached 90% classification ability. The mean field approximation was required to reach 100%.

Of particular note in Table 5.1 is (a) the average time required to learn the input space, and (b) the minimum time required by an individual simulation. Note that the Boltzmann Machine is a stochastic device and has a small level of thermal noise preventing it from reaching 100% classification ability (see 3.4.2).

5.3 *Experiment Two - Increasing Redundancy*

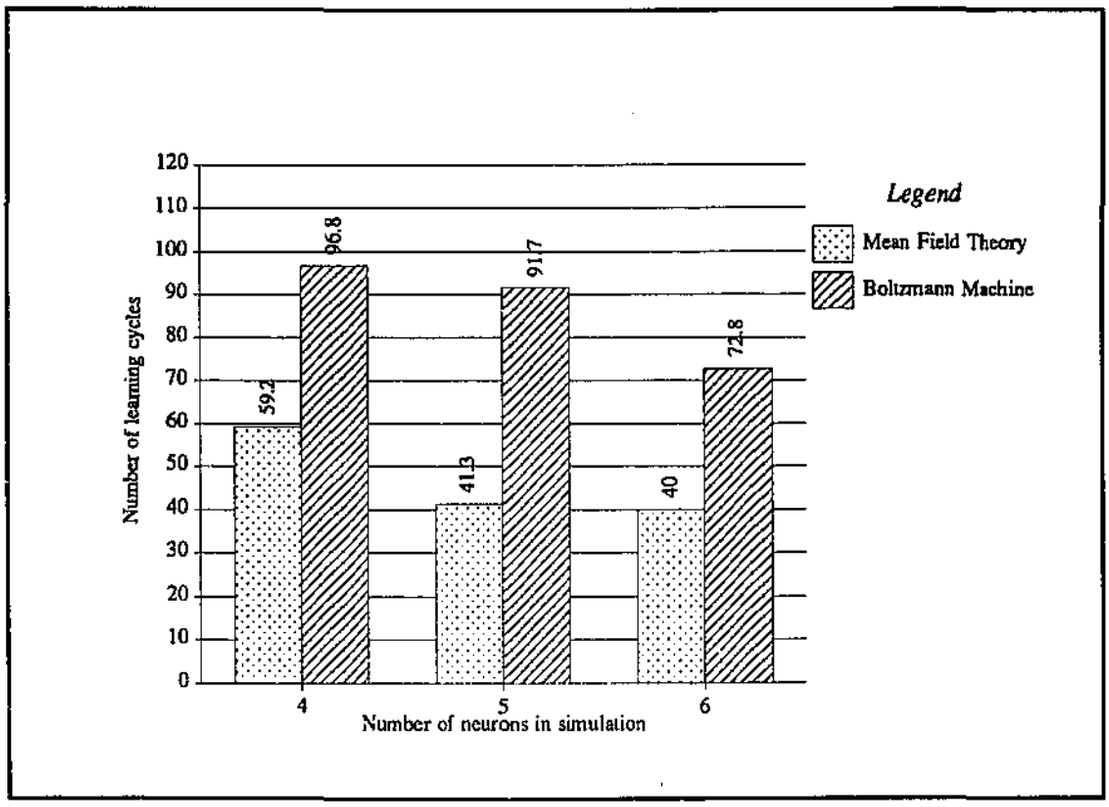

Figure 5.1. Average number of learning cycles required to learn the exclusive-Or function for the Boltzmann Machine and the mean-field theory approximation using three different sizes of network.

It can clearly be seen in Figure 5.1 that both networks respond positively to increased redundancy. It is interesting to note that the learning speed of the Boltzmann machine is continuing to improve whereas the mean field approximation seems to reach a plateau at 40 learning cycles. Further experimentation is required to determine if this trend continues, and to determine at what level the internal noise begins to dominate the training patterns - leading to suicidal behaviour (see 3.1).

