SELF-ORTHOGONALIZING STRATEGIES FOR ENHANCING HEBBIAN LEARNING IN RECURRENT NEURAL NETWORKS

By

Michael Richard Davenport

B. Sc. (Physics) University of Calgary

M. Sc. (Physics) University of British Columbia

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF PHYSICS

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

June 1993

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Physics
The University of British Columbia
2075 Wesbrook Place
Vancouver, Canada
V6T 1W5

Date: June 25, 1993

Abstract

A neural network model is presented which extends Hopfield's model by adding hidden neurons. The resulting model remains fully recurrent, and still learns by prescriptive Hebbian learning, but the hidden neurons give it power and flexibility which were not available in Hopfield's original network. The key to the success of the model is that it uses the emerging structure of its own memory space to establish a pattern in the hidden neurons such that each new memory is optimally orthogonal to all previous memories. As a result, the network actually learns a memory set which is "near-orthogonal", even though the visible components of the memories are randomly selected.

The performance of the new network is evaluated both experimentally, using computer simulations, and analytically, using mathematical tools derived from the statistical mechanics of magnetic lattices. The simulation results show that, in comparison with Hopfield's original model, the new network can (a) store more memories of a given size, (b) store memories of different lengths at the same time, (c) store a greater amount of information per neuron, (d) retrieve memories from a smaller prompt, and (e) store the XOR set of associations. It is also shown that the memory recovery process developed for the new network can be used to greatly expand the radius of attraction of standard Hopfield networks for "incomplete" (as opposed to "noisy") prompts.

The mathematical analysis begins by deriving an expression for the free energy of a Hopfield network when a near-orthogonal memory set has been stored. The associated mean-field equations are solved for the zero-temperature, single-recovered-memory case, yielding an equation for the memory capacity as a function of the level of orthogonality. A separate calculation derives a statistical estimate of the level of orthogonality that can be achieved by the roll-up process. When this is combined with the memory capacity-vs-orthogonality result, it yields a reasonable

estimate of the memory capacity as a function of the number of hidden neurons. Finally, the theoretical maximum information content of sets of near-orthogonal memories is calculated as a function of the level of orthogonality, and is compared to the amount of information that can be stored in the new network.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I am grateful to a number of people who contributed to my research and to the production of this thesis. By far the greatest contributer has been my research supervisor, Professor Geoffrey Hoffmann. Professor Hoffmann was the first to recognize the practical uses for peaks in the energy landscape, and many of the other concepts which are developed here were either suggested by him, or grew out of one of our frequent discussions. Thanks also to the other members of my advisory committee, Professors Gordon Semenoff, Luis deSobrino, and Dan Camporese, for reading the thesis and making many helpful suggestions. Professor Semenoff was especially helpful in providing direction for the mathematical derivations in Chapters 4 and 5, and in being available for discussions about the statistical mechanics of neural networks. I am also grateful to Professor Geoffrey Hinton for reading the thesis and making a number of helpful suggestions which have been integrated into the text. Thanks also to my colleague Shawn Day for comments on the text and for the loan of his computer, and to my family for their tolerance and corporeal support.

The research was funded by the Natural Sciences and Engineering Research Council of Canada, and the Science Council of British Columbia. Computer equipment and other financial support were provided by MPR Teltech Ltd., of Burnaby B.C.

Chapter 1

Introduction

The problem of gaining a comprehensive mathematical understanding of brain function is undeniably immense. The human brain is an extremely complex system, made up of at least 10^{11} neurons [42, pg 1], of which there are at least scores, and probably hundreds of different types [48, pg 318]. Every neuron connects to approximately 10^4 other neurons so there are on the order of 10^{15} synapses. The complexity, moreover, exists at all scales – inter-neural connections are no less complex than the connections between cortical columns or between larger functional areas. As a final obstacle to mathematical analysis, the dynamical equations for components of the brain tend to be both non-linear and time-delayed [24, 7]. Faced with such daunting obstacles, the research community has, in the time-honoured way, divided its resources between examining exact models of individual elements (neuro-physiology), more coarse-grained models of larger subsystems (neural networks), and behavioural models of the complete system (psychology), as illustrated in Figure 1.1.

The central role of neural network research is to discover how high-level behaviour such as memory storage and recovery, pattern recognition and, ultimately, intelligence can "emerge" from relatively simple interactions between neurons. The aesthetics of neural network research are therefore closely tied to this concept of "emergent" behaviour. A "good" network is one which is made of simple neural components, assembled and trained according to a simple principle, but which exhibits a complex computational ability. By contrast, a network is not as "good", even if it has the same computational ability, if its ability is only a result of intelligent

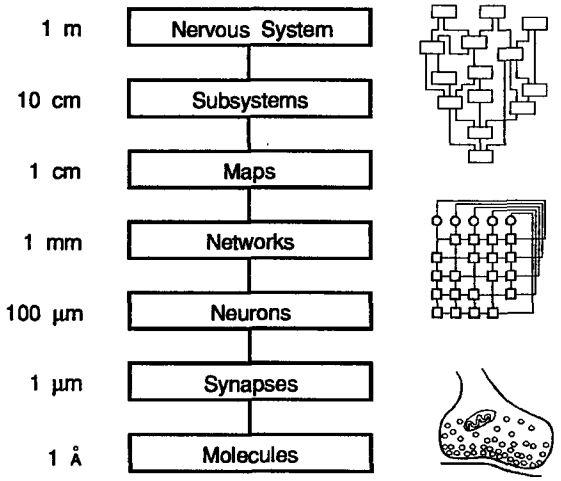


Figure 1.1: Relative scales of components of the nervous system. Neural network research examines structures which are typically between 1mm and 10cm in size. (Based on Sejnowski [48, pg 305])

interference from a human designer or brute force optimization of its parameters for the specified task. The behaviour of the latter network is better described as "engineered" rather than "emergent".

For many researchers, a second aesthetic is that the model incorporate as many aspects of the physiology of the brain as possible. There is some disagreement on this issue, because many neural network researchers are solely interested in the development of alternative computing devices. If a network does interesting things, what does it matter if it resembles the brain or not? The counter-argument to this is that the brain is a phenomenally successful computing device, capable of many functions which are far beyond the capability of current artificial machines. Surely there is a greater chance of reproducing these abilities in a machine which more closely resembles a real brain?

With the second aesthetic in mind, it is useful to enumerate some of the salient "design features" of the brain which might be important to its higher functions. A selected list is:

1. **Hebbian Learning:** learning occurs at each synapse according to a scheme first suggested by Hebb, as discussed below.

2. **Recurrence:** the output of almost every neuron indirectly feeds back into that neuron.

3. **Hidden Neurons:** virtually all neurons in the brain are neither sensory nor motor neurons.

4. **Low Fractional Fan-Out:** the number of neurons connected to a single neuron is much smaller than the total number of neurons in the brain.

5. **Time Delays:** Significant time delays are present in the interconnections between neurons.

Table 1.1 lists some important neural network models and indicates which of these features are associated with each.

Hopfield's network model [25], which will be described in detail in Chapter 2, uses a recurrent architecture, and a version of Hebbian learning. Each neuron in a Hopfield network is either

	Hebbian	recurrent	hiddens	low fan-out	time delays
Hopfield [25]	F	•			
Extended Hopfield (EH) [13]	F	•	•		
Boltzmann Machine [22]	S	•	•		
Time Delay Hopfield [51, 32, 2]	F	•			•
Willshaw [52]	F				
Kanerva [27, 9]	F		•		
Back-Propagation [45]			•	•	
Pineda [44]		•	•	•	
Dispersive [15]			•	•	•
Kohonen [33]		•	•	•	

Table 1.1: Some features of the neurophysiology of the brain cross-referenced to neural network models. Hebbian learning is categorized as fast (F) or slow (S), as discussed in Section 1.1.1. The EH network is the only one which combines recurrence, fast Hebbian learning, and hidden neurons.

"on" or "off". The state of the network is determined by the "on" and "off" states of all its neurons. A "memory" is a network state which has been "learned" by the network using a learning rule which is described below.

1.1 Important Features of the EH Network

The "Extended Hopfield" or "EH" network is the subject of this thesis, and is of particular interest because it is the only one to combine the features of recurrence, fast Hebbian learning, and hidden neurons. These three features of the brain's structure are discussed separately in the following subsections.

1.1.1 Hebbian Learning

The Hebbian learning principle was first suggested by D.O. Hebb in 1949 [19], without any experimental support. It can be summarized as follows [46, pg 36]:

> When neuron A and neuron B are simultaneously excited, increase the strength of the connection between them.

Until very recently experimental tools were not available to test Hebb's conjecture, but it remained of interest because it is a simple and fast way to train a network, and all learning is done locally; there is no need for an external "teacher" to influence the changes in synaptic strength.

In 1969, Willshaw *et al* [52] described a very simple associative memory network which uses a form of Hebbian learning. Their network consists of a set of input neurons b_j and output neurons a_i, connected by synapses c_{ij}. The network is trained by presenting a series of memory

pairs $\vec{a}^\alpha$ and $\vec{b}^\alpha$, and setting the synapses as follows:

$$c_{ij} = \begin{cases} 1 & \text{if } a_i^\alpha = b_j^\alpha = 1 \text{ for any } \alpha \\ 0 & \text{otherwise} \end{cases} .$$

After the network is trained, a memory a can be recalled from a prompt b as follows:

$$a_i = \frac{1}{2}\left[1 - \text{sign}\left(k - \sum_{j=1}^{N} c_{ij} b_j\right)\right]$$

where k is a positive constant determined by the specific memory set being stored. The capacity of Willshaw's network does not increase proportionally to N [20, pg 53], so its capacity is very limited. The network helped establish the concept of associative memories, however, and the viability of Hebbian learning for artificial neural networks.

In 1982, Hopfield [25] presented a neural network model which stored memories using a learning rule which was very similar to Hebb's, and which will be called "pseudo-Hebbian":

> During learning, if neuron A and neuron B are in the same state (either both excited or both quiescent), then increase the strength of the connection between them. If neuron A and neuron B are in opposite states, reduce the strength of the connection between them.

The ability of Hopfield's model to store many memories in a single network was a truly *emergent* property, and it generated a great deal of new interest in neural networks, and Hebbian learning in particular.

Some neural networks [22, 21, eg.] use an iterative form of Hebbian learning in which the training signal must be provided repeatedly over an extended period of time while the connections adapt. Early tests of the Boltzmann Machine, for example, required that each training item be presented 5000 times [22, pg 308] to eliminate the errors. Hopfield's version of Hebbian learning, by contrast, is prescriptive rather than iterative, and requires only a single

presentation of the training items. These two approaches are distinguished as "slow" and "fast" Hebbian learning, respectively, in Table 1.1. The EH network is a "fast" learner, although not as fast as Hopfield's because of time spent determining the optimum state of the hidden neurons.

Since the initial surge of interest in Hopfield's paper, much research activity has swung away from Hebbian learning toward iterative learning techniques such as back-propagation [46]. Because the back-propagation process maintains tight control of the behaviour of the network, its ability to recall memories is not really an "emergent" property. Such networks have been very successful, however, in industrial applications such as machinery control [39], signal processing [34], trend analysis [17, 11], speech recognition [37], and speech synthesis [47]. Recent improvements to back-propagation have included methods for improving learning speed [41], and for incorporating time delays in the synapses [14].

Hebbian learning has recently caught the interest of researchers again, as a result of important new experimental evidence. Thomas Brown's group at Yale [7, 8] have used microelectrodes to control the voltage above and below isolated synapses in the hippocampus, and have found that the synaptic conductance stays approximately constant until the pre- and post-synaptic membranes are stimulated *simulataneously*, just as Hebb had postulated. This is the first, and only, direct observation of learning at the cellular level, and it implies that new models of the brain will be far more likely to replicate brain-like behaviour if they incorporate some form of Hebbian learning.

1.1.2 Recurrence

Neural networks can be classified as either feed-forward or recurrent. Feed-forward networks are thought to be representative of sensory inputs and motor outputs. Signals in a feedforward network pass from the input neurons, often through layers of hidden neurons, and then arrive at the output neurons without ever looping back. Once trained, the behaviour of artificial

feedforward networks is therefore very simple. Most of their complexity lies in the methods used to train them.

In recurrent networks, signals going out from a neuron can eventually influence the input to that same neuron. Networks of this type are thought to be representative of the cognitive centers of the brain, where virtually all the neurons are of this class [42, pg 1]. Both the Hopfield model and the EH model are fully recurrent networks, in that every neuron can be connected to every other neuron.

The behaviour of recurrent networks can be quite complex, because of non-linearities and time delays in the signal loops. A network which begins in one state (a "state" is defined by the on/off status of all the neurons in the network), will typically evolve through a series of intermediate states as the neural signals loop around in the network. In both the Hopfield and the EH model, it will eventually reach a stable "attractor" state, and stop evolving. In the brain, the network never reaches a stable state – new thoughts are continually being created in the mind.

1.1.3 Hidden Neurons

The overwhelming majority of neurons in the brain are "hidden", meaning that they are not in contact with the outside world. The only neurons which are "visible", rather than hidden, are the sensory neurons and the motor neurons. In an artificial neural network, any neuron which is not initialized, and whose output is not seen, is considered to be a hidden neuron. Experience with feed-forward networks has shown that hidden neurons are of central importance for most applications. Minsky and Papert [40], for example, proved that one of the basic elements of logic, the XOR relationship, cannot be learned by a perceptron network with no hidden neurons. The XOR relationship can be learned by a network when hidden neurons are present or the

output units have non-monotonic transfer functions [1].

During learning, the on/off status of hidden neurons is determined by processes internal to the brain, rather than directly by the training signal. For neural networks to use hidden neurons, the design must include some internal procedure for determining their on/off status. Back-propagation, for example, manipulates the status of the hiddens by iteratively modifying the weights of the synapses leading into the hidden neurons.

The biggest limitation of Hopfield's network is that it has no mechanism for including hidden neurons. The model treats all neurons equally, as both inputs and outputs, and requires that all neural states be defined by the trainer. This makes for a very simple and elegant network, but it imposes the following limitations on the network's abilities:

- It cannot store the XOR relationship, or higher order parity.
- The length of each memory must be exactly equal to the number of neurons.
- The maximum number of memories which can be stored is strictly limited by the number of bits in the memories.

The EH network makes it possible to add hidden neurons to a Hopfield network without compromising Hopfield's use of Hebbian learning and recurrent architectures. During training, the EH network uses the emerging structure of its own memory space to determine the optimum on/off status of each hidden neuron, as described in Chapter 3. Because that optimum status is one where the memories are mutually orthogonal, the EH network can be thought of as a "self-orthogonalizing" network.

[1] The author thanks G. Hinton for pointing out that non-monotonic transfer functions in the output neurons make it possible to store XOR without hidden neurons.

1.2 Research Goals

The overall goals of this research are to better understand the structure and the behaviour of the brain, and to develop computing procedures which take advantage of those principles to perform novel computing tasks. The specific goals which led to the results presented here were:

- to extend Hopfield's model to include hidden neurons
- to demonstrate the improved performance resulting from the extension
- to gain "analytical control" [5, pg. 2293] of EH networks.
- to derive an analytical expression for the capacity of an EH network as a function of the number of hidden neurons

1.3 Summary

Chapters 2 through 6 describe the research that has been done in pursuit of the above goals. It can be summarized as follows:

- Chapter 2 reviews Hopfield's neural network and examines its performance with non-random memory sets. The chapter begins with a brief review of the structure of Hopfield networks, and the notation used to describe them. It then introduces the idea of "near-orthogonal" memory sets, describes how they can be stochastically generated, and examines how the memory capacity of the Hopfield network changes as the memory sets become more orthogonal. The final section of the chapter introduces an important working assumption which will be used in subsequent derivations.
- Chapter 3 defines and characterizes the new EH network. It begins by describing the EH network's memory storage procedure, and then presents measurements of how well

that procedure is able to orthogonalize the memory space. It describes two alternative methods for recalling memories when specific neurons (usually "hidden" neurons) are flagged as unknowns, and examines the effectiveness of the methods for near-orthogonal memory sets. Finally, in Section 3.4, many different aspects of the network's performance are examined, including its memory storage capacity, its information capacity, its ability to store correlated memory sets, and the size of its attractor basins.

- Chapter 4 develops a mathematical model for a Hopfield network in which near-orthogonal memories have been stored. The chapter begins by reviewing the analogy between lattices of Ising ferromagnets, and Hopfield-style neural networks. It then derives an expression for the free-energy, using the replica method and assuming replica symmetry. Finally, it derives a set of mean-field equations which are valid at the saddle point in the free energy.

- Chapter 5 derives an analytical estimate of the storage capacity of the EH network as a function of the fraction of visible neurons in the network. The chapter begins by using the mean-field equations from Chapter 4 to calculate the capacity of a Hopfield network as a function of the degree of orthogonality of the memories stored. Next, it uses a statistical argument to estimate the degree of orthogonality that can be achieved by the EH network's memory storage process. Finally, it combines these two results to estimate the memory capacity.

- Chapter 6 compares the theoretical maximum information capacity of near-orthogonal memories to the actual amount of information that can be stored in an EH network. It begins by calculating the information content of exactly orthogonal memory sets, using both a dimensional argument and a brute-force calculation. It then extends the result analytically to include near-orthogonal memory sets. Finally, a comparison is made between the amount of information which *can* be stored in a memory set of a certain orthogonality, and the amount of information which *is* stored in an EH network with the same degree of orthogonality.

- Chapter 7 gives a summary of the conclusions that have been reached in this research, and of further research that needs to be done.

Appendices A through D provide details of some of the calculations used in the main body of the thesis.

Chapter 2

The Hopfield Network and Non-Random Memory Sets

Before the EH network can be introduced, it is important to review the structure and dynamics of Hopfield's original network, and its behaviour for sets of memories which are not randomly generated. Section 2.1 summarizes the design of the network and introduces the notation which will be used in subsequent chapters. Section 2.3 defines the concept of a "near-orthogonal" memory set, and presents simulation results which show how the memory capacity of a Hopfield net varies as the memory sets become less random. Section 2.4 presents an important working assumption about the behaviour of Hopfield networks which will form the basis for mathematical derivations in subsequent chapters.

2.1 Review of Hopfield Neural Networks

A Hopfield net consists of N neurons, each of which is connected via synapses to every other neuron, as shown in Figure 2.1. The "synaptic strengths" J_{ij} are in general real numbers ranging from positive to negative. The output of the ith neuron is called the "state", S_i of that neuron. The Hopfield net uses bi-state neurons, each of which has only two possible states, $(S_i = +1)$ and $(S_i = -1)$. The set of N neural states together form an N-dimensional "network state" vector $\vec{S}$. A dot product between any two network states can be written as follows:

$$S^1 \cdot S^2 \equiv \sum_{i=1}^{N} S_i^1 S_i^2 \tag{2.1}$$

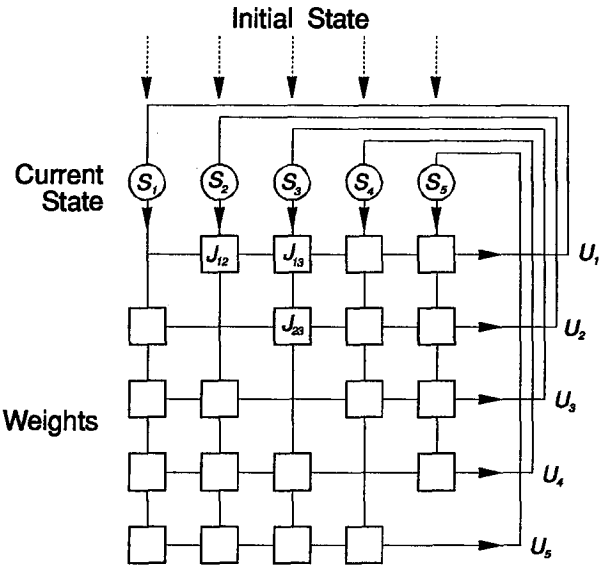


Figure 2.1: Diagram of a Hopfield network with 5 neurons. The output, S_i, from each neuron is multiplied by a weight J_{ij} and summed with other signals before being fed as an input to each of the other neurons. The initial states of the neurons are imposed by an external signal, which is removed once memory recovery begins.

State vectors can be written concisely using just the signs of their components, as in the following example:

$$S^3 \;=\; \texttt{++-+-+++--+--+++}$$

The input U_i to each neuron i is equal to the sum of the outputs of all the other neurons, weighted by a set of connection strengths J_{i1} through J_{iN}, where N is the number of neurons:

$$U_i \;=\; \sum_{j=1}^{N} J_{ij} S_j \tag{2.2}$$

Neurons do not synapse back onto themselves, so $J_{ii} = 0$. Non-zero J_{ii} terms would not affect the *location* of memories, but can influence the dynamics of the network. Large positive J_{ii}, for example, tend to stabilize states which would otherwise be unstable. Negative J_{ii} have the opposite effect, destabilizing some of the shallower attractor states.

The notation used here is that used by Amit [1], Lautrup [35], Mezard, and Parisi [38]. Hopfield used T_{ij} and V_i rather than J_{ij} and S_j in his original paper [25].

2.1.1 Storing Memories

The connection strengths J_{ij} are determined by the memories that have been stored in the network. A "memory", just like a network state, is an N-dimensional vector ξ_i whose components are (+1) or (-1). A set of P memories can be "stored" in a network by setting the strength of each connection as follows:

$$J_{ij} = \frac{1}{N}\left[\sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu - P\delta_{ij}\right],$$

where the δ_{ij} ensures that the diagonal elements of J are always zero. The $\frac{1}{N}$ term is often not included in the definition of J_{ij} but, as will become clear, it does not change the dynamics of the network, and it ensures that the connection strengths remain finite in the limit of very large P and N. The definition of J_{ij} implies that each new memory can be added using the following "outer product rule":

$$J'_{ij} = J_{ij} + \frac{1}{N}\left[\xi_i^\mu \xi_j^\mu - \delta_{ij}\right]. \tag{2.3}$$

The application of the outer product rule is variously referred to as "storing", "inserting", or "memorizing" a new memory ξ^μ. Note that the outer product rule is an implementation of the pseudo-Hebbian learning rule introduced in Section 1.1.1, in that if two neurons i and j are in the same state $\xi_i^\mu = \xi_j^\mu$, the connection J_{ij} between them increases, otherwise it decreases.

The "memory loading", α, of a network is equal to the ratio of the number of memories P and the number of neurons N:

$$\alpha \equiv \frac{P}{N} \tag{2.4}$$

2.1.2 Network Dynamics and Stable States

The evolution of a Hopfield network is determined by its initial state, and by the connection strength J_{ij}. In Hopfield's original network, the network evolves in the following way:

1. One of the neurons, i, is selected at random .
2. The input U_i for that neuron is calculated.
3. The state S_i is modified according to the following "forward update rule":

$$S_i = \begin{cases} +1 & \text{if } U_i > 0 \\ -1 & \text{if } U_i < 0 \end{cases} . \tag{2.5}$$

4. A new neuron is selected as in Step (1).

This is referred to as the "asynchronous" version of the forward update rule because each neuron is updated independently. Evolution continues until a stable state is reached in which all the inputs U_i have the same sign as the neural states S_i. Hopfield proved that the network will always reach such a stable state, and that the stable states tend to be attractors. If the final state is equal to one of the memories ξ^μ, then the network is said to have "recovered" or "recalled" that memory.

Hopfield networks are "autoassociative", meaning that they tend to recover the memory which is most similar to their initial state. In a typical application, the network is initialized to a "prompt" state S^0, which is an incomplete or noisy version of one of the memories, and then is allowed to evolve to one of the attractor states. Barring any problems, the final state is the memory state which most resembles the prompt state.

The "radius of attraction", R_ξ, of a memory ξ^μ is a measure of how many bits can be recovered during the memory recovery process. More precisely, it is the maximum number of

bits that can be wrong in the prompt before the probability of recovering the correct memory becomes less than 50%.

In some situations it is preferable for the network to evolve in the following way:

1. The inputs U_1 through U_N are all calculated.
2. The states S_1 through S_N are all modified using the forward update rule (Equation 2.5), using the pre-calculated U_i.
3. A new set of inputs U_i is selected as in Step (1).

This is referred to as the "synchronous" version of the forward update rule because all neurons are updated at the same time. A network evolving with the synchronous update rule does not always reach a stable state, but may instead enter a limit cycle in which it flips between two states [1, pg 78].

2.1.3 Overlap $m^{\mu\nu}$ and Orthogonality

The overlap $m^{\mu\nu}$ between two memories ξ^μ and ξ^ν is a measure of how similar they are to each other. It is defined as follows:

$$m^{\mu\nu} \equiv \frac{1}{N}\sum_{i=i}^{N} \xi_i^\mu \xi_i^\nu \tag{2.6}$$

Overlaps of 1, and -1 indicate that two memories are identical, or bit-inverse to each other, respectively. Two memories are said to be "orthogonal" to each other if their overlap is zero.

When m appears with a single superscript, it represents the overlap between the current state S and one of the memories:

$$m^\mu \equiv \frac{1}{N}\sum_{i=i}^{N} S_i \xi_i^\mu$$

It will be important in the following chapters to characterize memory sets by the magnitude of typical overlaps between the memories. The expected overlap $m^{\mu\nu}$ between two randomly selected memories is equivalent to the expected length of a random walk of N steps. The mean of $m^{\mu\nu}$ is therefore zero and the variance is $\frac{1}{N}$ [1, pg 175]. Overlaps for non-random memory sets are discussed in Section 2.3.

2.1.4 Energy Landscape

Hopfield noted that his network is "isomorphic with an Ising model" [25, pg 2556], a fact which will be discussed in detail in Section 4.1. This isomorphism led him to define the following "energy" E for every state S of the network:

$$E = -\frac{1}{2}\sum_{i,j=1}^{N} J_{ij}S_iS_j \tag{2.7}$$

It can be easily shown [20, pg. 21], when $J_{ij} = J_{ji}$, that this energy acts as a Liapunov function for the update rule in Equation 2.5. It is common, therefore, to speak of an "energy landscape" in which the network moves. When the forward update rule is used, the network never rolls uphill in this landscape, and always eventually reaches a stable state at the bottom of an "energy valley". When no memories have been stored in the network, and all the J_{ij} are zero, the landscape is flat. After one memory has been installed, there are two valleys: one at ξ^1 and one at $\overline{\xi^1}$.

2.1.5 The Meaning of Energy Peaks

If the valleys in the energy landscape represent stored memories and their bit-inverses, what do the peaks represent? If Equation 2.3 is replaced into Equation 2.7, the energy of a state S_i

can be written:

$$\begin{aligned} E &= -\frac{1}{2N}\sum_{i,j=1}^{N}\left[\sum_{\mu=1}^{P}\xi_i^\mu\xi_j^\mu - P\delta_{ij}\right]S_iS_j \\ &= -\frac{N}{2}\sum_{\mu=1}^{P}(m^\mu)^2 + \frac{P}{2}. \end{aligned} \tag{2.8}$$

Peak states are states where E is maximum, so they are also states where $\sum_{\mu=1}^{P}(m^\mu)^2$ is minimum. In other words the peaks mark states which are, in the least-squares sense, optimally orthogonal to all installed memories [1, pg 167]. "Optimally orthogonal" here means that the sum of the squares of the overlaps is minimum compared to all nearby states. If the states were exactly orthogonal, the sum of the squares would of course be zero.

2.1.6 False Attractors

There are always stable attractor states which were not explicitly stored as memories. Every time a memory ξ^μ is stored, for example, the vector $\overline{\xi^\mu}$ is automatically stored as well, where $\overline{\xi_i^\mu} \equiv -\xi_i^\mu$ for all i. This follows from the fact that J_{ij} in Equation 2.3 is unchanged when all the ξ_i^μs are negated. The vectors $\overline{\xi_i^\mu}$ are referred to as "bit inverse attractors".

When more than two memories have been installed, other "false" attractor states begin to appear which are not so simply related to the real memory states. The locations of many of these states are quite well understood analytically [1, pp 176-177]. The following "symmetric 3-states", for example, will always be false attractors:

$$\begin{aligned} S_i &= \text{sign}(\xi_i^1 + \xi_i^2 + \xi_i^3) \\ S_i &= \text{sign}(\xi_i^1 + \xi_i^2 - \xi_i^3) \\ S_i &= \text{sign}(\xi_i^1 - \xi_i^2 + \xi_i^3) \\ S_i &= \text{sign}(\xi_i^1 - \xi_i^2 - \xi_i^3) \\ S_i &= \text{sign}(-\xi_i^1 + \xi_i^2 + \xi_i^3) \end{aligned}$$

etc.

There are similar symmetric 5-states, 7-states, and so on.

False attractors are a serious problem for practical applications of attractor neural networks. As new memories are added, the number of false attractors grows far more rapidly than the number of true attractors, which reduces the radii of attraction of the true memories.

2.2 Limitations and Extensions of the Hopfield Network

There are a number of fundamental limitations of Hopfield's neural network which reduce its usefulness in many practical applications. These include:

1. The network is unable to store the XOR boolean relationship, for reasons discussed in Section 3.4.5. This fundamentally limits its usefulness as a computational device.
2. The number of synapses in the network is always equal to $N(N-1)$, where N is the number of neurons. Because the number of synapses determines the capacity of the network, this means that the capacity of the network is limited by the width of the stored memories.
3. The capacity of the network is proportional to the number of neurons, which is the square root of the number of synapses. As networks get bigger, therefore, the complexity increases as the square of the capacity.
4. The capacity of the network falls quickly when the memories become more correlated, as discussed in Section 3.4.3. Typical memory sets in the real world are often correlated, so this is an important limitation.

The EH network overcomes the first two of these limitations, as discussed in Chapter 3, but is not significantly better than the Hopfield network for the latter two. As discussed in

Chapter 5, although the capacity of an EH network is larger than that of a Hopfield network, it still increases linearly with N. Similarly, Section 3.4.3 shows that the capacity of the EH network falls almost as rapidly as the Hopfield network when the stored memories are correlated.

Baum and Moody [6] have described a feedforward network whose structure bears some similarity to Hopfield's, and which overcomes these limitations. They call the network a "unary Associative and Content Addressable Memory" (Unary ACAM). It has two layers of connections, with one layer of hidden neurons in between. The top set of connections, connecting the input neurons to the hidden neurons, form a Hamming net [36]. When an input is presented, the Hamming net activates exactly one hidden neuron – the one which corresponds to the pattern that most resembles the input pattern. The hidden neurons are therefore "grandmother" cells [20, pg 143] because each one corresponds to one specific memory. It is straightforward to set up connections from the grandmother cells to the output neurons which ensure that the desired output signal is generated in response to each grandmother cell. Baum and Moody point out that if the grandmother cells are replaced by unity gains, and the output is fed back into the input, their network is equivalent to Hopfield's and the training method is equivalent to the Outer Product Rule.

As a computing device the unary ACAM is, arguably, a better performer than Hopfield's. Its memory capacity is exactly equal to the number of hidden neurons, and so can be extended almost indefinately just by adding more hidden neurons. Furthermore, memory recall does not degrade as the network approaches capacity. The network also works well for correlated memories, and has no trouble storing the XOR set of associations.

As a model for human memory, however, the Unary ACAM is not very satisfactory because it is not trained with Hebbian learning and it is not recurrent. It is, essentially, a machine which has been designed to respond to specific inputs with a specific output. Its behaviour is therefore more engineered than emergent.

The most significant limitation of the Unary ACAM as a computing device is its lack of robustness. The failure of any grandmother cell, or any synapse in the output layer, causes an immediate failure in recalling the corresponding memory. This weakness can be addressed in a number of ways. The simplest way, as pointed out by Baum and Moody, is to add redundant grandmother cells. If two grandmother cells are used for each memory the failure of any single cell no longer degrades the performance of the network. Such a network is still far less robust than a Hopfield network, where more than half the neurons can typically be disabled without seriously degrading the performance.

Another network which is similar in structure to the Unary ACAM but somewhat more robust, has been described by Kanerva [27, 9]. Like the Unary ACAM, Kanerva's network is a feedforward device with two layers of connections and a single layer of hidden neurons. As in the Unary ACAM, each neuron in the hidden layer is trained to recognize a single vector input, but now the training vectors are randomly chosen rather than equal to the memory set. There is no "winner take all" circuitry on the hidden neurons, so more than one hidden neuron can be activated for each input pattern. The first layer of synapses now effectively encodes each N-bit input into an s-bit pattern, where s is the number of hidden neurons. The second layer of Kanerva's network is similar to a Hopfield network except that it has separate input and output neurons, and is not recurrent. The connections of the second layer are trained with the Outer Product Rule.

Kanerva's network performs comparably to the Unary ACAM, and its performance can be fine-tuned by adjusting its parameters. The robustness of the network, for example, can be increased by:

- Adding more hidden neurons, which adds redundancy to the network, at the price of added complexity; or

- Decreasing the thresholds on the hidden neurons, which brings more hidden neurons into action for each input pattern, at the price of more noise in the input to the second layer.

Like the Unary ACAM, the capacity of Kanerva's network grows in proportion to the number of hidden neurons, and hence can be expanded indefinately by adding more hardware.

Kanerva's memory, which has been described as a Hopfield network with a preprocessor [9, pg 285], is somewhat more satisfactory than the Unary ACAM as a model for biological memory. It is conceivable that a biological memory might have a hard-wired preprocessor network which encodes input signals into larger dimensional signals; and the Hopfield-like second layer is certainly plausible. Unlike the Hopfield model, however, Kanerva's network is not recurrent.

2.3 Non-Random Memory Sets

This section focusses on how the memory capacity of a Hopfield network is affected by the level of correlation between the memories that are being stored. The first step in quantifying that relationship is to define a measure for the level of correlation in a memory set. A natural measure is the root-mean-square of the overlap between all the memories ξ^μ. For random memories, the rms overlap is $\sqrt{N}$, so the following "normalized rms overlap" is used:

$$O_{\text{rms}} \equiv \frac{1}{\sqrt{N}} \left\langle\!\!\left\langle \left(\sum_{i=1}^{N} \xi_i^\mu \xi_i^\nu \right)^2 \right\rangle\!\!\right\rangle^{\frac{1}{2}} \tag{2.9}$$

The double brackets used here are Amit's notation [1, pg 187] for the average over an ensemble of memory sets, each randomly generated according to a given probability distribution. For example, if there are P memories in each memory set, and if all memories are equally likely, then the ensemble average of a quantity $G(\vec{\xi}^1 \cdots \vec{\xi}^P)$ is:

$$\left\langle\!\!\left\langle G(\{\vec{\xi}\}) \right\rangle\!\!\right\rangle \equiv 2^{-PN} \sum_{\{\vec{\xi}^1\}\cdots\{\vec{\xi}^P\}} G(\vec{\xi}^1 \cdots \vec{\xi}^P). \tag{2.10}$$

The ensemble average is included in the definition of $O_{\rm rms}$ so that it can be directly related to the probability distribution used to generate the memory set, rather than being unique for each set of memories.

In many of the calculations it is convenient to use the variance

$$V \equiv (O_{\rm rms})^2 \tag{2.11}$$

rather than $O_{\rm rms}$. Both $O_{\rm rms}$ and V range from 0 for exactly orthogonal memory sets to 1 for random memory sets. For memory sets which are correlated, such as those discussed in [5], they are both larger than unity.

2.3.1 Generating Near-Orthogonal Memory Sets

A "near-orthogonal" set of memories is a group of memories in which typical overlaps are smaller than they would be if the memories were randomly selected. One way to generate a set of near-orthogonal memories $\{\xi\}$ is to start with a set $\{\Gamma\}$ of *exactly* orthogonal memories and then progressively add random noise. Memories Γ^μ and Γ^ν are exactly orthogonal if and only if they satisfy the following equation:

$$\sum_{i=1}^{N} \Gamma_i^\mu \Gamma_i^\nu = N\delta^{\mu\nu}.$$

If b is the probability that a particular bit ξ_i^μ is not equal to Γ_i^μ, then the probability distribution of memory bits ξ_i^μ is:

$$P(\xi_i^\mu) = (1-b)\,\delta(\xi_i^\mu - \Gamma_i^\mu) + b\,\delta(\xi_i^\mu + \Gamma_i^\mu), \tag{2.12}$$

where $\delta(a-b) \equiv \delta^{ab}$. The value of b ranges from 0 (no noise) to 0.5 (maximum noise).

In order for Equation 2.12 to be useful, it is necessary to first have a set Γ^μ of exactly orthogonal memory vectors. Section A.1 describes how to generate one such set for any value

of N which is an integer power of 2. It is a "complete" set, in that the number of vectors is equal to the number of available dimensions. Complete orthogonal sets for other values of N are not possible because of the constraint that vector elements must be either +1 or -1.

The expected overlap between a state ξ^μ and its corresponding Γ^μ is:

$$\langle\langle \sum_{i=1}^{N} \xi_i^\mu \Gamma_i^\mu \rangle\rangle \;=\; \sum_{\{\xi\}} P(\xi_i^\mu)\xi_i^\mu\Gamma_i^\mu \;=\; (1-2b) \tag{2.13}$$

Consider now the average mean and variance of the overlaps between two different states ξ^μ and ξ^ν, as derived in Section A.2. The mean overlap is found to be zero, because there is no reason that a positive overlap should be more likely than a negative overlap. The variance in the overlap is, from Equation A.4, equal to:

$$\langle\langle \xi^\mu \cdot \xi^\nu \rangle\rangle \;=\; N\left[1-(1-2b)^4\right]$$

from which it follows immediately that:

$$V \;=\; 1-(1-2b)^4 \tag{2.14}$$

or

$$b \;=\; \frac{1}{2}\left[1-(1-V)^{\frac{1}{4}}\right]. \tag{2.15}$$

It is a simple matter, using Equation 2.15, to generate sets of memories for which V has a particular value. This was tested in simulations and proven to be accurate, as shown in Figure 2.2.

2.3.2 Generating Correlated Memory Sets

Amit [1][5] generates correlated memories by decreasing the probability of each neuron being +1. He uses a parameter a which varies from 0 (random memories) to 1 (completely correlated

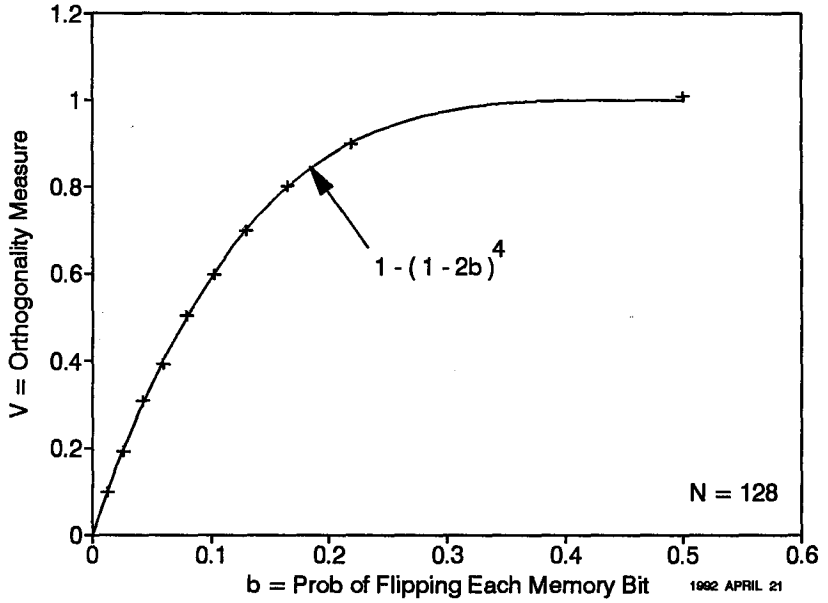


Figure 2.2: Measured variance in the overlaps between near-orthogonal memories, compared with the predicted variance. The line is a graph of Equation 2.14. The data points are from simulations of 100 memory sets, each with 128 neurons. The memories were generated by adding noise to exactly orthogonal memories, using Equation 2.12. The mean overlap between memories was measured and was close to zero.

memories), and writes the probability distribution as follows:

$$P(\xi) = \frac{1}{2}(1+a)\delta(\xi-1) + \frac{1}{2}(1-a)\delta(\xi+1) \tag{2.16}$$

The mean activation of the neurons is a, and the average overlap between two different memories is Na^2.

If a correlated memory set with zero mean activation is needed, it can be formed by making a cosmetic change to Amit's method. After each new memory is generated using Equation 2.16, a fair coin is flipped. If the coin comes up heads, every bit in the memory is inverted, otherwise the memory is left unchanged. This process creates memory sets with zero average activation, but with connection strengths J_{ij} which are exactly as they were using Amit's process, because of the symmetry in Equation 2.3.

If the mean activation is zero for Amit's correlated memories, then the variance V in the

overlaps can be used as a measure of the degree of orthogonality, as in Equation 2.11. The value of V is related to a as follows:

$$\begin{aligned}
\langle\langle V \rangle\rangle &= \frac{1}{N}\langle\left(\sum_{i=1}^{N}\xi_i^\mu\xi_i^\nu\right)^2\rangle \\
&= \frac{1}{N}\langle\sum_{i=1}^{N}\sum_{j=1}^{N}\xi_i^\mu\xi_i^\nu\xi_j^\mu\xi_j^\nu\rangle \\
&= \frac{1}{N}\langle\sum_{i=1}^{N}1+\sum_{i=1}^{N}\sum_{j\neq i}\xi_i^\mu\xi_i^\nu\xi_j^\mu\xi_j^\nu\rangle \\
&= \frac{1}{N}\left[N+\langle\sum_{i=1}^{N}\xi_i^\mu\xi_i^\nu\rangle\langle\sum_{j\neq i}\xi_j^\mu\xi_j^\nu\rangle\right] \\
&= 1+a^4(N-1) \qquad (2.17)
\end{aligned}$$

or

$$a = \left(\frac{V-1}{N-1}\right)^{\frac{1}{4}}.$$

2.3.3 Benefits of Near-Orthogonal Memory Sets

A memory ξ^ν is stable if the inputs U_i have the same sign as the neural states ξ_i^ν. This is equivalent to the following condition:

$$\begin{aligned}
0 &< U_i\xi_i^\nu && \forall i \\
0 &< \textstyle\sum_{j=1}^{N} J_{ij}\,\xi_i^\nu\,\xi_j^\nu && \forall i \\
0 &< \textstyle\sum_{\mu=1}^{P}\xi_i^\mu\xi_i^\nu\sum_{j=1}^{N}\xi_j^\mu\xi_j^\nu - P && \forall i \\
0 &< (N-P) + N\textstyle\sum_{\mu\neq\nu}\xi_i^\mu\xi_i^\nu m^{\mu\nu} && \forall i
\end{aligned} \qquad (2.18)$$

11.5

The first and second terms in this last equation can be thought of as the "signal" and the "noise", respectively. The sign of the noise term is random, so memories will tend to go unstable when the magnitude of the noise approaches $(N-P)$. This implies that smaller overlaps $m^{\mu\nu}$ will allow more memories to be stored before retrieval begins to deteriorate. The EH network is designed to capitalize on this fact.

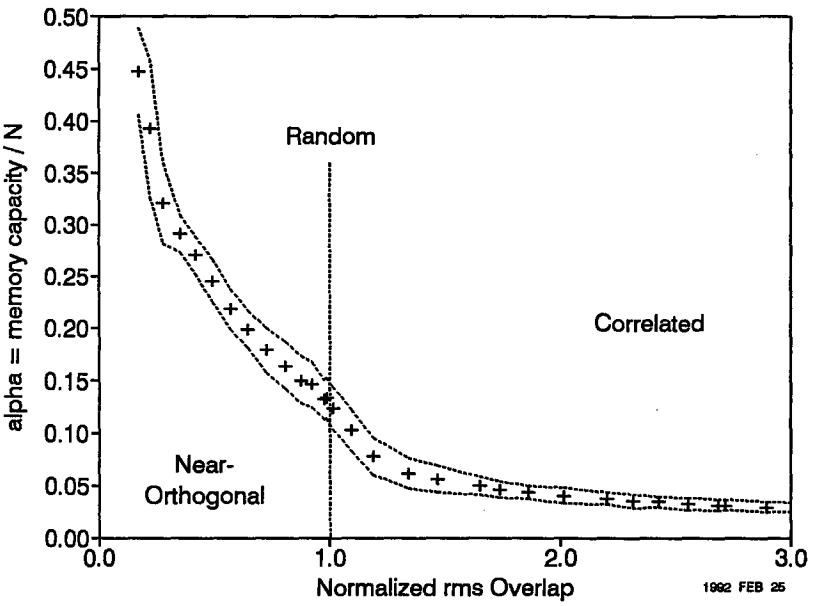


Figure 2.3: Memory capacity of a Hopfield network as a function of the level of orthogonality in the memory set. The orthogonality measure O_{rms} is defined in Equation 2.9. The memory capacity α_c is $\frac{1}{N}$ times the average number of memories which can be stored before one twentieth of the stored memories are no longer stable. Simulations were done on 100 different sets of random memories for each data point. The network consisted of 128 neurons. The dashed outlines show the standard deviation in memory capacities.

The derivation of an analytical relationship between storage capacity and O_{rms} is quite complex and will be postponed until Chapter 4, but it is appropriate in this chapter to present simulation results confirming that memory capacity improves with orthogonality. Figure 2.3 shows that the memory capacity increases dramatically as the rms overlap decreases, as expected. Also included in the figure are measurements of the network capacity for correlated memories, generated using the zero-mean-overlap version of Amit's process described in Section 2.3.2.

In addition to the need for large memory capacities, it is also important that neural network have basins of attraction which are as large as possible. Cottrell [10] showed that, for any network which uses the outer product rule, the radius of attraction is optimal if all the memories

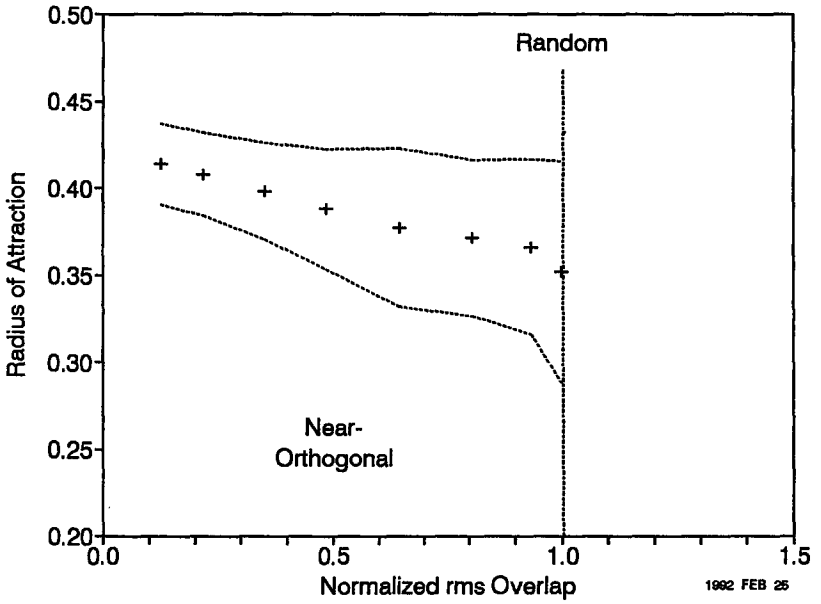


Figure 2.4: Radius of attraction of a Hopfield network after 10 memories have been stored, as a function of the level of orthogonality in the memory set. The normalized rms overlap O_{rms} is defined in Equation 2.9. The radius of attraction was measured as for Figure 2.3. Simulations were done on 100 different sets of random memories for each data point, using a network with 128 neurons. The dashed outlines show the standard deviation in radii of attraction. The memory capacity of the network falls below 10 just above ($O_{\text{rms}} = 1$), so it did not make sense to calculate the radii of attraction above that point.

are orthogonal. This suggests that the radii of attraction of the memories will improve as memory sets become nearly orthogonal. Figure 2.4 confirms that this is true, but the improvement in radius of attraction is not dramatic over the range that is plotted.

2.4 Working Premise: All Orthogonalizing Processes are Equivalent

Two different methods for generating near-orthogonal memory sets are investigated in this thesis. The first, which was introduced in Section 2.3.1, is quite compatible with mathematical analysis, but could not be used for storing real memories. The second, which will be introduced in Chapter 3 and is the core process in the EH network, lends itself very naturally to real-world implementations, but would be very difficult to analyse mathematically. The mathematical

analysis of EH networks therefore proceeds under the following working assumption:

- *ansatz*: the dependence of network behaviour on the orthogonality of the memory set is independent of the manner in which that orthogonality was achieved.

