Catastrophic Forgetting and the Pseudorehearsal Solution in Hopfield Networks

Simon McCallum

a thesis submitted for the degree of

Doctor of Philosophy

at the University of Otago, Dunedin,

New Zealand.

29 June 2007

Abstract

Most artificial neural networks suffer from the problem of catastrophic forgetting, where previously learnt information is suddenly and completely lost when new information is learnt. Memory in real neural systems does not appear to suffer from this unusual behaviour. In this thesis we discuss the problem of catastrophic forgetting in Hopfield networks, and investigate various potential solutions. We extend the pseudorehearsal solution of Robins (1995) enabling it to work in this attractor network, and compare the results with the unlearning procedure proposed by Crick and Mitchison (1983). We then explore a familiarity measure based on the energy profile of the learnt patterns. By using the ratio of high energy to low energy parts of the network we can robustly distinguish the learnt patterns from the large number of spurious "fantasy" patterns that are common in these networks. This energy ratio measure is then used to improve the pseudorehearsal solution so that it can store $0.3N$ patterns in the Hopfield network, significantly more than previous proposed solutions to catastrophic forgetting. Finally, we explore links between the mechanisms investigated in this thesis and the consolidation of newly learnt material during sleep.

Acknowledgements

There have been many people who have helped me over the course of this PhD. The completion of this thesis would not have been possible without the help, support, and friendship of my supervisor, Anthony Robins. He has helped me grow as a researcher and as a person. There are many other that I need to give thanks to. To all four of my parents who have loved me unconditionally. Special thanks go to my mother, who proof read my thesis at short notice. To my brother Peter who taught me the value of patience, and who has been a rock through many hard times. To my sisters, Ruth and Margaret who have grown up while I have been working. To Jayson Mackie for giving feedback on the final draft. And finally to the women who have shared this journey with me, Liana, Trish, and of course Rachael. It has been a long and winding road, but I finally made it. Thank you all.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

One of the most impressive characteristics found in mammals is their ability to adapt to new environments. This adaptability relies on learning and memory. The ability to generalise learnt responses to novel situations is of significant evolutionary advantage, allowing the animal to respond to changing environments and to extract the important similarities between situations. Without the ability to generalise, every situation appears to be unique and atomic, requiring a unique behavioural response. How is it that animals are able to learn that pressing a red lever will release food regardless of the context, and then later learn that the lever will only work after a bell is rung?

This ability to generalise about the results of actions and to select an appropriate action given a particular situation is vital to the survival of the individual. The ability to store the results of previous experience is called memory. The taxonomy of memory we will use is in line with the work of Tulving (1987) in which he breaks memory into three categories – procedural, semantic, and episodic.

Procedural memory is the most primitive of the memory types. This type of memory describes an animal's ability to learn to perform a task, a "procedure", allowing the learning of sequences of actions. Procedural memory does not require an explicit representation of time, it only needs to store the next action to perform. For example, the ability to hit a tennis ball is a procedural memory. The ability to hit the ball does not depend on memory of a particular instance in which you hit a tennis ball. Instead, the memory is stored as a sequence of actions coupled with a perceptual feedback system that keeps the racquet on target.

Semantic memory refers to the learning of facts and concepts, the meaning behind the structural relationships in the environment. This is the most abstract form of memory. Such memories do not elicit action, nor are they related to any distinct

event. However, semantic memory is very important when thinking and talking about the world around us. These memories allow us to extract meaning from repeated experience and to develop theories about the relationships between experiences. By testing these theories of association we can develop the fundamental structure needed for abstract thought.

Episodic memory is memory for specific events or episodes. This requires some form of temporal component to locate the event in time. This is the type of memory that is most commonly associated with the word "memory" and it provides us with our sense of identity, allowing us to feel the passage of time and to reflect on events in the past. This reflection on the past allows for learning to occur even after an event has passed.

Our memories are precious to each of us, they define who we are and how we respond to our environment. The study of memory and the mechanisms behind memory is critical both as an introspective analysis of ourselves and as a computational problem from a computer science perspective. By modelling biological memory we hope to learn more about how memory works in mammals and gain insight into how to develop software which can interact with humans in a more natural way.