5.4 Experiment Three - Increasing the Learning Rate

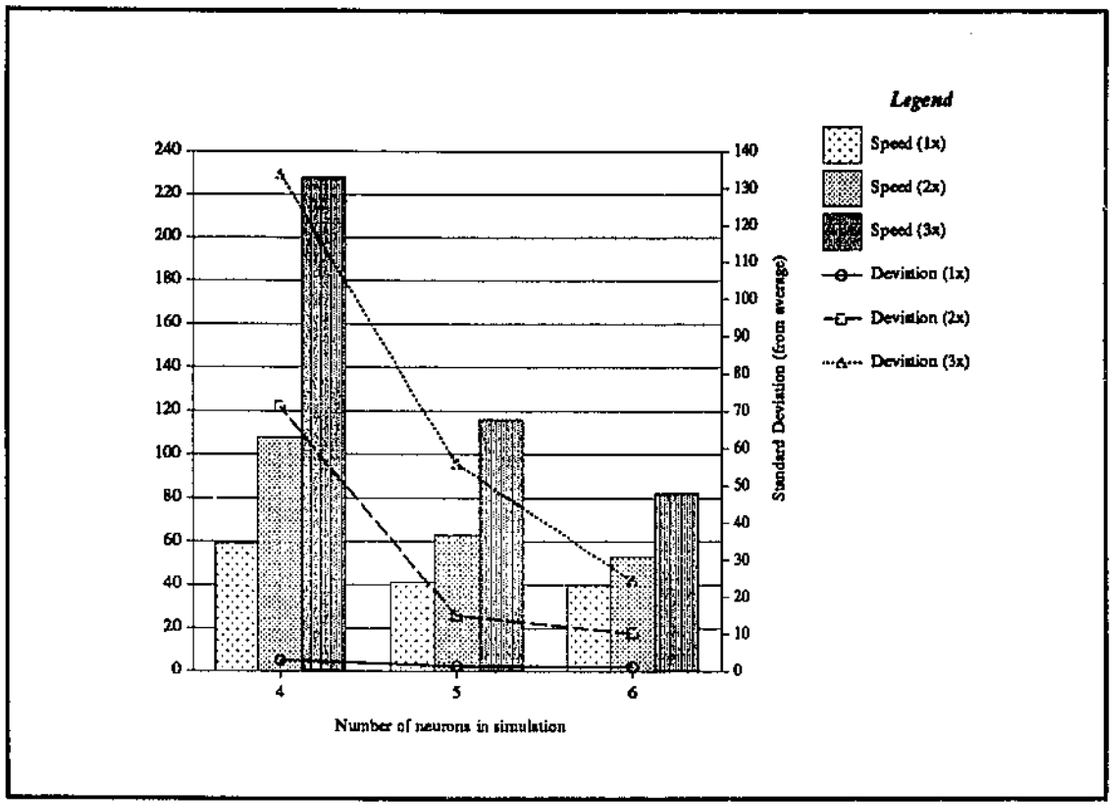

Figure 5.2. Average number of learning cycles required to learn exclusive-Or function for the mean field approximation for different learning rates, network sizes, and showing deviation from mean performance.

Figure 5.2 shows the behaviour of the mean field approximation using three different network sizes and three multiples of the estimated gradient. The mean field theory approximation decreases in speed and stability as the learning rate is increased, although increasing the level of redundancy improves this behaviour. The number of simulations that failed to learn the problem also increased.