Chapter 3

The Extended Hopfield Network

This chapter introduces the detailed design of the Extended Hopfield ("EH") network, and presents the results of a number of tests of its capacity and reliability. Some of these results have been published in earlier papers by Hoffmann and Davenport [12, 13].

The central design challenge is to find mechanisms for storing and retrieving memories, which make efficient use of hidden neurons. The following three points are of central importance in determining what direction to proceed:

- To add hidden neurons to a Hopfield network, some process must be used to determine the states of the hiddens before the outer product rule installs the memory (Section 1.1.3).
- More memories can be installed in a Hopfield network, and with better radii of attraction, if the installed memories are as nearly orthogonal to each other as possible (Section 2.3.3).
- Peaks in the energy landscape of a Hopfield network are located at states which are optimally orthogonal to all previously-installed memories (Section 2.1.5).

When stated in this way, the following conclusion seems obvious[1]:

> During memory storage, the states of the hidden neurons should be set in such a way that the network moves as close as possible to an energy peak. This will ensure

[1] This conclusion seems obvious in retrospect, but it wasn't obvious until Hoffmann pointed it out.

that the memory sets are as orthogonal as possible, which will improve the capacity and reliability of the network.

The technical details of how this is done are described in the following sections.

3.1 EH Network Terminology

The EH network requires some added notational conventions to accomodate the presence of hidden neurons, and the use of neurons with three possible states. These are introduced in subsections 3.1.1 through 3.1.5.

3.1.1 Hidden Neurons in an EH Network

Hidden neurons in an EH network are treated identically to visible neurons wherever possible. Each is fully connected to every other neuron, for example, and each obeys the same dynamical rules that the visibles obey. Just as in a Hopfield net, the hidden and visible neurons in an EH network behave in a manner which is completely independent of their position in the network, so for notational simplicity, the first R neurons are assumed to be visible, and the last $N - R$ neurons are assumed to be hidden. A state vector can be written as follows:

$$S^1 = \underbrace{+-+---+-++++-++-+--+--+}_{R \text{ visibles}} \underbrace{-++--+++--+--+++}_{(N-R) \text{ hiddens}}.$$

The letter M is used to represent the number of hidden neurons:

$$M \equiv (N - R)$$

3.1.2 Partial Overlaps in the EH Network

The overlap $m^{\mu\nu}$ between memories ξ^μ and ξ^ν can be split into two components in an EH network. The overlap between the *visible* neurons is written $m_{\rm v}^{\mu\nu}$ and is defined as:

$$m_{\rm v}^{\mu\nu} \equiv \frac{1}{R}\sum_{i=1}^{R}\xi_i^\mu\xi_i^\nu$$

The overlap between the *hidden* neurons is written $m_{\rm h}^{\mu\nu}$ and is defined as:

$$m_{\rm h}^{\mu\nu} \equiv \frac{1}{M}\sum_{i=R+1}^{N}\xi_i^\mu\xi_i^\nu$$

The total overlap between vectors is therefore:

$$m^{\mu\nu} = \frac{1}{N}\left(Rm_{\rm v}^{\mu\nu} + Mm_{\rm h}^{\mu\nu}\right)$$

3.1.3 Measuring "Stability" in an EH Network

The most common method for measuring the capacity of an attractor network is to measure how many stable memories can be stored. In a Hopfield network, the test for stability is to set all neurons equal to the memory being tested, and then allow the network to evolve using the update rule. If no evolution takes place, the network is stable.

In an EH network, stability is tested by setting only the *visible* neurons equal to the stored memories. The hidden neurons are treated as unknowns using one of the procedures discussed in Section 3.3. Memory recall then takes place in the normal way, and continues until an attractor is reached. If the state of the visible neurons after recall is equal to the memory state, then the memory is "stable".

3.1.4 "Level of Correlation" of Memory Sets

In Section 3.4, a number of tests are described in which memory sets with various statistical properties are tested. For those tests, the "level of correlation" of the memory set is determined by the rms overlap of the *visible* neurons only. So if a set of "random" memories are stored in a network with 50 hiddens and 50 visibles, it means that the visible bits have been selected at random, so:

$$(m_{\mathrm{v}}^{\mu\nu})_{\mathrm{rms}} = \frac{1}{R} \tag{3.1}$$

3.1.5 TriState Neurons Option

One of the options which will be discussed for the EH network is the use of tri-state instead of Hopfield's standard bi-state neurons. Tri-state neurons use the standard `+1` and `-1` states, and an additional "neutral" state `0`. A typical tri-state vector might look like:

$$S^1 = \underbrace{\texttt{+-+---+-++++-++-+--+--+}}_{R \text{ visible}}\ \underbrace{\texttt{-++00000-0+-000+}}_{(N-R) \text{ hiddens}}.$$

The advantage of using tri-state neurons is that when they are in the zero state, they have no influence on the other neurons, which is appropriate if, for example, the correct state of that neuron is completely unknown. The disadvantage, of course, is that tri-state neurons are more complicated, and are more difficult to implement as electronic hardware. It will be shown that everything possible with tri-state neurons can be achieved with bi-state neurons, albeit with somewhat more complicated procedures.

3.2 Storing Memories in an EH Network

An EH memory is stored using the outer product rule, just as in a Hopfield network, but before the memory is stored, the state of the hidden neurons is determined by letting the network "roll up" in the energy landscape. The rolling up process is very similar to the evolution procedures described in Section 2.1.2, except that the forward update rule (Equation 2.5) is replaced by the following "reverse update rule":

$$S_i = \begin{cases} -1 & \text{if } U_i > 0 \\ +1 & \text{if } U_i < 0 \end{cases} . \tag{3.2}$$

It is not difficult to prove that rolling up leads to an energy peak. From Equation 3.2, a stable state S satisfies $S_i = -\text{sign}(U_i)$. Using Equations 2.7 and 2.2, the energy E of a stable state S is therefore:

$$E = -\frac{1}{2}\sum_{i=1}^{N} U_i S_i = \frac{1}{2}\sum_{i=1}^{N} |U_i|$$

Inverting any bit j changes the energy to E':

$$E' = \frac{1}{2}\sum_{i=1}^{N} |U_i| - |U_j|,$$

which is always less than E, so the stable state is a point of (locally) maximum energy. The energy of a network with finite N is bounded from above, so a peak will always be encountered when rolling up.

The procedure for storing memories is shown schematically in Figure 3.1, and can be summarized as follows:

1. All known bits are set to their known values
2. All unknown or hidden neurons are set either to zero (if tri-state), or to +1 or -1 at random (if bi-state).

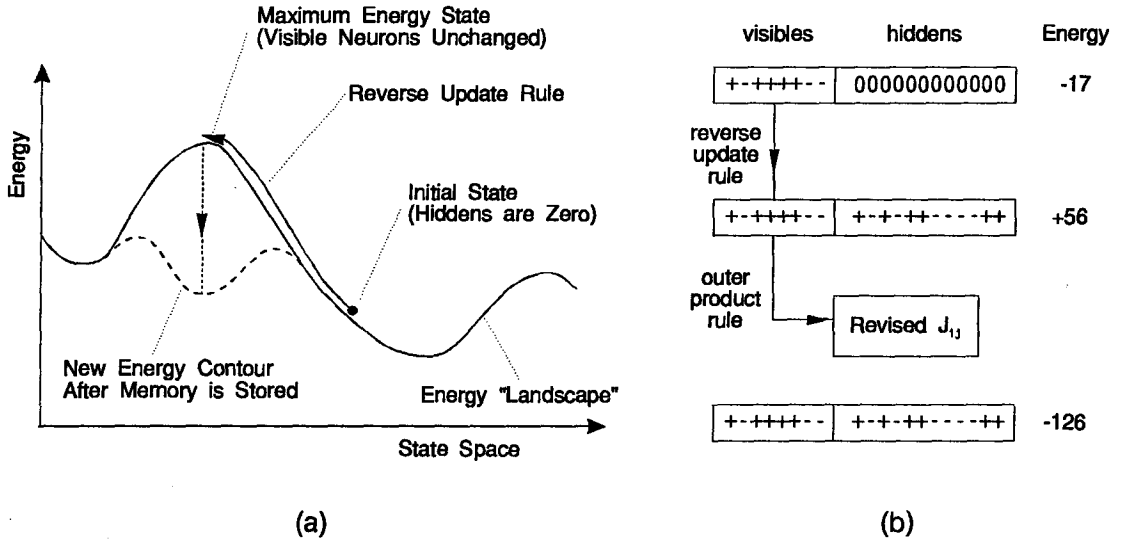


Figure 3.1: Schematic representation of the EH network memory storage process, (a) as a "roll-up" process in the energy landscape and (b) as an update process for bits in a state vector. The energy surface as a function of state space has peaks and valley-bottoms. The peaks are locally optimally orthogonal to the valleys. The location of a peak is found using the reverse update rule. The memory is installed, using the outer product rule, at the location of the peak, resulting in a new energy contour with an energy minimum at that point.

3. The reverse update rule (Equation 3.2) is applied, with the visible neurons clamped, until a stable state is reached. The term "clamped" means that the state of the neuron is not allowed to change.

4. The memory is installed using the outer product rule (Equation 2.3).

Note that when some of the neurons are clamped, the network evolves in a subspace of the complete network state space. The peaks that are reached in that subspace are not in general true peaks in the complete space, but experience has shown that they are close approximations to them.

A good indication of how well this memory storage process works is the degree of orthogonality that it achieves. Figures 3.2 and 3.3 show the rms overlap O_{rms} as a function of the

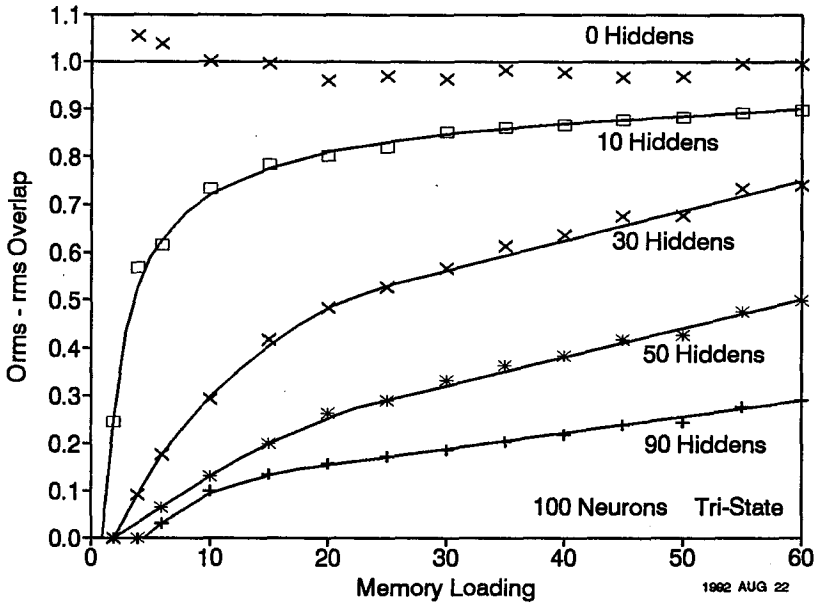


Figure 3.2: Degree of orthogonality achieved by the tri-state EH memory storage process, as a function of the number of memories added and for various numbers of hidden neurons. The visible neurons in each memory were set at random to either +1 or -1. The rms overlap O_{rms} was calculated using the complete memory vector, both visible and hidden, and was averaged over all pairs of memories that had been stored. The simulations were performed with a 100-neuron network, and each point represents the average result from 50 different sets of memories. The lines were drawn free-hand as a visual aid only.

number of memories installed and the number of hidden neurons available, for both tri-state and bi-state neurons. A comparison of the two figures reveals that tri-state and bi-state options are approximately equivalent. When no hiddens are present, the rms overlap is approximately unity, as expected. As a greater percentage of the neurons are hidden, the roll-up process is able to achieve better and better levels of orthogonality. This is examined again in Figure 3.4, where O_{rms} is plotted as a function of the number of hiddens.

3.3 Recovering Memories in an EH Network

During memory recovery in a standard Hopfield network, all bits are treated equally. There is no way to tell the network, for example, that you are very sure about bits 1 through 50, but

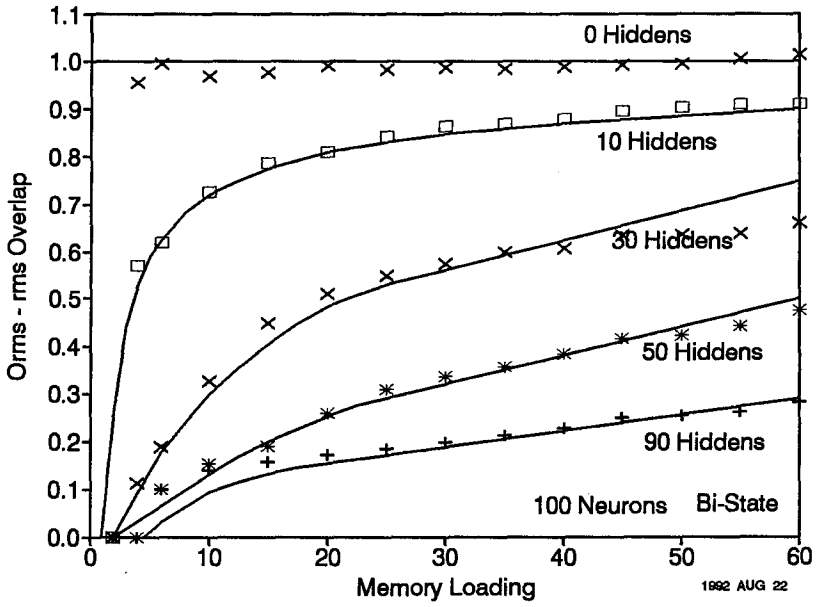


Figure 3.3: Degree of orthogonality achieved by the bi-state EH network memory storage process, as a function of the number of memories added, for various numbers of hidden neurons. The conditions for this simulation were identical to those in Figure 3.2. The lines are identical to those in Figure 3.2 to aid in comparison.

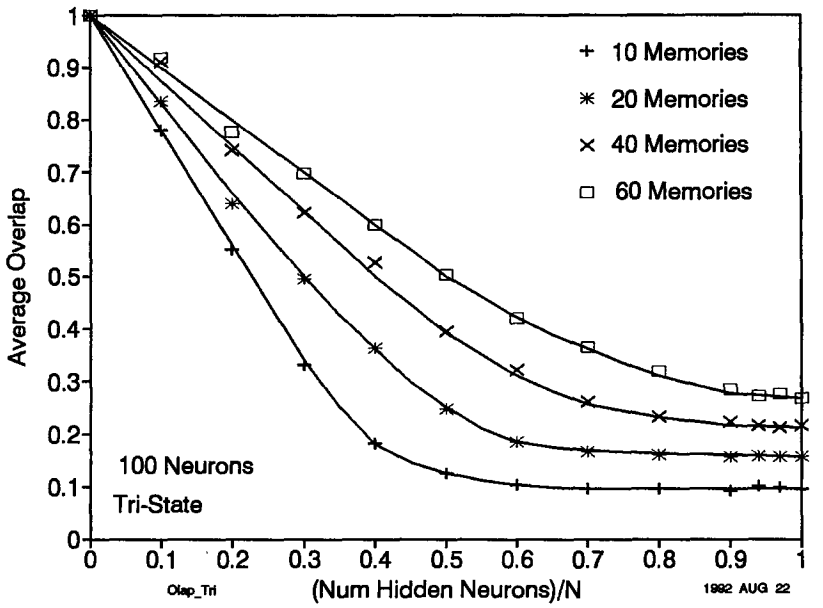


Figure 3.4: Degree of orthogonality achieved by the EH network memory storage process, as a function of the number of hidden neurons, for various levels of memory loading and for tri-state neurons. The conditions for this simulation were identical to those in Figure 3.2. The lines were drawn free-hand as a visual aid only.

have no information about bits 51 through 100. There are many applications, however, where the locations of the unknown bits is known. This is particularly true in an EH network, because the states of all of the hidden neurons are, by definition, completely unknown. It would also be true, for example, if a standard Hopfield network were being used to complete a bit-mapped image where the bottom half of the image was known to be obscured.

It would be useful to find a memory recovery process in which certain neurons could be flagged as "don't know", so that their initial values would have minimal influence on the final answer. Three different methods for doing this are discussed in Sections 3.3.1 through 3.3.3 and are compared in Sections 3.3.4 and 3.3.5.

All tests of the memory recovery process in Sections 3.3.1 through 3.3.5 use sets of memories which are randomly generated, with the bits equally likely to be +1 or -1. The tests described in Section 3.3.6 use near-orthogonal memory sets. None of the memory sets tested in these subsections were stored using the roll-up process.

3.3.1 "Random" Option

The first strategy, which is most commonly used with Hopfield networks, is to set all the "don't know" bits to +1 or -1 at random, and then apply the forward update rule (Equation 2.5) until an attractor is reached. This method relies on the randomly selected bits being uncorrelated with all the memories, so that their net influence is negligible. Unfortunately, when a large number of "don't know"s are present, or if the network is nearly at capacity, there is a high probability that the random bits will divert the network into a false attractor.

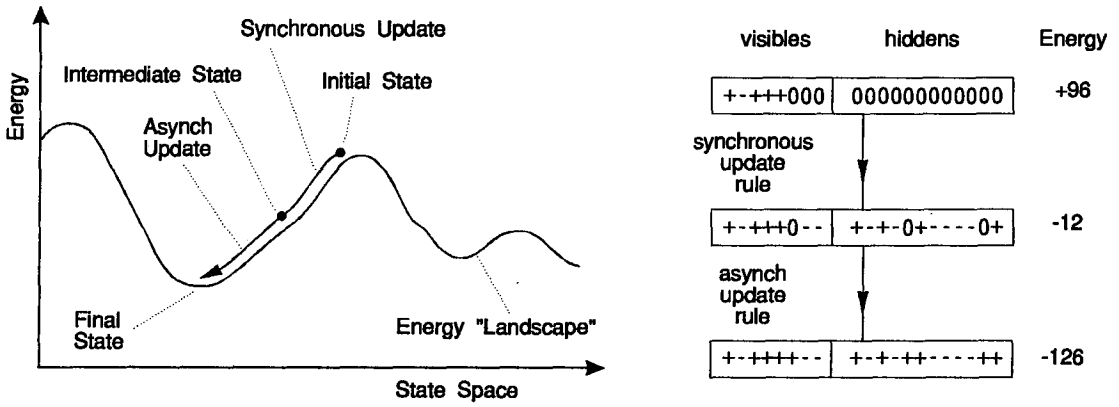


Figure 3.5: Schematic representation of the tri-state EH network memory retrieval process, (a) as a "roll-down" process in the energy landscape and (b) as an update process for bits in a state vector. Unknown bits, including all hidden neurons, are initially set to zero. The network then rolls down synchronously, with known neurons clamped, until the energy no longer decreases. All neurons are then unclamped, and the network rolls down asynchronously to an attractor state.

3.3.2 "TriState" Option

If tri-state neurons are used, a very successful memory recovery process is sketched in Figure 3.5, and can be summarized as follows [13, pg 136]:

1. All known neurons are set to their known values
2. All unknown or hidden neurons are set to zero
3. The synchronous forward update rule (Section 2.1.2) is applied, with all known neurons clamped. If some neurons are stuck in zero states, set one to `+1` or `-1` at random, and continue rolling down, until all neurons are non-zero and the energy no longer decreases.

4. The asynchronous forward update rule (Section 2.1.2) is applied, with no neurons clamped, until a stable state is reached.

It is an unfortunate added complexity that two different phases of roll-down are required, but tests show that both phases are needed. If asynchronous updates were used immediately, the first few neurons to be updated would exert disproportionate influences on the subsequent evolution of the network. If only synchronous updates were used, the network might settle into a limit cycle, and never reach a stable state.

3.3.3 "Bi-State" Option

If tri-state neurons are not used, comparable performance can be achieved with bi-state neurons, at the cost of an extra phase in the recall process. Bi-state recovery is shown schematically in Figure 3.6, and can be summarized as follows [12]:

1. All known bits are set to their known values
2. All unknown or hidden neurons are set to +1 or -1 at random.
3. The asynchronous reverse update rule (Equation 3.2) is applied, with known bits clamped, until a stable state is reached. This is the "roll-up" phase of bi-state memory recovery.
4. The synchronous forward update rule (Section 2.1.2) is applied with known bits clamped, until the energy no longer decreases.
5. The asynchronous forward update rule (Section 2.1.2) is applied with no bits clamped, until a stable state is reached.

The usefulness of step 3 is not immediately obvious, Isn't rolling up to a peak counter-productive when the goal is to arrive in a valley? And why will the peak be any closer to the

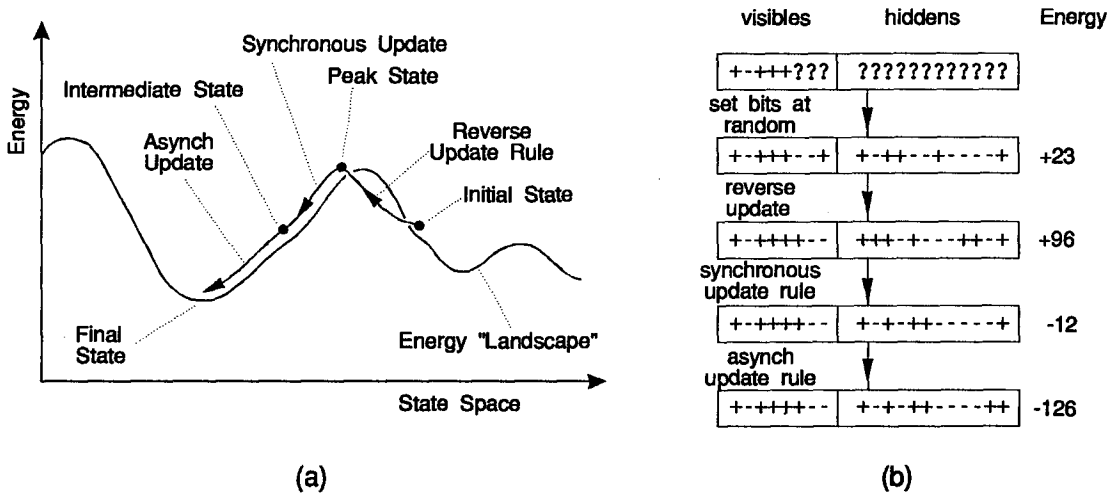


Figure 3.6: Schematic representation of the bi-state EH network memory retrieval process, (a) as a "roll-up-then-down" process in the energy landscape and (b) as an update process for bits in a state vector. Unknown bits, including all hidden neurons, are initially set to +1 or -1 at random. The network rolls up asynchronously using the reverse update rule, with known neurons clamped, until a peak is reached. It then rolls down synchronously, with known neurons clamped, until the energy no longer decreases, and finally, with all neurons unclamped, it rolls down asynchronously to an attractor state.

desired valley than to any other valley?

The explanation can be found using the following thought experiment. Let A be the known states of the visible neurons, and let B be the states of the hidden neurons when the network is at the peak, so the complete network state can be written $S^1 = (A + B)$. For example:

$$S^1 = \underbrace{\texttt{+-+---+-++++-++-+--+--+}}_{A}\ \underbrace{\texttt{-++--+++--+--+++}}_{B}.$$

By the symmetry of J_{ij}, the bit-inverse state $S^2 = (\overline{A} + \overline{B})$:

$$S^2 = \underbrace{\texttt{-+-+++-+----+--+-++-++-}}_{\overline{A}}\ \underbrace{\texttt{+--++---++-++---}}_{\overline{B}}$$

must also be a peak, and as such its overlap with all memories is as small as possible. Now invert all the visible neurons A:

$$S^3 = \underbrace{+-+---+-++++-++-+--+--+}_{A}\ \underbrace{+--++---++-++---}_{\overline{B}}.$$

The resulting state $(A + \overline{B})$ still has approximately zero overlap with all the memories, except for the one whose visible neurons are A. Its overlap with that memory is very large, which implies that the state $(A + \overline{B})$ is an ideal starting point from which to recover the desired memory.

Now consider what happens in the real network. When step 4 of the recovery begins, the network is in state $(A+B)$. All the hidden inputs U_i are the negative of the states S_i, because it is a peak. That means that the first synchronous update will immediately invert all the hidden neurons, and the network will be in state $(A + \overline{B})$, which was just shown to be a good starting point for memory recovery.

3.3.4 Retrieval Efficiency with 10 Memories Stored

Figure 3.7 shows the measured retrieval efficiency for a network in which 10 random memories have been stored. No roll-up procedure was used when the memories were stored. The figure indicates that the "tri-state" and the "bi-state" retrieval processes are comparable to each other, and are both superior to the "random" process. With the tri-state and bi-state options, there is a good chance of recalling a memory even when only 10 bits are known.

3.3.5 Radii of Attraction for Random Memories

Figure 3.8 shows how the radius of attraction of a typical memory varies as more and more random memories are added to the network. No roll-up procedure was used when the memories

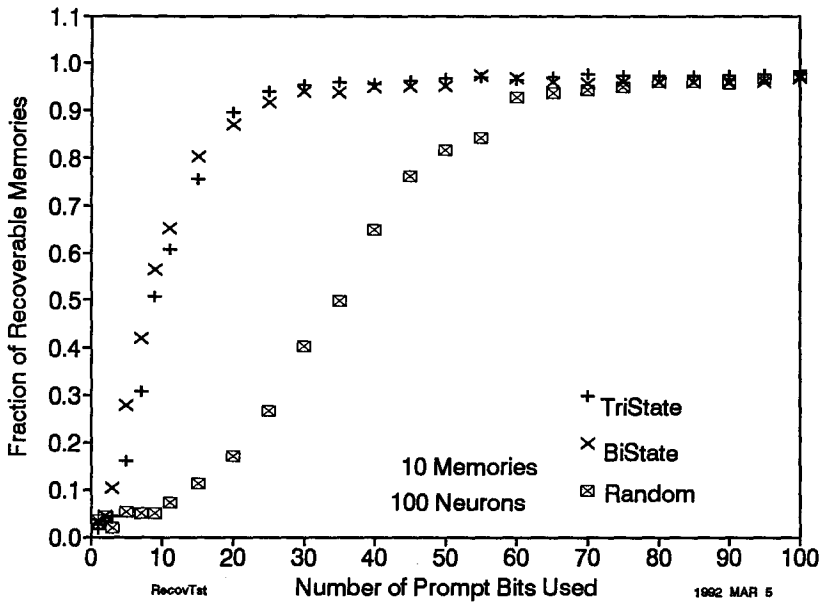


Figure 3.7: Comparison of three ways to handle "don't know" bits during memory recovery. In all three cases, ten random memories were stored in a standard Hopfield network. The network was then initialized to one of the memories ξ^μ, except that x of the bits were treated as "don't know" according to one of the three methods listed in the text, and memory recovery was performed. The graph plots the fraction of times that the recovered memory was exactly equal to ξ^μ. Fifty different memory sets were tested, and all 10 memories of each set were tested.

were stored. The figure shows that the radius of attraction gets smaller as memories are added. This is partly unavoidable, because when there are more memories to choose from, more information is required to specify which one is desired. It is also partly because the memory space becomes cluttered with false attractors, and this makes the *effective* number of memories much larger.

The first of these reasons can be quantified as described in Section B.1. Equation B.1 indicates that the average number of bits required to unambiguously specify one memory out of a set of P is:

$$M_{\min} = \log_2 4(P-1), \tag{3.3}$$

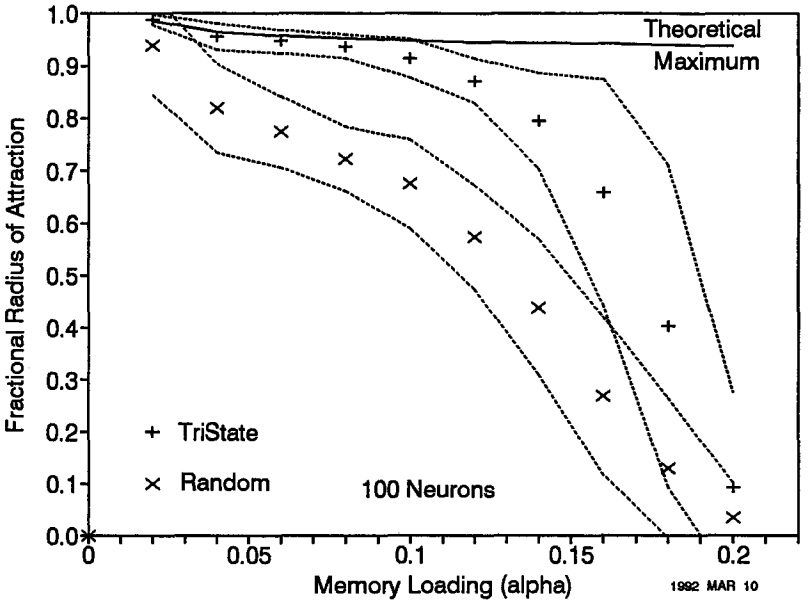


Figure 3.8: Comparison of radii of attraction for tri-state and random memory recovery procedures. Memory loading α is the number of stored memories divided by the number of neurons. The stored memories were completely random – no roll-up procedure was used to orthogonalize them. The fractional radius of attraction is $\frac{1}{N}$ times the number of bits which were necessary and sufficient to recover a memory, when the other bits were either set to zero (for "tri-state") or to random values (for "random"). The "theoretical maximum" was calculated using Equation 3.4. The dashed lines mark the standard deviation in the observed radii of attraction.

so the theoretical maximum fractional radius of attraction of a memory is:

$$r_{\max} = 1 - \frac{1}{N}\frac{\log 4(P-1)}{\log 2}. \tag{3.4}$$

This value is included in Figure 3.8 for reference.

The figure confirms that the tri-state retrieval process is much more effective than the random process, especially as the memory loading approaches capacity around $\alpha = 0.14$. For $\alpha < 0.1$, the tri-state network's performance is comparable to the theoretical upper limit. Bi-state data is not plotted because it was indistinguishable from the tri-state data.

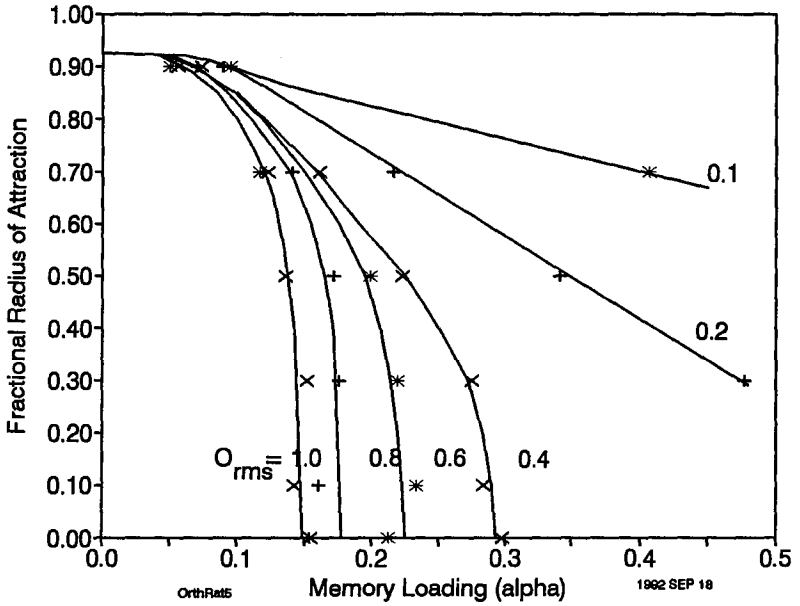


Figure 3.9: Radius of attraction as a function of memory loading for near-orthogonal memory sets, using the tri-state memory recovery procedure. Memory loading α is the number of stored memories divided by the number of neurons. Memory sets with normalized rms overlaps of 1.0, 0.8, 0.6, 0.4, 0.2, and 0.1 were generated as described in Section 2.3.1 and then stored. The fractional radii of attraction were calculated as the number of bits necessary and sufficient to recover 90% of the memories with fewer than 5% of the bits incorrect. Each data point is an average from 10 different memory sets. The lines were drawn freehand as visual aids only.

3.3.6 Radii of Attraction for Near-Orthogonal Memories

Figure 3.9 summarizes how the tri-state recovery process performs with near-orthogonal memory sets. The level of orthogonality makes no difference at low memory loads, but as the memory load increases, the radii of attraction of the more orthogonal networks are better.

3.4 Performance of the EH Network

The following subsections examine how well the EH network performs when the roll-up memory storage process described in Section 3.2 and the roll-up-then-down memory recovery process described in Section 3.3 work together. When they are both used some of the neurons can

be truly hidden, in that their states are neither imposed on the network when the memory is stored, nor used as prompts when the memory is recovered.

The presence of hidden neurons gives the EH network a number of advantages over a standard Hopfield network. Some of these are:

- More memories can be stored in a network of a given length, although the memories will be shorter.
- A network's capacity for memories of a given length can be increased
- The amount of stored information per neuron can be increased.
- The radii of attraction of stored memories can be increased.
- The XOR set of memories can be stored.
- Sets with memories of varying length can be stored.

These advantages are documented in Sections 3.4.1 through 3.4.6. In these tests, a memory is "stable" if it meets the criteria described in Section 3.1.3, and the "level of correlation" of a memory set is as defined in Section 3.1.4.

Unless otherwise noted, all simulations were done using tri-state neurons, using both the memory storage method from Section 3.2 and the memory recovery process from Section 3.3.

3.4.1 Memory Storage Capacity

An important advantage of EH networks is their ability to store more memories than a Hopfield network of similar size. The capacity of a network with a fixed number of neurons can be enhanced by designating some neurons as hiddens. Alternatively, the capacity of a network

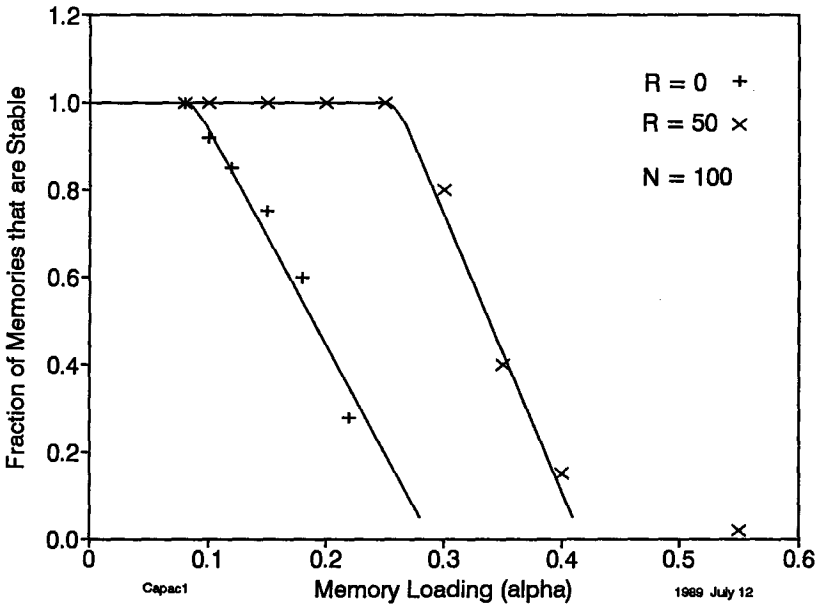


Figure 3.10: Comparison of an EH network with 50 hidden neurons and 50 visible neurons to a Hopfield network with 100 neurons. Random memory vectors were used for both networks, but the Hopfield memories were twice as long. Tri-state neurons were used for the EH network. Each point represents the average over five different sets of memories.

with fixed memory length can be increased by adding hidden neurons, and thus increasing the total number of neurons.

Figure 3.10 compares the recovery rates of a Hopfield network with 100 neurons to an EH network with 50 visible and 50 hidden neurons. The EH memories are only 50 bits long, but the EH network is able to store more than twice as many as the Hopfield network. In addition, memory stability remains 100% for up to 25 memories, as opposed to only 10 memories for the Hopfield network. If the "memory capacity" of each network is defined to be the load level for which the fraction of stable states falls below a certain threshold (say 0.9), then the Hopfield network has a capacity per visible neuron of 0.22 compared to 0.3 for the EH network.

Figure 3.11 shows the results of more extensive simulations where the memory capacity was measured as a function of R/N, the fraction of visible neurons in the network. The memory capacity increases steadily as R/N goes from 1.0 down to 0.5, because the network is using

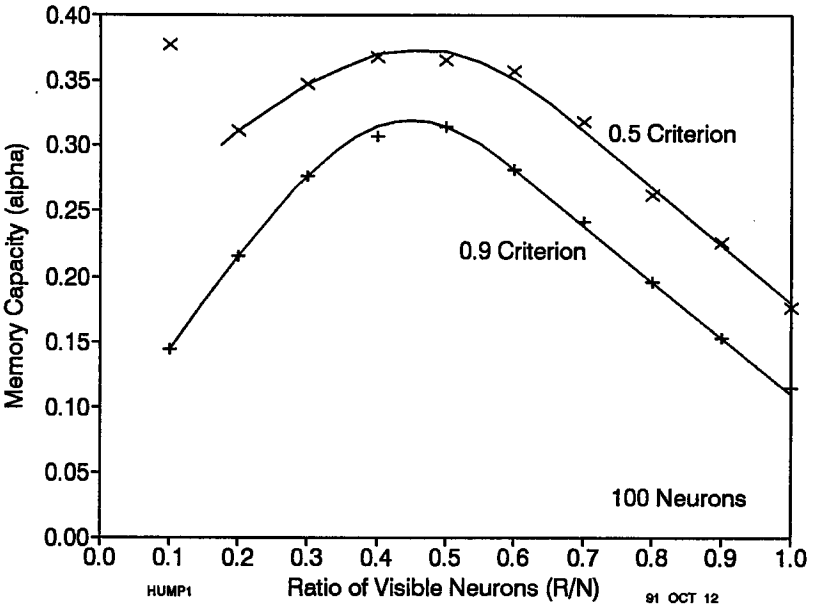


Figure 3.11: Network capacity as a function of the number of visible neurons. To measure the capacity, random memories were added one by one, using the EH storage method. After each memory was added, the number of stable memories was measured as described in Section 3.1.3. In this graph, the "memory capacity" is $\frac{1}{N}$ times the number of memories which can be stored before the fraction of stable memories fell below either 0.9 or 0.5. Each data point represents the average from ten different memory sets. The lines were drawn freehand as a visual aid only.

the hidden neurons to orthogonalize the memory space. As R/N becomes smaller than 0.5, however, the capacity decreases again, because it becomes more and more difficult to recall the memories with so few visible neurons.

3.4.2 Maximum Information Content

Even though more memories can be stored in a network with hidden neurons than in a standard Hopfield network, it is not immediately clear that the information capacity of the network increases or decreases. The reason for the confusion is that there are two competing effects as the ratio R/N decreases from unity:

- More memories can be stored, because the memory space can be orthogonalized.

- The amount of information in each memory decreases, because the hidden neurons contain no information.

The information capacity of an EH memory can be calculated quite easily. If the network has a total of N neurons, R of which are visible, and if the visible neurons are set at random, then the information content of the μth memory is

$$I_{EH}(\mu, R) \;=\; \log\left(2^R\right) - \log(\mu+1) \;=\; R\log 2 - \log(\mu+1). \tag{3.5}$$

The first term represents the information in the visible bits and the second term represents the information lost because the order in which the memories are stored is irretrievable. The mathematical justification for these terms is discussed in Section 6.1.1.

Figure 3.12 graphs the information capacity as a function of the ratio of visible neurons, and reveals that it can be increased by up to 45% by adding hidden neurons. In the limit of zero hiddens, the information capacity per neuron, per memory, is approximately 0.14 $\log 2$, which is consistent with results reported by Hopfield [25].

3.4.3 Correlated Memory Sets

A serious limitation in the performance of Hopfield networks is their very poor capacity for sets of correlated memories. The decrease in performance occurs because the correlations between the memories add together to form attractors which overwhelm the memories themselves. Amit has proposed a solution for one class of correlated memory sets, where the average activation of the bits is biased away from zero [5]. His solution involves restricting the networks during evolution to states which have a specified average activation.

The EH network's performance with correlated memories is of particular interest because its memory storage process very specifically tries to decorrelate the memory set. Simulations

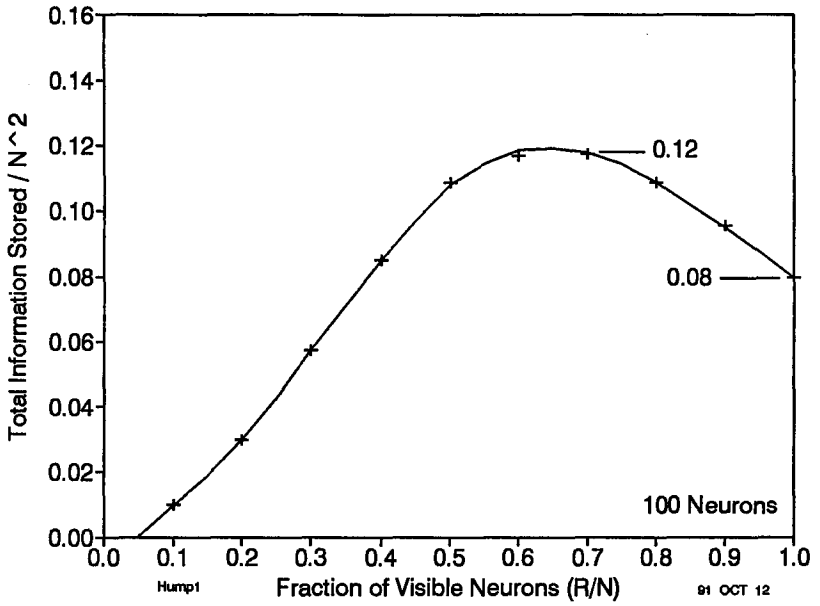


Figure 3.12: Information capacity of an EH network, as a function of the number of visible neurons used. Memory capacity was calculated using the "0.9 criterion" data described in Figure 3.11. The ordinate was then calculated as the number of memories stored, times the number of visible bits per memory, times $\log 2 \,/\, N^2$, so that the graphed value is equal to $\frac{\log 2}{N}$ times the number of data bits that can be stored for every neuron in the network. Each graph point is the average of tests from approximately 10 sets of memories. The line was drawn freehand as a visual aid only.

confirm that it is able to decorrelate the first few members of each memory set, and that this improves the storage capacity. Unfortunately, the decorrelating capacity of the hidden neurons is very quickly used up, so the performance of the network deteriorates quickly after the first few memories have been stored.

Figure 3.13 illustrates EH network capacity for memory sets with four different levels of correlation, where "level of correlation" refers only to the visible neurons, as discussed in Section 3.1.4. Notice that in the bottom left corner of the figure, where few memories are stored and many hidden neurons are available, memory capacities are the same for all levels of correlation. As the memory loading increases, the decorrelation process fails first for memory sets with the highest levels of correlation.

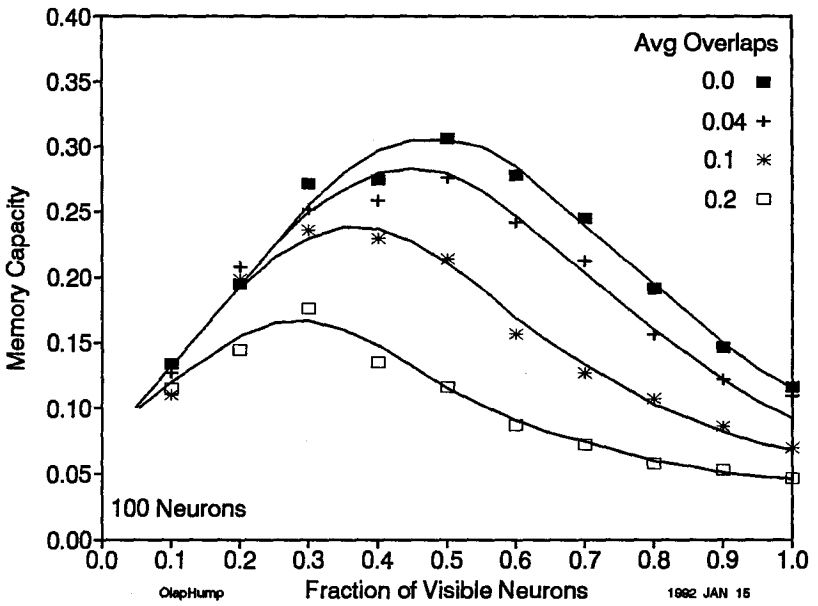


Figure 3.13: Storage capacity of an EH network for correlated memory sets with three different levels of correlation. The memory sets were generated as described in Section 2.3.2, and then stored and retrieved using a 100-neuron tri-state EH network. Memory capacity was calculated using the "0.9 criterion" data described in Figure 3.11. In the limit where the ratio of visible neurons is 1.0, this is the performance of a Hopfield network. Each point represents the average of 10 different memory sets. The lines were drawn freehand as a visual aid only.

Even when the EH network's performance is degraded because the memory set is correlated, it is still able to use the hidden neurons to advantage. An EH network with 50% hidden neurons can store more that twice as many memories as a Hopfield network, for example, when the memories have an average overlap of 0.1. Furthermore, the EH network has a greater capacity for correlated memories than the Hopfield network has for random memories.

3.4.4 Radii of Attraction

Most of the measurements of network capacity up to this point have focussed on the number of memories which can be stored before memories become unstable. To be *useful*, however, it is not good enough for a memory to be stable; it must also have a good radius of attraction.

In this section, two types of radii of attraction are measured:

1. **Type-1: Attraction from an incomplete prompt:** This is a measure of how many bits can be flagged as "don't know" (as discussed in Section 3.3) before the original memory can no longer be recovered.

2. **Type-2: Attraction from a noisy prompt:** This is a measure of how many bits can be set incorrectly, but not flagged as "don't know", before the original memory can no longer be recovered.

In both cases, the radius of attraction is equal to the number of visible bits which can be wrong (either "don't know" or inverted) before memory recovery begins to fail. If a network with 50 visible and 50 hidden neurons, for example, has a type-1 radius of "40", then recovery is possible when 40 of the visibles, as well as all 50 of the hiddens, are initially flagged as "don't know".

Figure 3.14 compares the type-1 radii of attraction of two different EH networks. The "100/0" network has 100 visible neurons and no hiddens, and the "50/50" network has 50 visibles and 50 hiddens. The 100/0 network is identical during memory storage to a 100-neuron Hopfield network. The radius of attraction shown in the figure could not, however, be achieved by a Hopfield network because it relies on the tri-state process described in Section 3.1.5.

The figure shows that the 50/50 network maintains a good radius of attraction for much higher memory loadings than the 100/0 network. This is not surprising, given that the 50/50 network has, according to Figure 3.11, a memory capacity of 30, while the 100/0 network's capacity is only 14. Figure 3.15 gives a head-to-head comparison of the two networks as they approach their nominal memory capacities. It shows that the 50/50 network has lost most of its basin of attraction by the time the 30th (and final) memory is added. By contrast, the 100/0 network still has a reasonable basin of attraction when the 14th memory is added.

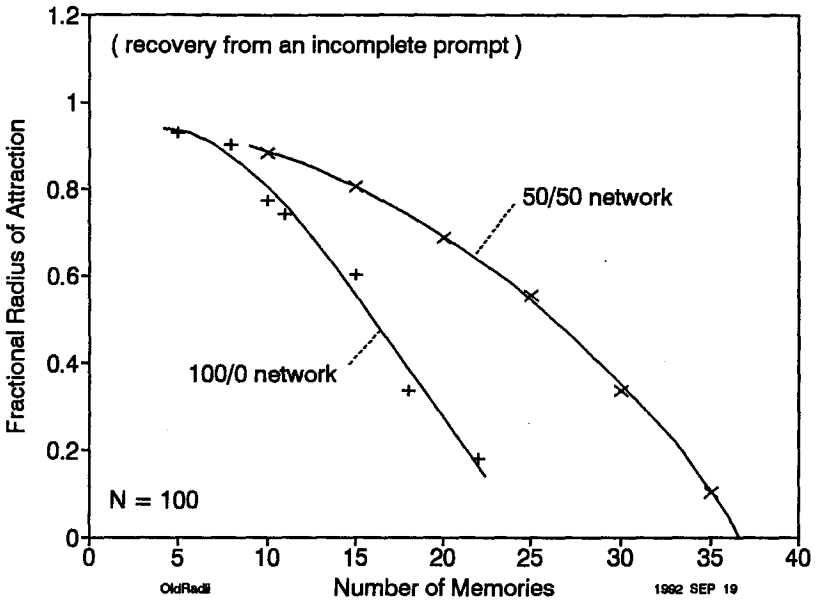


Figure 3.14: Type-1 radii of attraction of EH networks. The "100/0" network has 100 visible and 0 hidden neurons, and the "50/50" network has 50 visible and 50 hidden neurons. The ordinate is the maximum number of visible neurons which can be set to zero initially without hindering subsequent memory recovery, divided by the total number of visible neurons. Each marked point is the average from five different sets of memories. The solid lines were drawn freehand as a visual aid only.