This thesis is focused on episodic memory. We are interested in how memory is stored over time and in particular how mammals avoid the problems associated with information overload and the subsequent forgetting of old information. It is obvious that critical information is encoded early in an animal's life, and that certain responses learnt in these early years need to remain robust throughout the animal's lifetime. This poses a dilemma regarding how to protect the relevant information from the past while remaining flexible enough to adapt to new environments.

1.1 Thesis Scope

This thesis investigates the problem of catastrophic forgetting in content addressable memory (see Section 1.3–1.4). The scope of this investigation is limited to episodic memory, where each pattern is only available for the duration of a single learning episode, and tested after each subsequent pattern is learnt. The model of content addressable memory used is the Hopfield network with simple units in a fully connected architecture. The evaluation of the solutions to catastrophic forgetting is based on three parameters - the number of patterns that can be retrieved, the complexity of the solution, and the potential as an explanation of how biological systems solve this

problem. The exact process for memory consolidation in biological content addressable memory is unclear. This thesis presents a possible mechanism for consolidation, which performs better than competing theories.

1.2 Memory Mechanisms

The exact mechanism used to store long term memory in biological neural networks is still unknown. We know that there are certain areas of the brain that are vital for the development and storage of information. It has been shown that both the neocortex and the hippocampus are important for episodic memory (Atkinson and Shiffrin, 1968; Meeter, 2003). Models of short term memory (STM) and long term memory (LTM) rely heavily on lesion studies, CAT scans, and more recently fMRI, to give us an insight into the functioning of the mammalian brain. These methods investigate whole brain regions, each containing millions of individual neurons,the functional units of the brain, and hundreds of millions of synapses, the connections between the neurons. The results from these studies show us the areas of the brain that are important, and the structures that are involved in memory formation and recall. However, these methods are currently still too coarse to detect the minute and subtle changes that are occurring during memory formation.

In 1949, Donald Hebb suggested a mechanism that could alter the synaptic connections between neurons, so that structure would develop in the network of neurons. Hebb proposed that when two neurons were active at the same time, the likelihood of the same neurons firing at the same time in the future was increased.

> "When an axon of cell A is near enough to excite a cell B and repeatedly and persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased." Hebb (1949).

The biological basis for this process was described by Bliss and Lømo (1973). This work expanded on the experimental results from work done in 1966 by Lømo (Lømo, 2003; Bliss, Collingridge, and Morris, 2003). The long term change in the effectiveness of a neuron to activate another neuron is called Long Term Potentiation (LTP) (Bliss and Lømo, 1973). LTP has been further divided into early-phase LTP (E-LTP) and late-phase LTP (L-LTP). E-LTP seems to last less than an hour (Mayford, Abel, and Kandel, 1995) and so is unlikely to be responsible for the lifelong changes required

for long term memory. L-LTP however requires new protein synthesis and synaptic growth, and appears to last for years (Abel, Nguyen, Barad, Deuel, Kandel, and Bourtchouladze, 1997). This research provides a link between the microscopic changes occurring at a synaptic level and the behavioural phenomenon of memory. It is still not entirely clear whether L-LTP is solely responsible for lifelong memory, but it is likely that it has a significant role to play (Abraham, 2000).

The hippocampus is a focal point for the study of memory as it seems to be involved in most aspects of information storage, maintenance and retrieval. Damage to the hippocampus severely diminishes short term memory, and the ability to transform short term memories into long term memories. The hippocampus has been shown to be highly active when subjects are learning new material (Davachi, Mitchell, and Wagner, 2003). As well as serving an important role in memory the hippocampus has also been identified as important for navigation and learning the relationship between events and locations. This is most obvious in the studies of rat navigation in which rewards affect the firing patterns of cells in the hippocampus (Burgess, Reece, and O'Kefee, 1994).

The main feature that distinguishes human memory from computer memory is our ability to discover associations between events and to access information by analogy. This is extremely difficult with computers as they store information as discrete and isolated patterns of binary digits. Mammalian memories seem to be overlapping and they share resources with other memories forming interconnected representations. This overlapping of representation allows mammals to quickly access relationships between similar items and to respond to them in similar ways. When faced with a deadly carnivore it is a good idea to run away, irrespective of the exact number of teeth, the length of the legs or the position of the sun. To find similarities between events using a computer, it is necessary to explicitly work though all the combinations of features to detect the overlaps. This is an extremely complex task both algorithmically and computationally, requiring metrics to decide what is meant by similar and which of the millions of features of an event are relevant.