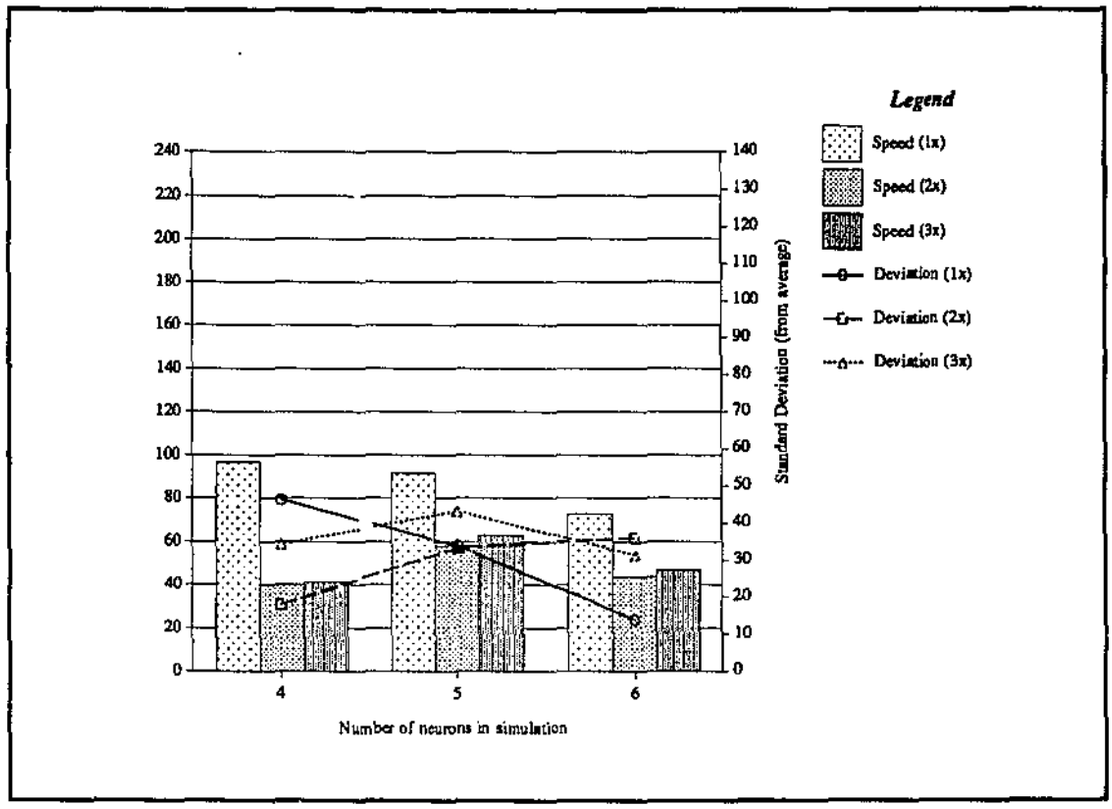

Figure 5.3. Average number of learning cycles required to learn exclusive-Or function for the Boltzmann Machine for different learning rates, network sizes, and showing deviation from mean performance.

The Boltzmann Machine simulations, see Figure 5.3, show an opposite trend to the mean field approximation. Learning speed and stability increases as the learning rate is increased. Tripling the learning rate does not seem to improve performance, in fact it leads to a slowing in learning speed.

5.5 Experiment Four - Dynamic Learning Rate

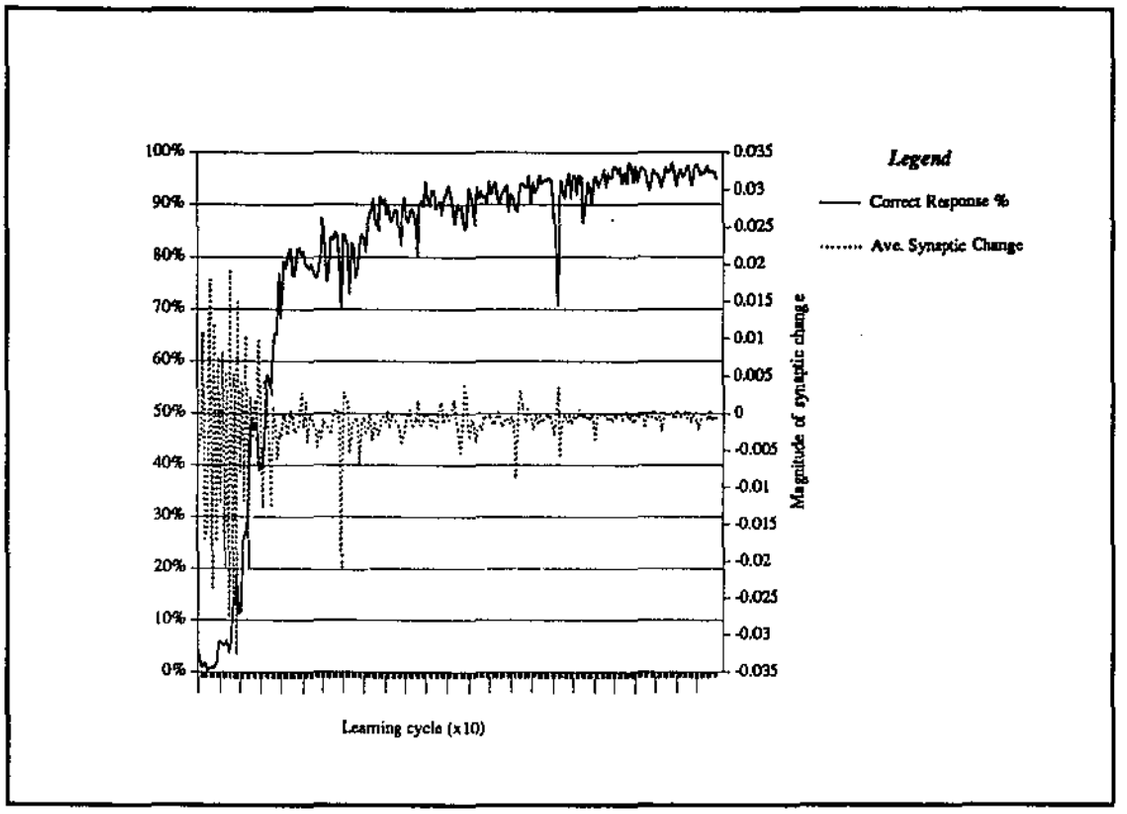

Figure 5.4. Behaviour of the Boltzmann Machine using five neurons and a fixed learning rate.