The type-1 radii of attraction of the 50/50 network were not only larger than those of the 100/0 network, they were also more consistent. For each of the points shown in Figure 3.14, both the average and the standard deviation of the fractional type-1 radii of attraction were measured, although only the average radius was plotted. The standard deviations averaged 0.13 for the 50/50 network, and 0.20 for the 100/0 network. This may have an important impact on system reliability for some applications.

The type-2 radii of attraction of the 100/0 and the 50/50 networks are shown in Figure 3.16. The figure shows the average maximum number of visible neurons that can be initially set to the bit-inverse of the memory state, without interfering with memory recovery. In this case, because none of the visible neurons were flagged as unknowns, the 100/0 network was entirely identical to a 100-neuron Hopfield network.

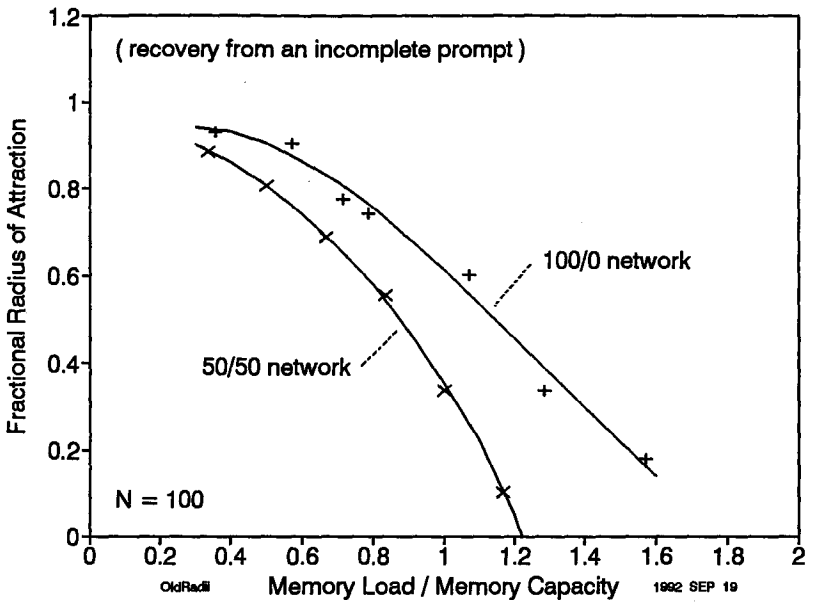


Figure 3.15: Type-1 radii of attraction of EH networks as a function of how near the network is to capacity. This figure is identical to Figure 3.14, except that the memory loads were divided by 14 for the 100/0 network, and by 30 for the 50/50 network.

A comparison of Figures 3.16 and 3.14 indicates that the type-2 radii of attraction are approximately one third the size of the type-1 radii. The type-2 basin of attraction of the 50/50 network is again larger for much higher memory loadings than the basin for the 100/0 network. The standard deviations of the fractional type-2 radii of attraction were measured and found to be approximately 0.08 for both the 50/50 and the 100/0 networks.

3.4.5 Storing the XOR Memory Set

One serious limitation of the Hopfield network is its inability to store the "not-equal" or "exclusive-or" (XOR) set of associations. This set of associations consists of the following

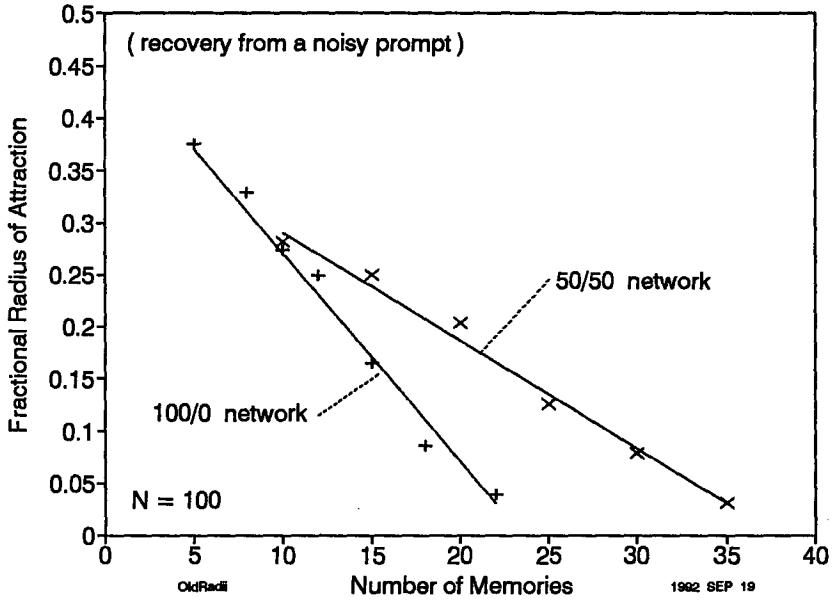


Figure 3.16: Type-2 radii of attraction of EH networks, using the same networks as in Figure 3.14. The plotted value is the maximum number of visible neurons which can be set to the inverse of the memory initially, without hindering memory recovery, divided by the total number of visible neurons. The 100/0 network can be thought of as either an EH network with no hidden neurons, or as a standard Hopfield network. Each marked point is the average from five different sets of memories. The solid lines were drawn freehand as a visual aid only.

four memories:

$$\begin{bmatrix} ++ & - \\ +- & + \\ -+ & + \\ -- & - \end{bmatrix} \qquad \begin{bmatrix} -- & + \\ -+ & - \\ +- & - \\ ++ & + \end{bmatrix}$$

(stored memories) (bit-inverse memories)

where the third bit of each stored memory is (-) if the first two bits are equal, and (+) if they are not. The goal is to store these memories in a network in such a way that when the first two "input" bits are provided, and the third "output" bit is flagged as "don't know", the network will roll down to an attractor in which the third bit is set correctly.

There are two severe problems with storing the XOR set in a Hopfield network:

1. For every stored memory of the XOR set (eg: -- -) with the correct answer in the third bit, a bit inverse attractor of a different memory (eg: -- +) is also stored which has the opposite answer in the third bit.

2. The sum of bits is zero in each of the three columns, so all the J_{ij}s are zero after the last memory has been stored.

The only conceivable solution with a standard Hopfield network is to add extra bits to each memory so that, for example, the memory set becomes:

$$\begin{bmatrix} \texttt{++ - +++++} \\ \texttt{+- + +++++} \\ \texttt{-+ + +++++} \\ \texttt{-- - +++++} \end{bmatrix} \qquad \begin{bmatrix} \texttt{-- + -----} \\ \texttt{-+ - -----} \\ \texttt{+- - -----} \\ \texttt{++ + -----} \end{bmatrix}$$

(stored memories) (bit-inverse memories)

The extra bits must be the same for all four memories, or else the user may end up "giving the answer" as part of the question. These extra bits clearly solve problem 2, because the bit inverse attractors can now be recognized by the incorrect values of the added bits.

Unfortunately problem 1 remains, and now applies to the extra bits as well. Because the extra bits bits are the same for all memories, the outer product rule

$$J_{ij} = \frac{1}{N} \sum_{\mu=1}^{4} \left(\xi_i^\mu \xi_j^\mu - \delta_{ij} \right),$$

zeroes every synapse that connects them to the "output" neuron, as can be readily confirmed by trying a few examples. The problem here is very similar to the problem of storing XOR in a single layer perceptron, as pointed out by Minsky and Papert [40].

The solution for the Hopfield network, just as for perceptron networks, is to add hidden neurons. The hidden neurons can do what the old extra neurons could not do, because the

hiddens change from one memory to another. This allows the J_{ij}s connecting the hiddens to the output neuron to be non-zero.

To test the EH network with the XOR set, the following four memories were stored, with varying numbers of hidden neurons:

$$\begin{bmatrix} + & ++ & - \\ + & +- & + \\ + & -+ & + \\ + & -- & - \end{bmatrix}$$

The first bit of each memory is a single symmetry breaking neuron, which is still required so that bit-inverse attractors can be avoided. The lower trace of Figure 3.17 shows the results of the tests, and indicates that approximately 11 hidden neurons are required before the correct answer is consistently achieved.

A disturbing feature of the XOR experiments was that even with quite large numbers of hidden neurons, the network occasionally made mistakes. Among 1200 tests of the network with 13 hiddens, for example, the network got the wrong answer 3 times. After careful tracing through of the network's behaviour, it was found that the errors always happened when the network reached a "plateau" in the energy landscape, where some of the hidden neurons had zero inputs, but were still in a zero state. The standard approach, at such a plateau, is to set one of the zero neurons to +1 or -1 at random and then continue.

The problem was solved by implementing a "tie-breaker heuristic" or "TBH". The input U_i to each neuron is the sum of $(N-1)$ signals $V_j J_{ij}$, not all of which have the same magnitude. If the sum of these signals is zero, the TBH counts whether there are more positive or more negative signals (ignoring their magnitude), and sets U_i to +1 or -1 respectively. With the TBH in place, only three hidden neurons were needed to store the XOR set, as shown in the upper trace of Figure 3.17. In 15,000 tests of the TBH network, with three hidden neurons or more,

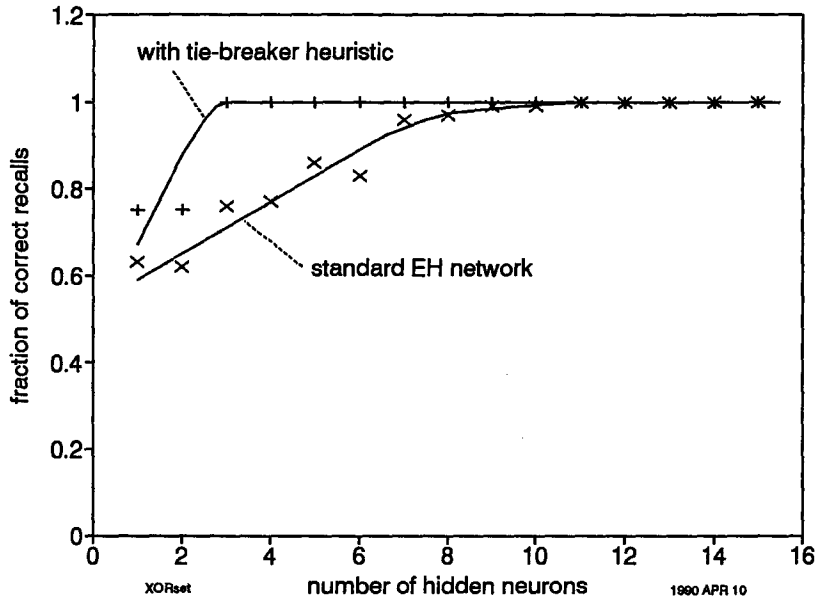


Figure 3.17: Retrieval rates for the XOR set of associations with varying numbers of hidden neurons. Each data point marks the average of a total of 1200 tests, during which the XOR set was stored 100 separate times. A test consisted of providing the two input bits, and then seeing if the network set the output bit correctly. For the lower trace, neural dynamics were as described in Section 3.3. For the upper trace, the tie-breaker heuristic was added, as described in the text. The lines were drawn freehand as a visual aid only.

zero errors occurred.

3.4.6 Flexible-Length Memories

The way that the EH network manages hidden neurons is ideally suited for applications where the memories are not all the same length. The network can be sized to accomodate the longest possible memory, and when a short memory is stored, the unused neurons can be put to good use orthogonalizing the network. The only minor hitch is that during recall, it may not be clear where the memory ends and the hidden neurons begin, but that can be solved quite simply as described below.

To demonstrate this and other capabilities of the EH network, a demonstration program was

written for use on a desktop computer. The program simulated a 162-neuron EH network, and stored memories which were entered and displayed as strings of characters. Each character was encoded as six bits, and the first six bits of each memory encoded the number of characters in the string, so strings of up to 26 characters could be entered. Any bits which were not required for the given character string were treated as hidden neurons.

A typical demonstration task was to store the titles of the works of Shakespeare. The titles ranged in length from 6 characters (*Hamlet*) to 24 characters (*Alls Well That Ends Well*), with an average size of about 16 characters. No systematic measurements were made of the performance of the network, but the following observations were made:

- All thirty five titles could be stored in approximately 95 seconds, when running on a 25 MHz '386 desktop computer.
- The network worked well with approximately 25 memories stored, in the sense that the radii of attraction were still very large. A prompt of "`Rome ?`", for example, would consistently elicit the response "`Romeo and Juliet`".
- Thirty titles could be stored without any becoming unstable, but the radii of attraction were noticeably reduced at that level of storage.
- When 35 memories were stored, approximately half the memories were no longer stable.
- The memories were surprisingly robust to "brain damage". Memories could be consistently recovered even when large percentages (70% to 90%) of the synapses were set to zero.

Chapter 4

The Free-Energy of a Network with Near-Orthogonal Memories

The most successful analytical tools for use with Hopfield-style neural networks are based on the statistical mechanics of Ising spin glasses [3, 4, 1, 38]. The formal ties between spin glasses and Hopfield networks were originally forged for networks in which all neurons were identical, and for memory sets which were completely random [3, 4]. Recent papers have extended the methods to include some forms of non-random memories [5, 43], and modifications to the learning rules [5].

Statistical mechanics methods are useful because they translate the "energy" landscape discussed in Section 2.1.4 into a "free energy" landscape, as described in Section 4.1. The energy landscape exists in the space of all possible states, and is uniquely determined by each set of memories that has been stored, as sketched in Figure 4.1(a). Its topology for one set of memories tends to bear no resemblance to its topology for another set of memories, so there is no "typical" energy landscape. The free energy landscape, on the other hand, exists in a space of order parameters, such as the overlap m^μ between a state S and memory ξ^μ, as sketched in Figure 4.1(b). Its topology tends to be very similar from one set of memories to another, so it can be averaged over all possible memory sets and then used to calculate properties of the network which are independent of the specific set of memories that has been installed.

This chapter derives the free energy for a network in which some of the neurons are clamped and in which the memory sets are near-orthogonal. The discussion and derivation proceed as

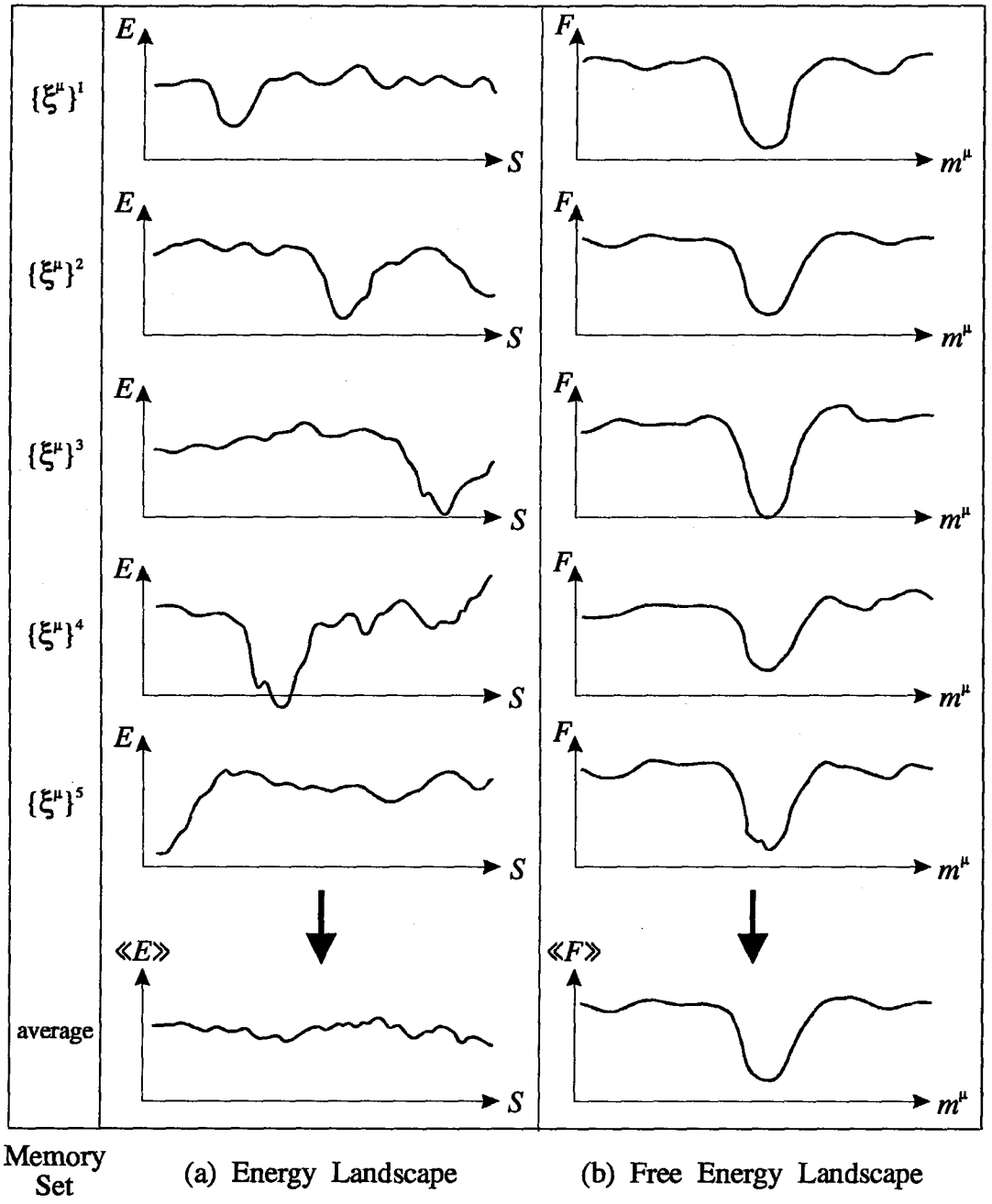


Figure 4.1: Conceptual sketch comparing *energy* landscapes to *free energy* landscapes. Energy landscapes are in the space of all possible states $\vec{S}$, and are completely different for every set of stored memories $\{\xi^\mu\}$. There is no interesting information in the average energy landscape. Free energy landscapes are in a space of order parameters such as m^μ, and tend to be similar from one set of memories to another. The free energy landscapes can be averaged to provide information about typical network behaviour, as discussed in Section 4.1.4.

follows:

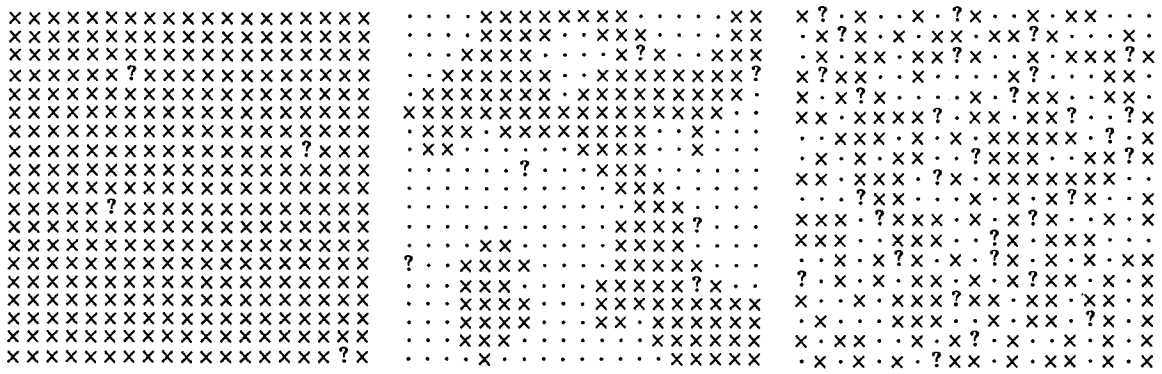
- Section 4.1 reviews the analogy between spin glasses and neural networks and justifies the use of a Gibbs distribution for the EH network with non-zero temperature.
- Section 4.2 derives a free energy equation for the most general case, and derives mean field equations for the expected overlap m^{μ}.
- Section 4.3 summarizes the results of the chapter.

4.1 Magnetic Lattice Analog for Neural Networks

Hopfield's neural network model [25] was intentionally designed to be analogous to a lattice of interacting magnetic dipoles. At low temperatures, each dipole in the lattice aligns itself with the total magnetic field at its location, which in turn is determined by the orientation of all the other dipoles. Such lattices have been shown [16, 50] to exhibit a "spin-glass" phase, in which the state of the lattice is stationary, but "frustrated", meaning that dipoles are aligned with some others, but misaligned with others.

One low-temperature stable state is the *ferromagnetic* state, where all dipoles are aligned in the same direction, as sketched in Figure 4.2(a). Each dipole is aligned not only with the local magnetic field, but also with dipoles which are very far from it.

Spin glass states occur at low temperatures when small groups of dipoles align themselves in one direction while other groups align themselves in the other direction, as in Figure 4.2(b). Although each dipole is aligned with the local field, it is anti-aligned with many of the dipoles in the lattice, so there is a high degree of frustration. These states are meta-stable (and hence glass-like) at finite temperatures, meaning that the lattice will stay in one state for a long period of time, but will eventually change to a new state.



(a) (b) (c)

Figure 4.2: Sketches of steady-state behaviour of (a) ferromagnetic, (b) spin-glass, and (c) paramagnetic phases of a two-dimensional lattice of Ising ferro-magnets. Sites marked × are spin-down, sites marked · are spin-up, and sites marked ? have recently changed orientation. Patterns in (a) and (b) are approximately stable in time, whereas the pattern in (c) is constantly changing.

At high temperatures, when the interactions between the particles are overwhelmed by thermal effects, the lattice is in a *paramagnetic* phase. The high temperature allows the dipoles to freely flip up and down, as sketched in Figure 4.2(c).

In a neural network at low loading, memory states resemble ferromagnetic states, in that few of the neurons are frustrated, and the energy of the network is low. In the analogy, the × symbols in Figure 4.2(a) represent neurons which are aligned with the recalled memory. Figure 4.3(a) is a sketch of the energy surface around a network in a ferromagnetic phase.

As memories are added, more and more false attractors appear, until eventually the space between memories is filled with large numbers of shallow false attractors, as sketched in Figure 4.3(b). A network in one of those states bears little resemblance to any of the memories, and has large numbers of frustrated neurons. If the "temperature" of the network is non-zero (as discussed in Section 4.1.2) then it will occasionally move between the false attractors, just like a spin-glass.

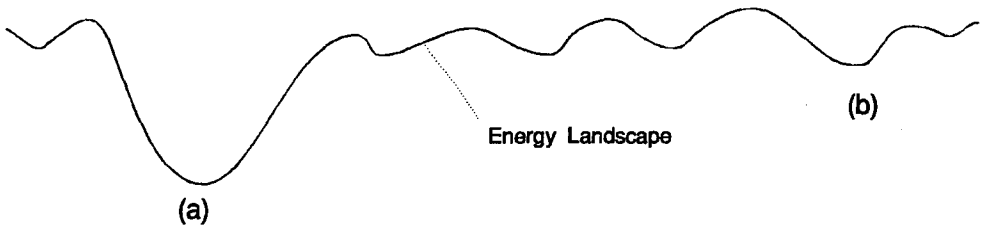


Figure 4.3: Sketch of the energy surface around (a) a ferromagnetic and (b) a spin-glass state. The ferromagnetic state is stable and strongly correlated to a memory. The spin-glass state evolves only very slowly, but has no recognizeable correlations to any memories.

4.1.1 Zero Temperature Ising Model

The simplest model which exhibits spin-glass phases is the Ising model of ferromagnetism [26]. It consists of N magnetic dipoles, each of which can be thought of as a charge spinning around an axis. The dipole moments are restricted to the $\hat{z}$ direction, so each lattice location i is allowed only two states, "spin up" ($S_i = 1$) or "spin down" ($S_i = -1$), The neural analog of the spin is the activation S_i of the ith neuron in a network, which is also allowed only two states, "on" ($S_i = 1$) or "off" ($S_i = -1$).

If an external field h_i is imposed at each lattice point i, then the total field at that point, U_i, is the sum of h_i and the sum of the magnetic fields from the other dipoles in the lattice:

$$U_i = h_i + \sum_{j=1}^{N} J_{ij} S_j, \tag{4.1}$$

where J_{ij} is the magnetic coupling coefficient between lattice locations i and j. The total energy of a network is equal to minus the sum of the dot products of the network states S_i with the local fields U_i:

$$E_i = -\sum_{i=1}^{N} \left[h_i S_i + \frac{1}{2} S_i \sum_{j=1}^{N} J_{ij} S_j \right] \tag{4.2}$$

The factor of $\frac{1}{2}$ is necessary because the potential energy between each interacting pair $S_i S_j$ is added twice. Note that this is the same as Equation 2.7, except that the external field h_i has

been added.

Equation 4.2 is only correct if the self-interaction terms J_{ii} are taken to be zero, and the magnetic coupling coefficients are symmetric ($J_{ij} = J_{ji}$). When the lattice is in thermal contact with a heat sink at absolute zero, the second law of thermodynamics ensures that E will decrease monotonically until a stable state is reached where virtually all the molecular dipoles are aligned with the local magnetic field.

Hopfield's model is analogous to the zero-temperature Ising lattice. The neural analog to the coupling coefficient J_{ij} is the synaptic strength J_{ij}, and the analog to the magnetic field U_i is the postsynaptic potential U_i. Hopfield was able to define the Lyapunov function or "energy" of his network using Equation 4.2 by imposing the restrictions ($J_{ii} = 0$) and ($J_{ij} = J_{ji}$), the latter of which certainly has no biological justification. The neural analog to the external fields h_i would be bias inputs provided at each neuron.

The major difference between magnetic lattice models and neural network models is the way that the J_{ij} are determined. In a magnetic lattice, they are determined by the proximity of the two lattice locations, and by the magnetic properties of the material forming the lattice. In the Sherrington and Kirkpatrick model [50, 30], for example, the values of J_{ij} are assumed to form a Gaussian distribution around zero. In a neural network, the J_{ij} are determined by a learning process, such as the outer product rule in Equation 2.3.

4.1.2 Non-Zero Temperature in Neural Models

Hinton and Sejnowksi [23] and Amit [3] extended Hopfield's model to networks in which neurons occasionally became mis-aligned with their postsynaptic potential. The appropriate analog for such a network is a magnetic lattice which is in contact with a heat bath at non-zero temperature. The temperature $1/\beta$ is a measure of the likelihood that individual molecular

dipoles will become misaligned with the local fields by absorbing energy from the heat bath. Following Glauber [18], if the Ising lattice is in contact with a heat bath at temperature $1/\beta$, and if all the dipoles except the ith are clamped in a state $\vec{S}$, then the probability that the ith dipole will be in state S_i is:

$$P(S_i|\vec{S}) = \frac{1}{2}\left[1 + \tanh\left(\beta S_i \tilde{U}_i\right)\right]. \tag{4.3}$$

In a neural network, this is the probability that neuron i will be in state S_i after it is updated, when all other neurons are kept fixed. In the zero temperature limit, $\beta = \infty$, this is identical to the update rule (Equation 2.5).

The "temperature" parameter β in neural network models determines the probability that a neuron will be in a state S_i which is contrary to its post synaptic potential U_i when the network is at equilibrium. It is not, of course, related to brain temperature. Following the magnetic analog, Glauber dynamics (Equation 4.3) are chosen to describe the temperature-dependence of the transition probability $P(S_i)$.

A neural network at non-zero temperature never reaches a perfectly stable state, or "fixed point". This lack of stability can be an advantage, because false attractors will no longer be fixed points of the system. The simulated annealing process [29], for example, uses non-zero temperatures in its search for the global minimum. When there are no fixed points, however, it is somewhat more difficult to recognize a recovered memory. The standard solution is to calculate the recovered memory as a time average $\langle\vec{S}\rangle$ of the network state.

At non-zero temperatures there is a finite probability that the network will evolve out of even the deepest attractor – a fact which seems to destroy the usefulness of the the network. If all final states are equally accessible from each initial state, then the system is ergodic, and there is no possibility of using the network to retrieve information. In practise, however, the time spent in an attractor is extremely long compared to the time spent to reach the attractor, so there is no problem recognizing the attractor. Analytically, ergodicity can be avoided by

carrying out all calculations in the thermodynamic limit, where the number of neurons goes to infinity and attractor states become infinitely deep wells in the energy landscape, so the probability of escaping goes to zero.

4.1.3 Justification for Using Thermodynamic Methods

When β is non-zero, the evolution of the neural network is stochastic and can no longer be described in terms of a Lyapunov function. Instead, if detailed balance [1, pg 109] is satisfied, the network can be modelled using the tools of statistical mechanics.

A system obeys detailed balance if transitions from state $\vec{S}^j$ to state $\vec{S}^k$ occur exactly as frequently as transitions from state $\vec{S}^k$ to state $\vec{S}^j$, in the thermodynamic limit. If $W(\vec{S}^j|\vec{S}^k)$ is the probability of going from state $\vec{S}^k$ to state $\vec{S}^j$, and $\rho(\vec{S}^k)$ is the probability of being in state $\vec{S}^k$, then this implies that:

$$W(\vec{S}^j|\vec{S}^k)\rho(\vec{S}^k) \;=\; W(\vec{S}^k|\vec{S}^j)\rho(\vec{S}^j). \tag{4.4}$$

If the system obeys Glauber dynamics (Equation 4.3), and if the only difference between $\vec{S}^j$ and $\vec{S}^k$ is that $S_i^j = -S_i^k$, then the requirement for detailed balance is:

$$\frac{W(\vec{S}^j|\vec{S}^k)}{W(\vec{S}^k|\vec{S}^j)} \;=\; \frac{P(S_i^j|\vec{S}^j)}{P(-S_i^j|\vec{S}^j)} \;=\; \frac{\exp(\beta\tilde{U}_i S_i)}{\exp(-\beta\tilde{U}_i S_i)} \;=\; \frac{\rho(\vec{S}^j)}{\rho(\vec{S}^k)}. \tag{4.5}$$

Any expression which satisfies 4.5 for all states $\vec{S}^j$ and $\vec{S}^k$ will suffice as an expression for $\rho(\vec{S})$. The natural choice is:

$$\rho(\vec{S}) \;\propto\; \exp\left(\beta\sum_{i=1}^{N}\tilde{U}_i S_i\right) \;=\; e^{-\beta E(\vec{S})}, \tag{4.6}$$

where $E(\vec{S})$ is Hopfield's energy for state $\vec{S}$, as in Equation 4.2. Note that this expression was derived without making any assumptions about the choice of connections J_{ij}, so it is equally applicable to the EH network as it is to the Hopfield network.

Equation 4.6 shows that finite-temperature neural networks conform to the Gibbs distribution of standard thermodynamics, with E playing the role of energy and β playing the role of inverse temperature. All the methods of statistical mechanics can therefore be applied to calculate the statistical behaviour of the networks at equilibrium. If changes in an observable O affect the energy of the network, then the mean value of O can be calculated using the partition function:

$$Z(h,\beta) = \sum_{\{S\}} \exp -\beta[E - hO(S)] \tag{4.7}$$

where the sum is over all accessible states of the network. Here h is the physical observable which must multiply O to calculate the energy. In a magnetic lattice, for example, if O is the magnetization, then h is the magnitude of an external field.

The time-average of O^n is:

$$\beta^n \langle O^n \rangle = \frac{1}{Z(h,\beta)} \frac{\partial^n Z(h,\beta)}{\partial h^n}\bigg|_{h=0} = -\beta \frac{\partial^n F}{\partial h^n}$$

$$\langle O^n \rangle = -\frac{1}{\beta^{n-1}} \frac{\partial^n F}{\partial h^n} \tag{4.8}$$

where

$$F \equiv -\frac{1}{\beta} \log Z \tag{4.9}$$

is the free energy.

In the following sections, F will be calculated as a function of a set of order parameters, such as $m^1 \ldots m^s$. This means that the sum in Equation 4.7 has been calculated over only those states which are compatible with the given order parameters. Such restricted sums are proportional to the probability that the network states are compatible with the given order parameters.

4.1.4 Quenched Random Memory States

There are, in a sense, two levels of randomness involved in the analysis of attractor neural networks: the random choice of a memory set $\{\xi^\mu\}$, and the random time-evolution of the neurons. To keep these levels distinct, the ξ^μs are treated as *quenched* variables. This means that, when time averages are being calculated, it is assumed that a single set of memories has been learned – no attempt is made to average over all possible memory sets. Equation 4.8, for example, can be used to calculate the time average of an observable for one specific set of memories $\{\xi^\mu\}$.

Once the time average is complete, a second average is calculated over an ensemble of all possible memory sets. From Equation 4.8, the ensemble average of a time-averaged observable O can be calculated as:

$$\langle\!\langle\, \langle O^n \rangle \,\rangle\!\rangle \;=\; \frac{1}{\beta^n}\left\langle\!\left\langle -\beta \frac{\partial^n F}{\partial h^n} \right\rangle\!\right\rangle \;=\; -\frac{1}{\beta^{n-1}} \frac{\partial^n \langle\!\langle F \rangle\!\rangle}{\partial h^n}. \tag{4.10}$$

where $\langle\!\langle\rangle\!\rangle$ is the same notation that was introduced in Section 2.3.

The central goal of the analysis which follows is therefore to calculate $\langle\!\langle F \rangle\!\rangle$, the ensemble average of the free energy F.

4.2 Ensemble Average of the Free Energy

In this section, the ensemble-averaged free energy $\langle\!\langle F \rangle\!\rangle$ is calculated for a network of N neurons, the first R of which are clamped, and the last $M \equiv N - R$ of which are not clamped. The ensemble average is calculated over all possible sets of memories $\{\xi^\mu\}$. The free energy F is a function of the overlaps m^μ, as discussed in the last paragraph of Section 4.1.3, so partition functions are summed only over states which are compatible with the given m^μ. The number

of unclamped neurons M is assumed to be proportional to N, with the ratio equal to γ:

$$\gamma \equiv \lim_{N\to\infty} \frac{M}{N} < 1. \tag{4.11}$$

In each memory set there are P memories $\vec{\xi}^1 \cdots \vec{\xi}^P$, which are near-orthogonal in the sense defined in Section 2.3.1. The probability $P(\xi_i^\mu)$ of a neural state ξ_i^μ is therefore:

$$P(\xi_i^\mu) = (1-b)\,\delta(\xi_i^\mu - \Gamma_i^\mu) + b\,\delta(\xi_i^\mu + \Gamma_i^\mu), \tag{4.12}$$

where Γ is a set of exactly orthogonal memories, and b determines the degree of orthogonality of ξ. Equivalently, the expected overlap between two memories ξ^μ and ξ^ν is:

$$\left\langle\!\!\left\langle \left(\sum_{i=R+1}^{N} \xi_i^\mu \xi_i^\nu \right)^2 \right\rangle\!\!\right\rangle = MV = M\left[1-(1-2b)^4\right], \tag{4.13}$$

as derived in Equation 2.14. The expression for the connection strength between neurons i and j is

$$J_{ij} \equiv \frac{1}{N} \sum_{\mu=1}^{P} \xi_i^\mu \xi_j^\mu - \alpha\delta_{ij}, \tag{4.14}$$

where the δ_{ij} term ensures that there are no self-interaction terms.

At each neuron S_i, let there be an external field h_i made up of P external fields which are proportional to the stored memories:

$$h_i = \sum_{\mu=1}^{P} h^\mu \xi_i^\mu$$

By increasing one of the h^μ, the network can thus be encouraged to recall one memory ξ^μ in particular.

4.2.1 The Energy per Neuron

The energy associated with the ith neuron is:

$$E_i = -\frac{1}{2} \sum_{j=1}^{N} J_{ij} S_j^\rho S_i^\rho - h_i S_i^\rho$$

$$= \frac{\alpha}{2} - \frac{1}{2N}\sum_{\mu=1}^{P}\sum_{j=R+1}^{N}\xi_i^\mu\xi_j^\mu S_i^\rho S_j^\rho - \sum_{\mu=1}^{P}\left[h^\mu + \frac{1}{2N}\sum_{j=1}^{R}\xi_j^\mu S_j^\rho\right]\xi_i^\mu S_i^\rho \qquad (4.15)$$

where the superscript ρ is used with the state $\vec{S}^\rho$ in anticipation of the case where more than one state will be considered in the same equation. As in Equation 2.4, the memory loading is represented by $\alpha = P/N$. It is convenient to work with $\mathcal{E}(S^\rho)$, the mean energy per neuron for a network in state S^ρ, rather than the energy of a specific neuron E_i. The mean is calculated over the last M neurons, which are the only ones participating in the dynamics:

$$\begin{aligned}
\mathcal{E}(S^\rho) &\equiv \frac{1}{M}\sum_{i=R+1}^{N} E_i(S^\rho) \\
&= \frac{\alpha}{2} - \frac{1}{2MN}\sum_{\mu=1}^{P}\sum_{i=R+1}^{N}\sum_{j=R+1}^{N}\xi_i^\mu\xi_j^\mu S_i^\rho S_j^\rho \\
&\qquad - \frac{1}{M}\sum_{\mu=1}^{P}\sum_{i=R+1}^{N}\xi_i^\mu S_i^\rho\hat{h}^\mu \\
&= \frac{\alpha}{2} - \frac{\gamma}{2}\sum_{\mu=1}^{P}\left(m_\rho^\mu\right)^2 - \sum_{\mu=1}^{P} m_\rho^\mu\hat{h}^\mu \qquad (4.16)
\end{aligned}$$

where

$$\hat{h}^\mu \equiv h^\mu + \frac{1}{2N}\sum_{j=1}^{R}\xi_j^\mu S_j \qquad (4.17)$$

and m_ρ^μ is the overlap between hidden neurons in state S^ρ and memory ξ^μ:

$$m_\rho^\mu \equiv \frac{1}{M}\sum_{i=R+1}^{N}\xi_i^\mu S_i^\rho. \qquad (4.18)$$

Not surprisingly, the only effect that the clamped neurons have on the dynamics of the unclamped neurons is to exert a constant external field, but that field will play a different role than the h^μ field when the limit $h^\mu \to 0$ is required (as in Section 5.1.2, for example). If S^ρ and ξ_i^μ are chosen at random, and $h^\mu = 0$, then the magnitude of $\hat{h}^\mu$ will be on the order of $\sqrt{M}/N = \sqrt{\gamma}/\sqrt{N}$.

4.2.2 Partition Function for n Replicas

The partition function for a network in which the average energy per neuron is given by Equation 4.16 is:

$$Z = \sum_{\{S\}} e^{-\beta M \mathcal{E}(\vec{S})}, \tag{4.19}$$

where the sum is over all accessible states S of the network. The ensemble average free energy per neuron is equal to:

$$\begin{aligned} \langle\langle f \rangle\rangle &= \frac{\langle\langle F \rangle\rangle}{N} \\ &= -\frac{1}{N\beta} \langle\langle \log Z \rangle\rangle, \end{aligned} \tag{4.20}$$

and is finite in the thermodynamic limit.

The ensemble average is calculated using the replica method [1, 35, 38],

$$\langle\langle f \rangle\rangle = -\frac{1}{\beta} \lim_{n \to 0} \lim_{N \to \infty} \frac{1}{Nn} (\langle\langle Z^n \rangle\rangle - 1) \tag{4.21}$$

where Z^n is the partition function of n non-interacting replicas of the neural network. Each replica has the same memories ξ^μ, and the same external fields h_i, but two replicas ρ and σ may be in different states S^ρ and S^σ. The replica method strategy is to find an analytical expression for the partition function Z^n as a function of n, and then calculate the formal algebraic limit ($n \to 0$). It is hard to imagine a physical interpretation of this limiting process – Amit refers to it as "an implausible exercise in a space of matrices with zero dimensions"[1, pg 337] – but it certainly is useful.

The exact expression for Z^n is:

$$Z^n = \left[\sum_{\{S\}} e^{-\beta M \mathcal{E}(S)} \right]^n$$

$$= \sum_{\{S^1\}} \cdots \sum_{\{S^n\}} \exp\left[-\frac{\beta\alpha\gamma N n}{2} + \frac{\beta\gamma^2 N}{2} \sum_{\rho=1}^{n} \sum_{\mu=1}^{P} \left(m_\rho^\mu\right)^2 + \beta\gamma N \sum_{\rho=1}^{n} \sum_{\mu=1}^{P} m_\rho^\mu \hat{h}^\mu \right]$$

$$= \sum_{\{S^1\}} \cdots \sum_{\{S^n\}} e^{-\frac{\beta\alpha\gamma N n}{2}} \prod_{\rho=1}^{n} \prod_{\mu=1}^{P} \exp\left[\frac{\beta\gamma^2 N}{2} \left(m_\rho^\mu\right)^2 + \beta\gamma N m_\rho^\mu \hat{h}^\mu \right],$$

where the sum is over all possible groups of n states.

The next step is to calculate the ensemble average $\langle\langle Z^n \rangle\rangle$. In the thermodynamic limit where $N \to \infty$, $\langle\langle Z^n \rangle\rangle$ will be dominated by the terms where the energy is minimum. In general there will be clusters of such terms around different memory states, and no single cluster of terms will dominate the sum. The external fields h^μ are used to break the symmetry between the attractors and force the solution to focus on the network's behaviour around a specific attractor. Assume that s of the external fields are non-zero. They might as well be the first s fields, $h^1 \ldots h^s$, because the numbering of the memories is arbitrary. If h^ρ is large enough, the attractor located near memories $\xi^1 \ldots \xi^s$ will certainly dominate $\langle\langle Z^n \rangle\rangle$. The corresponding memories $\xi^1 \ldots \xi^s$ are then referred to as condensed memories. Once the ensemble average has been calculated using the saddle point method, the external field can be removed by setting $h^\mu = 0$.

Following Amit [1, pg. 333] and Lautrup [35, pg 33], the ensemble average is split into two parts. An estimate of the amount of noise coming from the $(P - s)$ uncondensed memories is given by G_1:

$$G_1 \equiv \left\langle\left\langle \exp \sum_{\rho=1}^{n} \sum_{\mu=s+1}^{P} \left[\frac{\beta\gamma^2 N}{2} \left(m_\rho^\mu\right)^2 + \beta\gamma N m_\rho^\mu \hat{h}^\mu \right] \right\rangle\right\rangle_{\xi^{s+1} \ldots \xi^P} \tag{4.22}$$

where the $\xi^{s+1} \ldots \xi^P$ subscript indicates that the average is only over memory vectors $s + 1$ through P. The entire ensemble average can now be written:

$$\langle\langle Z^n \rangle\rangle = e^{-\frac{\beta\alpha\gamma N n}{2}} \left\langle\left\langle \sum_{\{S^1\}} \cdots \sum_{\{S^n\}} G_1 \exp \sum_{\rho=1}^{n} \sum_{\mu=1}^{s} \left[\frac{\beta\gamma^2 N}{2} \left(m_\rho^\mu\right)^2 + \beta\gamma N m_\rho^\mu \hat{h}^\mu \right] \right\rangle\right\rangle_{\xi^1 \ldots \xi^s} \tag{4.23}$$

4.2.3 Calculation of Average Over the Uncondensed Memories

This section calculates the average noise G_1 coming from the uncondensed memories, as defined in Equation 4.22. The quadratic m_ρ^μ terms in the exponent of G_1 can be removed using a Gaussian transform:

$$\exp\left(\frac{b^2}{4a}\right) = \sqrt{\frac{a}{\pi}}\int_{-\infty}^{\infty}\exp(-ax^2+bx)dx, \tag{4.24}$$

where:

$$\begin{aligned} a &= \frac{1}{2} \\ b &= \gamma\sqrt{\beta N}\, m_\rho^\mu \end{aligned}$$

This gives the following expression for G_1:

$$G_1 = \left(\frac{1}{2\pi}\right)^{\frac{(P-s)n}{2}}\int\!\!\cdots\!\!\int_{-\infty}^{\infty}\left(\prod_{\mu=s+1}^{P}\prod_{\rho=1}^{n}dx_\rho^\mu\right)\exp\sum_{\rho=1}^{n}\sum_{\mu=s+1}^{P}\left[-\frac{1}{2}(x_\rho^\mu)^2\right]G_1', \tag{4.25}$$

where

$$G_1' \equiv \left\langle\!\!\left\langle \exp\sum_{\rho=1}^{n}\sum_{\mu=s+1}^{P}\left[\left(\sqrt{\frac{\beta}{N}}x_\rho^\mu+\beta\hat{h}^\mu\right)\gamma N m_\rho^\mu\right]\right\rangle\!\!\right\rangle_{\xi^{s+1}\ldots\xi^p}. \tag{4.26}$$

The ensemble average in G_1' is calculated as follows;

$$G_1' = \left\langle\!\!\left\langle \prod_{i=R+1}^{N}\prod_{\mu=s+1}^{P}\exp\left[B_i^\mu\xi_i^\mu\right]\right\rangle\!\!\right\rangle_{\xi^{s+1}\ldots\xi^p} \tag{4.27}$$

where

$$B_i^\mu \equiv \sum_{\rho=1}^{n}\left[\left(\sqrt{\frac{\beta}{N}}x_\rho^\mu+\beta\hat{h}^\mu\right)S_i^\rho\right]$$

According to Equation 2.12, each ξ_i^μ must be either Γ_i^μ or $-\Gamma_i^\mu$, with probabilities $(1-b)$ and b respectively, so the average over all the ξs can be written as an average over all possible

choices of $\{\Gamma\}$:

$$\begin{aligned} G_1' &= \left\langle\!\!\left\langle A \prod_{i=R+1}^{N} \prod_{\mu=S+1}^{P} [(1-b)\exp(B_i^\mu \Gamma_i^\mu) + b\exp(-B_i^\mu \Gamma_i^\mu)] \right\rangle\!\!\right\rangle_\Gamma \\ &= \left\langle\!\!\left\langle A \prod_{i=R+1}^{N} \prod_{\mu=S+1}^{P} [\cosh B_i^\mu + (1-2b)\Gamma_i^\mu \sinh B_i^\mu] \right\rangle\!\!\right\rangle_\Gamma . \end{aligned} \tag{4.28}$$

The following expansion:

$$\log[\cosh x + a \sinh x] = ax + \frac{1}{2}(1-a^2)x^2 + O(x^3) \tag{4.29}$$

makes it possible to write G_1' as:

$$G_1' = A\left\langle\!\!\left\langle \exp \sum_{\mu=s+1}^{P} \sum_{i=R+1}^{N} \left\{(1-2b)\Gamma_i^\mu B_i^\mu + 2b(1-b)(B_i^\mu)^2 + O[(B_i^\mu)^3]\right\} \right\rangle\!\!\right\rangle_\Gamma . \tag{4.30}$$

In the limit $N \to \infty$, B_i^μ terms of order 3 and higher can be ignored, because $B_i^\mu \approx N^{-\frac{1}{2}}$. This leaves the following expression for G_1':

$$G_1' = A\exp\left\{\left[\sum_{\mu=s+1}^{P} \sum_{i=R+1}^{N} 2b(1-b)(B_i^\mu)^2\right] + \log\left\langle\!\!\left\langle e^{T_2} \right\rangle\!\!\right\rangle_\Gamma\right\}, \tag{4.31}$$

where

$$T_2 \equiv \sum_{\mu=s+1}^{P} \sum_{i=R+1}^{N} (1-2b)\Gamma_i^\mu B_i^\mu . \tag{4.32}$$

Digress here for a moment and consider the hypothetical case where $\{\Gamma\}$ is a set of random vectors. In that case, the parameter b should be irrelevent, because adding random noise to random vectors should have no effect on averaged behaviour. The value of $\left\langle\!\!\left\langle e^{T_2} \right\rangle\!\!\right\rangle$ is calculated for that case in Equation C.5 which, when replaced into Equation 4.31, gives the following expression for G_1':

$$G_1' = A\exp\left\{\frac{1}{2}\sum_{\mu=s+1}^{P} \sum_{i=R+1}^{N} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho}B^{\mu\sigma}S_i^\rho S_i^\nu\right\},$$

where

$$B^{\mu\rho} \equiv \left(\sqrt{\frac{\beta}{N}}x_\rho^\mu + \beta\hat{h}^\mu\right). \tag{4.33}$$

This expression is independent of b, which tends to confirm the accuracy of the derivation so far.