1.3 Content Addressable Memory

To enable computers to efficiently find associations between stored information and similar current events, the information needs to be stored in a way that allows comparison and association. Rather than being accessed purely by the location of the memory,

it needs to be recalled based on the content. The popularity of Google^{tm} is partly due to the natural way of accessing information. By providing text that is either on a web page or linked to the page, the Google^{tm} search returns pages that have content related to the search string. Google Desktop^{tm} performs an analysis of the computer's hard drive and stores content information in an index that can be searched quickly for relevant words and terms. These systems allow users to use textual context to access information, but they are extremely limited and require significant computational effort and additional memory. What is required is a memory that can be directly accessed by contents.

Content addressable memory provides the ability to access information from just a partial sample of the original content. For example, when trying to remember an event it is common to retrace your steps and present as much of the original context as possible. Mammals appear to have the same ability to remember behaviour based on contextual information. When developing a functional model of mammalian memory it is important to replicate this feature. Computer models of content addressable memory have been developing over many years (Hanlon, 1966; Hopfield, 1982; Kohonen, 1987; Pagiamtzis and Sheikholeslami, 2006). The model of content addressable memory that is the focus of this thesis is the Hopfield model. This model has a number of desirable features:

- A biologically plausible connection update rule, the Hebbian learning rule.
- A simple activation function than can be analysed and provide repeatable results.
- A large body of research for comparison of results.

The majority of this thesis will deal with variants of the Hopfield network (see Section 2.4) and analyse the performance of different learning methods applied to this model of content addressable memory.

The term content addressable memory is also applied to hardware which allows fast recovery of information based on partial content (Schultz and Gulak, 1996). These systems provide order of magnitude faster access to information than conventional digital memory. They are used in specialist areas such as telecommunications where speed is far more important that cost or energy efficiency. Hardware implementations are beyond the scope of this thesis. We will instead be focusing on software which simulates the properties of biological memory.

1.4 Flexible Representation

The brain is undergoing constant change. These changes are not just the transient electro-chemical signals that pulse through the brain, but changes to the shape and connectivity of neurons. There are also changes to the efficacy and speed of connections already present. Every neuron, and possibly every synapse, is making changes to its structure and behaviour. When the brain is damaged, it is often able to recover much of its original functionality. Computer memory on the other hand is fixed and inflexible. When errors occur the information in that area is usually completely lost, indeed even a single fault can cause the entire system to fail.

There are two reasons that we might want to reconcile these fundamentally different systems. The first is to help us more fully understand the nature of human memory, and the second is to improve the performance of the memory systems in computers, not to make them more accurate, but to make them more useful.

The task that we are particularly interested in is the mammalian ability to change behaviour in response to events that occur over an extended time frame. Mammalian brains are flexible enough to learn new responses to old situations and retain different behaviours for similar but distinct events. For example, when a rat is placed in an environment it can learn the location of food sources and is able to navigate to those locations weeks later, even after being exposed to many new environments.

The main problem posed by this type of learning is how to incorporate new information without destroying the memories that have already been stored. The animal could try to store everything that happens to it, forget those things that were not connected to survival, or just forget all the events and just remember simple relationships. Given that remembering everything is cognitively expensive, and that mammals seem to remember more than simple relationships, there must be a mechanism that allows smooth integration of new information without loosing too much old information.

Computer models of neural networks, artificial neural networks (ANNs), suffer from a peculiar problem when presented with new information that needs to be incorporated into an already functioning memory system. The old information disappears almost immediately when new information is learnt (McCloskey and Cohen, 1989). We will call this catastrophic loss of memory "catastrophic forgetting" (CF). Animals do not seem to suffer from this problem. What mechanisms are missing from our models of memory that prevent this loss in the face of new learning? Is it possible to create computer models that are as robust and flexible as the mammalian brain?