Figure 5.4 and Figure 5.5 show the magnitude of synaptic changes and the average levels of classification ability for sets of network simulations. Figure 5.4 shows the result of holding the learning rate constant throughout training. Figure 5.5 is the result of dynamically reducing this rate (see 5.1). The results of dynamically reducing the learning rate are also summarised in Figure 5.6.

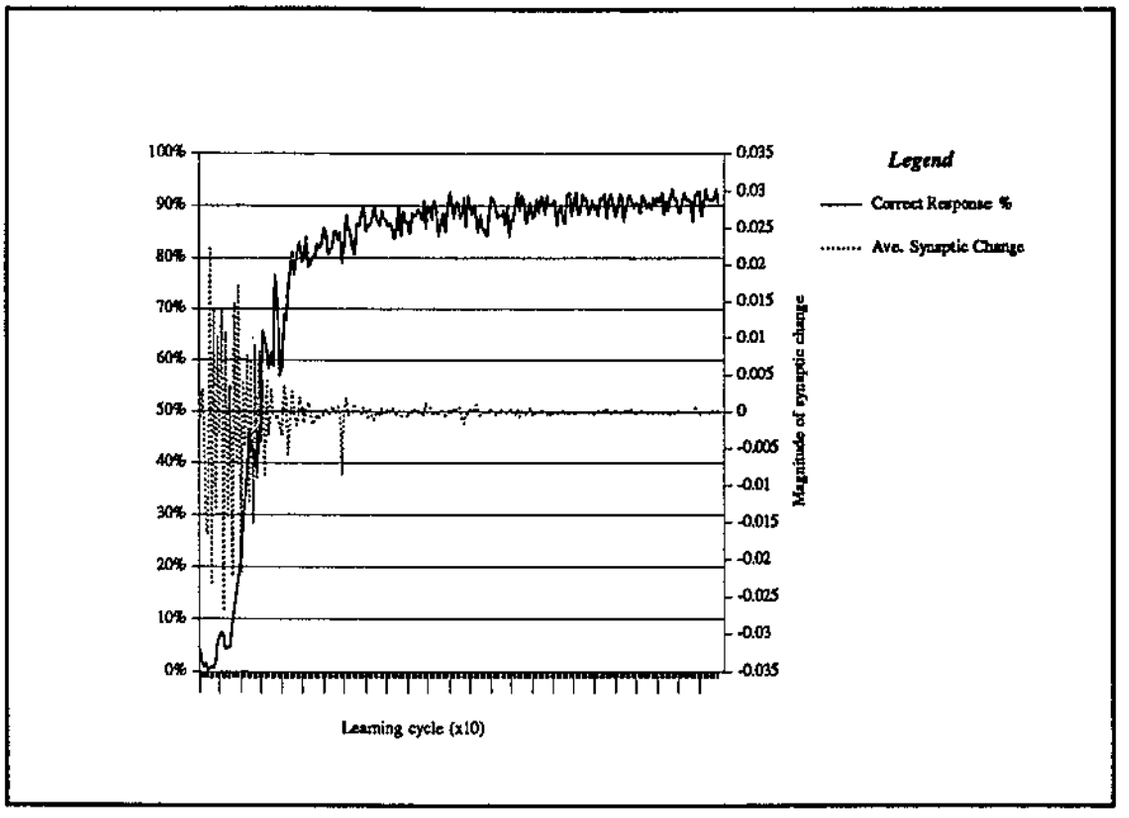

Figure 5.5. Behaviour of the Boltzmann Machine using five neurons and dynamically reducing the learning rate after discrete time intervals.

5.6 Experiment Five - Back Propagation

The back propagation network learnt the exclusive-Or function in an average of 53.6 learning cycles, with a standard deviation of 27.8 cycles. The number of failures was high - 40 simulations were required to collect 30 successful experiments.