Returning to the main calculation, consider the case where $\{\Gamma\}$ is a set of exactly orthogonal vectors. The calculation of $\left\langle\left\langle e^{T_2}\right\rangle\right\rangle$, which is much more difficult for this case, is described in Appendix C.2. In that calculation, $\left\langle\left\langle e^{T_2}\right\rangle\right\rangle$ is first expressed as a function of the overlaps $\Gamma^1\cdot S^\rho$ through $\Gamma^s\cdot S^\rho$ (Equation C.9), and then it is averaged over all possible vectors Γ^1 through Γ^s (Section C.4). The result (Equation C.12) is:

$$\left\langle\left\langle e^{T_2}\right\rangle\right\rangle = \exp\left\{\frac{1}{2}\sum_{\mu=s+1}^{P}\sum_{\rho,\sigma=1}^{n}\left[(1-2b)^2-(1-V^{\frac{1}{4}})\sum_{\nu=1}^{s}(m_\rho^\nu)^2\right]B^{\mu\rho}B^{\mu\sigma}S^\rho\cdot S^\sigma\right\},$$

where the dot product refers to a sum over the unclamped indices only:

$$S^\rho\cdot S^\sigma \equiv \sum_{i=R+1}^{N} S_i^\rho S_i^\sigma$$

Replacing the expression for $\left\langle\left\langle e^{T_2}\right\rangle\right\rangle$ into Equation 4.31 yields the following value of G_1':

$$\begin{aligned} G_1' &= A\exp\left\{\sum_{\mu=s+1}^{P}\sum_{\rho,\sigma=1}^{n}\left[2b(1-b)+\frac{1}{2}(1-2b)^2-\frac{1}{2}(m_\rho^\nu)^2(1-V^{\frac{1}{4}})\right]B^{\mu\rho}B^{\mu\sigma}S^\rho\cdot S^\sigma\right\} \\ &= A\exp\left\{\frac{1}{2}\sum_{\mu=s+1}^{P}\sum_{\rho,\sigma=1}^{n}C^\rho B^{\mu\rho}B^{\mu\sigma}S^\rho\cdot S^\sigma\right\}, \end{aligned} \tag{4.34}$$

where

$$C^\rho \equiv 1-(1-V^{\frac{1}{4}})(m_\rho^\nu)^2. \tag{4.35}$$

and where V is the orthogonality measure defined in Equation 2.11.

The scaling factor C^ρ will be used in all the derivations which follow to encapsulate information about the degree of orthogonality of the memory set and how it influences the free energy.

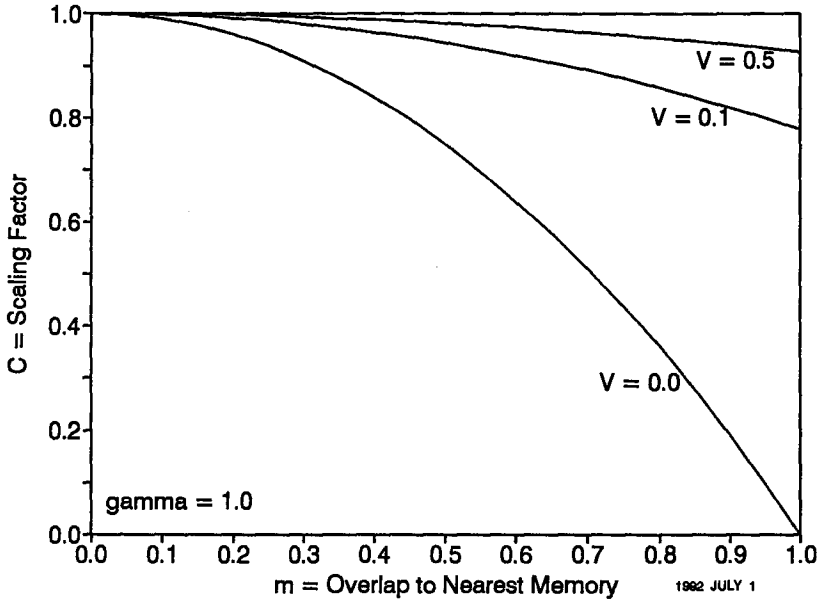


Figure 4.4: Plot of the scaling factor C^ρ as a function of the overlap m_ρ^ν, for three different degrees of orthogonality in the memory set. The ratio of visible bits, γ, is 1.0 for all three plots.

In the limit of random memories ($V = 1$), or zero correlation between the recalled memory and stored memories ($m_\rho^\nu = 0$), C^ρ becomes unity, and the equations reduce to the standard equations for a Hopfield network, as they should. In the limit of exactly orthogonal memories ($V = 0$), and exact recall ($m_\rho^\nu = 0$), C^ρ becomes zero and the noise from the uncondensed memories disappears, as it should. Figure 4.4 summarizes the variation in C^ρ over this range.

4.2.4 Inverse Gaussian Transforms

Now that the ensemble average over $\{\xi\}$ is complete, the Gaussian transform done in Section 4.2.2 must be reversed. In other words, G_1 must be integrated over the $z^{\mu\nu}$ and x_ρ^μ variables. Expanding Equation 4.34 in terms of x_ρ^μ:

$$G_1' = \left(\frac{1}{2\pi}\right)^{\frac{(P-s)n}{2}} \exp -\frac{1}{2} \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} \left\{ \left[\delta_{\rho\sigma} - \frac{\beta}{N} C^\rho \, S^\rho \cdot S^\sigma\right] x_\rho^\mu x_\sigma^\mu \right.$$

$$- \left[2C^\rho\beta\sqrt{\frac{\beta}{N}}\hat{h}^\mu\, S^\rho\cdot S^\sigma\right] x_\rho^\mu \; - \; \left[C^\rho\beta^2(\hat{h}^\mu)^2\, S^\rho\cdot S^\sigma\right] \Bigg\} \tag{4.36}$$

reveals the problem: there are mixed products of the integration variables x_ρ^μ in the exponent.

To remove the mixed products, a unitary transformation must be performed in the integration variables. This procedure is briefly described in [1, 35] for random memory sets. A modified form of the procedure that is suitable for near-orthogonal memories is described in Section C.6. The result of the inverse transform is:

$$G_1 \;=\; \exp\frac{N}{2}\left\{-\alpha\mathrm{Tr}(\log\mathbf{K}) \;+\; \tilde{H}\beta\sum_{\rho,\sigma=1}^{n}\left[\mathbf{K}^{-1}\right]_{\rho\sigma} - \beta\tilde{H}n\right\} \tag{4.37}$$

where $[\mathbf{K}]$ is the following $n \times n$ matrix:

$$[\mathbf{K}]_{\rho\sigma} \;\equiv\; \left(\delta_{\rho\sigma} - \frac{\beta}{N}C^\rho S^\rho\cdot S^\sigma\right) \tag{4.38}$$

and $\tilde{H}$ is the sum of the "crowd noises" from the clamped neurons:

$$\tilde{H} \;\equiv\; \sum_{\mu=s+1}^{P}(\hat{h}^\mu)^2. \tag{4.39}$$

The first term in the exponent of Equation 4.37 represents the noise from unclamped neurons in the uncondensed memories, and the second and third terms represent the noise from clamped neurons in the uncondensed memories.

4.2.5 Average Over Condensed Memories

The ensemble average over the condensed memories $\xi^1 \ldots \xi^s$ can now be calculated. Replacing Equation 4.37 into Equation 4.23 results in quite a complicated function of the overlaps m_ρ^μ, because of the m_ρ^μ term in C^ρ. It is therefore not possible to linearize the exponent using a Gaussian transform, as was done in Equation 4.25. Instead, all occurrences of m_ρ^μ and $S^\rho\cdot S^\sigma$

are replaced by new variables $\hat{m}_\rho^\mu$ and $q^{\rho\sigma}$ using the following δ-functions:

$$\begin{aligned}\delta\left(q_{\rho\sigma} - \frac{1}{M}S^\rho \cdot S^\sigma\right) &= \frac{\beta^2 P}{2\pi}\int_{-\infty}^{\infty} dr_{\rho\sigma} \exp\left[-\beta^2 \imath r_{\rho\sigma} P\left(q_{\rho\sigma} - \frac{1}{M}S^\rho \cdot S^\sigma\right)\right], \\ \delta\left(\hat{m}_\rho^\mu - m_\rho^\mu\right) &= \frac{\beta^2 P}{2\pi}\int_{-\infty}^{\infty} dy_\rho^\mu \exp\left[-\beta^2 P \imath y_\rho^\mu\left(\hat{m}_\rho^\mu - \frac{1}{M}\xi_i^\mu \cdot S_i^\rho\right)\right]. \end{aligned} \tag{4.40}$$

Terms that were previously functions of S^ρ and ξ^μ can now be written as functions of $\hat{m}_\rho^\mu$ and $q_{\rho\sigma}$:

$$\begin{aligned}\hat{C}^\rho &\equiv 1 - (\hat{m}_\rho^\nu)^2(1 - V^{\frac{1}{4}}) \\ [\hat{\mathbf{K}}]_{\rho\sigma} &\equiv \begin{cases} 1 - \beta\gamma\hat{C}^\rho & \forall\ \rho = \sigma \\ -\beta\gamma\hat{C}^\rho q_{\rho\sigma} & \forall\ \rho \neq \sigma \end{cases} \\ \hat{G}_1 &= \exp\frac{N}{2}\left\{-\alpha \mathrm{Tr}(\log\hat{\mathbf{K}}) + \tilde{H}\beta\sum_{\rho,\sigma=1}^{n}\left[\hat{\mathbf{K}}^{-1}\right]_{\rho\sigma} - \beta\tilde{H}n\right\} \end{aligned} \tag{4.41}$$

so that the averaged partition function, Equation 4.23, is equal to:

$$\begin{aligned}\langle\!\langle Z^n \rangle\!\rangle &= e^{-\frac{\beta\alpha\gamma N n}{2}}\left\langle\!\left\langle \sum_{\{S^1\}}\cdots\sum_{\{S^n\}}\int\int_{-\infty}^{\infty}\left(\prod_{\mu=1}^{s}\prod_{\rho=1}^{n} d\hat{m}_\rho^\mu dy_\rho^\mu\right)\int\int_{-\infty}^{\infty}\left(\prod_{\rho=1}^{n-1}\prod_{\sigma=\rho}^{n} dq_{\rho\sigma}dr_{\rho\sigma}\right)\right.\right. \\ &\times \left(\frac{\beta^2 P}{2\pi}\right)^{\frac{n(2s+n-1)}{2}} \hat{G}_1 \exp\sum_{\rho=1}^{n-1}\sum_{\sigma=\rho+1}^{n}\left[-\beta^2 \imath r_{\rho\sigma}\alpha N\left(q_{\rho\sigma} - \frac{1}{M}S^\rho \cdot S^\sigma\right)\right] \\ &\times \left.\left. \exp\sum_{\rho=1}^{n}\sum_{\mu=1}^{s}\left[\frac{\beta\gamma^2 N}{2}(\hat{m}_\rho^\mu)^2 + \beta\gamma N \hat{m}_h^\mu \hat{h}^\mu - \beta^2 \imath y_\rho^\mu \alpha N\left(\hat{m}_\rho^\mu - \frac{1}{M}\xi^\mu \cdot S^\rho\right)\right]\right\rangle\!\right\rangle_{\xi^1 \ldots \xi^s}. \end{aligned}$$

This leaves only two terms in the exponent which involve m_ρ^μ or S^ρ, so $\langle\!\langle Z^n \rangle\!\rangle$ can be written:

$$\langle\!\langle Z^n \rangle\!\rangle = \int\int_{-\infty}^{\infty}\left(\prod_{\mu=1}^{s}\prod_{\rho=1}^{n} d\hat{m}_\rho^\mu dy_\rho^\mu\right)\int\int_{-\infty}^{\infty}\left(\prod_{\rho=1}^{n-1}\prod_{\sigma=\rho}^{n} dq_{\rho\sigma}dr_{\rho\sigma}\right)\hat{G}_1\hat{G}_2 F_2, \tag{4.42}$$

where

$$F_2 \equiv \left\langle\!\left\langle \sum_{\{S^1\}}\cdots\sum_{\{S^n\}} \exp\left[\frac{\alpha\beta^2\imath}{\gamma}\left(\sum_{\rho=1}^{n-1}\sum_{\sigma=\rho+1}^{n} r_{\rho\sigma}S^\rho \cdot S^\sigma + \sum_{\mu=1}^{s}\sum_{\rho=1}^{n} y_\rho^\mu \xi^\mu \cdot S^\rho\right)\right]\right\rangle\!\right\rangle_{\xi^1 \ldots \xi^s},$$

and

$$\hat{G}_2 \equiv \left(\frac{\beta^2\alpha N}{2\pi}\right)^{\frac{n(2s+n-1)}{2}} \exp -N\left\{\frac{\beta\alpha\gamma n}{2} + \beta^2\alpha\sum_{\rho=1}^{n-1}\sum_{\sigma=\rho+1}^{n} \imath r_{\sigma\rho}q_{\rho\sigma}\right\}$$
$$\times \exp N\left\{\beta\sum_{\rho=1}^{n}\sum_{\mu=1}^{s}\left[\frac{\gamma^2}{2}(\hat{m}_\rho^\mu)^2 + \gamma\hat{m}_\rho^\mu\hat{h}^\mu - \beta\imath\alpha y_\rho^\mu\hat{m}_\rho^\mu\right]\right\}. \tag{4.43}$$

The ensemble average can be simplified in the standard way [1, pg. 336]:

$$\begin{aligned} F_2 &= \left\langle\!\!\left\langle \sum_{\{S^1\}}\cdots\sum_{\{S^n\}}\prod_{i=R+1}^{N} \exp\frac{\alpha\beta^2\imath}{\gamma}\left(\sum_{\rho=1}^{n-1}\sum_{\sigma=\rho+1}^{n} r_{\rho\sigma}S_i^\rho S_i^\sigma + \sum_{\mu=1}^{s}\sum_{\rho=1}^{n} y_\rho^\mu\xi_i^\mu S_i^\rho\right)\right\rangle\!\!\right\rangle_{\xi^1\ldots\xi^s} \\ &= \left\langle\!\!\left\langle \prod_{i=R+1}^{N}\sum_{S^1=\pm1}\cdots\sum_{S^n=\pm1} \exp\frac{\alpha\beta^2\imath}{\gamma}\left(\sum_{\rho=1}^{n-1}\sum_{\sigma=\rho+1}^{n} r_{\rho\sigma}S^\rho S^\sigma + \sum_{\mu=1}^{s}\sum_{\rho=1}^{n} y_\rho^\mu\xi_i^\mu S^\rho\right)\right\rangle\!\!\right\rangle_{\xi^1\ldots\xi^s} \\ &= \left\langle\!\!\left\langle \exp\sum_{i=R+1}^{N}\log\left\{\sum_{S^1=\pm1}\cdots\sum_{S^n=\pm1}\right.\right.\right. \\ &\qquad \left.\left.\left. \exp\frac{\alpha\beta^2\imath}{\gamma}\left(\sum_{\rho=1}^{n-1}\sum_{\sigma=\rho+1}^{n} r_{\rho\sigma}S^\rho S^\sigma + \sum_{\mu=1}^{s}\sum_{\rho=1}^{n} y_\rho^\mu\xi_i^\mu S^\rho\right)\right\}\right\rangle\!\!\right\rangle_{\xi^1\ldots\xi^s} \end{aligned} \tag{4.44}$$

Section C.8 shows that the expression inside the sum is self-averaging, even though the memory vectors are not entirely random. Let $\vec{T}$ be an s-dimensional column vector, with each element equal to either +1 or -1. The sum over i is proportional to an average over each of the 2^s possible vectors $\vec{T}_1$ through $\vec{T}_{2^s}$:

$$\begin{aligned} F_2 &= \exp N\left\langle\!\!\left\langle \log\sum_{S^1=\pm1}\cdots\sum_{S^n=\pm1}\right.\right. \\ &\qquad \left.\left. \exp\left[\alpha\beta^2\imath\left(\sum_{\rho=1}^{n-1}\sum_{\sigma=\rho+1}^{n} r_{\rho\sigma}S^\rho S^\sigma + \sum_{\mu=1}^{s}\sum_{\rho=1}^{n} y_\rho^\mu T^\mu S^\rho\right)\right]\right\rangle\!\!\right\rangle_{T_1\ldots T_{2^s}} \end{aligned} \tag{4.45}$$

4.2.6 Saddle Point Calculation

When the expressions for $\hat{G}_1$, $\hat{G}_2$, and F_2 are replaced into Equation 4.42, the result is a series of integrals over a single exponential in which the exponent is proportional to N. In the limit

of very large N, the integral is dominated by the value of the integrand at the point where the exponent of the integrand is maximum, so the saddle point method [1, pg. 138] can be used to solve the integral. Applying the saddle point method to Equation 4.42 gives:

$$\langle\langle Z^n \rangle\rangle \;=\; \hat{G}_1 \hat{G}_2 F_2,$$

where $\hat{G}_1$, $\hat{G}_2$, and F_2 are evaluated at the saddle point.

To find the saddle point, the following $\frac{1}{2}n(2s+n-1)$ equations must be solved simultaneously for $\frac{1}{2}n(2s+n-1)$ variables $q_{\rho\sigma}$, $r_{\rho\sigma}$, $\hat{m}_\rho^\mu$, and y_ρ^μ :

$$\left.\begin{aligned}
0 &= \frac{\partial \log \hat{G}_1}{\partial q_{\rho\sigma}} + \frac{\partial \log \hat{G}_2}{\partial q_{\rho\sigma}} && \{\forall\, \rho,\sigma : (1 \le \rho < n) \wedge (\rho < \sigma \le n)\} \\
0 &= \frac{\partial \log \hat{G}_2}{\partial r_{\rho\sigma}} + \frac{\partial \log F_2}{\partial r_{\rho\sigma}} && \{\forall\, \rho,\sigma : (1 \le \rho < n) \wedge (\rho < \sigma \le n)\} \\
0 &= \frac{\partial \log \hat{G}_1}{\partial \hat{m}_\rho^\mu} + \frac{\partial \log \hat{G}_2}{\partial \hat{m}_\rho^\mu} && \{\forall\, \mu,\rho : (1 \le \mu \le s) \wedge (1 \le \rho \le n)\} \\
0 &= \frac{\partial \log \hat{G}_2}{\partial y_\rho^\mu} + \frac{\partial \log F_2}{\partial y_\rho^\mu} && \{\forall\, \mu,\rho : (1 \le \mu \le s) \wedge (1 \le \rho \le n)\}
\end{aligned}\right\} \qquad (4.46)$$

Equations 4.46 only enforce the requirement that the free energy surface be flat. If the point of condensation, determined by the selection of h^μ, is chosen poorly, these equations may give a solution for a maximum or saddle point, rather than a minimum of the energy function. To confirm that the solution is a minimum, the second derivatives must be checked, a task which Lautrup found "quite complicated" [35, pg. 37], even for random memories with no clamped neurons, and which has not been done here.

4.2.7 Free Energy of the Replica Symmetric System

The established method for making the calculations manageable is to assume that the solutions will obey replica symmetry [1, 35, 38]. That is, assume that the values of $q_{\rho\sigma}$, $r_{\rho\sigma}$, $\hat{m}_\rho^\mu$, and

y_ρ^μ will be identical for all replicas. Under that assumption, the values of the variables at the saddle point can be written:

$$\begin{aligned} q_{\rho\sigma}\big|_{\text{saddle}} &= q \\ r_{\rho\sigma}\big|_{\text{saddle}} &= r \\ \hat{m}_\rho^\mu\Big|_{\text{saddle}} &= m_{\text{h}}^\mu \\ y_\rho^\mu\Big|_{\text{saddle}} &= y^\mu. \end{aligned}$$

This reduces the number of variables to $(2s+2)$, so Equations 4.46 reduce to $(2s+2)$ equations in q, r, m^μ, and y^μ.

Rather than n scaling factors C^ρ there is now only one:

$$C = 1-(1-V^{\frac{1}{4}})(m_{\text{h}}^\nu)^2, \tag{4.47}$$

where ν is the index of the memory which is most similar to the condensed state.

Sections C.9, C.10, and C.11 derive expressions for $\hat{G}_1$, $\hat{G}_2$, and F_2 for the replica symmetric case and for the limit of small n. When these are used in Equation 4.42, the result is:

$$\langle\!\langle Z^n \rangle\!\rangle = \exp\left[-nN\beta\,\langle\!\langle f \rangle\!\rangle\right] \tag{4.48}$$

where:

$$\begin{aligned} \langle\!\langle f \rangle\!\rangle = \Bigg\{ & \frac{\alpha}{2\beta}\log\left[1+\beta\gamma C(q-1)\right] - \frac{\alpha\gamma C q + 4\tilde{H}}{2\left[1+\beta\gamma C(q-1)\right]} \\ & + \frac{1}{2}\left[\tilde{H} + \alpha\gamma - \alpha\beta r(q-1)\right] - \sum_{\mu=1}^{s}\left[\frac{1}{2}\gamma^2(m_{\text{h}}^\mu)^2 + \gamma m_{\text{h}}^\mu \hat{h}^\mu - \beta\alpha y^\mu m_{\text{h}}^\mu\right] \\ & - \frac{1}{\beta}\left\langle\!\left\langle\!\left\langle \log 2\cosh\beta\left(\sqrt{\alpha r}z + \alpha\beta\sum_{\mu=1}^{s} y^\mu T^\mu\right)\right\rangle\!\right\rangle\!\right\rangle_{T_1 \ldots T_{2^s}} \Bigg\}. \end{aligned} \tag{4.49}$$

Replacing this into Equation 4.21 and using $(e^{nx} = 1+nx)$ in the limit $n \to 0$ confirms that the expression in Equation 4.49 is the average free energy per neuron.

Equations for q, r, m_{h}^{μ}, and y^{μ} are found by requiring that the partial derivatives of $\langle\langle f \rangle\rangle$ with respect to all variables are zero, as in Equations 4.46. This ensures a solution which is at a saddle point, but it does not ensure a minimum in the free energy. The calculations are described in Section C.12, and result in the following mean field equations:

$$m_{\mathrm{h}}^{\mu} = \left\langle\!\!\left\langle\!\!\left\langle T^{\mu} \tanh\beta \left[\sqrt{\alpha r}\, z + \sum_{\nu=1}^{s} \left(\gamma^2 m_{\mathrm{h}}^{\nu} + \gamma \hat{h}^{\nu} - G_3(q, m_{\mathrm{h}}^{\mu})\right) T^{\nu}\right] \right\rangle\!\!\right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \tag{4.50}$$

$$q = \left\langle\!\!\left\langle\!\!\left\langle \tanh^2\beta \left[\sqrt{\alpha r} z + \sum_{\mu=1}^{s} (\gamma^2 m_{\mathrm{h}}^{\mu} + \gamma \hat{h}^{\mu} - G_3(q, m_{\mathrm{h}}^{\mu})) T^{\mu}\right] \right\rangle\!\!\right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \tag{4.51}$$

$$r = \frac{\gamma C(\alpha\gamma C q + 4\tilde{H})}{\alpha\left[1 + \beta\gamma C(q-1)\right]^2} \tag{4.52}$$

where the $\langle\!\langle\!\langle \cdots \rangle\!\rangle\!\rangle$ symbol represents a combined ensemble average and an average over z using a Gaussian window, as defined in Equation C.57, and where

$$G_3(q, m_{\mathrm{h}}^{\mu}) \equiv \frac{\gamma^2 m_{\mathrm{h}}^{\mu}(1 - V^{\frac{1}{4}})}{\left[1 - \beta\gamma C(1-q)\right]^2} \left[\alpha - \alpha\beta\gamma C(1-q)^2 - 4\beta(q-1)\tilde{H}\right] \tag{4.53}$$

In the limits $V = 1$ and $M = N$, the network should behave as a standard Hopfield network, because there is effectively no correlation between the memories and the set of orthogonal memories. These values of V and m imply that $\gamma = 1$, $C = 1$, $\hat{h}^{\mu} = h^{\mu}$, $G_3(q, m_{\mu}) = 0$, and $\tilde{H} = 0$, so the mean-field equations become:

$$\begin{aligned} q &= \left\langle\!\!\left\langle\!\!\left\langle \tanh^2\beta \left[\sqrt{\alpha r} z + \sum_{\mu=1}^{s} (m_{\mathrm{h}}^{\mu} + h^{\mu}) T^{\mu}\right] \right\rangle\!\!\right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \\ r &= \frac{q}{\left[1 + \beta(q-1)\right]^2} \\ m_{\mathrm{h}}^{\mu} &= \left\langle\!\!\left\langle\!\!\left\langle T^{\mu} \tanh\beta \left[\sqrt{\alpha r}\, z + \sum_{\nu=1}^{s} (m_{\mathrm{h}}^{\nu} + h^{\nu}) T^{\nu}\right] \right\rangle\!\!\right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} . \end{aligned}$$

Thus the new equations are equivalent, in the limit, to the standard equations for a Hopfield network [1, pg. 293].

4.3 Summary of the Free-Energy Calculation

The calculations in Sections 4.1 and 4.2 can be summarized as follows:

- The update rule rule given in Equation 2.5 was modified so that neural evolution became probabilistic, with probabilities determined by Glauber dynamics (Equation 4.3). Neural states were allowed to evolve contrary to the postsynaptic potential with a probability which depends on a "temperature" parameter.

- The probability of a network at equilibrium and with temperature $\frac{1}{\beta}$ being in a state with energy E was shown to be proportional to $e^{-\beta E(\vec{S})}$.

- The free energy F was defined to be a function of a set of "order parameters" by setting it equal to the logarithm of the partition function Z, where Z is summed over only those states which are compatible with the given order parameters.

- The ensemble average partition function, Z^n, of a set of n identical networks (replicas) was calculated as a function of the order parameters. It was calculated in two parts: first for "uncondensed" memories, which are not associated with a large overlap order parameter m^μ, and second for "condensed" memories.

- It was shown that the "crowd noise" from the uncondensed memories diminishes in proportion to a scaling factor C^ρ (Equation 4.35) as the stored memory sets become more orthogonal.

- The exact expression for Z^n is given in Equation 4.42 in the form of a series of nested integrals of a complicated exponential function, and was solved using the saddle point method. The location of the saddlepoint is determined by a set of mean field equations in the integration variables, and can be translated into a set of mean field equations in the order parameters.

- The mean field equations were simplified by assuming that all the order parameters are identical for each of the n replicas ("replica symmetry"). The resulting equations, 4.50 through 4.52, are an important result in this chapter.

- The ensemble average free energy $\langle\langle f \rangle\rangle$ was calculated from Z^n using the standard replica "trick", as described in Equation 4.21.

Chapter 5

Analytical Estimate of Network Storage Capacity

This chapter derives an analytical estimate of the storage capacity of the EH network. The calculation is done in three stages:

- Section 5.1 calculates how many near-orthogonal memories can be successfully stored in a Hopfield network.
- Section 5.2 estimates the degree of orthogonality which can be achieved using the EH network's roll-up procedure, as a function of the number of memories installed and the number of hidden neurons.
- Section 5.3 combines these results to achieve an analytical estimate of the capacity of an EH network as a function of the number of hidden neurons.

5.1 Hopfield Network Capacity for Near-Orthogonal Memories

This section uses the mean-field equations derived in Section 4.2.7 to predict the memory capacity of a Hopfield network for near-orthogonal memory sets. The calculation parallels the standard solution for random memories [1, 35], but with some added complexities because of new terms in the mean-field equations.

5.1.1 Mean Field Equations at Zero Temperature

A very useful simplification of the mean-field equations (Equations 4.50, 4.51, and 4.52) is to take the zero-temperature limit. The first step is to calculate $B \equiv \beta(1-q)$ in the limit $\beta \to \infty$. This is done in Section D.1, and results in the following implicit equation for B:

$$\begin{aligned} B &\equiv \lim_{\beta\to\infty} \beta(1-q) \\ &= \frac{\sqrt{2}}{\sqrt{\alpha\pi r}} \left\langle\!\!\left\langle \exp - \left\{ \frac{\gamma}{\sqrt{2\alpha r}} \sum_{\mu=1}^{s} \left[\gamma m_{\rm h}^{\mu} + \hat{h}^{\mu} - \gamma(1-V^{\frac{1}{4}}) \left(\frac{\alpha m_{\rm h}^{\mu} + 4\tilde{H}B}{(1-\gamma CB)^2} \right) \right] \right\}^2 \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \end{aligned} \tag{5.1}$$

It follows immediately that the zero-temperature mean-field equations can be written in terms of B, using Equation D.8, as follows:

$$m_{\rm h}^{\mu} = \left\langle\!\!\left\langle T^{\mu} \mathrm{erf} \left\{ \frac{1}{\sqrt{2\alpha r}} \sum_{\nu=1}^{s} \left(\gamma^2 m_{\rm h}^{\nu} + \gamma \hat{h}^{\nu} - \gamma^2 (1-V^{\frac{1}{4}}) \frac{\alpha m_{\rm h}^{\mu} + 4\tilde{H}B}{[1-\gamma CB]^2} \right) \right\} \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \tag{5.2}$$

$$q = 1 \tag{5.3}$$

$$r = \frac{\gamma C(\alpha\gamma C q + 4\tilde{H})}{[1-\gamma CB]^2} \tag{5.4}$$

5.1.2 Memory Capacity with No Clamped Neurons

Equations 5.2, 5.3, and 5.4, which in general must be solved numerically, can be solved analytically in certain relatively simple cases. One such case, the subject of this section, is the calculation of the memory capacity for a network in which none of the neurons are clamped ($\gamma = 1$) and the temperature is zero ($\beta = \infty$).

The memory capacity is equal to the largest value of α for which the expected overlap is close to unity when a single state is recovered ($s = 1$). In terms of the parameters used in the mean-field equations, these conditions can be written:

$$\gamma = 1$$

$$\begin{aligned} s &= 1 \\ \beta &= \infty \\ \hat{h}^\mu &= 0 \qquad \forall\, \mu \\ \tilde{H} &= \sum_{\mu=1}^{P} \left(\hat{h}^\mu\right)^2 = 0. \end{aligned}$$

With only a single recovered state, the ensemble averages in Equations 5.1 and 5.2 can be calculated explicitly. Equation 5.1 becomes:

$$\begin{aligned} B &= \frac{\sqrt{2}}{\sqrt{\alpha\pi r}} \exp - \left\{ \frac{1}{2\alpha r} \left(1 - \frac{(1-V^{\frac{1}{4}})\alpha}{(1-CB)^2} \right)^2 (m_{\mathrm{h}}^\mu)^2 \right\} \\ &= \frac{\sqrt{2}}{\sqrt{\alpha\pi r}} e^{-y^2}, \end{aligned} \tag{5.5}$$

where

$$y \equiv \left(1 - \frac{(1-V^{\frac{1}{4}})\alpha}{(1-CB)^2} \right) \frac{m_{\mathrm{h}}^\mu}{\sqrt{2\alpha r}}. \tag{5.6}$$

Note that the variable y has nothing to do with the integration parameter y^μ that was used in Chapter 4. The mean-field equations, Equations 5.2, 5.3, and 5.4, reduce to:

$$\begin{aligned} m_{\mathrm{h}}^\mu &= \mathrm{erf}(y) \\ q &= 1 \\ r &= \frac{C^2}{(1-CB)^2} \end{aligned}$$

Section D.2 describes the derivation of a simultaneous solution for these three equations. The result is the following equation for the capacity α, as a function of y and V:

$$\alpha = \frac{1}{2} \left[\frac{1-CB}{(1-V^{\frac{1}{4}})\,\mathrm{erf}(y)} \left(\sqrt{y^2C^2 + 2(\mathrm{erf}\, y)^2(1-V^{\frac{1}{4}})} - Cy \right) \right]^2. \tag{5.7}$$

where:

$$B = \frac{2(1-V^{\frac{1}{4}})\,\mathrm{erf}(y)\, e^{-y^2}}{\sqrt{\pi}C \left\{ \sqrt{y^2C^2 + 2(\mathrm{erf}\, y)^2(1-V^{\frac{1}{4}})} - Cy \right\}}. \tag{5.8}$$

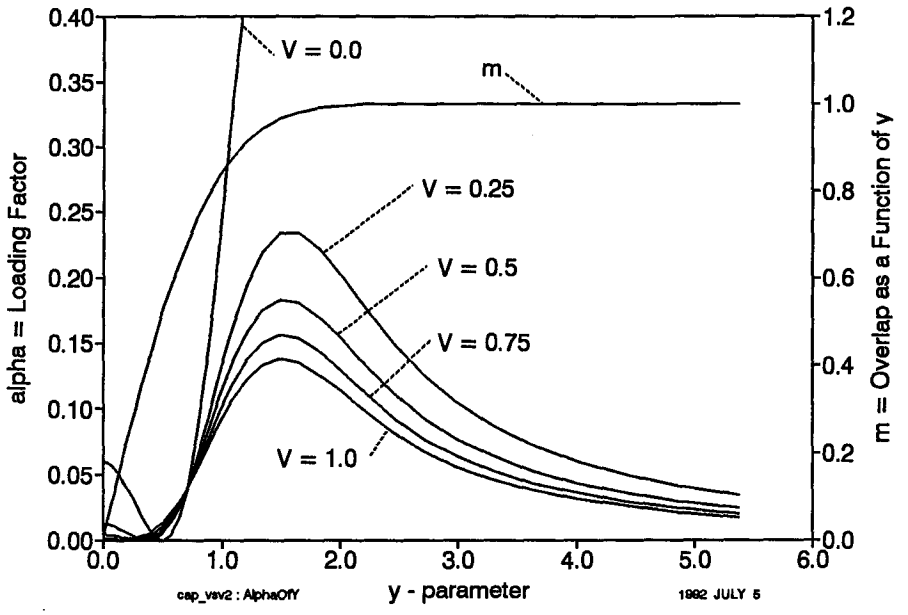


Figure 5.1: Plots of Equation 5.7, showing that for each level of orthogonality, solutions to the mean field equations exist only when memory loading is below a critical level. The y-parameter is defined in Equation 5.6 and is only useful as an aid in the solution of the mean-field equations.

Figure 5.1 is a plot of Equation 5.7 for various values of the orthogonality measure V. The plot for $V = 1$ is identical to Figure 5.5.1 in Lautrup's notes [35, pg. 48], as expected. The plots illustrate that for each level of orthogonality, there is a loading factor $\alpha_c(V)$ above which no solution exists. The memory capacity of the network is equal to $\alpha_c(V)$. A graph of $m_{\rm h}^{\mu}(y)$ is included in the figure for reference, and to illustrate that $m_{\rm h}^{\mu}$ is always close to unity when α is maximum. This means that the state of the network which is compatible with maximum loading factor is also a good-quality retrieval state.

The relationship between $m_{\rm h}^{\mu}$ and α can be seen more clearly in Figure 5.2, where the parameter y has been eliminated. As before, no solutions are available for $[\alpha > \alpha_c(V)]$. The figure shows that as the memory loading α gets smaller, solutions are available in which $m_{\rm h}^{\mu}$ is closer and closer to unity. This means that recovered memories tend to be exactly equal to stored memories when memory loading is much lower than memory capacity, which is consistent with experiments.

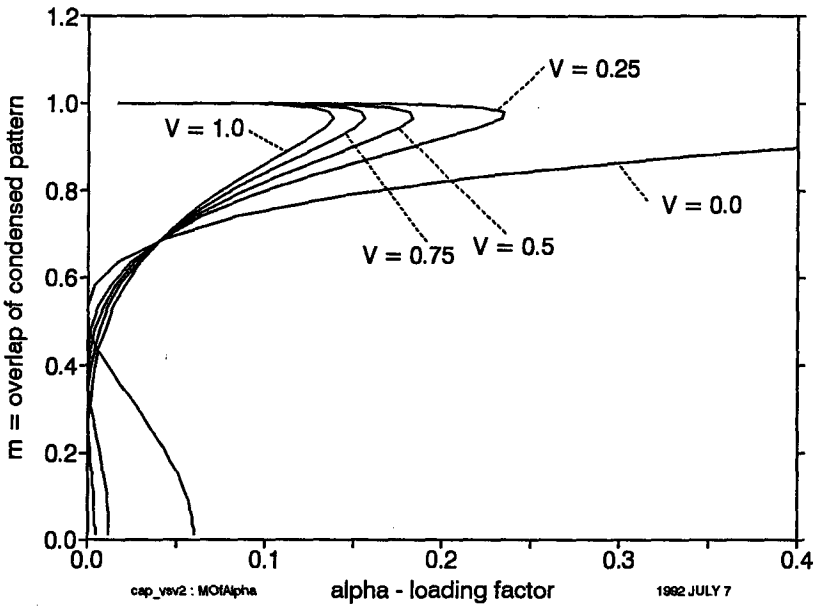


Figure 5.2: Plots of Equation 5.7, showing the retrieval quality m as a function of the load factor α. This figure is comparable to Amit's Figure 6.11 [1].

If the values of $\alpha_c(V)$ are plotted versus V, the result is the graph shown in Figure 5.3. It shows the predicted memory capacity of the network as a function of the orthogonality measure V. It is important to keep in mind that this calculation of "memory capacity" assumes that there are no hidden neurons, so all neurons are initialized to the memory state when stability is measured. This form of memory capacity was measured in Section 2.3.3, and the calculated memory capacity matches those measurements very nicely.

It may be possible, in future research, to extend the above calculations to predict the stability of near-orthogonal memories in the sense defined in Section 3.1.3, where hidden neurons do not play an "unfair" role in contributing to the stability. The goal of such research would be to analytically predict the locations of the curves in Figure 3.9, rather than just the values of their x-axis intersections.

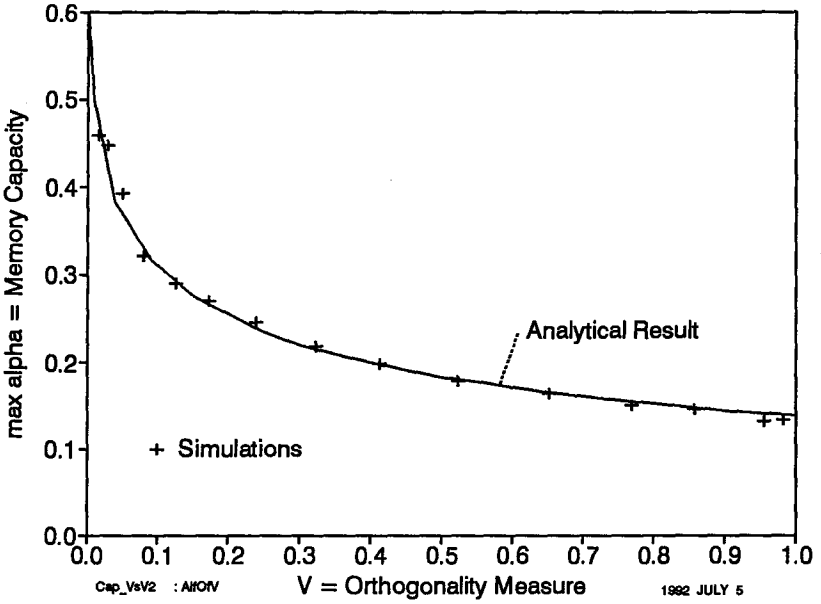


Figure 5.3: Comparison of predicted capacity $\alpha_c(V)$ to measured capacity. The line is a graph of the maximum value of α which is a solution of Equation 5.7. The data points are the same data points as in Figure 2.3.

5.2 Statistical Estimate of Overlap After Roll-Up

The following sections derive an approximate expression for the degree of orthogonality which can be achieved by the EH network's roll-up process. The calculations assume that the network is rolling up approximately as described in Section 3.2. The only difference is that all updates are assumed to be asynchronous – the initial synchronous updates described in Section 3.2 are ignored.

The presentation proceeds as follows. Sections 5.2.1 and 5.2.2 review the notation and introduce a new variable, the "neural alignment" X_i. Section 5.2.3 introduces $P(X,t)$, the statistical distribution of X_i, and Section 5.2.4 examines how $P(X,t)$ changes as the network rolls up. The resulting equations are solved numerically in Section 5.2.5 to predict the rms overlap between the most recent memory and all previous memories. Section 5.2.6 shows how this can be used to calculate the average overlap between all memories.

5.2.1 Notation

The notation used in the following sections is an extension of that introduced in Chapter 3. The network has N neurons, R of which are visible and therefore not free to change, and $M = (N - R)$ of which are hidden and free to change. Because the purpose of the calculation is to determine the mean overlap of the μth memory, it is assumed that there are $(\mu - 1)$ memories already installed when the roll-up process begins. The following two ratios are useful:

$$\begin{aligned} \alpha &\equiv \frac{\mu}{N} \approx \frac{(\mu - 1)}{N} \\ \gamma &\equiv \frac{M}{N}. \end{aligned}$$

Although the time-evolution of the network is not the focus of these calculations, some concept of "time" needs to be introduced so that time-varying characteristics can be expressed. The natural measure of time, represented by the symbol t, starts at zero before roll-up begins, and increases by one whenever a neuron in the network changes state. In addition, a scale-invariant time τ, defined as:

$$\tau \equiv \frac{t}{N}, \tag{5.9}$$

is used for some calculations.

The overlap calculated in this section is given a new symbol, O^{μ}_{rms}, defined as:

$$O^{\mu}_{\text{rms}} \equiv \sqrt{N} \left\langle \left\langle \frac{1}{\mu - 1} \sum_{\nu=1}^{\mu-1} (m^{\mu\nu})^2 \right\rangle \right\rangle^{\frac{1}{2}}. \tag{5.10}$$

This is the rms overlap between one newly-added memory ξ^{μ}, and each previous memory, and so is not the same as O_{rms}, the normalized rms overlap between *all* previous memories. The relationship between O_{rms} and O^{μ}_{rms} is discussed in Section 5.2.6.

5.2.2 Overlap as a Function of Neural Alignments

This section introduces the "neural alignment" X_i and shows how the expected normalized rms overlap can be calculated using it. X_i is a measure of how well the alignments between bit i of each memory and bit i of the current state $\vec{S}$ are matched to the overlaps m^ν between each memory ξ^ν and the current state, and is therefore closely related to the negative of the neural energy E_i. It is defined as follows:

$$X_i \equiv \sum_{\nu=1}^{\mu-1} m^\nu S_i \xi_i^\nu = [-2E_i + \alpha S_i] = \frac{1}{N} \sum_{j=1}^{N} \sum_{\nu=1}^{\mu-1} S_i S_j \xi_i^\nu \xi_j^\nu. \tag{5.11}$$

The value of X_i for each neuron S_i determines whether that neuron will be inverted during roll-up. This can be seen by expanding the input, U_i, to each neuron as follows:

$$\begin{aligned} U_i &= \sum_{j=1}^{N} J_{ij} S_j \\ &= \frac{1}{N} \sum_{j=1}^{N} \left[\sum_{\nu=1}^{\mu-1} \xi_i^\nu \xi_j^\nu - (\mu - 1)\, \delta_{ij} \right] S_j \\ &= S_i\, (X_i - \alpha). \end{aligned} \tag{5.12}$$

The roll-up process described in Section 3.2 inverts only neurons where the state S_i is the same sign as the input U_i, and hence from Equation 5.12, only neurons where $(X_i > \alpha)$. The roll-up process can thus be viewed as a process for shifting all the X_i which are above α until they are all below α.

There is a simple relationship between the average neural alignment and the rms overlap. The sum of all neural alignments X_i is equal to:

$$\sum_{i=1}^{N} X_i = \frac{1}{N} \sum_{\nu=1}^{\mu-1} \sum_{i=1}^{N} \sum_{j=1}^{N} S_i S_j \xi_i^\nu \xi_j^\nu = N \sum_{\nu=1}^{\mu-1} (m^\nu)^2.$$

A comparison of this to Equation 5.10 leads to the following exact relationship:

$$O_{\text{rms}}^{\mu} = \frac{1}{\sqrt{(\mu - 1)}} \left\langle \left\langle \sum_{i=1}^{N} X_i \right\rangle \right\rangle^{\frac{1}{2}}. \tag{5.13}$$

5.2.3 Probability Distribution of the Neural Alignments

To proceed further, it is necessary to define a probability distribution, $P(X,t)$, for the neural alignments X_i. The value of $P(X,t)$ is proportional to the probability that X_i, for some i chosen at random, will be equal to X. $P(X,t)$ acts as a probability density, so that the probability that the value of X_i will be between a and b is equal to:

$$P_{ab}(t) = \int_{X=a}^{b} P(X,t)\, dX.$$

Strictly speaking, $P(X,t)$ can only be considered a probability density in the limit ($N \to \infty$), where the allowed values of X_i approach a continuum.

Calculations of the dynamics of $P(X,t)$ are based on the following *ansatz*:

Working Assumption: $P(X,t)$ is assumed to be Gaussian at all times during roll-up, although the mean, $\bar{X}(t)$, and variance, $V_X(t)$, may change.

The general form for $P(X,t)$ is therefore

$$P(X,t) = \frac{1}{\sqrt{2\pi V_X(t)}} \exp\left[\frac{-(X-\bar{X}(t))^2}{2V_X(t)}\right]. \tag{5.14}$$

Section D.3 shows that the mean and the variance are both equal to α initially:

$$\bar{X}(0) = V_X(0) = \alpha.$$

A sketch of the initial distribution for $\alpha = 0.2$ is shown in Figure 5.4.

5.2.4 Dynamics of the Probability Distribution

This section derives the equations which will be used to calculate the rms overlap between a new memory and all previous memories. Two equations will be derived: one which describes how $\bar{X}$ varies with time, and another which calculates how long the roll-up process takes.

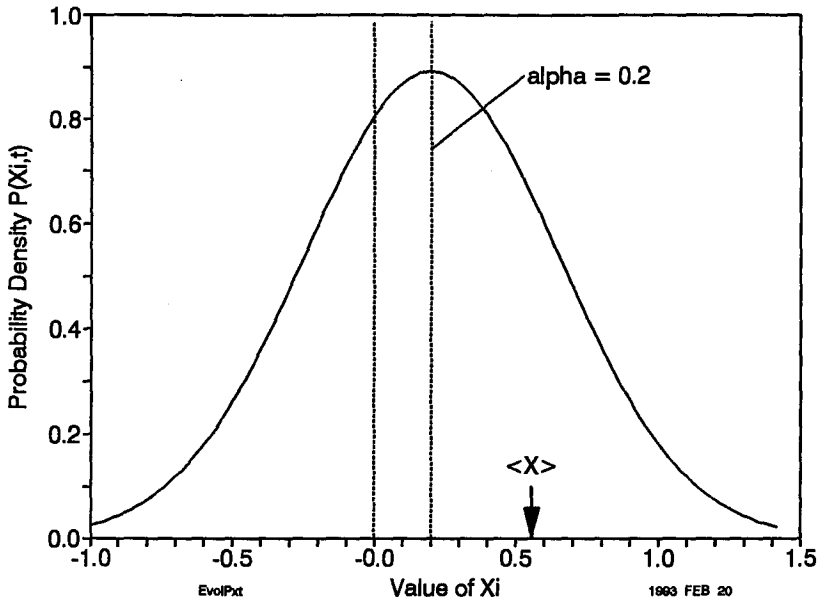


Figure 5.4: Sketch of the distribution of X_i for $\alpha = 0.2$, before the network rolls up. The vertical scale is $P(X,0)$, the probability density of X_i. Neurons for which $(X_i > \alpha)$ are candidates for being inverted during roll-up. The average value of all such candidates is marked $\langle X \rangle$.

The time evolution of $P(X,t)$ is sketched in Figure 5.5, and can be summarized as follows. Initially, $P(X,0)$ is a Gaussian distribution centered at $X = \alpha$, and with variance α. As each neuron is inverted, the Gaussian gets narrower, and its center, $\bar{X}(t)$, moves to the left. $\langle X \rangle$, the average of all the X_i above α, also moves to the left. The motion of the Gaussian is self-limiting because its speed is proportional to $(\langle X \rangle - \alpha)$, and so diminishes with distance travelled. The Gaussian continues to shift to the left and get narrower until all unclamped neurons have been used up.