We suggest that mammals have solved the problem of catastrophic forgetting by consolidating newly learnt information in an off-line process during sleep. The role of sleep as part of memory consolidation has been widely explored in the literature with promising results (Pearlman, 1971; Pearlman and Becker, 1973, 1974; Pearlman, 1979; Fishbein and Gutwein, 1981; Smith and Lapp, 1991; Smith, 1993, 1995; Hennevin, Hars, Maho, and Bloch, 1995; Hennevin, Hars, and Maho, 1995; Smith, 1996; Fishbein, 1997; Meeter and Murre, 2004; Muzur, 2005; Stickgold, 2005). Even with many decades of research the process that protects memory is not fully understood. There is even debate about the necessity of sleep for memory protection (Siegel, 2001; Vertes, 2004; Siegel, 2005) and many questions have arisen regarding the nature of these memory mechanisms. For example, what is the nature of the processing of information, and what might be happening during sleep that allows for improvements in certain types of memory (Stickgold, Hobson, Fosse, and Fosse, 2001)?

The approach that we will take to investigate some of these questions, is to develop a memory system that shares some of the features of mammalian memory and then to experiment with several different possible mechanisms for preventing the catastrophic forgetting that is common in computational versions of these systems. The Hopfield network model that we will investigate has two of the key features that we are interested in. It is a content addressable memory and it suffers from catastrophic forgetting.

1.5 Thesis Structure

The background of the network used is presented in Chapter 2. Chapter 3 describes the catastrophic forgetting problem in multilayer perceptron networks and some of the proposed solutions. Chapter 4 moves the investigation of catastrophic forgetting into Hopfield networks and describes previous solutions to this class of problems as well as our pseudorehearsal procedure (Robins and McCallum, 1998, 1999). One of the features of pseudorehearsal is the selection of the patterns/memories to rehearse. Chapter 5 investigates ways to distinguish between real memories and spurious memories, and presents a new technique which involves the comparisons of the inputs to individual units (Robins and McCallum, 2004). This technique is then applied to both the pseudorehearsal process and the unlearning process (Hopfield, Feinstein, and Palmer, 1983; van Hemmen, 1997) in Chapter 6, to establish if performance can be improved. Finally Chapter 7 discusses future directions for this research, and the link to sleep and dreaming.

Chapter 2

Background

Artificial neural networks (ANNs) have proven to be extremely effective in several problem domains. Feed forward neural networks, in which activation is passed forward through the network, have been shown to be able to approximate any functional mapping from a set of real numbers to another set of real numbers with arbitrary precision (Hornik, Stinchcombe, and White, 1989; Stinchcombe, 1999).

Most ANN learning algorithms are based on *concurrent* learning, in which the entire training set is presented and trained as a single, complete entity. Once the network has reached a set level of performance, no further information is learnt by the network. The system moves from training into a performance phase.

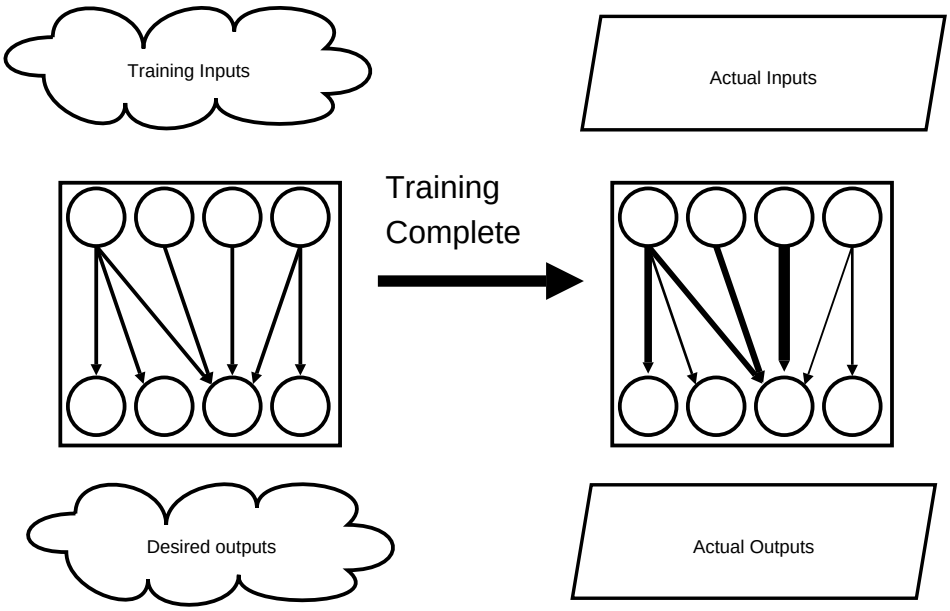


Figure 2.1: Training a traditional network.