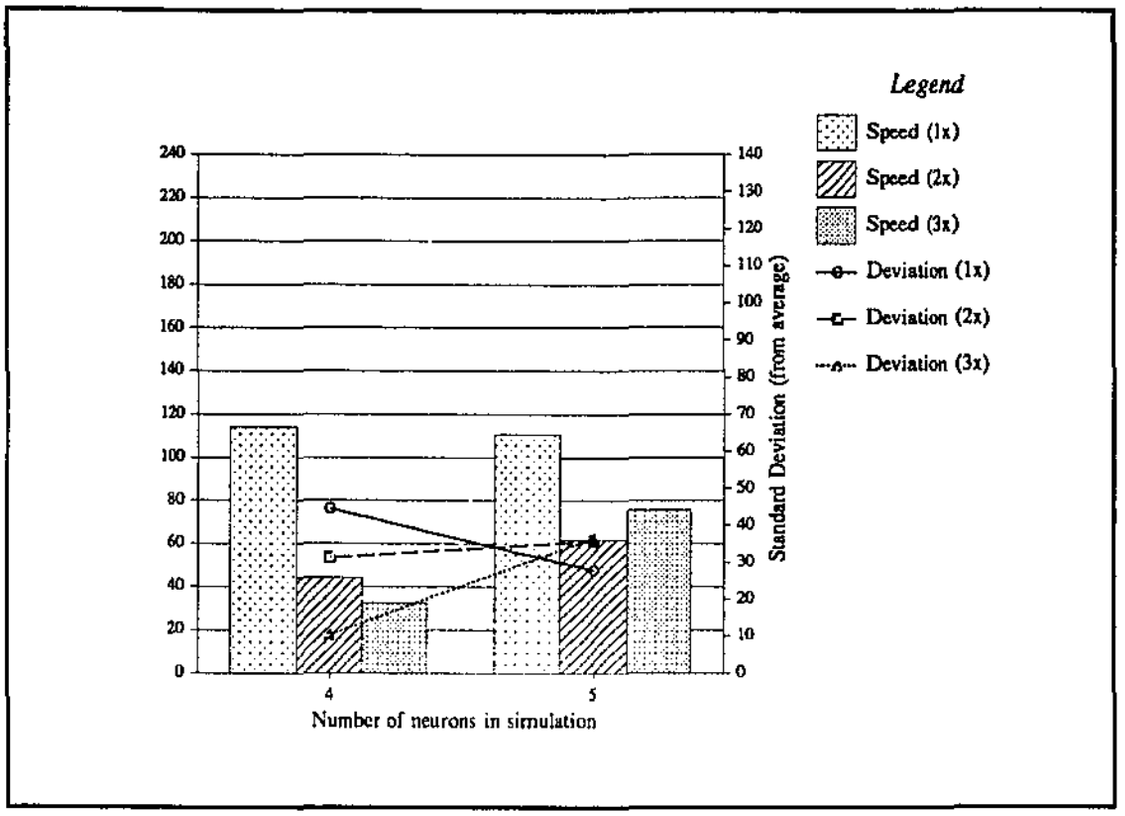

Figure 5.6. Average number of learning cycles required to learn exclusive-Or function for the Boltzmann Machine using dynamically reducing learning rates, two network sizes, and showing deviation from mean performance.

5.7 Interpretation of the Data

5.7.1 Performance of the Learning Algorithm

Experiment one and two clearly show that the mean field approximation learns in approximately half the number of learning cycles required by the Boltzmann Machine. When the number of sweeps made at each temperature is taken into consideration the mean field approximation learns over 100 times faster than the Boltzmann Machine.

Table 5.2 summarises the performance ratios for the experiments. In all cases the mean field approximation learnt faster than the Boltzmann Machine.

Table 5.2. *Performance ratios for the Boltzmann Machine compared to the mean-field theory approximation for different network sizes.*

	Performance Ratio (R)	
Network Size	Without Annealing Schedule	With Annealing Schedule
Four Neurons	1.6	109.6
Five Neurons	2.2	148.8
Six Neurons	1.8	121.9

5.7.2 Increasing Network Size

The results shown in Figure 5.1 indicate that the speed of learning improves when redundant neurons are introduced. Whether this trend continues is not clear from the available data, however the Boltzmann Machine simulations appear to be continuing to improve with each level of redundancy. Figure 5.3 indicates that the stability of the mean field approximation is improving as additional neurons are added to the network. This improvement is to be expected as the approximation becomes increasingly accurate (see 2.4.2).

5.7.3 Increasing the Learning Rate

Figure 5.3 shows that the learning speed of the mean field approximation *does not* improve when the learning rate is increased. This result is surprising, although the increased instability indicated by the clearly separated standard deviations, was expected. There are two possible reasons for these results, (a) the estimation of *G*-space used to approximate the learning was not valid for the mean field approximation, or (b) the approximation is too inaccurate for small networks to remain stable when the learning rate is increased.

The results shown in Figure 5.2 for the Boltzmann Machine are as expected, although the behaviour of network stability is difficult to decipher. The estimate of the gradient improves in accuracy as network size increases, this is indicated in Figure 5.2 by the linear reduction in the deviation from average learning times for 1X the estimate. The estimate can clearly be doubled, reducing the number of cycles required by half, however, stability seems to be decreasing. This trend is also shown when the rate is tripled.

5.7.4 Dynamically Reducing the Learning Rate

The difference between Figure 5.4 and Figure 5.5 indicates that dynamically reducing the learning rate does lead to smoother learning. Surprisingly, the average time required to reach a level of 90% was not significantly affected by reducing the learning rate, although the final level of classification ability was reduced.