All changes in $P(X,t)$ are driven by changes in $\bar{X}(t)$. Section D.4 derives the following expression for the rate of change of $\bar{X}(t)$, averaged over many updates:

$$\frac{d\bar{X}}{dt} = \frac{4}{N}(\alpha - \langle X \rangle) \tag{5.15}$$

The following equivalent expression using time τ is independent of N:

$$\frac{d\bar{X}}{d\tau} = 4(\alpha - \langle X \rangle) \tag{5.16}$$

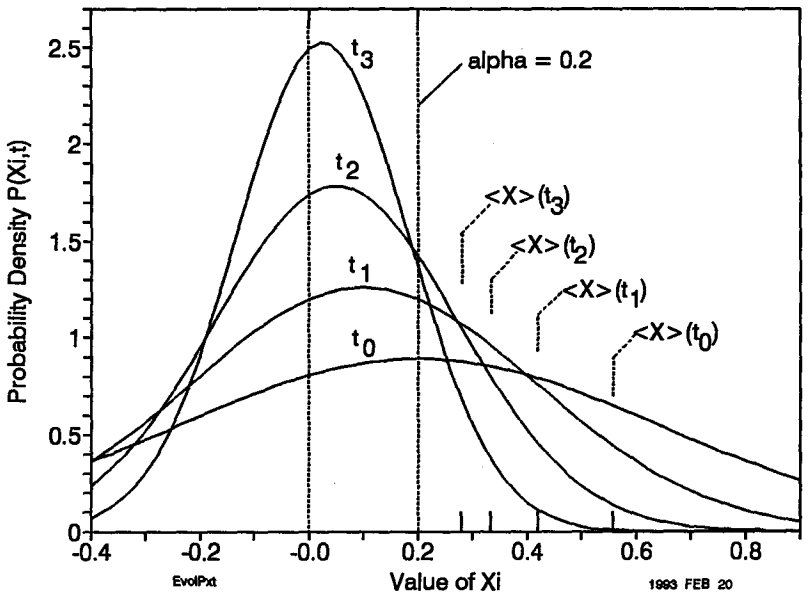


Figure 5.5: Sketch of the evolution of the distribution of X_i as the network rolls up, for a network with $\alpha = 0.2$. The initial probability distribution is labelled as t_0, and examples of subsequent distributions are labelled t_1, t_2, and t_3. The average of all $X_i > \alpha$ are marked as $\langle X \rangle(t_0)$ through $\langle X \rangle(t_3)$. As the network rolls up, the distribution becomes narrower, its mean moves toward zero, and $\langle X \rangle$ approaches α.

Section D.5 shows that $\langle X \rangle$ is equal to:

$$\langle X \rangle = \bar{X} + \sqrt{\frac{2V_X}{\pi}} \frac{e^{-y^2}}{1 - \mathrm{erf}(y)} \tag{5.17}$$

where

$$y \equiv \frac{(\alpha - \bar{X})}{\sqrt{2V_X}}.$$

The following estimate for the variance V_X is derived in Section D.6:

$$V_X \approx \bar{X}\left[1 + \alpha(O_{\mathrm{rms}})^2 - \bar{X}\right]$$

where O_{rms} is the rms overlap between all previously-stored memories ξ^ν. If the O_{rms} term were explicitly kept in, the equations describing $\bar{X}$ for each memory loading would become coupled to all the equations for smaller memory loadings. This would increase the complexity of the mathematical analysis enormously, and for very little gain.

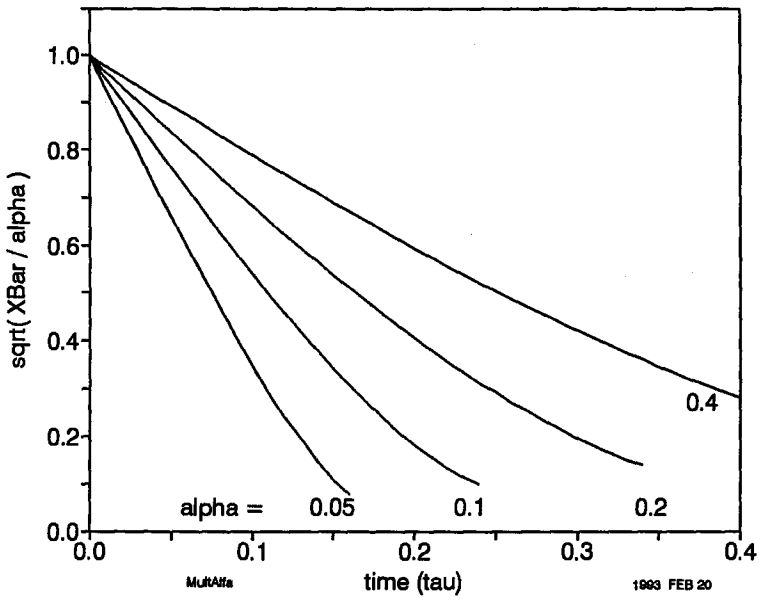


Figure 5.6: Numerical solutions for the time evolution of the mean neural alignment $\bar{X}$, for $\alpha = 0.05, 0.1, 0.2$, and 0.4. The value plotted is $(\bar{X}/\alpha)^{\frac{1}{2}}$. The time variable τ is $1/N$ times the number of neurons which have been inverted since roll-up began. The graphs are solutions of Equation 5.15, with the initial condition $(\bar{X}(0) = \alpha)$.

For simplicity, a constant has been used in place of O_{rms} for the calculations that follow. The experimental results in Chapter 3, such as those shown in Figure 3.3, suggest that a typical value for O_{rms} is 0.5, so V_X is set equal to:

$$V_X = \bar{X}\left[1 + (0.5)^2\alpha - \bar{X}\right] \tag{5.18}$$

Equations 5.16, 5.17, and 5.18 can be combined to form a single differential equation in $\bar{X}$, which can be solved numerically to yield $\bar{X}$ as a function of time. The functional form of $\bar{X}(t)$ depends on α, both because α appears explicitly in the equations, and because $\bar{X}$ is initialized to α. Figure 5.6 shows solutions of this equation for four values of α.

Before the calculation of O_{rms} can be completed, the length of time required for roll-up must be determined. It is clear from Figure 5.6 that the longer the roll-up takes, the smaller $\bar{X}$ will be when roll-up stops. The time required to roll up will in turn depend on what fraction of the

neurons are unclamped. If, for example, 90% of the neurons are clamped, the roll-up process will truncate after a short period of time, and the final average overlap will be large.

Section D.7 derives a differential equation which describes how the number of "invertible" neurons changes with time. To be invertible, a neuron S_i must be unclamped and must have $(X_i > \alpha)$. The total number of invertible neurons is represented by the symbol $B(t)$ and obeys the following differential equation:

$$\frac{dB(t)}{dt} = -1 + \frac{B(t)}{P_T}\left[\frac{dP_T}{dt} + 1\right], \tag{5.19}$$

where P_T is the total probability that X_i is greater than α, and is equal to:

$$P_T \equiv \frac{1}{2}\left[1 - \mathrm{erf}\left(\frac{\alpha - \bar{X}}{\sqrt{2\bar{X}}}\right)\right].$$

The initial number of invertible neurons in the network, $B(0)$, is set to $(\gamma N)/2$.

A scale-invariant version of $B(t)$ is:

$$b(\tau) \equiv \frac{B(t)}{N},$$

where τ was defined in Equation 5.9. Equation 5.19 can then be written using $b(\tau)$ in a manner which is independent of N, as follows:

$$\frac{db(\tau)}{d\tau} = -1 + \frac{b}{P_T}\left[\frac{dP_T}{d\tau} + 1\right], \tag{5.20}$$

Figure 5.7 shows four solutions of this equation on the same time scale as a solution for $\bar{X}(\tau)$.

5.2.5 Numerical Calculations of Average Overlaps

The solutions of Equations 5.15 and 5.20 can be used together to predict O^{μ}_{rms}, as illustrated in Figure 5.7. The time τ_f at which the graph of $b(\tau)$ crosses zero in Figure 5.7 is the amount of time available for the network to roll up. The value of $\bar{X}(\tau_f)$ is then the average neural

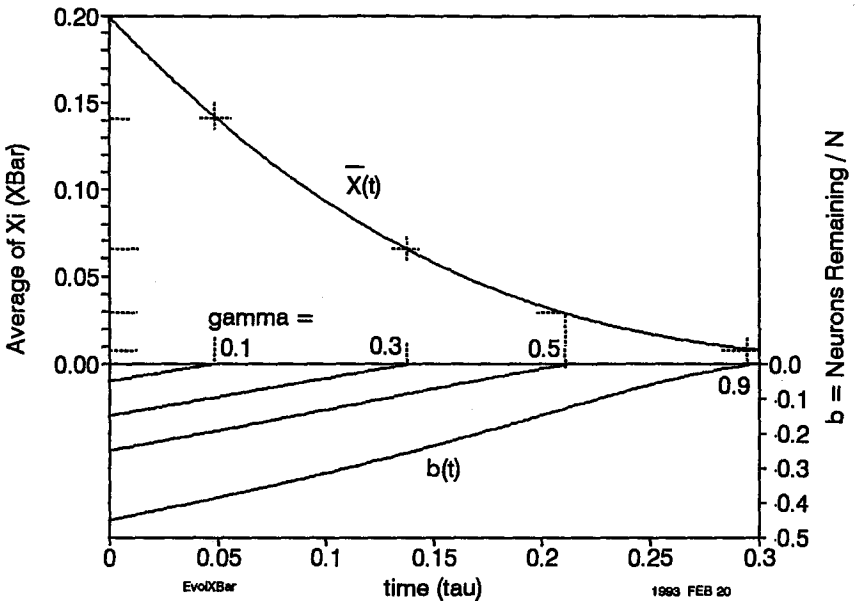


Figure 5.7: Illustration of how to calculate O^{μ}_{rms} from $\bar{X}(\tau)$ and $b(\tau)$, for $\alpha = 0.2$, and for $\gamma = 0.1$, 0.3, 0.5, and 0.9. The top graph shows the average neural alignment, $\bar{X}(\tau)$, and is a solution of Equation 5.15. The bottom graphs show $1/N$ times the number of invertible neurons, $b(\tau)$, and are solutions of Equation 5.20. The time variable τ is $1/N$ times the number of neurons which have been inverted since roll-up began. To calculate $\bar{X}$ for a given value of γ, draw a line up from the x-intersection of the graph of $b(\tau)$, and then across to the X_i axis. The rms overlap is then equal to $\sqrt{\bar{X}/\alpha}$.

alignment when roll-up is completed. It follows from Equation 5.13 that the rms overlap can be calculated from $\bar{X}(\tau_f)$ as follows:

$$O^{\mu}_{\text{rms}} = \left[\frac{\bar{X}(\tau_f)}{\alpha}\right]^{\frac{1}{2}} \tag{5.21}$$

Note that this solution is independent of N, and so is applicable to networks of any size.

Figure 5.8 compares the results of the numerical integration of Equations 5.15 and 5.20 to the average values of O^{μ}_{rms} seen in EH network simulations. The analytical solutions match the observed data very well for all values of α and γ, which confirms the integrity of the analysis.

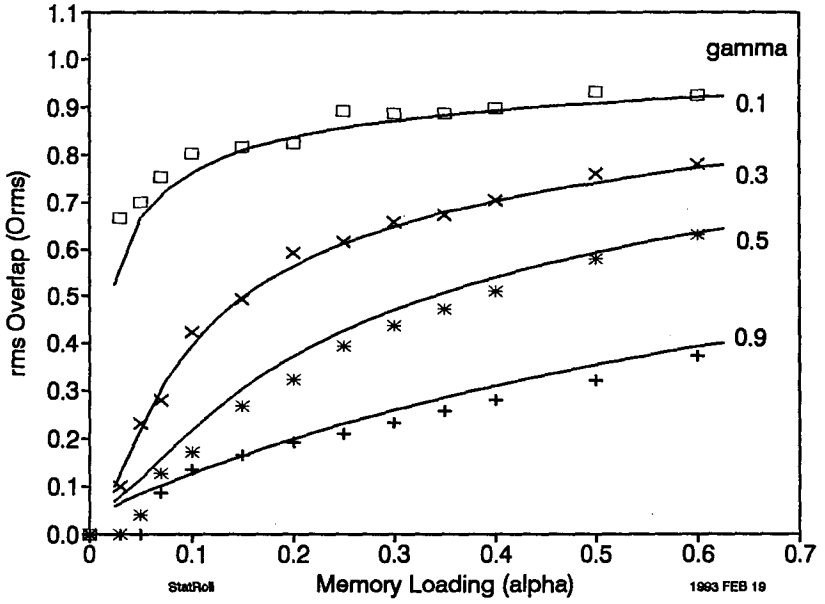


Figure 5.8: Predicted and observed overlap of the most recent memory to all previous memories, as a function of the number of memories stored. α is equal to the number of the current memory, divided by N. γ is equal to the number of neurons which are unclamped, divided by N. Solid lines are solutions of Equations 5.15 and 5.20 and marked points are averages from 100 different simulated memory sets. This figure shows the average overlaps between the *most recent* memory and all previous memories, which is in contrast to Figure 3.2, where the overlaps between *all* memories were included in the average.

5.2.6 Average Overlap of All Memories

An analytical estimate of $O^{\mu}_{\rm rms}$, the expected overlap between the most recent memory and each previous memory, has been derived, but an expression for $O_{\rm rms}$, the average overlap between *all* previous memories, would be more useful. The latter can be calculated from the former if $O_{\rm rms}$ is expanded using Equation 2.9:

$$(O_{\rm rms})^2 = \frac{1}{N}\left(\frac{1}{N\sum_{\nu=2}^{\mu}\sum_{\rho=1}^{\nu-1}1}\right)\sum_{\nu=2}^{\mu}\sum_{\rho=1}^{\nu-1}(m^{\nu\rho})^2 \tag{5.22}$$

This can be written in terms of $O^{\nu}_{\rm rms}$ as follows:

$$(O_{\rm rms})^2 = \frac{\sum_{\nu=2}^{\mu}(\nu-1)(O^{\nu}_{\rm rms})^2}{\frac{1}{2}\mu(\mu-1)}$$

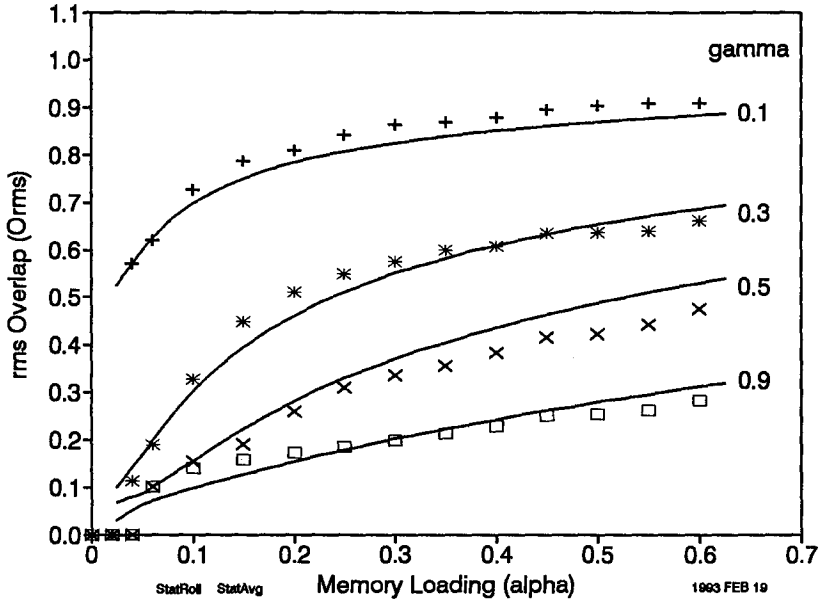


Figure 5.9: Predicted and observed overlaps of all memories, as a function of the number of memories stored. Solid lines are solutions of Equations 5.15, 5.20, and 5.23. Marked points are averages from 50 different memory sets in a bi-state network, and were previously presented in Figure 3.3.

$$= \frac{2}{\mu(\mu-1)} \sum_{\nu=2}^{\mu} (\nu-1)\,(O_{\text{rms}}^{\nu})^2 \tag{5.23}$$

Equation 5.23 makes it possible to translate the solution of Equations 5.15 and 5.20 into an expression for O_{rms} rather than O_{rms}^{μ}. The result, as shown in Figure 5.9, matches the simulation data quite well.

5.3 EH Network Storage Capacity

The analytical results from Sections 5.1 and 5.2 can now be combined to (almost) yield a prediction of the memory capacity of the EH network as a function of the number of hidden neurons. This is an "almost" prediction because it only estimates the number of memories that will be stable in the Hopfield sense – that is, will be stable when all the visible *and hidden*

neurons are initialized to the memory. This is a weaker form of stability than "true" EH network stability, defined in Section 3.1.3, where memories are stable when only the visible neurons are initialized to the memory. In the following paragraphs, capacities calculated using this weaker form of stability are referred to as "whole-vector" capacities.

Figure 5.10 shows graphically how to calculate the whole-vector capacity of the network using the analytical results from Sections 5.1 and 5.2.5. The line descending to the right is the solution of the mean-field equations in Section 5.1 and was previously presented in Figure 5.3. It delineates the maximum number of near-orthogonal memories that can be safely stored, as a function of their level of orthogonality. The lines ascending to the right were generated using the statistical estimate in Section 5.2.5 and were previously presented in Figure 5.9. They describe the level of orthogonality that can be attained by the roll-up process, for a given number of stored memories. If the *ansatz* made in Section 2.4 is correct, the intersection points on these two graphs mark the maximum number of memories that can be stored using the EH process without losing "whole vector" stability.

The locations of these intersections are plotted in Figure 5.11, and compared to the observed memory capacity of the EH network. When $R \gtrsim 0.4N$, the memory capacity approximately matches the analytical prediction, but the prediction diverges from the observed for smaller values of R. An explanation for this is that networks with large numbers of visible neurons are limited by their storage capacity, whereas networks with small numbers of visible neurons are limited by their recall capability. "Whole vector" stability ignores the recall process because it assumes that the memory has already been "recalled" and only tests for stability. This explains why the prediction in Figure 5.11 fails for $(R/N < 0.4)$. Better results will only be achieved when an analytical description of the recall process is developed.

Three important conclusions can be reached from the results of this section:

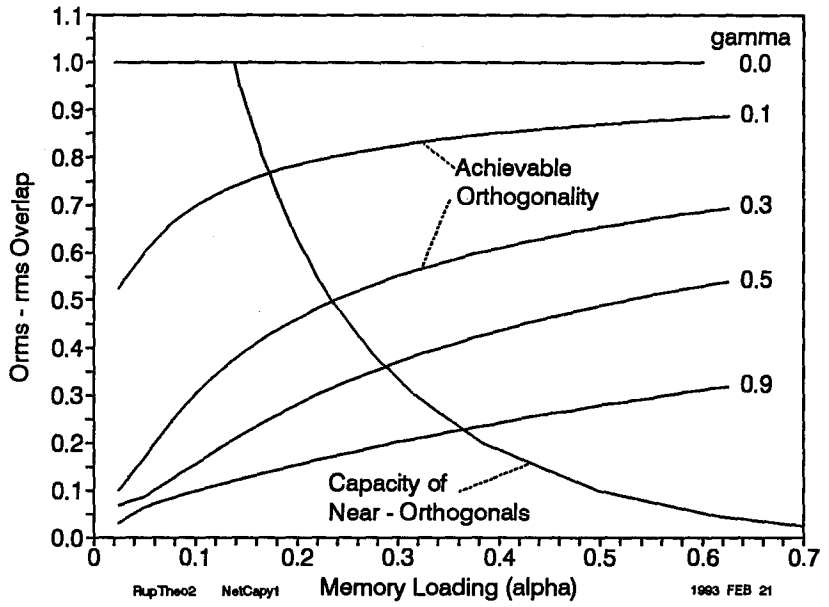


Figure 5.10: Graphical estimate of EH network capacity for "whole-vector" stability. The line curving down from left to right is identical to Figure 5.3, except that the axes are reversed and the data is plotted versus the rms overlap O_{rms} rather than the orthogonality measure V. The lines rising to the right are identical to those in Figure 5.9. If the *ansatz* in Section 2.4 is correct, the intersections of the lines will be the memory capacities of the network for each number of visible neurons.

- The good match between the analytical predictions of network capacity and the measured capacity adds to the confidence that a good mathematical model of the EH network has been achieved.

- The success of the technique illustrated by Figure 5.10 suggests that the *ansatz* made in Section 2.4 was a good one, and may be used for further calculations. Further research into the recall process needs to be done to clarify what the slight misalignment between the predicted and observed capacity for large R/N means. It may indicate that the equivalance principle introduced in the *ansatz* is inexact; or it may be an effect due to the unmodelled recall process.

- The analytical predictions indicate that roll-up effectiveness and network capacity are independent of the size of the network. This suggests that the performance benefits

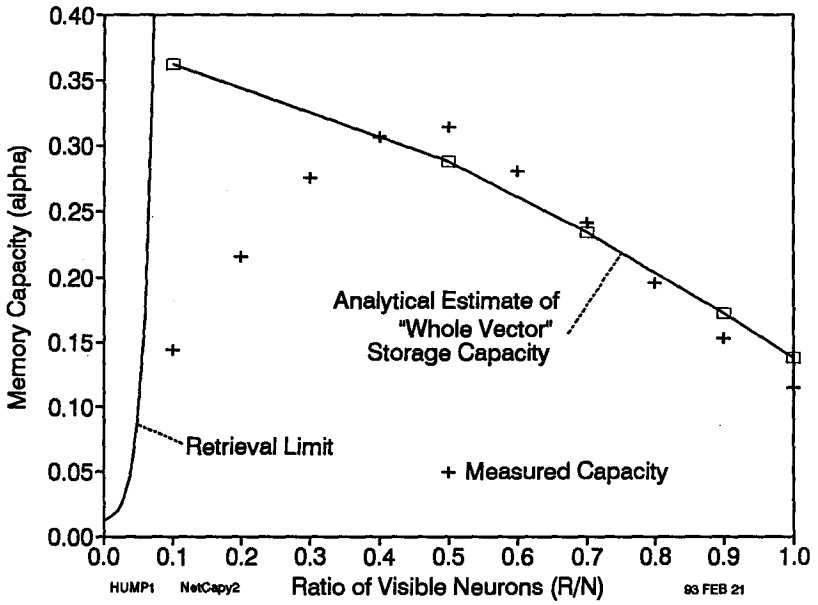


Figure 5.11: Estimate of the true EH network capacity. The solid line going down to the right is the analytical prediction of memory capacity, as determined by the intersections in Figure 5.10. The derivation leading to this result considers only the storage capacity of the network, as measured by "whole vector" stability. The "retrieval limit" line marks the theoretical minimum number of visible neurons required to distinguish one memory, as derived in Equation 3.4. Marked points are the observed capacity of the network, and were first presented in Figure 3.11.

documented in Chapter 3 for networks of 100 neurons will be observed in networks of all sizes.

Chapter 6

The Information Content of Near-Orthogonal Memories

This chapter calculates the information content of memory sets which are exactly orthogonal or near-orthogonal. The derivation proceeds as follows:

- **6.1 : Information in Exactly Orthogonal Memories.** An expression for the information content of a set of exactly orthogonal memories is derived, and confirmed by an exhaustive "brute force" calculation.
- **6.2 : Information in Near-Orthogonal Memories.** The information content of memory sets which are not exactly orthogonal is derived.
- **6.3 : Comparing Information Content and Capacity.** The information capacity of the EH network is compared to the information content of near-orthogonal memory sets with the same number of memories and the same average orthogonality.

6.1 Information Content of Exactly Orthogonal Memories

This section focuses on the problem of calculating the information content of a set of exactly orthogonal memories. The motivation for this undertaking is that it will serve as a starting point for calculating the information content of near-orthogonal memory sets, as described in Section 6.2. Section 6.1.1 briefly reviews the problems inherent in achieving an analytical

expression for the information content of orthogonal memories, and discusses some possible analytical approaches and their limitations. Section 6.1.2 describes a numerical method for exactly calculating the information content, based on an apparently reasonable simplifying assumption. In Section 6.1.3 the numerical results are compared to analytical predictions and conclusions are drawn.

6.1.1 Analytical Calculation of Information Content

When someone adds a new memory to a set of memories, the amount of information in that new memory is determined by how much freedom that person had to select the memory. This makes intuitive sense – one can communicate more effectively with a large vocabulary than with only five or ten words. Specifically, if $\mathcal{Q}$ is the set of all possible combinations of memory vectors which could be selected, then the information contained in the selected set is the logarithm of the size of $\mathcal{Q}$ [49][5]:

$$S = \log(g(\mathcal{Q})) \tag{6.1}$$

There is a direct analogy with a statistical mechanical system where $\mathcal{Q}$ is the space of all accessible states of the system and S is its entropy.

It is also common in the literature [28, 49, 1] to define the information in terms of the probability $P(\xi^\mu)$ of each vector ξ^μ being selected:

$$S = \sum_\mu -P(\xi^\mu)\log P(\xi^\mu) \tag{6.2}$$

In a system with G possible states, all equally probable, the probability of any state is $\frac{1}{G}$, so:

$$S = \sum_{\mu=1}^{G} -\frac{1}{G}\log\left(\frac{1}{G}\right)$$
$$S = \log G$$

which is consistent with Equation 6.1. For systems where not all vectors are equally probable, Equation 6.2 must be used.

It would be natural to calculate the size of $\mathcal{Q}$ as the product of the sizes of the state space for each memory in the set. In that case, the information in a set of memories would be the sum of the information in each memory. This would be over-counting, however, because the order in which the memories are added to the set is irrelevant. In other words, if two memories A and B are to be stored, it would count (A,B) as a different memory set from (B,A). To compensate, the size of $\mathcal{Q}$ must be divided my $M!$, where M is the number of memories, with the result that the degeneracy $g(\mathcal{Q})$ is:

$$\begin{aligned} g(\mathcal{Q}) &= \frac{1}{M!}\prod_{\mu=0}^{M-1} g(N,\mu) \\ &= \prod_{\mu=0}^{M-1}\frac{g(N,\mu)}{\mu+1} \end{aligned} \tag{6.3}$$

The information content in the most recently added memory is:

$$S(N,\mu) = \log(g(N,\mu)) - \log(\mu+1) \tag{6.4}$$

where μ is the index of the memory and, for reasons which become clear in Section 6.1.2, the initial memory is referred to as the zeroeth.

It is simple to calculate the information content of randomly chosen memories because each bit is equally likely to be +1 or -1. There are 2^N ways to select each memory, so the information in each new memory is $[N\log 2 - \log(\mu+1)]$.

Consider now the case where all elements of a set of memories are required to be mutually orthogonal. If this is the only condition imposed on the memory set, then each bit is still equally likely to be +1 or -1. When the first memory is selected, orthogonality is no constraint at all, so the accessible space for the first memory is:

$$g(N,0) = 2^N$$

The only constraint on the second memory is that exactly half of its bits be identical to the first memory. If N is odd, this is immediately impossible – so memory sets with odd numbers of neurons are immediately eliminated from the discussion. If N is even, the accessible space is:

$$g(N,1) = \binom{N}{N/2}$$

Similar arguments lead to the following expression for the third memory:

$$g(N,2) = \binom{N/2}{N/4}^2$$

It is useful to view each memory as one vertex of an N-dimensional hypercube, as is commonly done in the literature [1, pg 32]. According to that view, the first three memories make up a triad of orthogonal vectors in the N-dimensional space. The expressions for g for the first three memories are simple, because all possible sets of three vectors are equivalent, up to a rotation or reflection in the N-dimensional space. Reflections and rotations are achieved by either a gauge transform [1, pg 184], or by reordering the axes. In a gauge transform, a subset of bits in each memory ξ^μ is inverted using the transform-vector ξ^g:

$$\xi_i^{\mu'} = \xi_i^g \xi_i^\mu$$

It is possible, by rotations and reflections, to transform any set of three orthogonal vectors into the following set:

```
++++++++++++++++++++++++++++++++
++++++++++++++++----------------
+++++++++-------++++++++--------
```

The expression for the size of the accessible space for ξ^3 is more complicated:

$$g(N,3) = \sum_{i=0}^{N/4} \binom{N/4}{i}^4 \tag{6.5}$$

This increased complexity reflects the fact that there are many ways to generate ξ^3 which are not identical up to a rotation or reflection. The measure of the information in ξ^3 must take into account all of these possible "isomers" of the basis vectors. For the fifth and higher memories, each isomer spawns a new family of isomers, and the complexity of the problem quickly becomes intolerable.

There are a number of ways to proceed from here. The brute force approach is to do numerical calculations of $g(N,\mu)$. The complexity of this task is considerable, and one approach is described in 6.1.2. The other approaches involve some form of approximation.

The simplest and most successful analytical estimate of the information capacity is based on the N-dimensional hypercube interpretation of the vector space. In an N-dimensional network, which corresponds to an N-dimensional space, there are 2^N ways to insert ξ^0. In order to be orthogonal to ξ^0, ξ^1 is restricted to an $N-1$ dimensional subspace. If that subspace is topologically similar to the full space, it can be expected that there will be 2^{N-1} ways to form ξ^1. Following this line of reasoning, the number of accessible states for memory ξ^μ is:

$$g(N,\mu) \approx 2^{N-\mu} \tag{6.6}$$

In fact the $(N-1)$-dimensional subspace is not very similar to the full space. In geometrical terms, although the reduced-dimension space left over after ξ^0 has a "volume" of 2^{N-1}, a large part of that volume is not accessible to ξ^1 because of the constraint that ξ^1 lie on the vertices of the hypercube. Equation 6.6 is therefore an overestimate of the number of accessible states. There are two reasons why it is still an acceptable approximation:

- Section 6.1.3 shows, for one example set of orthogonal vectors, that the error due to over-estimation is not large.
- The argument gains strength when the requirement that all memories be strictly orthogonal is relaxed, because all the "hidden corners" of the space become accessible.

Combining Equations 6.4 and 6.6, the predicted information content of the μth memory is:

$$S(N,\mu) \approx (N-\mu)\log(2) - \log(\mu+1) \tag{6.7}$$

6.1.2 Brute Force Calculation of Information Content

A method has been developed for the exact calculation of the information contained in each new memory of a memory set which is constrained to be exactly orthogonal. This method assumes that N, the size of the network, is an integer power of two, so that a set of memories can be created which is both exactly orthogonal and complete. That is, it will be possible to generate one memory for every dimension in the hyperspace.

The information content $S(N,\mu)$ of the μth memory is determined by the size $g(N,\mu)$ of the vector space from which potential vectors can be chosen, as in Equation 6.1. It has already been shown that the complexity of the expressions for $g(N,\mu)$ begins to blow up when μ reaches three (Equation 6.5). The problem can be made tractable by assuming that the information contained in each new memory is independent of the specific set of memories stored previously. Under that assumption, it is acceptable to assume a specific set of exactly orthogonal previous memories $\xi^0 \ldots \xi^{M-1}$ when calculating the information in the Mth memory ξ^M. Because those previous memories are known precisely, the size of only one vector subspace needs to be calculated.

The canonical memory set Γ defined in Section A.1 was used as the set of exactly orthogonal memories. There is a direct correspondence between the set of memories constituting the nth level of $\Gamma^{(M)}$ and the set $\Gamma^{(n)}$, for all m. Consider, for example, the level 4 memories in table A.1. When the block ++++---- is mapped to a +1 bit, and the block ----++++ is mapped to a -1

bit, then level 4 of $\Gamma^{(32)}$ corresponds exactly to $\Gamma^{(4)}$, the canonical set of 4-neuron memories:

$$\left[\begin{array}{l} \texttt{++++----++++----++++----++++----} \\ \texttt{++++----++++--------++++----++++} \\ \texttt{++++--------++++++++--------++++} \\ \texttt{++++--------++++----++++++++----} \end{array}\right] \Leftrightarrow \left[\begin{array}{l} \texttt{++++} \\ \texttt{++--} \\ \texttt{+-+-} \\ \texttt{+--+} \end{array}\right] \tag{6.8}$$

These features of $\Gamma^{(N)}$ simplify the calculation of the information content in some important ways. The most important is that they permit the imposition of progressively stronger restrictions on the sum of the bits in each block of the new memories. Specifically:

- Consider a memory as being divided into 2^n blocks of width $N/2^n$. For all memories numbered $\mu = 2^n$ or higher, the sum of the bits within each of these blocks must be zero. In other words, there must be an equal number of +1 and -1 bits within each block.

In mathematical terms, this restriction is:

$$\sum_{i=1}^{N/2^n} \xi^{\mu}_{i+m\cdot 2^n} = 0 \qquad \begin{cases} \forall\, n : 1 < n < \log_2(N) \\ \forall\, \mu : \mu > 2^n \\ \forall\, m : 0 \leq m < 2^n \end{cases} \tag{6.9}$$

In table A.1, for example, the sum of bits 9 through 16 must be zero in any memory numbered 4 or higher.

The degeneracy for any memory ξ^{μ}, where μ is an integer power of two, is therefore:

$$g(N,\mu) = \left(\frac{B^{\mu}}{\frac{B^{\mu}}{2}}\right)^{L^{\mu}} \qquad \text{for all } \mu = L^{\mu} \tag{6.10}$$

where B^{μ} and L^{μ} are the block width and level number defined in Section A.1. Note that this is the number of vector states which *could* be chosen to be ξ^{2^n}. For the purposes of calculating the degeneracy of later memories, this memory is allowed to be only one state – the one defined by Equation A.1.

States ξ^μ where $\mu > L^\mu$ must satisfy the further restriction that they have zero overlap with memories Γ^ν where $\nu \geq L^\mu$. The overlap with each of these previous memories can be calculated as a weighted sum of k_0 through $k_{L^\mu-1}$, where the k_i are defined as:

$$k_i = \frac{1}{4}\sum_{j=1}^{B^\mu} \xi^\mu_{i\cdot B^\mu+j}\Gamma^{L^\mu}_{i\cdot B^\mu+j} \qquad \text{where } 0 \leq i < L^\mu \tag{6.11}$$

Each k_i is the quarter-overlap of the ith block in ξ^μ with the ith block in Γ^{L^μ}. The block furthest to the left is the zeroeth. The number of ways to arrange the bits within a block so that the quarter-overlap with Γ^{L^μ} equals k_i is:

$$g(B^\mu, k_i) = \begin{cases} \left(\frac{B^\mu/2}{\frac{B^\mu}{4}-k_i}\right)^2 & \text{for } B > 2 \\ 2 & \text{for } B = 2 \end{cases} \tag{6.12}$$

Each k_i can range from $-\frac{1}{4}B^\mu$ to $+\frac{1}{4}B^\mu$ (they must all be zero when $B = 2$), but as a set, the k_i's must satisfy the restrictions which ensure zero overlaps.

There are L^μ blocks, each with the degeneracy given by Equation 6.12, so the total degeneracy is:

$$g(N,\mu) = \prod_{j=0}^{L^\mu-1} \sum_{k_j=-B^\mu/4}^{B^\mu/4} \left(\frac{B^\mu}{\frac{B^\mu}{4}-k_j}\right)^2 D(\mu,\{k\}) \qquad \text{for all } \mu \neq L^\mu \tag{6.13}$$

where $D(\mu,\{k\})$ enforces the restrictions on the set of k_i's.

The restriction factors $D(\mu,\{k\})$ always take the form of a product of one or more delta functions. The following formula for $D(\mu,\{k\})$ is derived quite directly using the equivalence property from Equation 6.8:

$$D(\mu,\{k\}) = \prod_{j=1}^{\mu-L^\mu} \delta\left(\sum_{i=0}^{L^\mu-1} \Gamma^{(L^\mu)j-1}_{i+1} k_i\right) \tag{6.14}$$

Consider for example $g(32,5)$, the degeneracy of the fifth memory in table A.1. L^μ is 4, so there are four blocks, with quarter-overlaps of k_0 through k_3. k_0, for example, is the quarter

overlap between the first eight bits of ξ^5 and the first eight bits of Γ^4. Using Equation 6.14, the restriction factor is found to be:

$$D(5,\{k\}) \;=\; \delta(k_0+k_1+k_2+k_3)$$

and, using Equation 6.13, the total degeneracy is:

$$g(32,5) \;=\; \sum_{k_0,k_1,k_2,k_3=-8}^{8} \binom{4}{2-k_0}^2 \binom{4}{2-k_1}^2 \binom{4}{2-k_2}^2 \binom{4}{2-k_3}^2 \delta(k_0+k_1+k_2+k_3)$$

Once the degeneracy $g(N,\mu)$ has been calculated, the information content of the most recent memory ξ^μ can be calculated using Equation 6.4.

6.1.3 Calculated Information Content of Exactly Orthogonal Memories

The information content of networks with 8, 16, 32, and 64 neurons were calculated using the brute force method derived in Section 6.1.2. Figure 6.1 shows how the information in the most recently added memory decreases as more and more memories are added. The solid lines in the figure show the information content predicted by Equation 6.7. As expected, Equation 6.7 has overestimated the information content, for reasons discussed in Section 6.1.1. A closer approximation would be to replace $g(N,\mu)$, as defined in Equation 6.6, with an empirical function $g(N,\mu,k)$ that is defined as follows:

$$g(N,\mu,k) \;\equiv\; \frac{1}{k}2^{k(N-\mu)}. \tag{6.15}$$

This is equivalent to $g(N,\mu)$ when $k = 1.0$. From Equation 6.4, it follows that the information in the $(\mu+1)$th memory is:

$$S(N,\mu,k) \;=\; k(N-\mu)\log(2) - \log(k(\mu+1)). \tag{6.16}$$

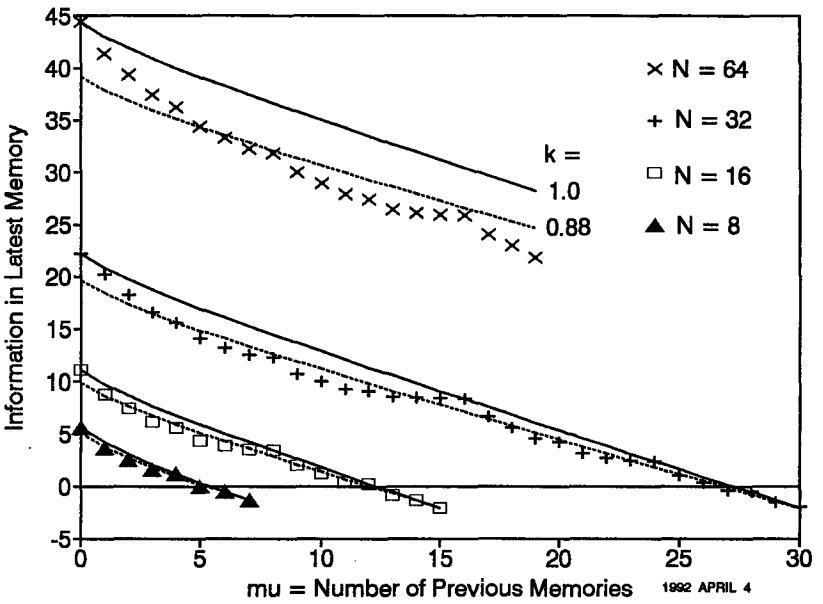


Figure 6.1: Information content of the $(\mu+1)$th memory in sets of exactly orthogonal memories with 8, 16, 32, and 64 neurons. The marked data points are from calculations performed using the "brute force" method of Equations 6.10, 6.13, and 6.4. The information capacity predicted by Equation 6.7 is shown as a continuous line above each set of points. The dashed line through the points is a plot of Equation 6.16, with $k = 0.88$.

The dashed line in Figure 6.1 shows $S(N, \mu, 0.88)$. This new estimate of the information is a compromise – it is worse than the original for small numbers of memories, but averages out better for large numbers of memories.

An interesting feature of Figure 6.1 is that the information content is negative for the last ten percent of the memories. This means that information is actually lost when each of those memories is added. This loss of information has nothing to do with the way the memories are stored, because storage of the memories has not even been considered yet. The explanation can be found in Equation 6.7: the second term, which represents the information lost due to the indistinguishable order of the memories, becomes larger than the first term, which represents the information inherent in the choice of the next vector.

Consider, for example, a situation where 63 memories have been added to an orthogonal set

of 64 neuron memories. One more memory can be added to make a complete set of memories. The person choosing that memory may only choose between two vectors, which are the bit inverse of each other, so the information in the 64th vector would be $\log 2$. Once that memory was added to the list, it would be indistinguishable from the other 63 memories. On the other hand if the 64th memory is not added, a careful observer could determine, up to a sign, exactly which memory was missing. The amount of information encoded in the choice of the missing memory is just the log of the number of memories which could have been selected – log 64. Thus more information can be conveyed by *not* adding the last memory, so the last memory contains a negative amount of information.

A second interesting feature of Figure 6.1 is that the error in $S(N,\mu)$ is zero when $(1-\frac{\mu}{N}) = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$, etc. This is a result of the structure of $\Gamma(N,\mu)$ which, as noted in Equation 6.8, ensures an exact isomorphism between the second half of the $\Gamma^{(N)}$ and the memory set with half as many neurons, $\Gamma^{(N/2)}$. The fourth quarter of $\Gamma^{(N)}$ can therefore be thought of as the second half of $\Gamma^{(N/2)}$, and so on. The brute force calculation of degeneracies at these points will therefore correspond exactly to those estimated in Equation 6.6. In a less highly structured set of exactly orthogonal memories, it is likely that the observed degeneracies would lie along a smoother curve, perhaps similar to the $S(N,\mu,0.88)$ curve.

The total amount of information I in the network is the sum of the information added by each new memory. Using the approximation from Equation 6.16, this is:

$$I(N,M,k) \;=\; kM(N+\frac{1}{2}-\frac{M}{2})\log 2 - M\log k - \log M! \qquad (6.17)$$

Figure 6.2 shows how the total information calculated using the brute force approach compares to the amounts predicted by these two expressions. The fact that graphs using $k = 0.88$ correspond so well to the data for all four memory sizes suggests that $k = 0.88$ should be used for all estimates of information content.

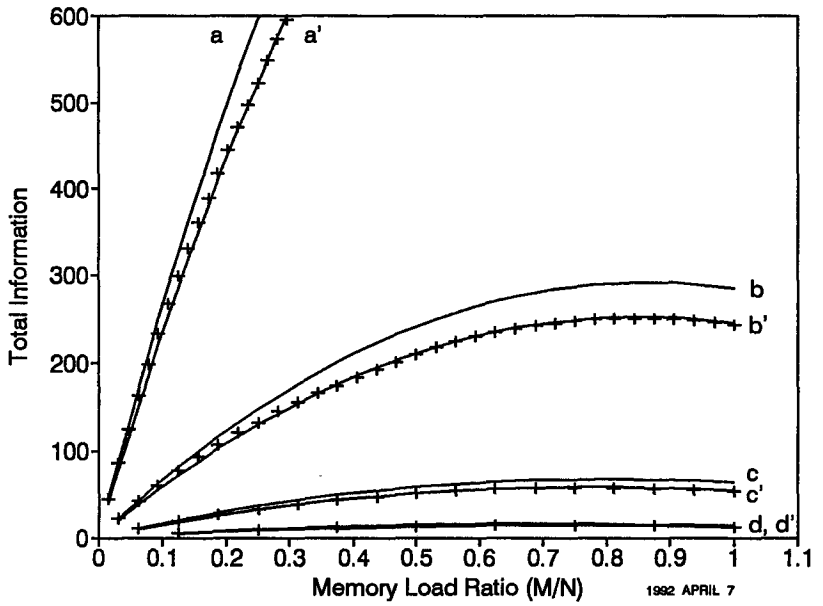


Figure 6.2: Total information content of a set of exactly orthogonal memories with 64 neurons. The marked data points are summations of the data shown in Figure 6.1. Lines marked a, b, c, and d are graphs of Equation 6.17 with $k = 1.0$, for $N = 64$, 32, 16, and 8, respectively. Lines marked with primes show the case where $k = 0.88$.

6.2 Information Content in Near-Orthogonal Memories

The information content in a near-orthogonal vector set can now be estimated. A "near-orthogonal" vector set is made up of vectors whose bits differ from one of the exactly orthogonal vectors Γ^μ according to the probability distribution defined in Equation 2.12:

$$P(\xi_i^\mu) = (1-b)\delta(\xi_i^\mu - \Gamma_i^\mu) + b\delta(\xi_i^\mu + \Gamma_i^\mu) \tag{6.18}$$

In the discussion which follows, it will be useful to have an estimate of how many vectors can be classified as "near" a particular vector Γ^μ. There is a finite chance that any vector in the whole vector space might be generated from Γ^μ using Equation 6.18, so it would not be possible to just count the number of "near" vectors. Instead, the number of "near" vectors is estimated using a statistical method, based on the amount of information added when the new vector is generated.

When a new vector ξ^μ is generated using Equation 6.18 the information in ξ^μ exceeds the information in Γ^μ by an amount equal to the sum of the information in each of its bits:

$$S^+ = -N[b\log b + (1-b)\log(1-b)] \tag{6.19}$$

The information is understood to be the logarithm of the number of accessible vectors, so the inverse is also true: the number of vectors which are "near" Γ^μ is the exponential of the information added when generating them:

$$g^+(N,b) \equiv e^{S^+} = \exp\{-N[b\log b + (1-b)\log(1-b)]\} \tag{6.20}$$

The next step is to estimate the total size, $g(N,\mu,b)$, of the vector space from which near-orthogonal memories can be selected. A new near-orthogonal vector ξ^μ is generated in two steps: first an exactly orthogonal vector Γ^μ is selected at random, and then noise is added according to Equation 6.18. The space of all possible vectors ξ^μ is therefore the space of all vectors which are "near" any one of the Γ^μ vectors. The number of possible Γ^μ vectors is $g(N,\mu,k)$, as derived in Section 6.1. It would be tempting to calculate $g(N,\mu,k,b)$ as simply the product of $g(N,\mu,k)$ and $g^+(N,b)$:

$$g(N,\mu,k,b) \stackrel{?}{=} g(N,\mu,k)\cdot g^+(N,b) \tag{6.21}$$

The problem with this is that it would count the vectors around each possible Γ^μ as being distinct from those around every other Γ^μ. Any vectors which were "near" more than one of the possible Γ^μ would be counted more than once. This is a reasonable approximation when both g and b are small, but it is possible to derive an expression which is correct in the more general case.

The number of vectors which are nearly orthogonal to a set of previously-installed vectors is calculated as the total number of vectors that are close to each possible new Γ^μ vector. There are $g(N,\mu,k)$ possible vectors Γ^μ, numbered from $\Gamma^{\mu[1]}$ through $\Gamma^{\mu[g]}$. The number of vectors

which are close to $\Gamma^{\mu[1]}$ was calculated in Equation 6.20:

$$g_1^+ = \exp\{-N[b\log b + (1-b)\log(1-b)]\} \tag{6.22}$$

The size of the complete vector space is 2^N, so the fraction of all states which are near $\Gamma^{\mu[1]}$ is

$$\theta = \frac{g_1^+}{2^N} = \left[2\, b^b\, (1-b)^{(1-b)}\right]^{-N} \tag{6.23}$$

The definition of g_2^+, the number of uncounted states close to $\Gamma^{\mu[2]}$, must take into account the fact that there are now only $2^N(1-\theta)$ states which are still eligible to be counted. The simplest assumption is that, on average, the memories around $\Gamma^{\mu[2]}$ are evenly distributed between the reduced space and the excluded space, in which case:

$$g_2^+ = g_1^+(1-\theta)$$

This leaves $2^N(1-\theta)^2$ states eligible for counting in g_3^+, and so on:

$$\begin{aligned} g_3^+ &= \theta\, 2^N(1-\theta)^2 \\ g_i^+ &= \theta\, 2^N(1-\theta)^{i-1} \end{aligned}$$

The total number of possible memory states is the sum of g_1^+ through g_g^+:

$$\begin{aligned} g(N,\mu,k,b) &= \sum_{i=1}^{g(N,\mu,k)} g_i^+ = \sum_{i=1}^{g} \theta\, 2^N(1-\theta)^{i-1} \\ &= \theta\, 2^N \sum_{i=0}^{\infty}(1-\theta)^i - \theta\, 2^N \sum_{i=g}^{\infty}(1-\theta)^i \\ &= 2^N - \theta\, 2^N(1-\theta)^{g(N,\mu,k)} \sum_{i=0}^{\infty}(1-\theta)^i \\ &= 2^N\left[1-(1-\theta)^{g(N,\mu,k)}\right] \end{aligned} \tag{6.24}$$

This is a difficult expression to simplify analytically, but it is tractable in the limits $b \to 0$ and $b \to \frac{1}{2}$. As b goes to zero, θ also goes to zero, so Equation 6.24 can be simplified as follows:

$$g(N,\mu,k,b) \approx 2^N\left[1-(1-g(N,\mu,k)\theta)\right]$$

$$= g(N,\mu,k)b^{-Nb}(1-b)^{-N(1-b)} = g(N,\mu,k)\cdot g_1^+$$

$$\log g(N,\mu,k,b) = \log g(N,\mu,k) - Nb\log b - N(1-b)\log(1-b) \quad (6.25)$$

This result is accurate when $(\theta\, g(N,\mu,k)) \ll 1$, and matches Equation 6.21 as expected. In the domain where it is accurate, the information in the near-orthogonal vectors is just the information in the exactly orthogonal vectors plus the information from inverting some of the bits.