In contrast, human learning is an ongoing, lifelong process. It is *sequential* in that we can learn new information at any time and integrate it into what we already know. Most ANN learning algorithms are not capable of sequential learning because of a fundamental underlying problem: when new information is learnt by a network it significantly disrupts or even eliminates information that has been previously learnt. This problem has been identified in many forms in memory literature – it is commonly called the "catastrophic forgetting", "catastrophic interference", or "serial learning" problem (Hetherington and Seidenberg, 1989; McCloskey and Cohen, 1989; Lewandowsky, 1991; French, 1992; McRae and Hetherington, 1993; Sharkey and Sharkey, 1995; Robins, 1995; French, 1999; Ans, Rousset, French, and Musca, 2002; Robins and McCallum, 2004).

2.1 Memory

Digital computers store data as discrete packets of information. These are stored in a medium designed to prevent one piece of information disrupting or interfering with the information currently stored in other parts of the medium. This type of memory has many desirable properties. The lack of interference means that new memories can be stored without fear of corruption, and there is an *a priori* limit to the amount of data this type of memory can store. This limit is defined by the number of individual memory areas available and the size of the memories to store. These systems are often very stable where a single bit of data, a 1 or a 0, will remain in the same state for years. The lack of interference means that the data stored (memory) remains stable as long as the medium remains stable. These properties are desirable for situations where each individual event contains distinct information that needs to be recalled *exactly*. An example of the type of system would be a banking transaction system.

There are however some situations where this sort of discrete memory does not perform well. The real world does not provide clear separation between events or categories of objects. The discrete storage used in computers does not support building associations between similar events or memories. For example "buying apples from the corner store" is as similar to "buying oranges from the corner store" as to "hunting wildebeest in Mozambique". Indeed, it would be difficult to develop the idea of prototypes or to generalize learning without the ability to see the similarities present in the real world (Medin and Smith, 1984).

It is precisely the fact that having seen a lion kill a family member, we run from

any large furry animal, that helped us survive. If we stored information so separately that there was no way of telling when events were similar we would soon fall victim to similar but distinct predators. Humans instinctively identify similarities and this can make it difficult to understand why computers can not make these "obvious" connections between very similar events. Ideally computer systems would have two types of memory – a precise system for situations like banking transactions, and an overlapping similarity system for dealing with situations in the real world in a way that humans can understand. There are a number of problems that need to be overcome in order to create a memory system that is functionally similar to animals. These include: catastrophic forgetting, where old information is forgotten suddenly; information overload, where the system cannot store new items and memory breaks down; and spurious memory, where events that never happened are "recalled".

The input to the sensory system of most animals contains far too much information to be able to store each individual sensory event as a separate memory. The amount of storage required to deal with just a few hours of experience would entirely overwhelm most uncompressed discrete storage media. Consider the human visual system, with approximately one million optic nerve cells leaving the retina (Jonas, Schmidt, Muller-Bergh, Schldrzer-Schrehardr, and Naumann, 1992), each of which has a maximum firing rate of about 100 Hz (Kim and Dudek, 1993). This gives a maximum data rate of about 100,000,000 bits per second (100Mbit or about 12MB per second). In one minute this would be 750MB, in one hour 45GB and for a 16 hour waking day 720GB i.e., in the order of 10^{12} bytes.

Calculating the storage capacity of the human brain is very difficult. There is no clear distinction between stored information and the processing of that information, or indeed what exactly it means to store information. We can make claims about the maximum information complexity of an animal's brain based on the number of individual components and their connectivity. This upper bound in complexity is likely to be much higher than the actual capacity as it ignores redundancy, relay and amplification, and other processing that would not increase the total amount of information stored. Even with these caveats the capacity of the human brain is not large compared to the sensory input. The human brain is estimated to contain 10^{11} neurons each with an average of about 10^3 connections. Thus the maximum amount of information that could be stored is in the order of 10^{14} bytes given one byte per synapse. If our *entire* brain was dedicated to storing every bit of data coming from our eyes we would be able to store about 100 days of vision. Given that there are millions of neurons for

audition, olfaction, and somatosensory systems, this estimate is very conservative and we would probably only be able to store a full sensory record for a few weeks.