Changes to the synaptic matrix are greatly reduced in later learning cycles and, consequently, there are no significant changes in classification performance in later learning cycles. The result of adjusting the learning rate is to restrict the influence of stochastic and sampling noise - thus making learning stable. The standard deviation shown in Figure 5.6 indicates that there was no significant change in the stability of the algorithm.

5.7.5 Comparisons with Back Propagation

Comparison of the average number of learning cycles required by the mean field approximation using five neurons to that of the back propagation experiment produces a performance ratio of 8.2 (see 4.5.2). This indicates that the back propagation network learns a factor 8 faster than the mean field approximation. However, the following points must be considered:

1. The mean field approximation was fully connected, performance studies conducted by Peterson (1991, p. 27) indicate that performance ratios of between two and five can be achieved using restricted connectivity.

2. The mean field theory approximation is naturally asynchronous and is easily transferred to a parallel implementation - the back propagation network is serial and would not gain the same benefits from such an implementation.

3. The annealing schedule used for the simulations was primarily selected to ensure that the Boltzmann Machine obtained thermal equilibrium. The mean field approximation may not require such extensive annealing, hence reducing the number of synaptic updates required and improving the performance ratio.

4. The reliability of the mean field approximation is higher than the back propagation network - no experiments failed unless the learning rate was increased beyond the original estimate.

It can be concluded that the mean field approximation learns more slowly than back propagation, but would be competitive in a suitable environment.

5.8 Conclusions

The experimental hypothesis described in the previous section (see 4.2) have proven to be correct, with few exceptions. The following conclusions can be drawn from the results of the simulations:

1. The mean field approximation learns 2-100 times faster than the Boltzmann Machine.

2. Redundancy in the hidden neurons makes the learning task easier.

3. Increasing the learning rate beyond the estimated gradient improves the performance of the Boltzmann Machine but damages the performance of the mean field approximation.

4. Reducing the learning rate during learning leads to very stable learning by reducing the effects of stochastic and sampling noise.

5. The back propagation network learns up to 8 times faster than the mean field approximation but it is very unstable.

Section 6: Summary

6.1 Generalisation of Results

6.1.1 Objectives

The original hypothesis (see 1.3.1) was successfully tested and shown to be a reasonable generalisation. The Boltzmann Machine learning algorithm, when used to train the mean field approximation, is efficient, reliable, and flexible.

Efficiency. The mean field approximation learns at least twice as fast as the Boltzmann Machine (see Table 5.2). This is a conservative estimate that is lower than the difference reported by Peterson and Anderson (1987, see 4.4.3). However, this ratio ignores the number of update sweeps required by the Boltzmann Machine and the measure used by Peterson and Anderson (1987) is not fully defined.

The mean field approximation learns slower than back propagation by a factor of eight (see 5.7.5). This is a pessimistic estimate as the annealing schedule used in the experiments allowed the Boltzmann Machine to reach thermal equilibrium. Given the advantages provided by the mean field approximation the additional computational cost might be considered negligible.

Reliability. The algorithm is extremely stable when applied to the mean field approximation. The deviation from average learning times and the number of failed simulations under normal circumstances was very low (see 5.7.1). Stability improved as the approximation became more accurate, i.e., as the size of the network was increased. The results of Hartman (1991) suggests that stability remains high as the network becomes larger.

Flexibility. The networks used in the simulations were fully connected, while the results discussed in the literature were for layered architectures. Obviously the algorithm can train at least two different architectures. The use of synaptic dilution techniques for Hopfield networks (see 3.4.5) suggests that the algorithm could train networks of any connectivity.

Unfortunately, the mean field approximation seems very sensitive to the learning rate (see 5.7.3). As the network becomes larger the approximation becomes more accurate, and sensitivity to the learning rate diminishes. The cause of this behaviour cannot be determined due to the limited nature of this study.

6.1.2 Generalised Conclusions

The following conclusions can be made because of this study:

1. The Boltzmann Machine is more accurate and robust than the mean field approximation but is too slow for large applications.

2. The Boltzmann Machine learning algorithm is competitive with back propagation when used with the mean field approximation.

3. The estimate of the learning rate is crucial to the performance of the learning algorithm, especially for the mean field approximation.

4. The learning algorithm can train layered, or fully connected, networks.

6.1.3 Implications

The learning rate and the architecture are crucial to the performance of the learning algorithm. Reliable estimates are available for the learning rate, however the architecture requires the network designer to encode *a priori* knowledge. The range of architectures that can be trained indicates that the algorithm might work with a structural adaption algorithm.