As b approaches 0.5, the memory set approaches complete randomness, so the amount of information in each new memory should approach $(N\log 2 - \log(\mu+1))$. If b is written in terms of a small amount δ:

$$b = \frac{1-\delta}{2}$$

then θ can be written:

$$\begin{aligned}
\theta &= 2^{-N}\left(\frac{1-\delta}{2}\right)^{-\frac{N}{2}(1-\delta)}\left(\frac{1+\delta}{2}\right)^{-\frac{N}{2}(1+\delta)} \\
&= (1-\delta)^{-\frac{N}{2}}(1+\delta)^{-\frac{N}{2}}\left(\frac{1+\delta/2}{1-\delta/2}\right)^{2\delta N} \\
&\approx \left(1-\delta^2\right)^{-\frac{N}{2}}\left(1+\frac{\delta}{2}\right)^{2\delta N}\left(1+\frac{\delta}{2}+\frac{\delta^2}{4}\right)^{2\delta N} \\
&\approx \left(1-\frac{N}{2}\delta^2\right)\left(1+2\delta^2 N\right) \\
&\approx \left(1+\frac{3}{2}N\delta^2\right) \qquad (6.26)
\end{aligned}$$

This approximation is accurate if $N\delta^2 \ll 1$. Replacing this into Equation 6.24,

$$g(N,\mu,k,b) = 2^N\left[1-\left(\frac{3}{2}N\delta^2\right)^{g(N,\mu,k)}\right] \quad (6.27)$$

So when b gets close to 0.5, the information content approaches $N\log 2$, as required.

The information content of the $(\mu+1)$th near-orthogonal memory is, from Equations 6.4 and 6.24 equal to:

$$S(N,\mu,k,b) = \log\left[g(N,\mu,k,b)\right] - \log(\mu+1)$$

$$= N\log 2 + \log\left[1-(1-\theta)^{g(N,\mu,k)}\right] - \log(\mu+1) \tag{6.28}$$

where θ is defined exactly by Equation 6.23. This value was calculated numerically and is shown in Figures 6.3 and 6.4 for $k = 1.0$ and $k = 0.88$ respectively. In the $k = 0.88$ graph, the information content for $b = 0$ rather awkwardly does not match the other plots at $\mu = 0$. This is because $S(N,\mu,0.88)$ is a poor match to the true information content at $\mu = 0$, as noted in Figure 6.1. This awkwardness is compensated by the fact that it is a better match for large values of μ.

As expected, the near-orthogonal graph runs parallel to the exactly-orthogonal graph in the domain where $\theta\, g(N,\mu,k) \ll 1$. There is a sharp transition to the domain where the information is just $N\log 2 - \log(\mu+1)$ so, to a good approximation:

$$S(N,\mu,k,b) = \text{minimum of:} \begin{cases} S(N,\mu,k) - Nb\log b - N(1-b)\log(1-b) \\ N\log 2 - \log(\mu+1) \end{cases} \tag{6.29}$$

The location, μ_0, of the elbow in the graph of $S(N,\mu,k,b)$ is defined implicitly by the following equation:

$$N\log 2 - (\mu_0+1) = S(N,\mu_0,k) - Nb\log b - N(1-b)\log(1-b) \tag{6.30}$$

where $S(N,\mu,k)$ is defined in Equation 6.16.

Once a set of memories has been installed in a network, the order in which they were installed is irrecoverable, so it is not meaningful to regard early-installed memories as containing more information than later-installed ones. The average amount of information per neuron, after M memories have been installed, is:

$$\begin{aligned} \bar{s}(N,M,k,b) &= \sum_{\mu=0}^{M-1} \frac{S(N,\mu,k,b)}{MN} \\ &= \frac{1}{MN}\left\{\sum_{\mu=0}^{\mu_0}[N\log 2 - \log(\mu+1)] + \right. \end{aligned}$$

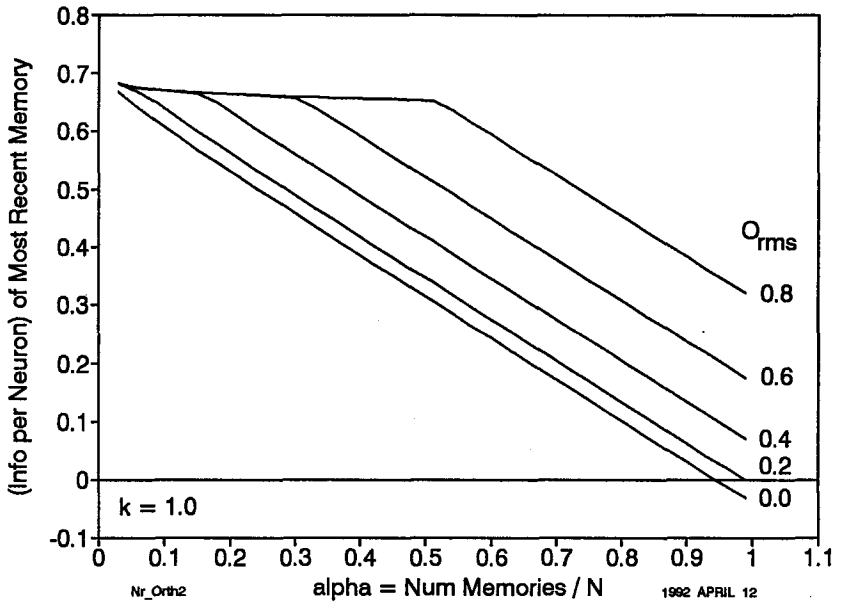


Figure 6.3: Numerical calculation of the information per neuron in the most recently added near-orthogonal memory, from Equation 6.28. The calculation assumed 100 neurons, and used $k = 1.0$. The relationship between O_{rms} and b is given in Equations 2.11 and 2.14.

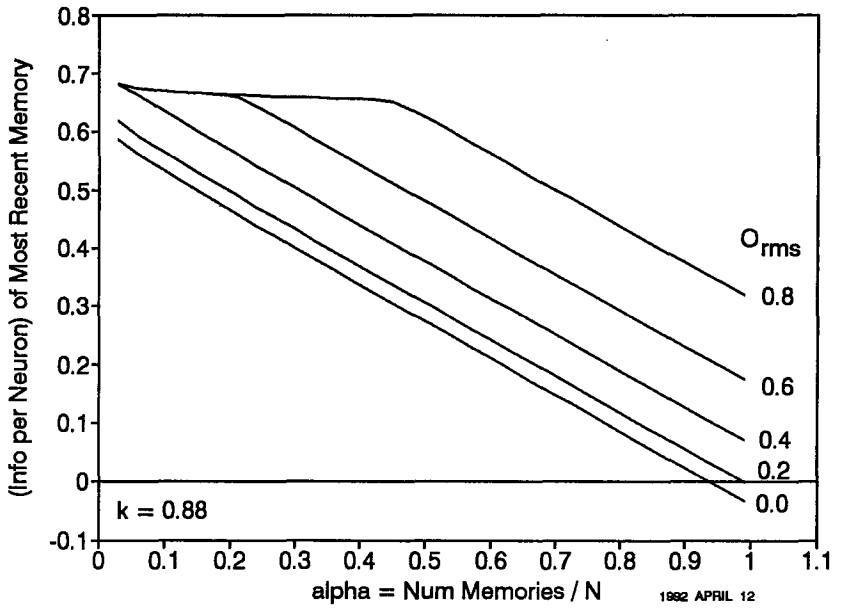


Figure 6.4: Numerical calculation of the information per neuron in the most recently added near-orthogonal memory. The graph is identical to Figure 6.3 except that $k = 0.88$.

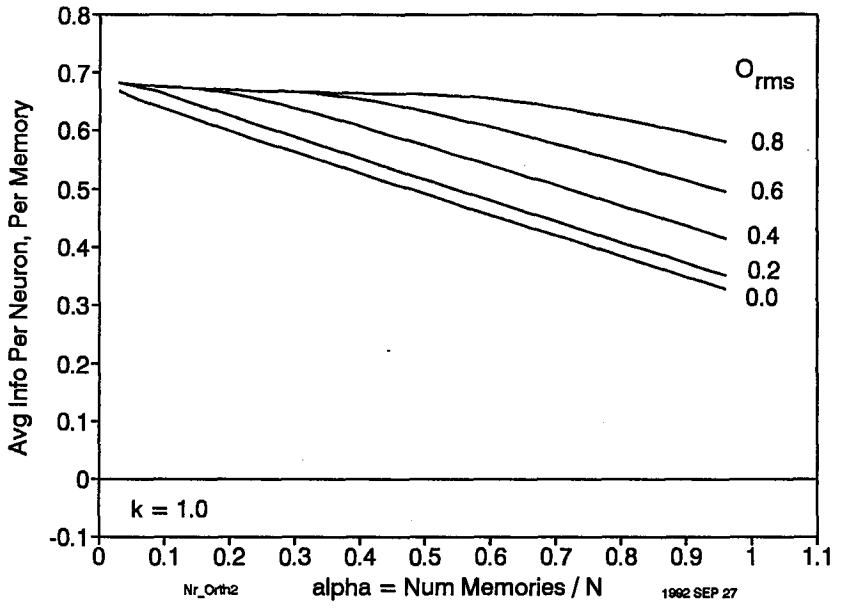


Figure 6.5: Numerical calculation of the average information per neuron, per memory, in a near-orthogonal memory set, as a function of the memory loading $\alpha = M/N$, for $k = 1.0$. The calculations were done by integrating and normalizing the data shown in Figure 6.3.

$$\left.\sum_{\mu=\mu_0+1}^{M-1}\left[k(N-\mu)\log 2 - \log(k(\mu+1)) + \log g_1^+\right]\right\} \qquad (6.31)$$

where g_1^+ is related to b via Equation 6.22. Figures 6.5 and 6.6 show numerical calculations of $\bar{s}(N, M, b)$ for various values of b, and for k equal to unity or 0.88.

6.3 Ideal Information Capacity vs Information Content in EH Networks.

This section relates the information *capacity* calculations of Section 6.2 to the information *content* of EH networks. The term "ideal capacity" is used to refer to the amount of information that can theoretically be stored in a set of memories which are generated as described in Section 2.3.1, and which have a given rms overlap O_{rms}. The term "information content" is used to refer to the amount of information represented by the visible neurons in the memory set of an EH network. It is not immediately obvious how the ideal capacity can be related to

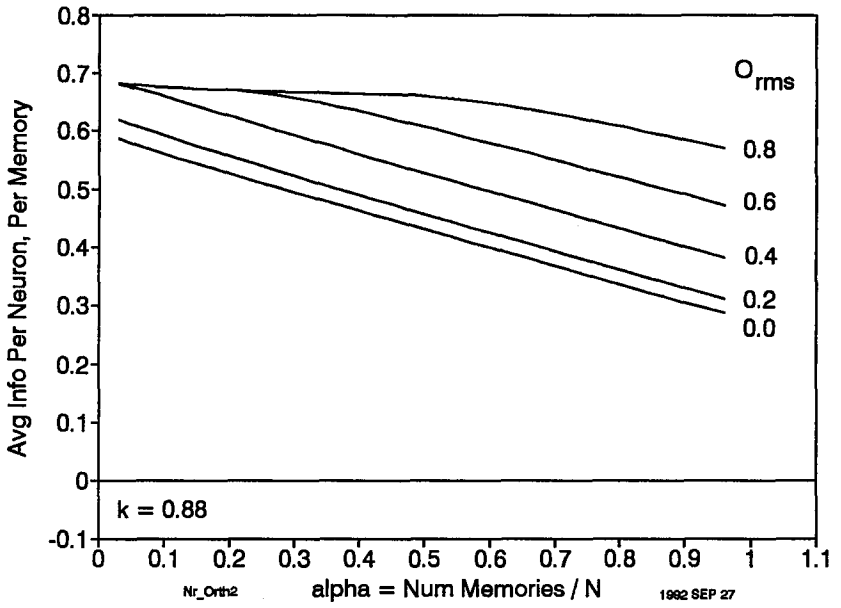


Figure 6.6: Numerical calculation of the average information per neuron, per memory, in a near-orthogonal memory set, as a function of the memory loading $\alpha = M/N$, for $k = 0.88$. The calculations were done by integrating and normalizing the data shown in Figure 6.4.

the information content, because the former was derived for memory sets generated in a very different way from the latter. According to the *ansatz* in Section 2.4, however, it is assumed that the network's behaviour will be independent of the way in which the orthogonality was achieved. This implies that the ideal information capacity is a theoretical upper limit on the allowed information content of EH memories.

Sections 6.3.1 and 6.3.2 discuss how efficiently the EH process matches the information content of each memory set to the ideal capacity of the network. For simplicity, all calculations assume that $k = 0.88$.

6.3.1 Limits on the Orthogonalization Process

Information theory imposes limits on the degree of orthogonalization that can be achieved by the EH network's roll-up process. The limit results from the fact that the more orthogonal

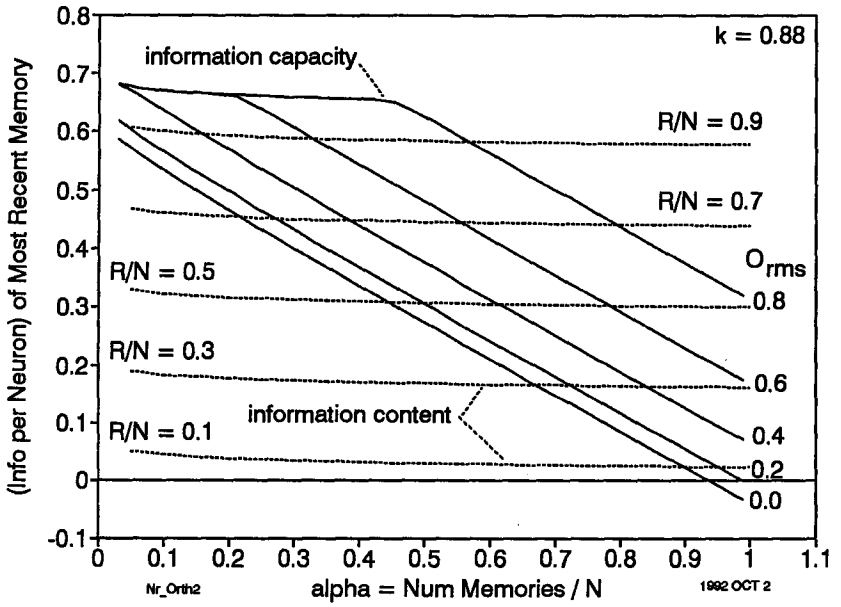


Figure 6.7: Ideal information capacity as a function of the memory loading, compared to the information content of an EH memory. The solid lines are identical to those in Figure 6.4 and show the amount of information that *can be* stored (the *ideal capacity*) in a new memory, for a set of near-orthogonal memories with overlap O_{rms}. The dashed lines are plots of Equation 3.5 for five different values of R/N, and show the amount of information that *is* stored (the *information content*) in the latest EH memory, in a network with R visible neurons.

a memory set becomes, the less information can be stored in it, as illustrated in Figures 6.3 and 6.4. An "optimum" orthogonalization process will therefore reduce O_{rms} only until the ideal capacity $S(N, \mu, k, b)$ of each newly-added memory is equal to its information content $I_{EH}(\alpha, R)$.

The rms overlaps which would be achieved by such an optimum orthogonalizer can be found using Figure 6.7, where the information content of one EH memory, as calculated in Equation 3.5, is superimposed on the ideal capacity from Figure 6.4. The intersections of these graphs indicate the lower limit on O_{rms}, for various numbers of visible neurons and various degrees of memory loading. If the memory loading is so low that the graphs do not intersect, the lower limit on O_{rms} is zero.

Figure 6.8 summarizes the locations of the intersections in Figure 6.7, and thus shows the

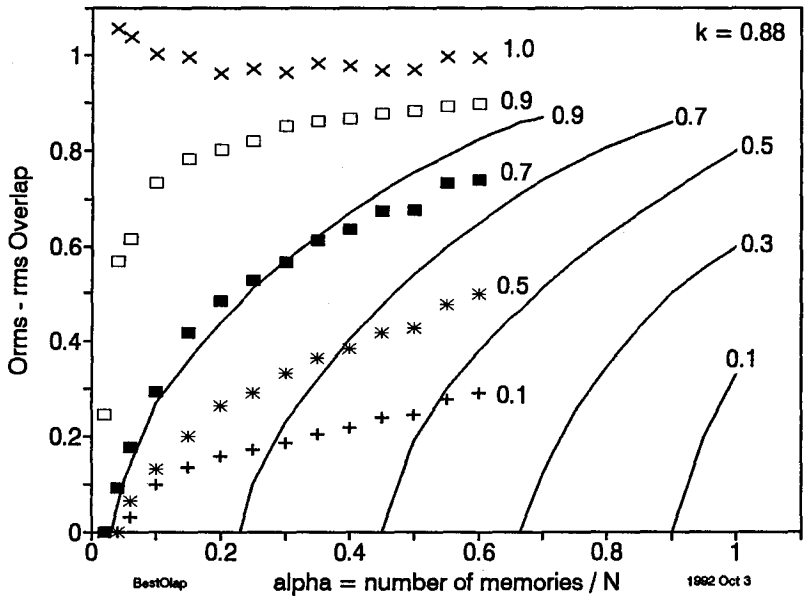


Figure 6.8: Comparison of the theoretical minimum overlap, to the overlap which is achieved by the roll-up process. The decimal numbers indicate the ratio of visible neurons, R/N. The lines plot the locations of the intersections in Figure 6.7, and indicate the theoretical minimum rms overlap required in order to store the given number of randomly-selected visible neurons. The marked data points are copied from Figure 3.2, and show the levels of overlap achieved by the EH network's roll-up process.

minimum value of O_{rms} that is required in order to store a given number of EH memories. The overlap achieved by the roll-up process is shown for comparison. Within the tested range, the achieved overlap is always greater than the theoretical minimum, as expected.

6.3.2 Information Content Near Capacity

This section compares the maximum information content of an EH network to the ideal information capacity of a set of memories with the same level of orthogonality. The comparison provides a measure of how efficiently the network is utilizing the hidden neurons.

The first step in the calculation is to determine how well the memory sets are orthogonalized when the network is filled to capacity, as shown in Figure 6.9. Given a value of R/N, the

memory capacity is given by Figure 3.11, and O_{rms} at capacity is given by Figure 6.9, so the ideal information capacity can be found using Figure 6.6. The result, a graph of the ideal information capacity per neuron at maximum memory loading as a function of R/N, is shown in Figure 6.10. When these values are multiplied by the memory capacity in Figure 3.11, the result is a graph of the ideal information capacity at maximum memory loading, as a function of R/N, as shown in Figure 6.11.

The plots in Figure 6.11 indicate that the information capacity of the EH network is less than the ideal information capacity, particularly for low numbers of visible neurons. In the zone where the memory capacity is highest, the EH network's capacity is between one half and two thirds as large as the ideal network.

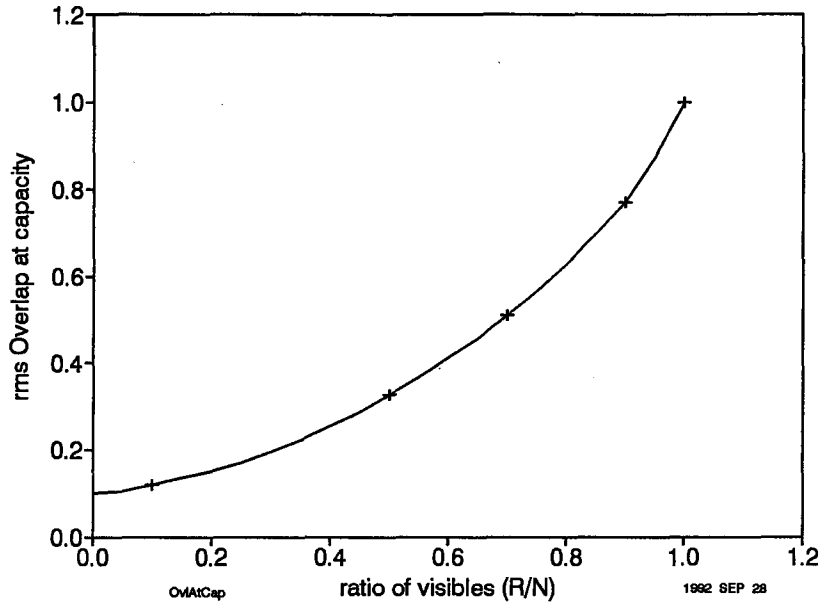


Figure 6.9: Orthogonality of an EH network at capacity, as a function of the ratio of visibles, R/N. The data points were calculated by first finding the capacity as a function of R/N, using Figure 3.11, and then finding the corresponding rms overlap, using Figure 3.2.

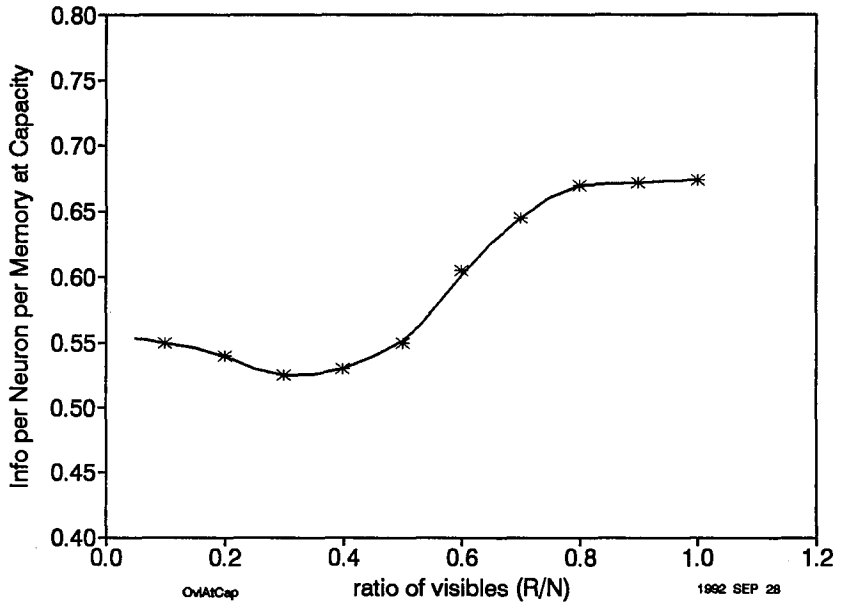


Figure 6.10: Ideal information capacity at maximum loading, as a function of the ratio of visibles. The data points were derived from Figures 3.11 and 6.9, as described in the text. The line was drawn freehand as a visual aid only.

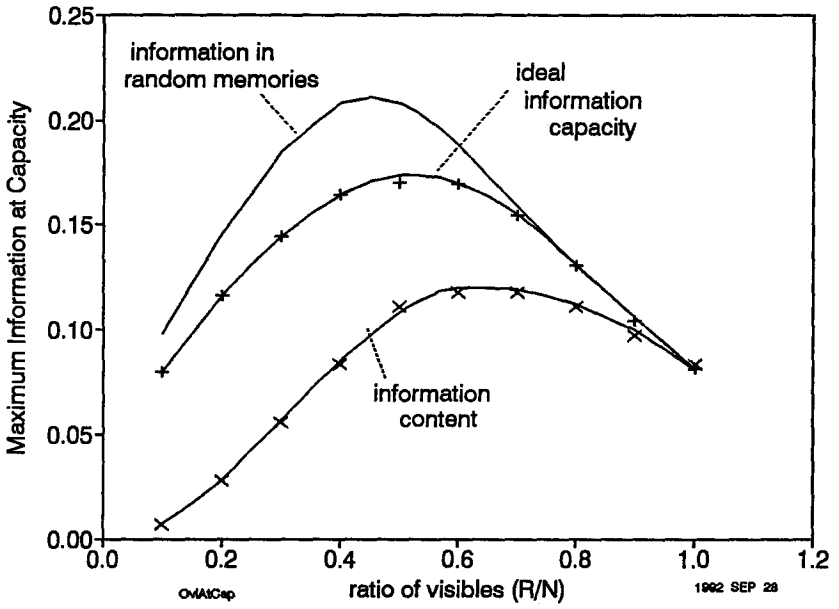


Figure 6.11: Comparison of ideal information capacity and actual information content, at maximum loading. The lowest plot is a copy of Figure 3.12, and shows the total information actually stored in an EH network at maximum loading α_c. The center plot shows the ideal information capacity of a set of α_c memories with rms overlap equal to the rms overlap of the EH network memories at maximum loading. The top plot shows, for reference, the information capacity of a set of $\alpha_c N$ completely random memories. All information capacities are divided by N^2, so that the plotted value is equal to $\frac{\log 2}{N}$ times the number of data bits that can be stored for every neuron in the network. The lower two lines were drawn freehand as visual aids only.

Chapter 7

Conclusions and Discussion

The following sections review the conclusions that have been reached in this research and propose a number of topics which may be pursued in further research.

7.1 Conclusions

All of the research goals listed in section 1.2 have been met. An "Extended Hopfield" or "EH" network has been described which adds hidden neurons to the Hopfield network. The performance of the EH network was compared to the Hopfield network, and it was shown that:

- 45% more information can be stored in an EH network with hidden neurons, compared to a Hopfield network with the same total number of neurons (Figure 3.12).
- for a fixed total number of neurons, up to twice as many memories can be stored and recalled in an EH network as in a Hopfield network (Figure 3.11).
- the radius of attraction for "incomplete" prompts in an EH network, even with no hidden neurons, far exceeds that of a Hopfield network (Figure 3.7).
- when recovering a memory from a "noisy" prompt, a greater percentage of visible bits can be initially wrong in an EH network than in a Hopfield network (Figure 3.16).

- the EH network is able to store the XOR set of associations, something that a Hopfield network cannot do (Figure 3.17).
- the EH network is able to efficiently store and recall sets of memories with variable lengths.

For sets of correlated memories, the capacity of the EH network was found to deteriorate just as rapidly as the Hopfield network as the correlation increased, as shown in Figure 3.13. The EH network was able to store more memories than the Hopfield network, but the performance of both networks was severely degraded compared to their performance with random memories.

"Analytical control" of the EH network was achieved on a number of fronts. Most importantly, an expression for the free-energy of a Hopfield network with a near-orthogonal memory set was derived (Equation 4.49), and an associated set of mean-field equations was found (Equations 4.50, 4.51, and 4.52). The mean-field equations were then used to predict the memory capacity of a Hopfield network as a function of the degree of orthogonality of the memory set (Figure 5.3). The success of these predictions is an encouraging indication that the mean-field equations are correct.

A mathematical description of the average performance of the roll-up process was also derived, resulting in a pair of coupled differential equations (Equations 5.15 and 5.20). These equations were used to predict how well new memories can be orthogonalized as a function of the memory loading and the number of hidden neurons. The analytical predictions matched the observed overlaps very accurately, as shown in Figure 5.8.

Analytical tools were developed to describe the information content of near-orthogonal memory sets. The result is Equation 6.31, which describes the average amount of information per neuron as a function of the number of neurons, the number of memories, and the degree of orthogonality of the memory set. The information capacity of an EH network was compared to the ideal information capacity for a set of memories with the same degree of orthogonality.

The results, shown in Figure 6.11, indicate that the EH network is able to store approximately two thirds of the theoretical maximum amount of information when one third of the neurons are hidden.

The final goal, to predict the memory capacity of the EH network as a function of the fraction of hidden neurons, was partially achieved in Section 5.3, as summarized in Figure 5.11. The analytical predictions were poor for networks made up of mostly hidden neurons, because it ignored the problems associated with recalling a memory from a small number of visible prompt bits. The analytical predictions were good, however, for networks where more than half the neurons were visible. Furthermore, the analysis showed that the improved performance observed in networks of 100 neurons, will be observed in networks of all sizes and will be independent of network size.

7.2 Further Research

The results of the present research suggest many possible topics for further work. There are, for example, many aspects of the behaviour of EH networks which haven't been explored. The most important, as highlighted in Section 5.3, is to analyse how the radius of attraction of EH networks varies with the memory loading and with the orthogonality of the memory set. One possible approach would be to clamp the "visible" neurons, represented by the ξ_j^μ $(j = 1, R)$ in Equation 4.17, at varying degrees of correlation with the memory being recalled, in order to simulate a recall process which begins far from the target memory. It might then be possible, by solving the mean field equations as in Section 5.1, to determine the memory capacity as a function of the radius of attraction.

Other research topics which would help define the behaviour of EH networks are:

- An evaluation of how robust the network is to synaptic damage.
- An analysis of how the EH network behaves at non-zero temperatures
- An investigation into whether time delay synapses can be introduced to an EH network
- A further analysis of why the tie-breaking heuristic works so well for the XOR set of associations, as discussed in Section 3.4.5, and whether it may be of use for more general sets of memories.

There are also a number of topics suggested by this research which do not directly involve the EH network architecture. One of the most interesting would be to investigate how the roll-up process, and hidden neurons, can be used to advantage with correlated sets of memories. It was shown in Section 3.4.3 that the EH network has difficulties with correlated memories, in much the same way that Hopfield's network does. Amit *et al* [5], have described a version of Hopfield's network which is tolerant of correlated memory sets. By rigidly constraining all network states to match the correlation in the memory set, they achieved a network with a memory capacity for correlated memories that exceeds Hopfield's capacity for random memories. A significant problem with their solution is that it imposes such tight constraints on the memory sets. A network with hidden neurons might be able to achieve the rigid constraints on the memory set while still allowing a broad range of memories to be stored.

Perhaps the most valuable result of this research is that it has demonstrated how a neural network can learn new memories, taking advantage of the accumulated experience of all the old memories, but without explicitly reviewing each old memory. This may be an important step forward – the energy landscape is not used just for memory recall, but is also used to optimize its own shape as that shape develops. Further research may find more general applications of this principle to a broader range of network architectures.

Bibliography

[1] Danial J. Amit. *Modeling Brain Function, The world of attractor neural networks.* Cambridge University Press, 1989.

[2] Daniel J. Amit. Neural network counting chimes. *Proceedings of the National Academy of Science, USA*, 85:2141, 1988.

[3] Daniel J. Amit and Hanoch Gutfreund. Spin-glass models of neural networks. *Physical Review A*, 32(2):1007–1018, August 1985.

[4] Daniel J. Amit and Hanoch Gutfreund. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, 55(14):1530–1533, September 1985.

[5] Daniel J. Amit, Hanoch Gutfreund, and H. Sompolinsky. Information storage in neural networks with low levels of activity. *Physical Review A*, 35(5):2293–2303, March 1987.

[6] E.B. Baum, J. Moody, and F. Wilczek. Internal representations for associative memory. *Biological Cybernetics*, 59:217–228, 1988.

[7] T.H. Brown, A.H. Ganong, E.W. Kariss, and C.L.Keenan. Hebbian synapses - computations and biophysical mechanisms. *Annual Review of Neuroscience*, 12:, 1989.

[8] Thomas Brown. Neuronal plasticity and learning. In *IJCNN-91 Tutorial Notes*, IEEE, Seattle, 1991. (bound separately from other tutorial notes).

[9] Philip A. Chou. The capacity of the Kanerva associative memory. *IEEE Transactions on Information Theory*, 35(2):281–298, March 1989.

[10] M. Cottrell. Stability and attractivity in associative memory networks. *Biol. Cyber.*, 58:129–139, 1988.

[11] Michael R. Davenport and Shawn P. Day. Chaotic signal emulation using a recurrent time delay neural network. In *Neural networks for signal processing 2 - proceedings of the 1992 IEEE workshop*, 1992.

[12] Michael R. Davenport and Geoffrey W. Hoffmann. Using hidden neurons with Hebbian learning in a bi-state Hopfield neural network. In *Proceeding of the third annual conference on advances in communication and control systems*, IEEE, Victoria, B.C., October 1991.

[13] M.R. Davenport and G.W. Hoffmann. A recurrent neural network using tri-state hidden neurons to orthogonalize the memory space. *Intl. J. Neural Systems*, 1(2):133–141, 1989.

[14] Shawn P. Day and Michael R. Davenport. Continuous-time temporal back-propagation with adaptable time delays. *I.E.E.E. Transactions on Neural Networks*, 4(2):348–354, March 1993.

[15] Shawn P. Day and Michael R. Davenport. Dispersive networks for nonlinear adaptive filtering. In *Neural networks for signal processing 2 - proceedings of the 1992 IEEE workshop*, 1992.

[16] S.F. Edwards and P.W. Anderson. Theory of spin glasses. *J. Phys*, F5:965–974, 1975.

[17] J.D. Farmer and J.J. Sidorowich. Predicting chaotic time series. *Physical Review Letters*, 59:845–848, 1987.

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

[19] D.O. Hebb. *The organization of behaviour.* Wiley, 1949.

[20] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation. Santa Fe Institute Studies in the Sciences of Complexity*, Addison Wesley, 1991.

[21] A. Herz, B. Sulzer, R. Kuhn, and J.L. van Hemmen. Hebbian learning reconsidered: representation of static and dynamic objects in associative neural nets. *Biological Cybernetics*, 60:457–467, June 1989.

[22] G.E. Hinton and T.J. Sejnowski. Learning and relearning in Boltzmann machines. In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing Explorations in the Microstructure of Cognition*, pages 282–317, MIT Press, 1986.

[23] G.E. Hinton and T.J. Sejnowski. Optimal perceptual inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 448–453, IEEE, Washington, D.C., 1983.

[24] A.L. Hodgkin, A.F. Huxley, and B. Katz. Currents carried by sodium and potassium ions through the membrane of the giant axon of loligo. *J. Physiology*, 116:463, 1952.

[25] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. (USA)*, 79:2554 - 2558, 1982.

[26] E. Ising. Bertrag zur Theorie des Ferromagnetismus. *Zeitschrift Für Physik*, 31:253, 1925.

[27] P. Kanerva. *Self-propagating search: A unified theory of memory.* Technical Report CSLI-84-7, Stanford Center of Language and Information, Stanford, California, March 1984.

[28] A.I. Khinchin. *Mathematical Foundations of Information Theory.* Dover, 1957.

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

[30] Scott Kirkpatrick and David Sherrington. Infinite-ranged models of spin-glasses. *Physical Review B*, 17(11):4384–4403, 1978.

[31] Charles Kittel. *Thermal Physics.* J. Wiley and Sons, 1969.

[32] David Kleinfeld. Sequential state generation by model neural networks. *Proc. Natl. Acad. Sci (USA)*, 83:9469, 1986.

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

[34] S.Y. Kung, F. Fallside, J.Aa. Sorenson, and C.A. Kamm, editors. *Neural networks for signal processing II – Proceedings of the 1992 IEEE workshop*, Princeton, NJ, 1992.

[35] Benny Lautrup. *Lecture notes on the theory of the Hopfield model.* Technical Report NBI-HE-88-06, Niels Bohr Institute, January 1988.

[36] Richard P. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, April 1987.

[37] Richard P. Lippmann. Review of neural networks for speech recognition. In Alex Waibel and Kai-Fu Lee, editors, *Readings in Speech Recognition*, pages 374–392, Morgan Kaufmann, San Mateo, 1990.

[38] M. Mézard, G. Parisi, and M.A. Virasoro. *Spin Glass Theory and Beyond.* World Scientific, 1987.

[39] W.T. Miller III and R.S. Sutton andP.J. Werbos. *Neural Networks for Control.* MIT Press, Cambridge MA, 1991.

[40] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1969.

[41] K.S. Narendra and K. Parthasarathy. Gradient methods for the optimization of dynamical systems containing neural networks. *I.E.E.E. Transactions on Neural Networks*, 2:252–262, March 1991.

[42] John Nolte. *The Human Brain.* C.V. Mosby, St. Louis, 1988.

[43] Conrado J. Perez Vicente and Daniel J. Amit. Optimized network for sparsely coded patterns. *J. Physics A: Math.*, 22:559–569, 1989.

[44] Fernando J. Pineda. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters*, 59(19):2229, 1987.

[45] David E. Rumelhart, Geoffrey E. Hinton, and R.J.Williams. Learning representations by back-propogating errors. *Nature*, 323:533 – 536, October 1986.

[46] David E. Rumelhart and James L. McClelland. *Parallel Distributing Processing: Explorations in the Microstructure of Cognition.* Volume 1: Foundations, MIT Press, 1986.

[47] T. Sejnowski and C.R. Rosenberg. *NETtalk: a parallel network that learns to read aloud.* Technical Report JHU/EECS-86/01, Johns Hopkins University, 1986.

[48] T.J. Sejnowski and P.S. Churchland. Brain and cognition. In M.I. Posner, editor, *Foundations of Cognitive Science*, MIT Press, Cambridge, MA, 1990.

[49] C.E. Shannon. The mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 1948.

[50] David Sherrington and Scott Kirkpatrick. Solvable model of a spin-glass. *Physical Review Letters*, 35:1792–1796, 1975.

[51] H. Sompolinsky and I. Kanter. Temporal association in asymmetric neural networks. *Phys. Rev. Lett.*, 57:2861, December 1986.

[52] D.J. Willshaw, O.P. Buneman, and H.C. Longuet-Higgins. Non-holographic associative memory. *Nature*, 222:960–962, June 1969.

Appendix A

Calculations from Chapter 2

The following sections provide detailed derivations of results which are used to generate near-orthogonal memories in Section 2.3.1.

A.1 The Canonical Set $\{\Gamma\}$ of Exactly Orthogonal Memories

This section defines a method for generating a "canonical" set of N-bit long memories, each of which is exactly orthogonal to all the others. This set is given the symbol $\Gamma^{(N)}$, where N refers to the number of neurons in the network. The μth memory in $\Gamma^{(N)}$ is $\Gamma^{(N)\mu}$, and the state of its ith neuron is $\Gamma_i^{(N)\mu}$. When there is no ambiguity about the number of neurons in the network, this is generally written Γ_i^μ. This canonical set is used to generate near-orthogonal memory sets, as in Equation 2.12, and is used in the derivation which begins at Equation 6.8.

The symbols B^μ and L_i^μ are used to represent the block width and level number in the definition of Γ. They do not correspond to symbols used elsewhere, except in the derivation immediately following Equation 6.8.

The structure chosen for $\Gamma^{(N)\mu}$ is illustrated in table A.1 and defined recursively as follows:

$$\Gamma_i^0 = +1 \qquad \text{for all } i$$

μ	$\Gamma^{(32)\mu}$	L^μ	B^μ	
0	++++++++++++++++++++++++++++++++			
1	++++++++++++++++---------------	1	32	} level 1
2	++++++++--------++++++++--------	2	16	
3	++++++++----------------++++++++	2	16	} level 2
4	++++----++++----++++----++++----	4	8	
5	++++----++++--------++++----++++	4	8	
6	++++--------++++++++--------++++	4	8	} level 4
7	++++--------++++----++++++++----	4	8	
	block 0 block 1 block 2 block 3			
8	++--++--++--++--++--++--++--++--	8	4	
9	++--++--++--++----++--++--++--++	8	4	
⋮	⋮	⋮	⋮	

Table A.1: Listing of the first ten "canonical" orthogonal memories $\Gamma^{(32)\mu}$, for a 32-neuron network. L^μ is the level number, and B^μ is the block width for the μth memory. The memories are numbered starting at zero.

$$
\begin{aligned}
\Gamma_i^1 &= \begin{cases} +1 & \text{if } i \le N/2 \\ -1 & \text{if } i > N/2 \end{cases} \\
L^\mu &= 2^{\text{trunc}[\log_2(\mu)]} \qquad \text{for all } \mu \\
B^\mu &= \frac{N}{L^\mu} \qquad \text{for all } \mu \\
\Gamma_i^\mu &= \begin{cases} \Gamma_i^{L^\mu/2}\Gamma_{[(i+B^\mu/2) \bmod N]}^{L^\mu/2} & \text{if } L^\mu = \mu \\ \Gamma_i^{L^\mu}\Gamma_i^{\mu-L^\mu} & \text{if } L^\mu \neq \mu \end{cases}
\end{aligned}
\tag{A.1}
$$

where "trunc[x]" means the largest integer less than or equal to x.

Table A.1 shows how each memory Γ^μ can be assigned a "level" L^μ, and a "block width" B^μ. Blocks are numbered from 0 through $(L^\mu - 1)$, as illustrated for $\Gamma^{(32)7}$.

A.2 Mean and Variance of $(\xi^\mu \cdot \xi^\nu)$

This section calculates the mean and variance of $(\xi^\mu \cdot \xi^\nu)$. The results are used to derive Equation 2.14.

The overlaps between two different states, ξ^μ and ξ^ν, may be analysed in terms of the following sets of bits:

symbol	definition	size
$\mathcal{A}$	$\{i : \xi_i^\mu = \Gamma_i^\mu\}$	$N(1-b)$
$\mathcal{B}$	$\{i : \xi_i^\nu = \Gamma_i^\nu\}$	$N(1-b)$
$\mathcal{F}$	$(\mathcal{A} \cap \mathcal{B}) \cup (\bar{\mathcal{A}} \cap \bar{\mathcal{B}})$	$N(1-2b+2b^2)$
$\mathcal{G}$	$(\mathcal{A} \cap \bar{\mathcal{B}}) \cup (\bar{\mathcal{A}} \cap \mathcal{B})$	$2Nb(1-b)$

where "size" is the expected size of the set in the limit $N \to \infty$. The mean overlap is expected to be zero because the states are generated with no preference for either +1 or -1. It is calculated by splitting the sum using $\mathcal{F}$ and $\mathcal{G}$:

$$\begin{aligned}
\langle\!\langle \sum_{i=1}^{N} \xi_i^\mu \xi_i^\nu \rangle\!\rangle &= \langle\!\langle \sum_{i \in \mathcal{F}} \xi_i^\mu \xi_i^\nu + \sum_{i \in \mathcal{G}} \xi_i^\mu \xi_i^\nu \rangle\!\rangle \\
&= \langle\!\langle \sum_{i \in \mathcal{F}} \Gamma_i^\mu \Gamma_i^\nu - \sum_{i \in \mathcal{G}} \Gamma_i^\mu \Gamma_i^\nu \rangle\!\rangle \\
&= N(1-2b+2b^2)\langle\!\langle \Gamma_i^\mu \Gamma_i^\nu \rangle\!\rangle - 2Nb(1-b)\langle\!\langle \Gamma_i^\mu \Gamma_i^\nu \rangle\!\rangle \\
&= N(1-4b+4b^2)\langle\!\langle \Gamma_i^\mu \Gamma_i^\nu \rangle\!\rangle \qquad \text{(A.2)}
\end{aligned}$$

The orthogonality of Γ^μ and Γ^ν ensures that $\langle\!\langle \Gamma_i^\mu \Gamma_i^\nu \rangle\!\rangle = 0$, so the mean overlap of pairs of memories is also zero.

The variance of $\xi^\mu \cdot \xi^\nu$ can be calculated as follows:

$$\mathrm{Var}(\xi^\mu \cdot \xi^\nu) = \left\langle\!\left\langle \left(\sum_{i=1}^{N} \xi_i^\mu \xi_i^\nu \right)^2 \right\rangle\!\right\rangle$$

$$= \left\langle\!\left\langle \left(\sum_{i \in \mathcal{F}} \Gamma_i^\mu \Gamma_i^\nu - \sum_{i \in \mathcal{G}} \Gamma_i^\mu \Gamma_i^\nu \right)^2 \right\rangle\!\right\rangle$$

$$= \left\langle\!\left\langle \left(-2 \sum_{i \in \mathcal{G}} \Gamma_i^\mu \Gamma_i^\nu \right)^2 \right\rangle\!\right\rangle \tag{A.3}$$

A general solution for expressions of this type is derived in Section A.3. In this case q in Equation A.9 is set to $2Nb(1-b)$, the size of $\mathcal{G}$, so:

$$\begin{aligned} \mathrm{Var}(\xi^\mu \cdot \xi^\nu) &= 4 \left\langle\!\left\langle \left(\sum_{i \in \mathcal{G}} \Gamma_i^\mu \Gamma_i^\nu \right)^2 \right\rangle\!\right\rangle \\ &= 8Nb(1-b)\left[1 - 2b(1-b)\right] \\ &= N\left[1 - (1-2b)^4\right] \end{aligned} \tag{A.4}$$

A.3 Expectation of Partial Overlaps in Orthogonal Memories

This section derives a probability distribution which is useful for calculating partial overlaps in orthogonal and near-orthogonal memory sets. The situation is equivalent to picking q balls out of a jar holding W white and $N - W$ black balls. The goal is to calculate the probability distribution of s, the number of white balls that are picked, as a function of W and q. The following symbols are local to this section, and do not correspond to symbols used elsewhere:

N = the total number of balls in the jar

W = the number of white balls in the jar

q = the total number of balls taken from the jar

s = the number of white balls taken from the jar

A, C = factors which do not depend on s

B = the factor which does depend on s

L = shifted version of s

The probability of choosing a set with s white balls is:

$$P(s,q,W) = \begin{cases} \binom{W}{s}\binom{N-W}{q-s}\binom{N}{q}^{-1} & \forall s : \begin{cases} 0 \le s < W \\ q \ge s > q+W-N \end{cases} \\ 0 & \text{otherwise} \end{cases}$$

The terms which are of interest are those involving s, so this can be written:

$$P(s,q,W) = A(N,W,q)B(N,W,q,s)$$

where

$$B(N,W,q,s) = \frac{1}{s!\,(W-s)!\,(q-s)!\,(N-W-q-s)!}$$

Notice that B is symmetric in the exchange of W and q, so $B(N,x,y,s) = B(N,y,x,s)$.