6.2 Limitations of the Study

The results of the study are limited because:

- Only supervised learning was considered.
- Only a simple function from a single problem domain, i.e., the parity problem, was examined.

- The exclusive-Or function, while being indicative of an ability to solve the more general parity problem, is small and the issues of scalability cannot be fully examined.
- The results have not been compared to alternative models of machine learning.
- The experimental hypotheses were limited by the computation time required by the Boltzmann Machine.

6.3 Future Research Directions

6.3.1 Structural Adaption

It is well known that biological neural systems use synaptic *and* structural adaption to respond to the environment. Most connectionist learning algorithms, including the Boltzmann Machine learning algorithm, are limited to synaptic adjustment. The domain independence, structural flexibility, and learning speed, shown by the mean field approximation suggests that it is a potential candidate for use as a structurally adaptive neural network.

6.3.2 Scalability

The mean field approximation has been implemented for a small problem domain; to find out if the results of Minsky and Papert (1969; see 4.4) are

unavoidable requires the implementation of much larger problems. Peterson and Anderson (1988) explore the use of the approximation for large optimisation problems and suggest that the convergence time, i.e., the number of learning cycles required, increases linearly with the size of the network (p. 4). Hartman (1991) has explored the use of the mean field approximation for content-addressable memories, showing that the capacity of the network scales linearly with the number of hidden neurons (p. 15).

6.3.3 Parameter Settings

Heuristics for specifying the learning rate, the annealing schedule, and the representation used for neuron states should be developed because they have such a major role in the performance of the algorithm. The results of the study show that the mean field approximation is very sensitive to the learning rate. Although an estimate using Derthick's formula can be used it is obviously not optimal for the network.

6.3.4 Synaptic Modelling

The techniques of synaptic clipping, dilution, and death by exhaustion should be applied to the mean field approximation. These techniques have been used with Hopfield networks (see 3.4.5), indicating that all recurrent networks may benefit from their use. These techniques may work well when used with structural adaption.

6.4 Conclusions

The study has been successful. The original hypothesis has proven to be reasonable, and some heuristics have been described for specifying the domain dependent parameters of the algorithm. These results should make it easier to apply the Boltzmann Machine learning algorithm to real problems. The flexibility and genericity of the algorithm makes it a more attractive option than back propagation, and it may have additional advantages for a structurally adaptive paradigm.

Section 7: Bibliography

Aleksander, I., & Morton, H. (1990). *An introduction to neural computing*. London: Chapman and Hall

Beale, R., & Jackson, T. (1990). *Neural computing: An introduction.* Bristol, Great Britain: Adam Hilger

Campbell, C., Sherrington, D., & Wong, K.Y.M. (1989). Statistical mechanics and neural networks. In I. Aleksander (Ed.), *Neural computing architectures: The design of brain-like machines* (pp. 239-257). London: North Oxford Academic

Cowan, J.D., & Sharp, D.H. (1988). Neural nets and artificial intelligence. In S. R. Graubard (Ed.), *The artificial intelligence debate: False starts, real foundations* (pp. 85-121), Cambridge, Massachusetts: MIT Press

Diederich, J. (Ed.). (1990). *Artificial neural networks: Concept learning.* Washington, DC: IEEE Society Press

Derthick, M. (1984). *Variations on the Boltzmann Machine learning algorithm.* Pittsburgh, PA, Technical Report CMU-CS-84-120, Carnegie-Mellon University, Dept. of Computer Science

Hampson, S.E. (1990). *Connectionistic problem solving: Computational aspects of biological learning*. Birkhauser: Boston

Hartman, E. (1991). A high storage capacity neural network content-addressable memory. *Network: Computation in neural systems, 2* (3), 315-334

Hecht-Nielsen, R. (1989). Neurocomputer applications. In R. Eckmiller and C. v. d. Malsburg (Eds.), *Neural computers* (pp.445-453). Berlin: Springer-Verlag

Hecht-Nielsen, R. (1990). *Neurocomputing*. Cambridge, Massachusetts: Addison-Wesley

Hinton, G.E. (1990a). Connectionist learning procedures. In J. Diederich (Ed.), *Artificial neural networks: Concept learning* (pp. 11-47). Washington, DC: IEEE Computer Society Press

Hinton, G.E. (1990b). Deterministic Boltzmann learning performs steepest descent in weight-space. In J. Diederich (Ed.), *Artificial neural networks: Concept learning* (pp.128-133). Washington, DC: IEEE Computer Society Press

Hinton, G.E., & Sejnowski, T.J. (1986). Learning and relearning in Boltzmann Machines. In D. E. Rumelhart, & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (vol. 1). Cambridge, Massachusetts: MIT Press