In the limit of large N, this can be approximated as a Gaussian distribution. In the special case where $W = \frac{1}{2}N$, B the calculation proceeds as follows:

$$\begin{aligned} B(N,\frac{1}{2}N,q,s) &= \frac{1}{(\frac{q}{2}+L)!\,(\frac{N}{2}-\frac{q}{2}-L)!\,(\frac{q}{2}-L)!\,(\frac{N}{2}-\frac{q}{2}+L)!} \\ &= G\left(\frac{q}{2}\right) \times G\left(\frac{N}{2}-\frac{q}{2}\right) \end{aligned} \tag{A.5}$$

where

$$\begin{aligned} L &\equiv s - \frac{q}{2} \\ G(x) &\equiv \frac{1}{(\frac{q}{2}+L)!\,(\frac{q}{2}-L)!} \end{aligned}$$

The logarithm of the $G(x)$ functions can be simplified following Kittel [31, pp 19-20]. Note that $G(x)$ is an even function of L, so the derivation can assume $L \ge 0$ without any loss of generality:

$$\begin{aligned} \log G(x) &= -\log(x+L)! - \log(x-L)! \\ &= -\log(x!) - \sum_{i=1}^{L}\log(x+i) - \log(x!) + \sum_{i=1}^{L}\log(x-i+1) \end{aligned}$$

$$\begin{aligned} &\approx -2\log(x!) - \sum_{i=1}^{L} \log\left(\frac{x+i}{x-i}\right) \\ &\approx -2\log(x!) - \sum_{i=1}^{L} \log\left(\frac{1+i/x}{1-i/x}\right) \end{aligned}$$

In the limit where $x \gg L$, the logarithm can be approximated as follows:

$$\log G(x) \approx -2\log(x!) - \sum_{i=1}^{L} \frac{2i}{x}$$

Replacing this into Equation A.5, and introducing C and C' to represent terms which do not depend on s:

$$\begin{aligned} \log\left[B(N,\tfrac{1}{2}N,q,s)\right] &= -2\log(\tfrac{q}{2})! - 2\log(\tfrac{N}{2}-\tfrac{q}{2})! - \sum_{i=1}^{L} 4i\left[\frac{1}{q}+\frac{1}{N-q}\right] \\ &= C - 2L(L+1)\left[\frac{1}{q}+\frac{1}{N-q}\right] \\ &= C - 2L(L+1)\left[\frac{N}{q(N-q)}\right] \\ &= C - 2\left(s-\frac{q}{2}\right)\left(1+s-\frac{q}{2}\right)\left(\frac{N}{q(N-q)}\right) \\ &= C' - \left(\frac{2N}{q(N-q)}\right)\left[s+\frac{1}{2}(1-q)\right]^2 \\ &\approx C' - \left(\frac{2N}{q(N-q)}\right)\left[s-\frac{q}{2}\right]^2 \end{aligned}$$

Taking the exponential of both sides gives:

$$B(N,\tfrac{1}{2}N,q,s) = K(N,q)\exp\left[\frac{-(s- < s >)}{\mathrm{Var}(s)}\right]$$

where $K(N,q)$ is a normalizing constant chosen so that

$$\int_{-\infty}^{\infty} P(s)ds = 1,$$

and where:

$$\begin{aligned} \langle s \rangle &= \tfrac{q}{2} && \text{for } W = \tfrac{N}{2} \\ \mathrm{Var}(s) &= \tfrac{q(N-q)}{4N} && \text{for } W = \tfrac{N}{2} \end{aligned}$$

The dependence on W, which was ignored by setting W to $\frac{N}{2}$, can now be inferred from B's symmetry in the interchange of q and W. For this symmetry to be present in the mean and variance, they must have the following values:

$$\langle s \rangle = \frac{Wq}{N} \tag{A.6}$$

$$\mathrm{Var}(s) = \frac{q(N-q)W(N-W)}{N^3}. \tag{A.7}$$

The resulting expression for the probability of picking s white balls is:

$$P(s,q,W) \approx \begin{cases} K(N,q,W)\exp\left[\frac{-(s-\langle s\rangle)^2}{\mathrm{Var}(s)}\right] & \forall s : \begin{cases} 0 \leq s < W \\ q \geq s > q+W-N \end{cases} \\ 0 & \text{otherwise} \end{cases} \tag{A.8}$$

where $K(N,q,W)$ is a normalizing constant.

Figure A.1 illustrates $P(s)$ for various values of W and q, with the normalization calculated numerically. Except in the most extreme cases, the shapes of all the curves appear to be similar to a Gaussian.

The variance $V(q)$ in the overlap of a q-bit long subset of two exactly orthogonal vectors Γ^μ and Γ^ν can be calculated using this result. If "white balls" represent bits which are identical in the two vectors, then $W = \frac{N}{2}$ and the overlap between the vectors is $2s$. The variance in the overlap is therefore:

$$\mathrm{Var}(q) = 4\langle s^2 \rangle = \frac{q(N-q)}{N}. \tag{A.9}$$

Figure A.2 shows the mean measured overlaps between orthogonal vectors for values of q ranging from 0 to N, and confirms that Equation A.9 is correct.

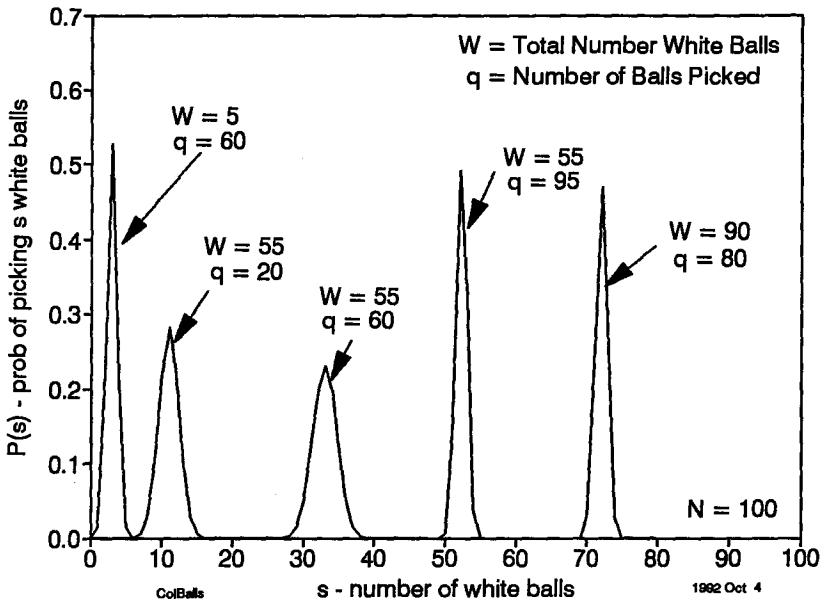


Figure A.1: Probability distributions calculated using Equation A.8, for example values of W and q. If there are N balls, and W of them are white, this is the probability that a random sample of q balls will include exactly s white balls.

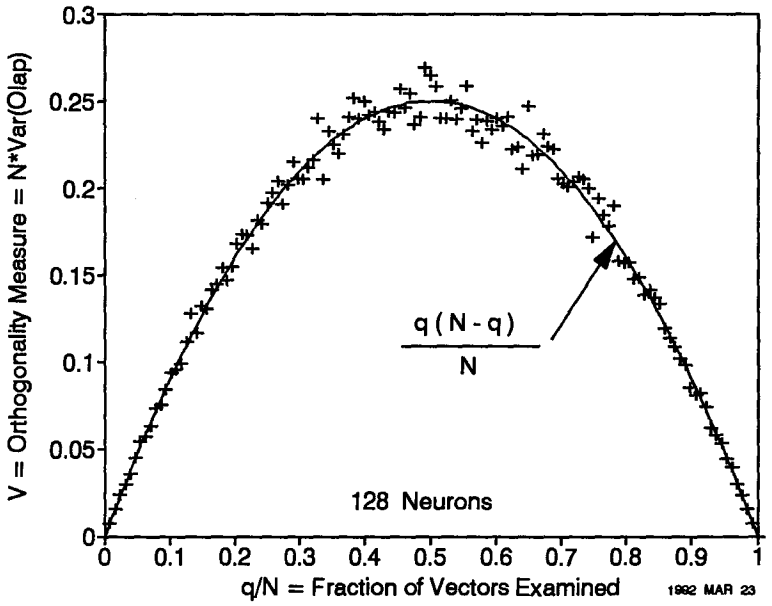


Figure A.2: Observed overlaps between subsets of two orthogonal memories of 128 neurons each. The horizontal scale shows what fraction of the bits were used to calculate the overlap. Each data point was generated using 1000 trials. The line is a graph of Equation A.9.

Appendix B

Calculations from Chapter 3

B.1 Theoretical Minimum Prompt Length

This section calculates the number of bits required to uniquely determine one particular vector in a set of P randomly-generated vectors. In other words, it answers the following question: "given a set of P vectors, on average how many bits are required to uniquely specify one of those vectors?" A reasonable interpretation of this question is: "how many bits must be given before the probability drops below 0.5 that more than one vector in the set will match?"

Assume that the prompt is M bits long. One prompt bit is immediately used up to distinguish between the memories and their bit-inverse images (ref Section 2.1.6). The probability that one $(M-1)$-bit substring will match another is equal to $2^{-(M-1)}$. One memory can be assumed to match the prompt, and the probability that none of the others match is:

$$\frac{(P-1)}{2^{M-1}} \quad < \quad 0.5$$

This simplifies to the following condition on M:

$$M \quad > \quad \log_2 4(P-1). \tag{B.1}$$

This is the number of bits typically required to select one memory from among P memories.

Appendix C

Calculations from Chapter 4

The following sections provide detailed derivations of results which are used to calculate the free energy in Section 4.2.

C.1 Sum-Product Inversions Equation

A common identity used in the literature [1, 35] provides a method for inverting the order of sums and products of exponentials under certain circumstances. The identity is re-stated here as a reference because it is used frequently in the main calculation. The identity can be stated as follows:

$$\sum_{\{S\}} \prod_{i=1}^{N} e^{\alpha_i S_i} = \prod_{i=1}^{N} \sum_{k \in \mathcal{C}} e^{\alpha_i k} \tag{C.1}$$

where the vector $\vec{S}$ is made up of N components S_i, and the possible values of each component make up the set $\mathcal{C}$. The sum on the left side must include all possible vectors $\vec{S}$. The sum on the right side is over all elements of the set $\mathcal{C}$. Most commonly, $\mathcal{C} = \{-1, +1\}$, and the identity can be written:

$$\sum_{\{S\}} \prod_{i=1}^{N} e^{\alpha_i S_i} = \prod_{i=1}^{N} \sum_{s=\pm 1} e^{\alpha_i s} = \prod_{i=1}^{N} 2\cosh(\alpha_i) \tag{C.2}$$

C.2 Ensemble Average of Noise Term over Orthogonal Sets

This section calculates the ensemble average $\left\langle\left\langle e^{T_2}\right\rangle\right\rangle_\Gamma$ from Equation 4.31, where:

$$T_2 \equiv (1-2b)\sum_{\mu=s+1}^{P}\sum_{i=R+1}^{N}\Gamma_i^\mu B_i^\mu$$

The result is used in Equation 4.34 to calculate the amount of noise coming from the uncondensed memories. The following symbols are local to this section, and do not correspond to symbols used elsewhere:

V_T = the variance of T_2

$\hat{V}_T$ = the variance of T_2 if the Γ vectors are random, not orthogonal

In this Appendix, and those following, the dot product refers to a sum over the unclamped indices only, as in for example:

$$X\cdot Y \equiv \sum_{i=R+1}^{N} X_i Y_i$$

The tactic for calculating $\left\langle\left\langle e^{T_2}\right\rangle\right\rangle_\Gamma$ is to calculate the average over $\Gamma^{s+1}\ldots\Gamma^p$ with an integral over all possible values of T_2. The result of that integral must then be averaged over all possible $\Gamma^1\ldots\Gamma^s$. The central limit theorem implies that T_2 will be distributed according to a Gaussian distribution in the $\{\Gamma\}$-space. The mean value of T_2 will be zero because there is no reason why positive or negative components should be favoured in its sum. The variance V_T is equal to the ensemble average of $(T_2)^2$:

$$\begin{aligned} V_T &\equiv \left\langle\left\langle (T_2)^2\right\rangle\right\rangle_{\Gamma^{s+1}\ldots\Gamma^P} \\ &= (1-2b)^2\left\langle\left\langle \sum_{\mu,\nu=s+1}^{P}\sum_{i,j=R+1}^{N} B_i^\mu B_j^\nu \Gamma_i^\mu \Gamma_j^\nu \right\rangle\right\rangle_{\Gamma^{s+1}\ldots\Gamma^P} . \end{aligned} \tag{C.3}$$

The average of e^{T_2} is calculated using the following Gaussian integral:

$$\begin{aligned} \left\langle\left\langle e^{T_2}\right\rangle\right\rangle_{\Gamma} &= \int_{-\infty}^{\infty} dT_2 \, \exp\left[\frac{-(T_2)^2}{2V_T} + T_2\right] \\ &= \left\langle\left\langle \exp\left(\frac{1}{2}V_T\right)\right\rangle\right\rangle_{\Gamma^1 \ldots \Gamma^s} \end{aligned} \tag{C.4}$$

where the average over $\Gamma^1 \ldots \Gamma^s$ is weighted according to the values of m_ρ^μ and $S^\rho \cdot S^\sigma$.

Consider first, as an aside, what the value of e^{T_2} would be if the Γ set was just randomly selected vectors. Nowhere yet in the calculation has the orthogonal nature of Γ been taken into account, so this is a reasonable exercise. Call the random vectors $\hat{\Gamma}$ to avoid confusion. The ensemble average over $\hat{\Gamma}$ in Equation C.3 could be moved inside the summation because all terms in the sum would be independently random:

$$\hat{V}_T = (1-2b)^2 \sum_{\mu,\nu=s+1}^{P} \sum_{i,j=R+1}^{N} B_i^\mu B_j^\nu \left\langle\left\langle \hat{\Gamma}_i^\mu \hat{\Gamma}_j^\nu \right\rangle\right\rangle_{\hat{\Gamma}^{s+1} \ldots \hat{\Gamma}^P}.$$

The ensemble average $\left\langle\left\langle \hat{\Gamma}_i^\mu \hat{\Gamma}_j^\nu \right\rangle\right\rangle$ would be zero unless the indices were identical, so:

$$\begin{aligned} \hat{V}_T &= (1-2b)^2 \sum_{\mu,\nu=s+1}^{P} \sum_{i,j=R+1}^{N} B_i^\mu B_j^\nu \delta^{\mu\nu} \delta_{ij} \\ &= (1-2b)^2 \sum_{\mu=s+1}^{P} \sum_{i=R+1}^{N} (B_i^\mu)^2. \end{aligned}$$

Replacing this into Equation C.4 would give:

$$\left\langle\left\langle e^{T_2}\right\rangle\right\rangle_{\hat{\Gamma}} = \exp\left[\frac{1}{2}(1-2b)^2 \sum_{\mu=s+1}^{P} \sum_{i=R+1}^{N} (B_i^\mu)^2\right]. \tag{C.5}$$

This result is used in Equation 4.33 to check the consistency of the derivation up to this point.

When each set Γ consists of exactly orthogonal memories, the ensemble average in Equation C.3 cannot be moved inside the summations so easily. It is possible , however, to simplify Equation C.3 using the fact that for every set of orthogonal vectors $\{\Gamma\}$, there is another set $\{\Gamma\}'$ whose vectors are all identical except for one, which is the bit-inverse of its counterpart. This

is illustrated by the following two sets of exactly orthogonal 8-vectors:

$$\begin{bmatrix} ++++---- \\ ++--++-- \\ +-+-+--+ \\ ++----++ \end{bmatrix} \Leftrightarrow \begin{bmatrix} ++++---- \\ ++--++-- \\ +-+-+--+ \\ --++++-- \end{bmatrix}$$

Because of this anti-symmetry, the only terms in Equation C.3 which will contribute to the average are those where $\mu = \nu$, so Equation C.3 reduces to:

$$\begin{aligned} V_T &= (1-2b)^2 \left\langle\!\!\left\langle \sum_{\mu=s+1}^{P} \sum_{i,j=R+1}^{N} B_i^\mu B_j^\mu \Gamma_i^\mu \Gamma_j^\mu \right\rangle\!\!\right\rangle_{\Gamma^{s+1}\ldots\Gamma^P} \\ &= (1-2b)^2 \left\langle\!\!\left\langle \sum_{\mu=s+1}^{P} (\Gamma^\mu \cdot B^\mu)^2 \right\rangle\!\!\right\rangle_{\Gamma^{s+1}\ldots\Gamma^P} . \end{aligned}$$

Note that the vectors B^{s+1} through B^P are distinct from each other – they are *not*, for example, the projection of a single vector onto the Γ^μ vectors.

The Γ vectors form a complete orthogonal set in the M-dimensional space of unclamped vector segments, so any vector can be expanded as a sum of its projections along each of the Γ vectors. Expanding B^μ in this way gives:

$$\begin{aligned} |B^\mu|^2 &= \frac{1}{M} \sum_{\nu=1}^{s} (\Gamma^\nu \cdot B^\mu)^2 + \frac{1}{M} \sum_{\nu=s+1}^{M} (\Gamma^\nu \cdot B^\mu)^2 \\ \sum_{\nu=s+1}^{M} (\Gamma^\nu \cdot B^\mu)^2 &= M\,|B^\mu|^2 - \sum_{\nu=1}^{s} (\Gamma^\nu \cdot B^\mu)^2 \end{aligned}$$

If nothing is known about vectors Γ^{s+1} through Γ^N, then each term in the sum has the same expected value, so:

$$\begin{aligned} \left\langle\!\!\left\langle (\Gamma^\alpha \cdot B^\mu)^2 \right\rangle\!\!\right\rangle_{\Gamma^{s+1}\ldots\Gamma^p} &= \frac{1}{M-s} \left[M\,|B^\mu|^2 - \sum_{\nu=1}^{s} (\Gamma^\nu \cdot B^\mu)^2 \right] \\ &\approx \frac{1}{M} \left[M\,|B^\mu|^2 - \sum_{\nu=1}^{s} (\Gamma^\nu \cdot B^\mu)^2 \right] \end{aligned} \tag{C.6}$$

The first term in this expression can be calculated directly:

$$\begin{aligned} |B^\mu|^2 &= \sum_{i=R+1}^{N} (B_i^\mu)^2 \\ &= \sum_{i=R+1}^{N} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho} B^{\mu\sigma} S_i^\rho S_i^\sigma \end{aligned} \tag{C.7}$$

where $B^{\mu\rho}$ is defined in Equation 4.33.

The second term in Equation C.6 can be written as a function of the overlaps in the condensed memories, as follows:

$$\sum_{\nu=1}^{s} (\Gamma^\nu \cdot B^\mu)^2 = \sum_{\rho,\sigma=1}^{n} B^{\mu\rho} B^{\mu\sigma} \sum_{\nu=1}^{s} (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\sigma)$$

so

$$\left\langle\!\left\langle (\Gamma^\alpha \cdot B^\mu)^2 \right\rangle\!\right\rangle_{\Gamma^{s+1} \ldots \Gamma^p} = \frac{1}{N} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho} B^{\mu\sigma} \left[M S^\rho \cdot S^\sigma - \sum_{\nu=1}^{s} (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\sigma) \right]$$

and

$$V_T = (1-2b)^2 \frac{1}{N} \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho} B^{\mu\sigma} \left[M S^\rho \cdot S^\sigma - \sum_{\nu=1}^{s} (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\sigma) \right].$$

Replacing this into Equation C.4 yields the following expression for $\left\langle\!\left\langle e^{T_2} \right\rangle\!\right\rangle_\Gamma$:

$$\begin{aligned} \left\langle\!\left\langle e^{T_2} \right\rangle\!\right\rangle_\Gamma = \Bigg\langle\!\Bigg\langle \exp \Bigg\{ & \frac{1}{2}(1-2b)^2 \frac{1}{M} \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} \\ & \times B^{\mu\rho} B^{\mu\sigma} \left[M S^\rho \cdot S^\sigma - \sum_{\nu=1}^{s} (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\sigma) \right] \Bigg\} \Bigg\rangle\!\Bigg\rangle_{\Gamma^1 \ldots \Gamma^s} \end{aligned} \tag{C.8}$$

To complete the calculation, it must be averaged over all possible orientations of Γ^1 through Γ^s, and expressed as a function of either $\xi^1 \ldots \xi^s$ or $m_\rho^1 \ldots m_\rho^s$, as follows:

$$\left\langle\!\left\langle e^{T_2} \right\rangle\!\right\rangle_\Gamma = \exp \left\{ \frac{1}{2}(1-2b)^2 \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho} B^{\mu\sigma} S^\rho \cdot S^\sigma \right\} \left\langle\!\left\langle e^{-T_3} \right\rangle\!\right\rangle_{\Gamma^1 \ldots \Gamma^s} \tag{C.9}$$

where

$$T_3 \equiv \frac{1}{2M}(1-2b)^2 \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho}B^{\mu\sigma} \sum_{\nu=1}^{s} (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\sigma) \tag{C.10}$$

The average $\left\langle\!\left\langle e^{-T_3} \right\rangle\!\right\rangle_{\Gamma^1 \ldots \Gamma^s}$ is calculated in Section C.3, with the following result from Equation C.16:

$$\left\langle\!\left\langle e^{-T_3} \right\rangle\!\right\rangle_{\Gamma^1 \ldots \Gamma^s} = \exp\left\{ -\frac{1}{2}(1-V^{\frac{1}{4}}) \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho}B^{\mu\sigma} \left[\sum_{\nu=1}^{s} (m_\rho^\nu)^2 \right] S^\rho \cdot S^\sigma \right\} \tag{C.11}$$

When this is replaced into Equation C.9, it reduces to:

$$\left\langle\!\left\langle e^{T_2} \right\rangle\!\right\rangle_{\Gamma} = \exp\left\{ \frac{1}{2} \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} \left[(1-2b)^2 - (1-V^{\frac{1}{4}}) \sum_{\nu=1}^{s} (m_\rho^\nu)^2 \right] B^{\mu\rho}B^{\mu\sigma} S^\rho \cdot S^\sigma \right\} \tag{C.12}$$

C.3 Average of e^{-T_3} for Condensed Vectors

This section calculates the average of e^{-T_3} over Γ^1 through Γ^s, where T_3 is defined in Equation C.10 as

$$T_3 \equiv \frac{1}{2M}(1-2b)^2 \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho}B^{\mu\sigma} \sum_{\nu=1}^{s} (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\sigma).$$

The result is used in Equation C.11 as part of the calculation of the total noise generated by the uncondensed memories. The following symbols are local to this section, and do not correspond to symbols used elsewhere:

$\overline{T_3}$ = the average value of T_3

V_T = the variance of T_3 in the space of Γ^1 through Γ^s

A = a central factor in $\left\langle\!\left\langle e^{-T_3} \right\rangle\!\right\rangle$

The tactic for calculating $\left\langle\left\langle e^{-T_3}\right\rangle\right\rangle_{\Gamma^1 \ldots \Gamma^s}$ is similar to that used for calculating $\left\langle\left\langle e^{T_2}\right\rangle\right\rangle$ in Section C.2. As before, the sum over all possible vectors $\Gamma^1 \ldots \Gamma^s$ is replaced with an integral over all possible values of T_3. Although the central limit theorem does not apply in this case, T_3 is assumed to be distributed according to a Gaussian distribution in the $\{\Gamma\}$-space. This time, however, the mean value is non-zero. If the mean and variance of T_3 are:

$$\begin{aligned} \overline{T_3} &= \langle\langle T_3 \rangle\rangle_{\Gamma^1 \ldots \Gamma^s} \\ V_T &= \left\langle\left\langle (T_3)^2 \right\rangle\right\rangle_{\Gamma^1 \ldots \Gamma^s} - (\overline{T_3})^2 \end{aligned}$$

then the ensemble average will take the following form:

$$\begin{aligned} \left\langle\left\langle e^{-T_3}\right\rangle\right\rangle_{\Gamma^1 \ldots \Gamma^s} &= \int_{-\infty}^{\infty} dT_3 \exp\left[\frac{-(T_3 - \overline{T_3})^2}{2V_T} - T_3\right] \\ &= \exp\left(\frac{1}{2}V_T - \overline{T_3}\right). \end{aligned} \tag{C.13}$$

The mean value of T_3,

$$\overline{T_3} = \langle\langle T_3 \rangle\rangle_{\Gamma^1 \ldots \Gamma^s} \equiv \frac{1}{2M}(1-2b)^2 \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho} B^{\mu\sigma} \sum_{\nu=1}^{s} \langle\langle (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\sigma) \rangle\rangle_{\Gamma^1 \ldots \Gamma^s},$$

is calculated using the average of $(\Gamma^\mu \cdot S^\rho)(\Gamma^\mu \cdot S^\rho)$ derived in Section C.4. In regions where $m \gg \frac{1}{M}$ and $V \ll (1 - \frac{1}{M})$, which covers virtually all the range of interest, Equation C.20 gives the following expression for $\overline{T_3}$:

$$\overline{T_3} = \frac{1}{2}(1-V) \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho} B^{\mu\sigma} \sum_{\nu=1}^{s} (m_\rho^\nu)^2 \, S^\rho \cdot S^\sigma. \tag{C.14}$$

Equation C.23 from Section C.5 presents the following expression for the variance of T_3:

$$\mathrm{Var}(T_3) = (V^{\frac{1}{4}} - V) \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho} B^{\mu\sigma} \sum_{\nu=1}^{s} (m_\rho^\nu)^2 \, S^\rho \cdot S^\sigma. \tag{C.15}$$

Equation C.13 can now be used to calculate $\left\langle\left\langle e^{-T_3}\right\rangle\right\rangle_{\Gamma^1 \ldots \Gamma^s}$:

$$\left\langle\left\langle e^{-T_3}\right\rangle\right\rangle_{\Gamma^1 \ldots \Gamma^s} = \exp\left\{-\frac{1}{2} \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho} B^{\mu\sigma} \sum_{\nu=1}^{s} (m_\rho^\nu)^2 (1 - V^{\frac{1}{4}}) S^\rho \cdot S^\sigma\right\} \tag{C.16}$$

C.4 Average of $(\Gamma^\mu \cdot S^\sigma)(\Gamma^\mu \cdot S^\rho)$ over Condensed Vectors

This section calculates the mean of $(\Gamma^\mu \cdot S^\rho)(\Gamma^\mu \cdot S^\sigma)$ in the space of all orthogonal vectors $\Gamma^1 \ldots \Gamma^s$, with probabilities determined by the values of m_ρ^μ and $S^\rho \cdot S^\sigma$. The result is used to calculate the mean of T_3 in Equation C.14. The following symbols are local to this section, and do not correspond to symbols used elsewhere:

$$
\begin{aligned}
L_i^\mu &= \text{the weighted sum of } S^\rho \text{ vectors} \\
Q &= \text{the value of } \tfrac{1}{M^2}(\Gamma^\mu \cdot S^\rho)(\Gamma^\mu \cdot S^\sigma) \\
m_\rho^\mu &= \text{the overlap between } S^\rho \text{ and } \xi^\mu \\
\mathcal{M}, \mathcal{A}, \ldots \mathcal{G} &= \text{the sets defined below} \\
M &= \text{the number of elements in } \mathcal{M} \\
A, \ldots G &= \text{the number of elements in each set, divided by } M
\end{aligned}
$$

sym	definition	$\frac{1}{M} \times$ avg size	$\frac{1}{M} \times$ variance
$\mathcal{M}$	set of all indices	$1 = \frac{1}{M}(N - R)$	0
$\mathcal{A}$	$\{i \in \mathcal{M} : S_i^\rho = \xi_i^\mu\}$	$A = \frac{1}{2}(1 + m_\rho^\mu)$	0
$\mathcal{B}$	$\{i \in \mathcal{M} : \Gamma_i^\mu = \xi_i^\mu\}$	$\langle B \rangle = (1 - b)$	$b(1-b)$
$\mathcal{C}$	$\{i \in \mathcal{M} : S_i^\rho = \Gamma_i^\mu\}$	$\langle C \rangle = m_\rho^\mu B + \frac{1}{2}(1 - m_\rho^\mu)$	$\left[1 - (m_\rho^\mu)^2\right] b(1-b)$
$\mathcal{D}$	$\{i \in \mathcal{M} : S_i^\rho = S_i^\sigma\}$	$D = \frac{1}{2}(1 + \frac{1}{M} S^\rho \cdot S^\sigma)$	0
$\mathcal{F}$	$\{i \in \mathcal{C} \cap \mathcal{D}\}$	$\langle F \rangle = CD$	$D(1-D)C(1-C)$
$\mathcal{G}$	$\{i \in \mathcal{C} \cap (\mathcal{M} - \mathcal{D}\}$	$G = C - F$	0

The averages and variances in this table were calculated using Equations A.6 and A.7. The probability distributions of the sizes of the sets are assumed to be Gaussian, as for example:

$$P(X) = \exp\left(\frac{-(X-\langle X\rangle)^2}{2\mathrm{Var}(X)}\right) \tag{C.17}$$

The calculation of $(\Gamma^\mu \cdot S^\rho)(\Gamma^\mu \cdot S^\sigma)$ begins with the observation that:

$$\begin{aligned} Q &\equiv \frac{1}{M^2}(\Gamma^\mu \cdot S^\rho)(\Gamma^\mu \cdot S^\sigma) \\ &= \frac{1}{M^2}\left(\sum_{i\in\mathcal{D}} \Gamma_i^\mu S_i^\rho + \sum_{i\in\mathcal{M}-\mathcal{D}} \Gamma_i^\mu S_i^\rho\right)\left(\sum_{i\in\mathcal{D}} \Gamma_i^\mu S_i^\sigma + \sum_{i\in\mathcal{M}-\mathcal{D}} \Gamma_i^\mu S_i^\sigma\right) \\ &= \frac{1}{M^2}\left(\sum_{i\in\mathcal{D}} \Gamma_i^\mu S_i^\rho + \sum_{i\in\mathcal{M}-\mathcal{D}} \Gamma_i^\mu S_i^\rho\right)\left(\sum_{i\in\mathcal{D}} \Gamma_i^\mu S_i^\rho - \sum_{i\in\mathcal{M}-\mathcal{D}} \Gamma_i^\mu S_i^\rho\right) \\ &= \frac{1}{M^2}\left(\sum_{i\in\mathcal{D}} \Gamma_i^\mu S_i^\rho\right)^2 - \frac{1}{M^2}\left(\sum_{i\in\mathcal{M}-\mathcal{D}} \Gamma_i^\mu S_i^\rho\right)^2 \end{aligned} \tag{C.18}$$

The size of these two terms can be calculated using the sets defined above:

$$\begin{aligned} \frac{1}{M}\sum_{i\in\mathcal{D}} \Gamma_i^\mu S_i^\rho &= F-(D-F) = 2F-D \\ \frac{1}{M}\sum_{i\in\mathcal{M}-\mathcal{D}} \Gamma_i^\mu S_i^\rho &= G-(1-D-G) = 2C-2F+D-1. \end{aligned}$$

so:

$$\begin{aligned} Q &= \iiint dF\,dC\,dB\,P(F)\,P(C)\,P(B)\,\left[(2F-D)^2-(2C-2F+D-1)^2\right] \\ &= \iiint dF\,dC\,dB\,P(F)\,P(C)\,P(B) \\ &\qquad \left[(8C-4)F+\,(2-4C)D\,-\,4C^2+4C-1\right] \end{aligned}$$

When this is integrated over F it reduces to:

$$\begin{aligned} Q &= \iint dC\,dB\,P(C)\,P(B) \\ &\qquad \left[(8C-4)\,(CD)+(2-4C)D-4C^2+4C-1\right] \\ &= \iint dC\,dB\,P(C)\,P(B)\,(2D-1)\left[4C^2\,-\,4C\,+\,1\right]. \end{aligned}$$

The second integration is over C, and yields:

$$\begin{aligned} Q &= \int dB\ P(B)\ (2D-1)\ \left[4(\mathrm{Var}(C)+\langle C\rangle^2)\ -\ 4\langle C\rangle\ +\ 1\right] \\ &= \int dB\ P(B)\ (2D-1)\ \left\{\frac{4}{M}(1-m^2)b(1-b)\ +\ 4\left[mB+\frac{1}{2}(1-m)\right]^2\right. \\ &\qquad \left.-\ 4\left[mB+\frac{1}{2}(1-m)\right]\ +\ 1\right\}. \end{aligned}$$

The terms proportional to $\frac{1}{M}$ are negligible except when all other terms are very small, but this is an important exception, so they are kept in:

$$\begin{aligned} Q &= \int dB\ P(B)\ (2D-1)\ \left[\frac{4}{M}(1-m^2)b(1-b)\ +\ m^2(4B^2-4B+1)\right] \\ &= \int dB\ P(B)\ \frac{1}{M}(S^\rho\cdot S^\sigma)\ \left[\frac{4}{M}(1-m^2)b(1-b)\ +\ m^2(4B^2-4B+1)\right] \\ &= \frac{1}{M}(S^\rho\cdot S^\sigma)\ \left[\frac{4}{M}(1-m^2)b(1-b)\ +\ m^2(4\mathrm{Var}(B)+4\langle B\rangle^2-4\langle B\rangle+1)\right] \\ &= \frac{1}{M}(S^\rho\cdot S^\sigma)\ \left\{\frac{4}{M}\left[(1-m^2)b(1-b)+m^2b(1-b)\right]\ +\ m^2\left[4(1-b)^2-4(1-b)+1\right]\right\} \\ &= \frac{1}{M}(S^\rho\cdot S^\sigma)\ \left\{\frac{4}{M}b(1-b)\ +\ m^2(1-2b)^2\right\} \end{aligned} \tag{C.19}$$

The average of $(\Gamma^\mu\cdot S^\rho)(\Gamma^\mu\cdot S^\sigma)$ is therefore:

$$\langle\!\langle(\Gamma^\mu\cdot S^\rho)(\Gamma^\mu\cdot S^\sigma)\rangle\!\rangle_{\Gamma^1\ldots\Gamma^s}\ =\ M(m^\mu_\rho)^2(1-2b)^2\ S^\rho\cdot S^\sigma\ +\ 4b(1-b)\ S^\rho\cdot S^\sigma \tag{C.20}$$

This can be qualitatively checked for some simple cases, and with the aid of the sketch in Figure C.1. When $b=\frac{1}{2}$, the Γ vectors are not correlated with the ξ vectors. In that situation the m^μ_ρ overlaps should be irrelevent, which is indeed true in Equation C.20. If $b=0$, so that the Γ vectors are exactly aligned with the ξ vectors, and if $\rho=\sigma$, so the two S vectors are aligned, then Equation C.20 is equal to the square of $\xi^\mu\cdot S^\rho$, which is correct.

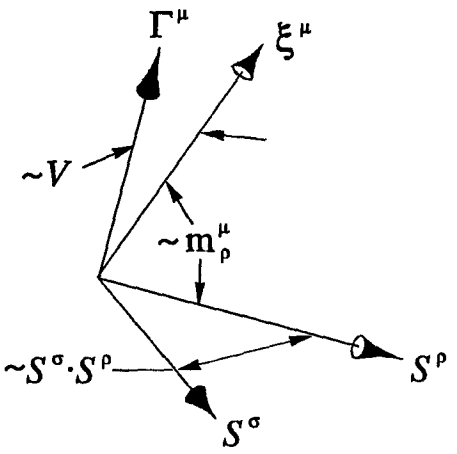


Figure C.1: Sketch of the vectors which make up T_3, for a case where $n = 2$ and $s = 1$. The exact value of T_3 is the product of the projections of S^ρ and S^σ onto Γ^μ. The mean value and variance of T_3 can be estimated as a function of m^μ_ρ, $S^\rho \cdot S^\sigma$, and V.

C.5 Variance of T_3 in the Space of Condensed Vectors

This section estimates the variance of T_3 in the space of all orthogonal vectors $\Gamma^1 \ldots \Gamma^s$. The result is used to calculate the mean of e^{T_3} in Equation C.15.

T_3, as defined in Equation C.10, is:

$$T_3 \equiv \frac{1}{2M}(1-2b)^2 \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho} B^{\mu\sigma} \sum_{\nu=1}^{s} (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\sigma),$$

which is a sum of products of projections of two vectors, S^σ and S^ρ, onto a reference vector Γ^μ. Figure C.1 helps to visualize this.

The variance in T_3 is an estimate of how much the sum of these projection products varies away from the mean value, calculated in Section C.4, as the Γ vectors are rotated in all possible directions. There are only $(n^2 s)$ terms which vary with $\Gamma^1 \ldots \Gamma^s$ and are added together inside T_3. Because this number of terms is small compared to M, the variance in T_3 will be of

the same order as T_3 for $M \to \infty$, and so cannot be ignored.

The exact calculation of $\mathrm{Var}(T_3)$ proceeds as follows:

$$\begin{aligned}
\mathrm{Var}(T_3) &= \left\langle\!\left\langle (T_3)^2 \right\rangle\!\right\rangle_{\Gamma^1 \ldots \Gamma^s} - (\overline{T_3})^2 \\
&= \frac{1}{4M^2}(1-2b)^4 \sum_{\theta,\phi=s+1}^{P} \sum_{\rho,\sigma,\alpha,\beta=1}^{n} B^{\theta\rho} B^{\theta\sigma} B^{\phi\alpha} B^{\phi\beta} \\
&\quad \times \sum_{\mu,\nu=1}^{s} \left\langle\!\left\langle (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\sigma)(\Gamma^\mu \cdot S^\alpha)(\Gamma^\mu \cdot S^\beta) \right\rangle\!\right\rangle_{\Gamma^1 \ldots \Gamma^s} - (\overline{T_3})^2 \qquad \text{(C.21)}
\end{aligned}$$

This can be simplified by expanding only terms up to $n = 2$, because the resulting expression will be used in the limit where $n \to 0$. With that simplification, there are only four different types of terms to be averaged:

$$\begin{aligned}
Q_1^{\nu\mu} &= \langle\!\langle (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\rho)(\Gamma^\mu \cdot S^\rho)(\Gamma^\mu \cdot S^\rho) \rangle\!\rangle_{\Gamma^1 \ldots \Gamma^s} \\
Q_2^{\nu\mu} &= \langle\!\langle (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\rho)(\Gamma^\mu \cdot S^\rho)(\Gamma^\mu \cdot S^\sigma) \rangle\!\rangle_{\Gamma^1 \ldots \Gamma^s} \\
Q_3^{\nu\mu} &= \langle\!\langle (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\rho)(\Gamma^\mu \cdot S^\sigma)(\Gamma^\mu \cdot S^\sigma) \rangle\!\rangle_{\Gamma^1 \ldots \Gamma^s} \\
Q_4^{\nu\mu} &= \langle\!\langle (\Gamma^\nu \cdot S^\rho)(\Gamma^\nu \cdot S^\sigma)(\Gamma^\mu \cdot S^\rho)(\Gamma^\mu \cdot S^\sigma) \rangle\!\rangle_{\Gamma^1 \ldots \Gamma^s} .
\end{aligned}$$

When each of these is simplified separately, the result is a series of terms of a form similar to:

$$\begin{aligned}
\mathrm{Var}(T_3) \approx\ & f_1(V) \sum_{\theta,\phi=s+1}^{P} \sum_{\rho,\sigma,\alpha,\beta=1}^{n} B^{\theta\rho} B^{\theta\sigma} B^{\phi\alpha} B^{\phi\beta} \\
& \times \sum_{\mu,\nu=1}^{s} (m_\rho^\mu)^{n_1} \, (S^\rho \cdot S^\sigma) \qquad \text{(C.22)}
\end{aligned}$$

where n_1 and $f_1(V)$ are constants and functions as required. If this calculation was carried out, it would result in a complicated expression in which every term includes a product of four $B^{\mu\rho}$ factors. Such a calculation was not done in this research. Instead, the value of $\mathrm{Var}(T_3)$ was approximated using an expression which is very similar to Equation C.14, as described below. The result is an equation for $\left\langle\!\left\langle e^{T_3} \right\rangle\!\right\rangle$ that is much simpler than the full calculation would have been, and is accurate in the domain of interest.

The expression for the variance is derived by first establishing its values at the limits $V = 0$ and $V = 1$, and then proposing a simple form for the full range. At $V = 0$, the Γ^μ vector in Figure C.1 is always exactly aligned with ξ^μ, and so has a variance of zero in the space of all allowed Γ vectors. When $V = 1$, the $(1 - 2b)$ factor in T_3 ensures that it is always zero, so the variance is again zero. A simple expression for the functional dependence on $\mathrm{Var}(T_3)$, which is zero in these two limits, is:

$$\mathrm{Var}(T_3) \quad \propto \quad V^{n_1}(1 - V^{n_2})$$

where n_1 and n_2 are positive exponents to be determined. It was found that $n_1 = 0.25$ and $n_2 = 0.75$ work very well.

Following the form of Equation C.14, the variance is therefore written:

$$\mathrm{Var}(T_3) \quad = \quad (V^{\frac{1}{4}} - V) \sum_{\mu=s+1}^{P} \sum_{\rho,\sigma=1}^{n} B^{\mu\rho} B^{\mu\sigma} \sum_{\nu=1}^{s} (m_\rho^\nu)^2 S^\rho \cdot S^\sigma. \tag{C.23}$$

C.6 Details of the Inverse Gaussian Transform

The inverse Gaussian transform of F_1, as defined in Equation 4.36, is most easily calculated if there are no mixed products of the form $x_\rho^\mu x_\sigma^\mu$. Such mixed products can be removed by doing a unitary transformation before the integration, as follows.

Define an $n \times n$ matrix $[\mathbf{K}]$:

$$[\mathbf{K}]_{\rho\sigma} \quad \equiv \quad \left(\delta_{\rho\sigma} - \frac{\beta}{N} C^\rho S^\rho \cdot S^\sigma\right) \tag{C.24}$$

The diagonal elements of $\mathbf{K}$ are $(1 - \gamma\beta C^\rho)$. There must exist a unitary transformation $U_{\nu\rho}$ which diagonalizes $\mathbf{K}$:

$$\sum_{\rho\sigma=1}^{n} U_{\nu\rho} K_{\rho\sigma} U_{\sigma\gamma}^{-1} \quad = \quad \lambda_\nu \delta_{\nu\gamma} \tag{C.25}$$

where λ_ν are the eigenvalues of $\mathbf{K}$. From the definition of $\mathbf{K}$ it follows that:

$$\begin{aligned}\sum_{\rho\sigma=1}^{n} U_{\nu\rho}\delta_{\rho\sigma}U^{-1}_{\sigma\gamma} - \sum_{\rho\sigma=1}^{n} U_{\nu\rho}\frac{\beta}{N}[C^\rho S^\rho \cdot S^\sigma]\,U^{-1}_{\sigma\gamma} &= \lambda_\nu\delta_{\nu\gamma} \\ \sum_{\rho\sigma=1}^{n} U_{\nu\rho}[C^\rho S^\rho \cdot S^\sigma]\,U^{-1}_{\sigma\gamma} &= \frac{N}{\beta}(1-\lambda_\nu)\delta_{\nu\gamma}, \qquad \text{(C.26)}\end{aligned}$$

so U also diagonalizes $[C^\rho S^\rho \cdot S^\sigma]$.

The coordinate transformation proceeds as follows:

$$\begin{aligned}F_1 &= \left(\frac{1}{2\pi}\right)^{\frac{(P-s)n}{2}} \exp -\frac{1}{2}\sum_{\mu=s+1}^{P}\left\{\left[\cdots x^\mu_\rho\cdots\right]\mathbf{K}\begin{bmatrix}\vdots\\ x^\mu_\sigma\\ \vdots\end{bmatrix}\right. \\ &\qquad \left. -2\beta\sqrt{\frac{\beta}{N}}\hat{h}^\mu\sum_{\rho=1}^{n}[C^\rho S^\rho\cdot S^\sigma]\begin{bmatrix}\vdots\\ x^\mu_\sigma\\ \vdots\end{bmatrix} - \beta^2(\hat{h}^\mu)^2\sum_{\rho,\sigma=1}^{n} C^\rho S^\rho\cdot S^\sigma\right\}\end{aligned}$$

$$\begin{aligned}&= \left(\frac{1}{2\pi}\right)^{\frac{(P-s)n}{2}} \exp -\frac{1}{2}\sum_{\mu=s+1}^{P}\left\{\left[\cdots x^\mu_\rho\cdots\right]U^{-1}U\mathbf{K}U^{-1}U\begin{bmatrix}\vdots\\ x^\mu_\sigma\\ \vdots\end{bmatrix}\right. \\ &\qquad \left. -2\beta\sqrt{\frac{\beta}{N}}\hat{h}^\mu\sum_{\rho=1}^{n}U^{-1}U[C^\rho S^\rho\cdot S^\sigma]U^{-1}U\begin{bmatrix}\vdots\\ x^\mu_\sigma\\ \vdots\end{bmatrix} - \beta^2(\hat{h}^\mu)^2\sum_{\rho,\sigma=1}^{n} C^\rho S^\rho\cdot S^\sigma\right\}\end{aligned}$$

$$\begin{aligned}&= \left(\frac{1}{2\pi}\right)^{\frac{(P-s)n}{2}} \exp -\frac{1}{2}\sum_{\mu=s+1}^{P}\left\{\sum_{\rho,\sigma=1}^{n}\lambda_\rho\hat{x}^\mu_\rho\hat{x}^\mu_\sigma\delta_{\rho\sigma}\right. \\ &\qquad \left. -2\sqrt{\beta N}\hat{h}^\mu\sum_{\rho,\sigma=1}^{n}U^{-1}_{\rho\sigma}(1-\lambda_\sigma)\hat{x}^\mu_\sigma - \beta^2(\hat{h}^\mu)^2\sum_{\rho,\sigma=1}^{n} C^\rho S^\rho\cdot S^\sigma\right\} \qquad \text{(C.27)}\end{aligned}$$

where the transformed integration variables $\hat{x}$ are defined as:

$$\begin{bmatrix}\vdots\\ \hat{x}^\mu_\sigma\\ \vdots\end{bmatrix} \equiv U\begin{bmatrix}\vdots\\ x^\mu_\sigma\\ \vdots\end{bmatrix}. \qquad \text{(C.28)}$$

Because this transformation is unitary its Jacobian is 1, and the integral over $\hat{x}$ will be identical to the integral over x. F_1 can be written more simply as:

$$F_1 = \left(\frac{1}{2\pi}\right)^{\frac{(P-s)n}{2}} \exp\left\{-\frac{1}{2}\sum_{\mu=s+1}^{P}\sum_{\rho=1}^{n}\left[\lambda_\rho(\hat{x}_\rho^\mu)^2 - 2\sqrt{\beta N}\hat{h}^\mu\tilde{U}_\rho(1-\lambda_\rho)\hat{x}_\rho^\mu - \beta^2(\hat{h}^\mu)^2\sum_{\sigma=1}^{n}C^\rho S^\rho\cdot S^\sigma\right]\right\}, \tag{C.29}$$

where

$$\tilde{U}_\rho \equiv \sum_{\sigma=1}^{n} U_{\sigma\rho}^{-1}. \tag{C.30}$$

All mixed products of the form $x_\rho^\mu x_\sigma^\mu$ have been removed from this expression, so the inverse Gaussian transformation can now be calculated. Using Equation 4.24, the integral of F_1 is equal to:

$$\begin{aligned} G_1 &\equiv \int\!\!\int_{-\infty}^{\infty}\left(\prod_{\mu=s+1}^{P}\prod_{\rho=1}^{n}d\hat{x}_\rho^\mu\right)F_1 \\ &= \left(\frac{1}{2\pi}\right)^{\frac{(P-s)n}{2}}\prod_{\mu=s+1}^{P}\prod_{\rho=1}^{n}\sqrt{\frac{2\pi}{\lambda_\rho}}\exp\left\{\frac{\beta N(\hat{h}^\mu)^2\tilde{U}_\rho^2(1-\lambda_\rho)^2}{2\lambda_\rho} + \frac{\beta^2}{2}(\hat{h}^\mu)^2\sum_{\sigma=1}^{n}C^\rho S^\rho\cdot S^\sigma\right\} \\ &= \prod_{\mu=s+1}^{P}\frac{1}{\sqrt{\det(\mathbf{K})}}\exp\left\{\frac{\beta N}{2}(\hat{h}^\mu)^2\left[\sum_{\rho=1}^{n}\tilde{U}_\rho^2\frac{(1-\lambda_\rho)^2}{\lambda_\rho} + \frac{\beta}{N}\sum_{\rho,\sigma=1}^{n}C^\rho S^\rho\cdot S^\sigma\right]\right\}. \end{aligned} \tag{C.31}$$

Equation C.31 can be written more simply in terms of the matrix $\mathbf{K}$. The exponent is written in terms of $\mathbf{K}$ using the expression derived in Section C.7, the determinant of $\mathbf{K}$ is moved into the exponent using the identity

$$\log(\det(\mathbf{K})) = \mathrm{Tr}(\log(\mathbf{K})), \tag{C.32}$$

and the sum of all the external field effects $\hat{h}$ are grouped under a single symbol $\tilde{H}$:

$$\tilde{H} \equiv \sum_{\mu=s+1}^{P}(\hat{h}^\mu)^2. \tag{C.33}$$

In addition, following [35], $P-s$ can be replaced by P for very large N, because P scales with N while s remains finite. The result is the following expression for G_1:

$$G_1 \;=\; \exp\frac{N}{2}\left\{-\alpha\mathrm{Tr}(\log\mathbf{K}) \;+\; \tilde{H}\beta\sum_{\rho,\sigma=1}^{n}\left[\mathbf{K}^{-1}\right]_{\rho\sigma} - \beta\tilde{H}n\right\}$$

C.7 Simplification of Exponent in G_1

This section describes how to simplify the exponent in Equation C.31, which is referred to as "X" for this derivation only:

$$X \;\equiv\; \frac{\beta\tilde{H}N}{2}\left[\sum_{\rho=1}^{n}\tilde{U}_\rho^2\frac{(1-\lambda_\rho)^2}{\lambda_\rho} + \frac{\beta}{N}\sum_{\rho,\sigma=1}^{n}C^\rho\, S^\rho\cdot S^\sigma\right].$$

The matrix $\tilde{U}$ is a contraction of a unitary matrix U, and $U_{\rho\nu} = U_{\nu\rho}^{-1}$ for any unitary matrix, so:

$$\tilde{U}_\rho \;\equiv\; \sum_{\sigma=1}^{n}U_{\sigma\rho}^{-1} \;=\; \sum_{\sigma=1}^{n}U_{\rho\sigma}. \tag{C.34}$$

Using this, and the definition of $\mathbf{K}$ from Equation 4.38, X can be written:

$$X \;=\; \frac{\beta\tilde{H}N}{2}\left[\sum_{\rho,\sigma,\nu=1}^{n}(U^{-1})_{\sigma\rho}\left[\frac{1}{\lambda_\rho}-2+(\lambda_\rho)\right]U_{\rho\nu} \;+\; \sum_{\rho,\sigma=1}^{n}[\mathbf{I}-\mathbf{K}]_{\rho\sigma}\right]. \tag{C.35}$$

These are the inverse of the transformations done in Equation C.25, so:

$$\sum_{\nu=1}^{n}(U^{-1})_{\rho\nu}\,\frac{1}{\lambda_\nu}\,U_{\nu\sigma} \;=\; [\mathbf{K}^{-1}]_{\rho\sigma}$$

and

$$\sum_{\nu=1}^{n}(U^{-1})_{\rho\nu}\,\lambda_\nu\,U_{\nu\sigma} \;=\; [\mathbf{K}]_{\rho\sigma}.$$

This means that X is equal to:

$$\begin{aligned} X \;&=\; \frac{\beta\tilde{H}N}{2}\left[\sum_{\rho,\sigma=1}^{n}\left\{\frac{(1-\mathbf{K})^2}{\mathbf{K}}\right\}_{\rho\sigma} + \sum_{\rho,\sigma=1}^{n}[\mathbf{I}-\mathbf{K}]_{\rho\sigma}\right] \\ &=\; \frac{\beta\tilde{H}N}{2}\sum_{\rho,\sigma=1}^{n}\left[\mathbf{K}^{-1}\right]_{\rho\sigma} - \frac{\beta\tilde{H}Nn}{2} \end{aligned} \tag{C.36}$$

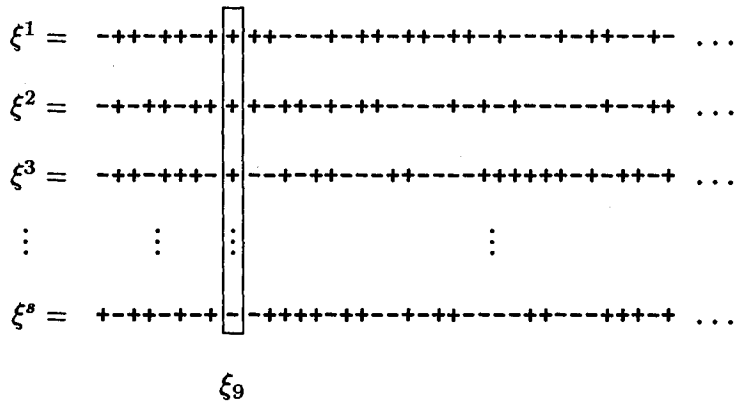
C.8 Self-Averaging in Condensed Near-Orthogonal Memories

This section demonstrates that a sum of the form

$$\sum_{i=1}^{N} f(\xi_i^1 \ldots \xi_i^s) \tag{C.37}$$

is self-averaging when $s \ll N$. This conclusion is not immediately obvious in this case, because the elements of each ξ^μ are generated by adding noise to a set of exactly orthogonal memories Γ, rather than randomly. It will be shown, however, that this structured process for generating memories actually works in favour of self-averaging.