Hinton, G.E., Sejnowski, T.J., & Ackley, D.H. (1984). *Boltzmann Machines: Constraint satisfaction networks that learn.* Pittsburgh, PA, Technical Report CMU-CS-84-119, Carnegie-Mellon University, Dept. of Computer Science

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Science (U.S.), 79* (April), 2554-2558

Hopfield, J.J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Science (U.S.), 81* (May), 3088-3092

Hopfield, J.J., & Tank, D.W. (1985). "Neural computation of decisions in optimization problems. *Biological Cybernetics, 52*, 141-152

Kamp, Y., & Hasler, M. (1990). *Recursive neural networks for associative memory.* Chichester, Great Britain: John Wiley and Sons

Kibler, D., & Langley, P. (1988). Machine learning as an experimental science. In J. Shavlik, and T. Dietterich (Eds.) (1990), *Readings in machine learning* (pp.38-43). San-Mateo, California: Morgan Kaufmann

Killeen, P.R. (1989). Behaviour as a trajectory through a field of attractors. In J. R. Brink and C. R. Haden (Eds), *The computer and the brain: Perspectives on human and artificial intelligence* (pp.53-82). Amsterdam: North-Holland

Klimasauskas, C.C. (Ed.). (1989). *The 1989 neuro-computing bibliography.* (2nd ed.). Cambridge, Massachusetts: MIT Press

Kosko, B. (1992). *Neural networks and fuzzy systems.* Englewood Cliffs: Prentice-Hall

Lee, T. (1991). *Structure level adaption for artificial neural networks.* Boston, Massachusetts: Kulwer Academic Publishers

Minsky, M.L., & Papert, S.A. (1969). *Perceptrons: An introduction to computational geometry* (2nd ed.). Cambridge, Massachusetts: MIT Press

Miller, R.K., Walker, T.C., & Ryan, A.M. (1990). *Neural net applications and products.* SEAI Technical Publications

Müller, B., & Reinhardt, J. (1990). *Neural networks: An introduction.* Berlin: Springer-Verlag

Narendra, K., & Thathachar, M.A.L. (1989). *Learning automata: An introduction.* Englewood-Cliffs: Prentice-Hall

Nelson, M.M., & Illingworth, W.T. (1991). *A practical guide to neural nets.* Massachusetts: Addison-Wesley

Peterson, C. (1991). Mean field theory neural networks for feature recognition, content addressable memory and optimization. *Cognitive Science, 3* (1), 3-33

Peterson, C., & Anderson, J.R. (1987). A mean field theory learning algorithm for neural networks. *Complex Systems, 1* (1987), 995-1019

Peterson, C., & Anderson, J.R. (1988). Neural networks and NP-complete optimization problems; A performance study on the graph bisection problem. *Complex Systems, 2* (1988), 59-89

Peterson, C., & Hartman, E. (1989). Explorations of the mean field theory learning algorithm. *Neural Networks, 2* (1989), 475-494

Rumelhart, D.E., Hinton, G.E, & Williams, R.J. (1986). Learning internal representations by error propagation, In D. E. Rumelhart, & J. L. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (pp.318-362). Cambridge, Massachusetts: MIT Press

Schwartz, J.T. (1988). The new connectionism: Developing relationships between neuroscience and artificial intelligence. In S. R. Graubard (Ed.), *The artificial intelligence debate: False starts, real foundations* (pp. 123-141), Cambridge, Massachusetts: MIT Press

Shavlik, J.W., & Dietterich, T.G. (Eds.) (1990). *Readings in machine learning*. San Mateo, California: Morgan Kaufmann

Sejnowski, T. (1989). "The computer and the brain" revisited. In J. R. Brink, and C. R. Haden (Eds.), *The computer and the brain: Perspectives on human and artificial intelligence* (pp.45-52). Amsterdam: North-Holland

Tsang, C.P., & Bellgard, M.I. (1990). Sequence generation using a network of Boltzmann Machines. In C. P. Tsang (Ed.), *AI'90 - Proceedings of the 4th Australian Joint Conference on Artificial Intelligence* (pp. 224-233). Singapore: World Scientific

Wassermann, P.D. (1989). *Neural computing: Theory and practice.* New York: Van Nostrand Reinhold

Xu, L., & Oja, E. (1990). Improved simulated annealing, Boltzmann Machine, and attributed graph matching. In L.B.Almeida and C.J.Wellekens (Eds.), *Neural Networks: EURASIP workshop proceedings* (pp. 150-160). Berlin: Springer-Verlag

Zeidenberg, M. (1990). *Neural networks in artificial intelligence.* London: Ellis Horwood