Self-averaging is best understood if the set of s memory vectors are placed together to form an $s \times N$ array of bits. A sketch of such an array is shown here, with the 9th column vector, ξ_9, highlighted:



When $N \gg 2^s$, every possible column vector occurs many times within the sum in expression C.37, and each time it occurs, $f(\xi_i^1 \ldots \xi_i^s)$ contributes the same amount to the sum. If the frequency of each column vector can be estimated accurately, then the sum over all bits i can be replaced by a "self-averaged" weighted sum over all possible column vectors $\{\xi_i\}$ [1, pg. 188].

The first step is to establish the expected frequency of each possible column vector. Let $\{T\}$ be the set of all possible column vectors, and number its members from T_1 to T_{2^s}. Let ϕ_j be the number of times that vector T_j occurs in a given memory set $\{\xi^\mu\}$. All column vectors are

equally likely for near-orthogonal memory sets because they are generated by adding noise to a set of randomly-seeded orthogonal memory sets, so the expected frequencies of all column vectors are the same:

$$\bar{\phi} \equiv \langle\langle \phi_j \rangle\rangle_{\{\xi^\mu\}} = \frac{N}{2^s}.$$

The next step is to show that $N/2^s$ is an increasingly accurate estimate of ϕ_j in the thermodynamic limit. More specifically, it must be shown that the standard deviation in ϕ_j becomes insignificant compared to $\bar{\phi}$ as $(N \to \infty)$. The expected variance in ϕ_j can be written as:

$$\langle\langle \mathrm{Var}(\phi_j) \rangle\rangle \{\xi^\mu\} = \left\langle\left\langle (\hat{\phi}^j)^2 \right\rangle\right\rangle_{\{\xi^\mu\}}$$

where the frequency deviation $\hat{\phi}^j$ is:

$$\hat{\phi}^j \equiv \phi^j - \bar{\phi}. \tag{C.38}$$

The overlap $(\xi^\mu \cdot \xi^\nu)$ between two memories is equal to a weighted sum of the frequency deviations:

$$\begin{aligned} \xi^\mu \cdot \xi^\nu &= \sum_{i=1}^{N} \xi_i^\mu \xi_i^\nu \\ &= \sum_{j=1}^{2^s} \phi_j T_j^\mu T_j^\nu \\ &= \sum_{j=1}^{2^s} \hat{\phi}_j T_j^\mu T_j^\nu + \bar{\phi} \underbrace{\sum_{j=1}^{2^s} T_j^\mu T_j^\nu}_{=0} \end{aligned}$$

The $T_j^\mu T_j^\nu$ factor is $+1$ half the time and -1 half the time, so this sum is equivalent to a random walk of 2^s steps, with step size equal to the standard deviation in ϕ_j. Thus the variance in $(\xi^\mu \xi^\nu)$ is related to the variance in ϕ_j by:

$$\mathrm{Var}(\xi^\mu \cdot \xi^\nu) = 2^s \mathrm{Var}(\phi_j)$$

Using Equation 2.14 as the variance in $\xi^\mu \cdot \xi^\nu$, this leads to:

$$\mathrm{Var}(\phi_j) \;=\; \frac{N}{2^s}\left[1-(1-2b)^4\right]$$

The standard deviation in ϕ_j will therefore be proportional to $\sqrt{N}$, and will become insignificant as N becomes very large. As $b \to 0$ and the memories become more orthogonal, the variance in ϕ_j diminishes, so near-orthogonal memories are even better self-averagers than random memories.

In summary, it has been shown that self-averaging does occur when the memories are near-orthogonal, so the sum in expression C.37 can be replaced as follows:

$$\sum_{i=1}^{N} f(\xi_i^1 \ldots \xi_i^s) \;=\; N \left\langle\!\!\left\langle f(T_j^1 \ldots T_j^s) \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \tag{C.39}$$

C.9 Replica Symmetric Value of $\hat{G}_1$

This section derives an expression for $\hat{G}_{11}$ in the case of replica symmetry, and in the limit of small n. Equation 4.41 indicates that the determinant and trace of $\mathbf{K}$ will be required, so these are calculated first.

The matrix $\mathbf{K}$ for replica symmetric solutions is of the form:

$$\mathbf{K} \;=\; \begin{bmatrix} A & B & B & \cdots \\ B & A & B & \cdots \\ B & B & A & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{C.40}$$

where

$$\begin{aligned} A &\equiv 1-\beta\gamma C \\ B &\equiv -\beta\gamma C q \end{aligned} \tag{C.41}$$

Its eigenvectors are:

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \\ \vdots \\ 0 \end{pmatrix} \cdots \tag{C.42}$$

and the corresponding eigenvalues are:

$$\begin{aligned} \lambda_1 &= A + (n-1)B \\ \lambda_2 \ldots \lambda_n &= (A - B) \end{aligned} \tag{C.43}$$

It follows immediately that the determinant and trace of $\mathbf{K}$ are:

$$\det(\mathbf{K}) = (A-B)^{n-1}(A - B + nB) \tag{C.44}$$

$$\mathrm{Tr}(\mathbf{K}) = A - B + An \tag{C.45}$$

The inverse of $\mathbf{K}$ is calculated as follows:

$$\left[\mathbf{K}^{-1}\right]_{\rho\sigma} \equiv \frac{1}{\det(\mathbf{K})} \frac{\partial \det(\mathbf{K})}{\partial [K]_{\sigma\rho}} \tag{C.46}$$

but:

$$\begin{aligned} \frac{\partial}{\partial A} \det \mathbf{K} &= (A-B)^{n-2}(2A - 2B + nB) \\ \frac{\partial}{\partial B} \det \mathbf{K} &= -(A-B)^{n-2}(2B - 2A - 2Bn + An) \end{aligned} \tag{C.47}$$

so:

$$\begin{aligned} \left[\mathbf{K}^{-1}\right]_{\rho\rho} &= \frac{2A - 2B + Bn}{(A-B)(A - B + nB)} \\ \left[\mathbf{K}^{-1}\right]_{\rho\sigma} &= \frac{2B - 2A - 2Bn + An}{(A-B)(A - B + nB)}. \end{aligned} \tag{C.48}$$

Equation 4.41 requires the following contraction:

$$\sum_{\rho,\sigma=1}^{n} \left[\mathbf{K}^{-1}\right]_{\rho\sigma} = \frac{n(2A - 2B + Bn) \; + \; n(n-1)(2B - 2A - 2Bn + An)}{(A-B)(A - B + nB)}. \tag{C.49}$$

In the limit $n \to 0$ this becomes:

$$\begin{aligned} \sum_{\rho,\sigma=1}^{n} \left[\mathbf{K}^{-1}\right]_{\rho\sigma} &= \frac{4n}{A-B} + O(n^2) \\ &= \frac{4n}{1-\beta\gamma C(1-q)} + O(n^2) \end{aligned} \tag{C.50}$$

Using Equation C.32, the trace of the log of $\mathbf{K}$ can be simplified as follows:

$$\begin{aligned} \text{Tr } \log(\mathbf{K}) &= \log\left[(A-B)^{n-1}(A-B+nB)\right] \\ &= n\log(A-B) + \log\left(1 + n\frac{B}{B-A}\right) \end{aligned} \tag{C.51}$$

In the limit $n \to 0$ this becomes:

$$\begin{aligned} \text{Tr } \log(\mathbf{K}) &= n\log(A-B) + n\frac{B}{B-A} + O(n^2) \\ &= n\left\{\log\left[1+\beta\gamma C(q-1)\right] - \frac{\beta\gamma Cq}{1+\beta\gamma C(q-1)}\right\} + O(n^2) \end{aligned} \tag{C.52}$$

Replacing C.50 and C.52 into Equation 4.41 yields the following expression for $\hat{G}_1$:

$$\hat{G}_1 = \exp Nn\left\{\frac{-\alpha}{2}\log\left[1+\beta\gamma C(q-1)\right] + \frac{\beta(\alpha\gamma Cq + 4\tilde{H})}{2+2\beta\gamma C(q-1)} - \frac{1}{2}\beta\tilde{H}\right\} \tag{C.53}$$

C.10 Replica Symmetric Value of $\hat{G}_2$

The value of $\hat{G}_2$ for a replica-symmetric system is calculated from Equation 4.43. In the limit $n \to 0$ the initial factor becomes unity:

$$\lim_{n\to 0}\left(\frac{\beta^2\alpha N}{2\pi}\right)^{\frac{n(2s+n-1)}{2}} = 1.$$

The sums over ρ and σ in the exponent are straightforward. In the limit of small n, the first sum can be approximated as follows:

$$\begin{aligned} \sum_{\rho=1}^{n-1}\sum_{\sigma=\rho+1}^{n} rq &= \frac{1}{2}n(n-1)rq \\ &\approx -\frac{1}{2}nrq \end{aligned}$$

The resulting expression for $\hat{G}_2$ is:

$$\hat{G}_2 = \exp Nn\beta \left\{ \frac{-\alpha\gamma}{2} + \frac{\beta\alpha rq}{2} + \sum_{\mu=1}^{s} \left[\frac{1}{2}\gamma^2 (m_{\mathrm{h}}^{\mu})^2 + \gamma m_{\mathrm{h}}^{\mu} \hat{h}^{\mu} - \beta\alpha y^{\mu} m_{\mathrm{h}}^{\mu} \right] \right\} \quad \text{(C.54)}$$

C.11 Replica Symmetric Value of F_2

Following Amit [1], the expression for F_2 from Equation 4.45 is simplified for the replica symmetric case as follows:

$$F_2 = \exp N \left\langle\!\!\left\langle \log \sum_{S^1=\pm 1} \cdots \sum_{S^n=\pm 1} \exp \left\{ \alpha\beta^2 \left[\frac{r}{2} \left(\sum_{\rho=1}^{n} S^{\rho} \right)^2 - \frac{rn}{2} + \sum_{\mu=1}^{s} y^{\mu} T^{\mu} \left(\sum_{\rho=1}^{n} S^{\rho} \right) \right] \right\} \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}}.$$

Equation 4.24 is now used to linearize the expression with regard to S^{ρ}:

$$\begin{aligned} F_2 &= \exp N \left\langle\!\!\left\langle \log \sum_{S^1=\pm 1} \cdots \sum_{S^n=\pm 1} e^{-\frac{\alpha\beta^2 rn}{2}} \int_{-\infty}^{\infty} \frac{dz}{\sqrt{2\pi}} \exp - \left[\frac{1}{2} z^2 + F_3 \left(\sum_{\rho=1}^{n} S^{\rho} \right) \right] \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \\ &= \exp N \left\langle\!\!\left\langle \log \left\{ e^{-\frac{\alpha\beta^2 rn}{2}} \int_{-\infty}^{\infty} \frac{dz}{\sqrt{2\pi}} e^{-z^2/2} \sum_{S^1=\pm 1} \cdots \sum_{S^n=\pm 1} \prod_{\rho=1}^{n} e^{-S^{\rho} F_3} \right\} \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \end{aligned}$$

where:

$$F_3 \equiv \beta \left(\sqrt{\alpha r} z + \alpha\beta \sum_{\mu=1}^{s} y^{\mu} T^{\mu} \right) \quad \text{(C.55)}$$

Using Equation C.2, this reduces to:

$$\begin{aligned} F_2 &= e^{-\frac{N\alpha\beta^2 rn}{2}} \exp N \left\langle\!\!\left\langle \log \int_{-\infty}^{\infty} \frac{dz}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \prod_{\rho=1}^{n} \sum_{S=\pm 1} e^{-SF_3} \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \\ &= e^{-\frac{N\alpha\beta^2 rn}{2}} \exp N \left\langle\!\!\left\langle \log \int_{-\infty}^{\infty} \frac{dz}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} (2\cosh F_3)^n \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \\ &= e^{-\frac{N\alpha\beta^2 rn}{2}} \exp N \left\langle\!\!\left\langle \log \int_{-\infty}^{\infty} \frac{dz}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \exp \left[n \log(2\cosh F_3) \right] \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \end{aligned}$$

In the limit $n \to 0$, $e^{nX} = 1 + nX$, and $\log(1 + nX) = nX$, so:

$$\begin{aligned} F_2 &= e^{-\frac{N\alpha\beta^2 rn}{2}} \exp N \left\langle\!\!\left\langle \log \int_{-\infty}^{\infty} \frac{dz}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} [1 + n \log(2\cosh F_3)] \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \\ &= e^{-\frac{\alpha\beta^2 rn}{2}} \exp N \left\langle\!\!\left\langle \log \left[1 + n \int_{-\infty}^{\infty} \frac{dz}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \log(2\cosh F_3) \right] \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \\ &= e^{-\frac{\alpha\beta^2 rn}{2}} \exp Nn \left\langle\!\!\left\langle \int_{-\infty}^{\infty} \frac{dz}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} \log(2\cosh F_3) \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \\ &= \exp Nn \left[\langle\!\langle\!\langle \log(2\cosh F_3) \rangle\!\rangle\!\rangle_{T_1 \ldots T_{2^s}} - \frac{\alpha\beta^2 r}{2} \right] \end{aligned} \tag{C.56}$$

where the triple brackets $\langle\!\langle\!\langle \; \rangle\!\rangle\!\rangle$ represent both an ensemble average over the column vectors T_i and an average over a Gaussian window in z:

$$\langle\!\langle\!\langle X(z, T_i) \rangle\!\rangle\!\rangle \equiv \left\langle\!\!\left\langle \int_{-\infty}^{\infty} \frac{dz}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} X(z, T_i) \right\rangle\!\!\right\rangle_{T_1 \ldots T_{2^s}} \tag{C.57}$$

C.12 Detailed Calculation of the Mean Field Equations

The mean field equations are calculated by requiring that the derivative of the free energy $\langle\!\langle F \rangle\!\rangle$, Equation 4.49, is zero for a set of variables $m_{\rm h}^{\mu}$, y^{μ}, q, and r. The following symbols are local to this section, and do not correspond to symbols used elsewhere:

$$\begin{aligned} A &= \text{denominator in the expression for } y^{\mu} \\ C' &= \text{derivative of } C \text{ with respect to } m_{\rm h}^{\mu} \end{aligned}$$

Setting the derivative with respect to $m_{\rm h}^{\mu}$ to zero provides an equation for y^{μ}:

$$\begin{aligned} \frac{\partial \langle\!\langle F \rangle\!\rangle}{\partial m_{\rm h}^{\mu}} = 0 = \frac{\alpha}{2\beta} \left(\frac{-\beta\gamma(1-q)\,C'}{A} \right) - \frac{\alpha\gamma q C'}{2A} \\ - \frac{(\alpha\gamma C q + 4\tilde{H})\beta\gamma(1-q)C'}{2A^2} - \gamma^2 m_{\rm h}^{\mu} - \gamma \hat{h}^{\mu} + \beta\alpha y^{\mu}, \end{aligned}$$

where:

$$\begin{aligned} C' &\equiv -2\gamma m_{\rm h}^{\mu}(1 - V^{\frac{1}{4}}) \\ A &\equiv 1 - \beta\gamma C(1-q) \end{aligned} \tag{C.58}$$

The equation reduces to:

$$y^\mu = \frac{1}{\alpha\beta}\left[\gamma^2 m_{\rm h}^\mu + \gamma\hat{h}^\mu - G_3(q, m_{\rm h}^\mu)\right] \tag{C.59}$$

where, as in Equation 4.53:

$$G_3(q, m_{\rm h}^\mu) \equiv \frac{\gamma^2 m_{\rm h}^\mu (1 - V^{\frac{1}{4}})}{A^2}\left[\alpha - \alpha\beta\gamma C(1-q)^2 + 4\beta(1-q)\tilde{H}\right] \tag{C.60}$$

Zeroing the derivative with respect to y^μ provides an equation for $m_{\rm h}^\mu$:

$$\begin{aligned}
\frac{\partial \langle\!\langle F \rangle\!\rangle}{\partial y^\mu} &= 0 = \beta\alpha m_{\rm h}^\mu - \frac{1}{\beta}\left\langle\!\!\left\langle\!\!\left\langle \tanh\beta\left(\sqrt{\alpha r}\, z + \alpha\beta\sum_{\nu=1}^{s} y^\nu T^\nu\right)\alpha\beta^2 T^\mu \right\rangle\!\!\right\rangle\!\!\right\rangle \\
m_{\rm h}^\mu &= \left\langle\!\!\left\langle\!\!\left\langle T^\mu \tanh\beta\left(\sqrt{\alpha r}\, z + \alpha\beta\sum_{\nu=1}^{s} y^\nu T^\nu\right)\right\rangle\!\!\right\rangle\!\!\right\rangle
\end{aligned} \tag{C.61}$$

Inserting the value of y^μ, this becomes:

$$m_{\rm h}^\mu = \left\langle\!\!\left\langle\!\!\left\langle T^\mu \tanh\beta\left[\sqrt{\alpha r}\, z + \sum_{\nu=1}^{s}\left(\gamma^2 m_{\rm h}^\nu + \gamma\hat{h}^\nu - G_3(q, m_{\rm h}^\nu)\right)T^\nu\right]\right\rangle\!\!\right\rangle\!\!\right\rangle \tag{C.62}$$

Zeroing the derivative of $\langle\!\langle F \rangle\!\rangle$ with respect to r generates the mean field equation for q:

$$\frac{\partial \langle\!\langle F \rangle\!\rangle}{\partial r} = 0 = \beta\alpha(q-1) + \left\langle\!\!\left\langle\!\!\left\langle \tanh\left(\underbrace{\beta\sqrt{\alpha r}}_{A} z + \underbrace{\alpha\beta^2\sum_{\mu=1}^{s} y^\mu T^\mu}_{B}\right) z\sqrt{\frac{\alpha}{r}}\right\rangle\!\!\right\rangle\!\!\right\rangle$$

$$\beta\sqrt{\alpha r}(1-q) = \int_{-\infty}^{\infty}\frac{dz}{\sqrt{2\pi}} z\, e^{-z^2/2}\,\langle\!\langle \tanh(Az+B)\rangle\!\rangle$$

This can be integrated by parts, using $d/dx\ \tanh x = \mathrm{sech}^2 x$:

$$\begin{aligned}
\beta\sqrt{\alpha r}(1-q) &= \frac{1}{\sqrt{2\pi}}\left\langle\!\!\left\langle \tanh(Az+B)\, e^{-z^2/2}\right\rangle\!\!\right\rangle\Big|_{-\infty}^{\infty} + \int_{-\infty}^{\infty}\frac{dz}{\sqrt{2\pi}} e^{-z^2/2}\left\langle\!\!\left\langle A\,\mathrm{sech}^2(Az+B)\right\rangle\!\!\right\rangle \\
&= \beta\sqrt{\alpha r}\left\langle\!\!\left\langle\!\!\left\langle \mathrm{sech}^2(Az+B)\right\rangle\!\!\right\rangle\!\!\right\rangle \\
q &= 1 - \left\langle\!\!\left\langle\!\!\left\langle \mathrm{sech}^2(Az+B)\right\rangle\!\!\right\rangle\!\!\right\rangle \\
q &= \left\langle\!\!\left\langle\!\!\left\langle \tanh^2\beta\left[\sqrt{\alpha r}\, z + \alpha\beta\sum_{\mu=1}^{s} y^\mu T^\mu\right]\right\rangle\!\!\right\rangle\!\!\right\rangle \\
q &= \left\langle\!\!\left\langle\!\!\left\langle \tanh^2\beta\left[\sqrt{\alpha r}\, z + \sum_{\mu=1}^{s}\left(\gamma^2 m_{\rm h}^\mu + \gamma\hat{h}^\mu - G_3(q, m_{\rm h}^\mu)\right)T^\mu\right]\right\rangle\!\!\right\rangle\!\!\right\rangle
\end{aligned} \tag{C.63}$$

Finally, when the derivative of the free energy with respect to q is set to zero, the result is an equation for r:

$$\begin{aligned} \frac{\partial \langle\langle F \rangle\rangle}{\partial q} &= 0 = \frac{-\beta^2 \gamma C}{2A^2} \left(\alpha \gamma C q + 4\tilde{H} \right) - \frac{\beta \alpha r}{2} \\ r &= \frac{\gamma C \left(\alpha \gamma C q + 4\tilde{H} \right)}{\alpha A^2} \end{aligned} \tag{C.64}$$

Appendix D

Calculations from Chapter 5

The following sections provide detailed derivations of results which are used to calculate the network capacity in Chapter 5.

D.1 Limiting Behaviour of $\beta(1-q)$

The following calculation of $B \equiv \beta(1-q)$ in the limit $\beta \to \infty$ is based on Lautrup's derivation [35, pg. 45]. The following symbols are local to this section, and do not correspond to symbols used elsewhere:

$$\begin{aligned} Y &= \text{the tanh expression in Equations 4.50 and 4.51} \\ a &= \text{the factor multiplying } z \text{ in the argument of } Y \\ b &= \text{the factor not multiplying } z \text{ in the argument of } Y \end{aligned}$$

Define Y as follows:

$$Y \equiv \tanh \beta(az + b)$$

where

$$\begin{aligned} a &\equiv \sqrt{\alpha r} \\ b &\equiv \sum_{\nu=1}^{s} \left\{ \gamma^2 m^\nu + \gamma \hat{h}^\nu - \gamma^2 (1 - V^{\frac{1}{4}}) \left[\frac{\alpha m^\mu + 4\tilde{H}B}{(1-\gamma C B)^2} \right] \right\} T^\nu \end{aligned}$$

Note that:

$$\frac{\partial Y}{\partial b} = \frac{\beta}{\cosh^2 \beta(az+b)} = \beta(1-Y^2) \tag{D.1}$$

The limit of $\beta(1-q)$ is calculated as follows:

$$\begin{aligned} \lim_{\beta\to\infty} \beta(1-q) &= \lim_{\beta\to\infty} \beta \left\langle\!\!\left\langle\!\!\left\langle 1-Y^2 \right\rangle\!\!\right\rangle\!\!\right\rangle \\ &= \lim_{\beta\to\infty} \frac{d}{db} \left\langle\!\!\left\langle\!\!\left\langle Y \right\rangle\!\!\right\rangle\!\!\right\rangle . \end{aligned} \tag{D.2}$$

where $\langle\!\langle\!\langle Y \rangle\!\rangle\!\rangle$ can be written as:

$$\begin{aligned} \lim_{\beta\to\infty} \left\langle\!\!\left\langle\!\!\left\langle Y \right\rangle\!\!\right\rangle\!\!\right\rangle &= \lim_{\beta\to\infty} \frac{-1}{2\beta} \frac{\partial}{\partial b} \left\langle\!\!\left\langle\!\!\left\langle \log(1-Y^2) \right\rangle\!\!\right\rangle\!\!\right\rangle \\ &= \lim_{\beta\to\infty} \frac{1}{\beta} \frac{\partial}{\partial b} \left\langle\!\!\left\langle\!\!\left\langle \log\cosh\beta(az+b) \right\rangle\!\!\right\rangle\!\!\right\rangle . \end{aligned} \tag{D.3}$$

It is possible to integrate out the z-dependence of this expression in the limit of large β by first eliminating the transcendental functions using the following identity:

$$\begin{aligned} \lim_{\beta\to\infty} \log\cosh\beta(az+b) &= \lim_{\beta\to\infty} \left[\log\left(e^{\beta(az+b)} + e^{-\beta(az+b)} \right) - \log 2 \right] \\ &= \lim_{\beta\to\infty} \left[\beta|az+b| - \log 2 \right] . \end{aligned}$$

The average over z is calculated as follows:

$$\begin{aligned} \langle\, |az+b| \,\rangle_z &= \int_{-\infty}^{\infty} \frac{dz}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} |az+b| \\ &= -\int_{-\infty}^{-b/a} \frac{dz}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} (az+b) + \int_{-b/a}^{\infty} \frac{dz}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} (az+b) \\ &= \frac{a}{\sqrt{2\pi}} \left[e^{-\frac{1}{2}z^2} \right]_{-\infty}^{-b/a} - \frac{a}{\sqrt{2\pi}} \left[e^{-\frac{1}{2}z^2} \right]_{-b/a}^{\infty} \\ &\qquad - b\int_{-\infty}^{-b/a} \frac{dz}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} + b\int_{-b/a}^{\infty} \frac{dz}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} \end{aligned} \tag{D.4}$$

The solution of these integrals involves the error function erf(x). The following two properties of erf(x) will be useful:

$$\begin{aligned} \int_x^\infty e^{-u^2/c^2}du &= \frac{c\sqrt{\pi}}{2}\left[1-\mathrm{erf}\left(\frac{x}{c}\right)\right] \\ \frac{d}{dx}\mathrm{erf}(x) &= \frac{2}{\sqrt{\pi}}e^{-x^2} \end{aligned} \tag{D.5}$$

Equation D.4 reduces to:

$$\begin{aligned} \langle\, |az+b|\,\rangle_z &= a\sqrt{\frac{2}{\pi}}\exp\left(\frac{-b^2}{2a^2}\right) - \frac{b}{2}\left[\left(1-\mathrm{erf}\left(\frac{b}{\sqrt{2}\,a}\right)\right)-\left(1-\mathrm{erf}\left(\frac{-b}{\sqrt{2}\,a}\right)\right)\right] \\ &= a\sqrt{\frac{2}{\pi}}\exp\left(\frac{-b^2}{2a^2}\right) + b\,\mathrm{erf}\left(\frac{b}{\sqrt{2}\,a}\right), \end{aligned} \tag{D.6}$$

and its derivatives are:

$$\begin{aligned} \frac{\partial}{\partial b}\langle\, |az+b|\,\rangle_z &= \mathrm{erf}\left(\frac{b}{\sqrt{2}\,a}\right) \\ \frac{\partial^2}{\partial b^2}\langle\, |az+b|\,\rangle_z &= \frac{1}{a}\sqrt{\frac{2}{\pi}}\exp\left(\frac{-b^2}{2a^2}\right). \end{aligned} \tag{D.7}$$

Replacing the first derivative into Equation D.3 gives:

$$\begin{aligned} \lim_{\beta\to\infty}\langle\!\langle\!\langle Y\rangle\!\rangle\!\rangle &= \left\langle\!\left\langle \lim_{\beta\to\infty}\mathrm{erf}\left(\frac{b}{\sqrt{2}\,a}\right)\right\rangle\!\right\rangle \\ &= \left\langle\!\left\langle \mathrm{erf}\left\{\frac{1}{\sqrt{2\alpha r}}\sum_{\nu=1}^{s}\left(\gamma^2 m^\nu + \gamma\hat{h}^\nu - \gamma^2(1-V^{\frac{1}{4}})\frac{\alpha m^\mu + 4\tilde{H}B}{[1-\gamma CB]^2}\right)\right\}\right\rangle\!\right\rangle \end{aligned} \tag{D.8}$$

and when the second derivative is replaced into Equation D.2 the result is:

$$\begin{aligned} B &= \lim_{\beta\to\infty}\beta(1-q) = \lim_{\beta\to\infty}\frac{1}{\beta}\left\langle\!\left\langle \frac{\beta}{a}\sqrt{\frac{2}{\pi}}\exp\left(\frac{-b^2}{2a^2}\right)\right\rangle\!\right\rangle \\ &= \lim_{\beta\to\infty}\left\langle\!\left\langle \frac{1}{a}\sqrt{\frac{2}{\pi}}\exp\left(\frac{-b^2}{2a^2}\right)\right\rangle\!\right\rangle_{T_1\ldots T^{2^s}} \end{aligned} \tag{D.9}$$

D.2 Solution of Mean-field Equations for No Clamped Neurons

In this section an analytical solution is found for the mean-field equations in Section 5.1.2. The equations are

$$\begin{aligned} m &= \mathrm{erf}(y) \\ r &= \frac{C^2}{(1-CB)^2} \end{aligned} \tag{D.10}$$

where:

$$y \equiv \left(1 - \frac{(1-V^{\frac{1}{4}})\alpha}{(1-CB)^2}\right)\frac{m}{\sqrt{2\alpha r}},$$

and

$$B = \frac{\sqrt{2}}{\sqrt{\alpha\pi r}}e^{-y^2}. \tag{D.11}$$

B can be eliminated from the definition of y using Equation D.10

$$\sqrt{2\alpha r}y = m - \frac{(1-V^{\frac{1}{4}})m}{C^2}\alpha r.$$

This is a quadratic equation in $\sqrt{\alpha r}$ whose solution is

$$\sqrt{\alpha r} = \frac{C^2}{2(1-V^{\frac{1}{4}})m}\left[\pm\sqrt{2y^2 + \frac{4m^2(1-V^{\frac{1}{4}})}{C^2}} - \sqrt{2}y\right].$$

Choosing a positive sign will give the correct value in the limit $V \to 0$, so:

$$\sqrt{\alpha r} = \frac{C}{\sqrt{2}(1-V^{\frac{1}{4}})m}\left[\sqrt{y^2C^2 + 2m^2(1-V^{\frac{1}{4}})} - yC\right] \tag{D.12}$$

Replacing this into Equation D.11 gives

$$B = \frac{2(1-V^{\frac{1}{4}})\,\mathrm{erf}(y)\,e^{-y^2}}{\sqrt{\pi}C}\left\{\sqrt{y^2C^2 + 2(\mathrm{erf}\,y)^2(1-V^{\frac{1}{4}})} - Cy\right\}^{-1}. \tag{D.13}$$

The mean-field equation in r can be written:

$$\frac{1}{\sqrt{r}} = \frac{1-CB}{C} \tag{D.14}$$

so the square of Equation D.12 is equal to:

$$\alpha = \frac{1}{2}\left[\frac{1-CB}{(1-V^{\frac{1}{4}})\,\mathrm{erf}(y)}\left(\sqrt{y^2C^2+2(\mathrm{erf}\,y)^2(1-V^{\frac{1}{4}})}-Cy\right)\right]^2. \tag{D.15}$$

When Equation 5.5 is replaced into Equation D.15, the result is an equation for α as an explicit function of y.

D.3 Distribution of X_i Before Roll-Up

This section calculates the mean and the variance of the neural alignments X_i of a randomly selected memory vector before roll-up has occurred. A definition of the neural alignments is given in Equation 5.11.

The ensemble average variance, V_X is, by definition:

$$V_X = \left\langle\!\left\langle X^2 \right\rangle\!\right\rangle - \left\langle\!\left\langle X \right\rangle\!\right\rangle^2.$$

The first term $\langle\!\langle X^2 \rangle\!\rangle$ can be calculated as follows:

$$\begin{aligned}
\left\langle\!\left\langle X^2 \right\rangle\!\right\rangle &= \left\langle\!\left\langle \sum_{\rho=1}^{\mu-1}\sum_{\nu=1}^{\mu-1} m^\rho m^\nu \xi_i^\nu \xi_i^\rho \right\rangle\!\right\rangle \\
&= \sum_{\rho=1}^{\mu-1}\sum_{\nu=1}^{\mu-1}\frac{1}{N^2}\left\langle\!\left\langle \sum_{j=1}^{N}\sum_{k=1}^{N} S_j S_k \xi_j^\rho \xi_k^\nu \xi_i^\rho \xi_i^\nu \right\rangle\!\right\rangle \\
&= \sum_{\rho=1}^{\mu-1}\sum_{\nu=1}^{\mu-1}\frac{\xi_i^\rho \xi_i^\nu}{N^2}\left\langle\!\left\langle \sum_{j=1}^{N}\xi_j^\rho \xi_j^\nu + \underbrace{\sum_{j=1}^{N}\sum_{k\neq j} S_j S_k \xi_j^\rho \xi_k^\nu}_{\approx 0} \right\rangle\!\right\rangle \\
&= \sum_{\rho=1}^{\mu-1}\sum_{\nu=1}^{\mu-1}\frac{1}{N^2}\left\langle\!\left\langle 1 + \sum_{j\neq i}\xi_i^\rho \xi_i^\nu \xi_j^\rho \xi_j^\nu \right\rangle\!\right\rangle \\
&= \frac{(\mu-1)^2}{N^2} + \frac{1}{N^2}\sum_{j\neq i}\sum_{\rho=1}^{\mu-1}\left[1 + \underbrace{\left\langle\!\left\langle \sum_{\nu\neq\rho}\xi_i^\rho \xi_i^\nu \xi_j^\rho \xi_j^\nu \right\rangle\!\right\rangle}_{\approx 0}\right] \\
&= \frac{(\mu-1)^2}{N^2} + \frac{(N-1)(\mu-1)}{N^2}.
\end{aligned} \tag{D.16}$$

The second term can be calculated as follows:

$$\begin{aligned}
\langle\langle X \rangle\rangle &= \left\langle\left\langle \frac{1}{N} \sum_{\nu=1}^{\mu-1} \sum_{j=1}^{N} S_j S_i \xi_i^\nu \xi_j^\nu \right\rangle\right\rangle \\
&= \frac{1}{N} \sum_{\nu=1}^{\mu-1} 1 + \underbrace{\left\langle\left\langle \sum_{\nu=1}^{\mu-1} \sum_{j \neq i}^{N} S_i S_j \xi_i^\nu \xi_j^\nu \right\rangle\right\rangle}_{\approx 0} \\
&= \frac{\mu - 1}{N} = \alpha
\end{aligned} \tag{D.17}$$

So the variance in X_i is:

$$V_X = \frac{(N-1)(\mu-1)}{N^2} \approx \alpha \tag{D.18}$$

If, as assumed in Section 5.2.3, the probability distribution is Gaussian, then the probability distribution of the X_i at time $t = 0$ can be written as follows:

$$P(X,0) = \frac{1}{\sqrt{2\pi\alpha}} \exp\left[\frac{-(X_i - \alpha)^2}{2\alpha}\right] \tag{D.19}$$

D.4 Effect of Inverting One Bit

This section calculates an exact expression for the change in $\sum X_i$ when one neuron in the network is inverted, and uses it to derive an expression for the average time rate of change of $\bar{X}$ as the network rolls up.

The exact expression is derived as follows. The initial states of all the neurons are written S_i, and the final states are written $\hat{S}_i$. Similarly X_i and $\hat{X}_i$ are used to represent the initial and final neural alignments, as defined in Equation 5.11. Let k be the index of the inverted neuron S_k. The new value of X_i, for $(i \neq k)$, is:

$$\hat{X}_{i(\neq k)} = \frac{1}{N} \sum_{\nu=1}^{\mu-1} \sum_{j \neq k} S_i S_j \xi_i^\nu \xi_j^\nu - \frac{1}{N} \sum_{\nu=1}^{\mu-1} S_i S_k \xi_i^\nu \xi_k^\nu$$

$$\begin{aligned} &= X_i - \frac{2}{N}\sum_{\nu=1}^{\mu-1} S_i S_k \xi_i^\nu \xi_k^\nu \\ &= X_i - 2Z_{ik} \end{aligned}$$

where:

$$Z_{ij} \equiv \frac{S_i S_j}{N}\sum_{\nu=1}^{\mu-1} \xi_i^\nu \xi_j^\nu. \tag{D.20}$$

The new value of X_i, for $(i = k)$ is:

$$\begin{aligned} \hat{X}_k &= \frac{-1}{N}\sum_{\nu=1}^{\mu-1}\sum_{j\neq k} S_k S_j \xi_k^\nu \xi_j^\nu + \frac{\mu-1}{N} \\ &= -X_k + 2\alpha \end{aligned}$$

The sum of $\hat{X}_i$ includes $(N-1)$ terms where $i \neq k$, and one term where $(i = k)$. The sum is exactly equal to:

$$\begin{aligned} \sum_{i=1}^{N}\hat{X}_i &= \sum_{i=1}^{N} X_i - 2\sum_{i\neq k} Z_{ik} - 2X_k + 2\alpha \\ &= \sum_{i=1}^{N} X_i - 2\sum_{i=1}^{N} Z_{ik} + 2Z_{kk} - 2X_k + 2\alpha \\ &= \sum_{i=1}^{N} X_i - 4[X_k - \alpha]. \end{aligned} \tag{D.21}$$

This result was derived without making any assumptions about the initial distribution of X_i.

It is now convenient to bring the notation for X_i back in line with what is used in Section 5.2, so that $\hat{X}_i$ becomes $X_i(t+1)$, and X_i becomes $X_i(t)$. It follows immediately from Equation D.21 that the change in the average value of X_i in one time step is equal to:

$$\begin{aligned} \frac{d\bar{X}(t)}{dt} &= \frac{1}{N}\left[\sum_{i=1}^{N} X_i(t+1) - \sum_{i=1}^{N} X_i(t)\right] \\ &= \frac{4}{N}[\alpha - X_k] \end{aligned}$$

Because this is linear in X_k, the average rate of change in $\bar{X}$ over many updates will be

proportional to the average of X_k:

$$\left\langle \frac{\bar{X}(t)}{dt} \right\rangle = \frac{4}{N}[\alpha - \langle X_k \rangle]. \tag{D.22}$$

D.5 Mean Value of X_i for $X_i > \alpha$

This section calculates the average value of all the X_is which are greater than α. The desired value is

$$\langle X \rangle = C_1 \int_\alpha^\infty dX\, X\, P(X,t)$$

where $P(X,t)$ is the Gaussian distribution given in Equation 5.14, and where C_1 is a constant which satisfies the following normalization condition:

$$1 = C_1 \int_\alpha^\infty dX\, P(X,t) \tag{D.23}$$

Both of the above integrals are standard, so their solutions need only be briefly summarized here. The normalization constant is:

$$\frac{1}{C_1} = \sqrt{\frac{V_X \pi}{2}}[1 - \mathrm{erf}(y)]$$

where

$$y \equiv \frac{\alpha - \bar{X}}{\sqrt{2V_X}}$$

The average value of X_i is:

$$\begin{aligned}
\langle X \rangle &= C_1 \int_\alpha^\infty dX\, X \exp\left[\frac{-(X-\bar{X})^2}{2V_X}\right] \\
&= C_1 \int_{\alpha-\bar{X}}^\infty dX(X+\bar{X}) \exp\left[\frac{-X^2}{2V_X}\right] \\
&= C_1 \bar{X} \sqrt{\frac{V\pi}{2}}[1 - \mathrm{erf}(y)] + C_1 \left[-V_X \exp\left(\frac{-X^2}{2V_X}\right)\right]_{\alpha-\bar{X}}^\infty .
\end{aligned}$$

Replacing the value for C_1 gives:

$$\langle X \rangle = \bar{X} + \sqrt{\frac{2V_X}{\pi}} \frac{e^{-y^2}}{1 - \mathrm{erf}(y)}$$

D.6 Estimate of the Variance $V_X(t)$

This section derives a relationship between the average neural alignment $\bar{X}(t)$, and the variance in the neural alignments, $V_X(t)$. The derivation begins by examining the ensemble average of $(X_i)^2$:

$$\begin{aligned}
\left\langle\left\langle (X_i)^2 \right\rangle\right\rangle &= \left\langle\left\langle \frac{1}{N}\sum_{i=1}^{N}(X_i)^2 \right\rangle\right\rangle \\
&= \left\langle\left\langle \frac{1}{N^3}\sum_{i,j,k=1}^{N}\sum_{\rho,\nu=1}^{\mu-1} S_i S_j S_i S_k \xi_i^\nu \xi_j^\nu \xi_i^\rho \xi_k^\rho \right\rangle\right\rangle \\
&= \left\langle\left\langle \frac{1}{N^2}\sum_{j,k=1}^{N}\sum_{\nu=1}^{\mu-1} S_j S_k \xi_j^\nu \left(\sum_{\rho=1}^{\mu-1}\xi_k^\rho m^{\nu\rho}\right) \right\rangle\right\rangle
\end{aligned} \tag{D.24}$$

The overlap $m^{\nu\rho}$ between two memories can be used to calculate the probability that any two bits ξ_k^ν and ξ_k^ρ are identical, as follows:

$$\begin{aligned}
P(\xi_k^\nu = \xi_k^\rho) &= \frac{1+m^{\nu\rho}}{2} \\
P(\xi_k^\nu \neq \xi_k^\rho) &= \frac{1-m^{\nu\rho}}{2}
\end{aligned}$$

The average value of the parenthesized sum in Equation D.24 can therefore be written as follows:

$$\begin{aligned}
\left\langle\left\langle \sum_{\rho=1}^{\mu-1} m^{\nu\rho}\xi_k^\rho \right\rangle\right\rangle_\xi &= \left\langle\left\langle \sum_{\rho=1}^{\mu-1}\left[m^{\nu\rho}\left(\frac{1+m^{\nu\rho}}{2}\right)(\xi_k^\nu) + \left(\frac{1-m^{\nu\rho}}{2}\right)(-\xi_k^\nu)\right] \right\rangle\right\rangle_\xi \\
&= \left\langle\left\langle \sum_{\rho=1}^{\mu-1}(m^{\nu\rho})^2\xi_k^\nu \right\rangle\right\rangle_\xi \\
&= \left\langle\left\langle \sum_{\rho\neq\mu}^{\mu-1}(m^{\nu\rho})^2 \right\rangle\right\rangle_{\nu\neq\rho} \xi_k^\nu + \xi_k^\nu \\
&= \left[(\mu-2)\left\langle\left\langle (m^{\nu\rho})^2 \right\rangle\right\rangle_{\nu\neq\rho} + 1\right]\xi_k^\nu \\
&\approx k_1\,\xi_k^\nu
\end{aligned} \tag{D.25}$$

where

$$k_1 \equiv \alpha(O_{\text{rms}})^2 + 1$$

and where $O_{\rm rms}$ is the rms overlap between previously-installed memories.

When Equation D.25 is used to replace the parenthesized factor in Equation D.24, the result is:

$$\begin{aligned} \left\langle\left\langle (X_i)^2 \right\rangle\right\rangle &= k_1 \frac{1}{N^2} \sum_{j,k=1}^{N} \sum_{\nu=1}^{\mu-1} S_j S_k \xi_j^\nu \xi_k^\nu \\ &= k_1 \bar{X} \end{aligned}$$

where $\bar{X}$ is the average value of X_i, as used in Equation 5.14. The variance in X_i is therefore:

$$\begin{aligned} V_X &= \left\langle\left\langle (X_i)^2 \right\rangle\right\rangle - \langle\langle X_i \rangle\rangle^2 \\ &= k_1 \bar{X} - \bar{X}^2 \\ &= \bar{X} \left[\alpha (O_{\rm rms})^2 + 1 - \bar{X} \right] \end{aligned} \tag{D.26}$$

D.7 Time Evolution of the Number of Invertible Neurons

This section derives a differential equation which describes how the number of invertible neurons varies with time. The result is used in Section 5.2.4 to calculate the rms overlap of the network after roll-up.

An "invertible" neuron is one which might be inverted as the network rolls up. For that to be possible, it must be unclamped, and it must have ($X_i > \alpha$). The symbol $B(t)$ represents the total number of invertible neurons at time t.

Let $\mathcal{F}$ be the set of all neurons, clamped or unclamped, for which $X_i > \alpha$. The population of $\mathcal{F}$ is:

$$|\mathcal{F}| = N P_T$$

where P_T is the total probability that any neuron has $X_i > \alpha$. The value of P_T is equal to the area under the graph of $P(X,t)$ and to the right of α in Figure 5.4, and can be calculated as

follows:

$$\begin{aligned} P_T &= \int_\alpha^\infty dX\, P(X,t) \\ &= \frac{1}{2}\left[1 - \mathrm{erf}\left(\frac{\alpha - \bar{X}}{\sqrt{2V_X}}\right)\right] \end{aligned} \tag{D.27}$$

There are two methods by which neurons leave $\mathcal{F}$:

1. they may be inverted (this happens to exactly one neuron in each time step), or
2. their neural alignments may become smaller than α due to the changes in $\bar{X}$ and V_X.

The total number of neurons that leave $\mathcal{F}$ by the second method is equal to one less than the total number that leave:

$$\text{total loss by second method} = -\frac{d}{dt}(NP_T) - 1.$$

It is assumed that invertible neurons are just as likely to leave by the second method as any other neurons, so the reduction in $B(t)$ by the second method is proportional to the total loss by the second method:

$$\text{loss of invertibles by second method} = -\frac{B}{NP_T}\left[\frac{d}{dt}(NP_T) + 1\right]$$

The total change in the number of invertibles is negative, and is the sum of the amounts lost by the two methods:

$$\frac{dB}{dt} = -1 + \frac{B}{NP_T}\left[\frac{d}{dt}(NP_T) + 1\right] \tag{D.28}$$

This expression can be made independent of the size of the network N by defining:

$$b \equiv \frac{B}{N} \qquad\qquad \tau \equiv \frac{t}{N}.$$

The differential equation can then be written:

$$\frac{db}{dt} = -1 + \frac{b}{P_T}\left[\frac{dP_T}{d\tau} + 1\right] \tag{D.29}$$