It is obvious from these figures that much of the information coming into the brain must be either discarded, filtered or compressed, or indeed all three of these in combination. The most obvious compression available is related to time. The fact that the world remains similar from one instant to the next can be used to reduce the amount of information that needs to be stored. Rather than storing each micro-second as a separate event, systems that can identify objects in the environment and store view dependant models (Riesenhuber and Poggio, 2002) of the objects, only need to store a small amount of information about where the object is and if it has changed in some way. The consistency of gaze and flow of information into the nervous system naturally lends itself to this sort of compression.

It is extremely useful to be able to identify an object from many angles and have some form of content addressable memory to access information about objects that have similar features. Accessing information about objects and behaving in an environmentally appropriate way can be aided by both prototypical objects and object classes. Prototypes emphasise the similarities between a set of objects in a class, where object classes provide boundaries between types of objects.

To take advantage of these similarities, similar memories could be stored so that the information that is similar is only stored once in a shared area of the medium. Learning to identify objects in a view independent way relies on object persistence so that different views can be bound to the same object. Without some degree of view independent memory for objects it would be difficult to form any consistent object based behaviours.

One of the earliest identified learning procedures for memory is reinforcement learning, as described by Pavlov (1927). His studies of classical conditioning required animals to be able to identify an association between the ringing of a bell and the presentation of food. Given that the bell was manually rung the generated sounds would be similar but not identical. Thus the animals were identifying similar rather than identical sounds and responding in an appropriate way.

Both classical conditioning and operant conditioning demonstrate animals' ability to associate rewards with objects or events. Even bees can learn to associate fragrance with food reward (Mercer, 2001). The evolutionary advantage of associating behaviour with sensory stimuli is reflected in the memory systems that have evolved in animals.

2.2 Single Neuron Models

To understand the nature of the mammalian memory system it is essential to look in to its underlying structure. Just as the performance of computer memory is related to the structure and nature of the single binary unit storage, so real neural networks have some features which are directly related to the nature of the base units, the individual neurons.

We can analyse the behaviour of a biological neural network at many levels of abstraction. The most abstract is at the functional level of *input* $\rightarrow$ *processing* $\rightarrow$ *output*. This abstraction can be applied, in a reductionist way, to any part of the network from the whole network down to the propagation of signals down the axon of the neuron. In the construction of artificial models of neural networks the building blocks (units) of the network may represent synapses, individual neurons, a number of neurons or a whole cerebral lobe. When discussing behaviour at this level of abstraction we will refer to the components as units rather than neurons. For the rest of this thesis 'units' will refer to components with functional behaviour similar to that of individual neurons rather than neuron clusters or entire lobes.

There are a range of unit models at the level of individual neurons, the simplest being proposed by McCulloch and Pitts (1943) which describes units whose function is that of traditional logical circuits. Units have only two binary inputs and a single output. The output depends on an activation function applied to the two inputs. The function tests the summed input against a threshold and gives a true/false, 1/0 response.

$$\psi = (A + B) > T \tag{2.1}$$

This can be visualised as a unit with connections as shown in Figure 2.2.

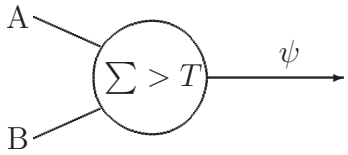


Figure 2.2: McCulloch and Pitts unit.

A much more complex model for neuronal activation was proposed by Hodgkin and Huxley (1952) describing the activation of a squid giant axon. This model is widely used as it is regarded as an accurate model of the propagation of the action potential of a biological neuron. The equation for the activation and propagation of signals is

relatively complex (see Equation (2.2)).

$$I = C_m \cdot \Delta V_m / \Delta t + I_{Na} + I_K + I_L \tag{2.2}$$

where

$$\begin{aligned} I_{Na} &= G_{Na} \cdot m^3 \cdot h \cdot (V_m - E_{Na}) \\ I_K &= G_K \cdot n^4 \cdot (V_m - E_K) \\ I_L &= G_L \cdot (V_m - E_L) \end{aligned}$$

where I is the total ionic current across the membrane of the neuron, C_m is the probability of there being enough activation to open the channel, I_{Na}, I_K, I_L are the individual currents for sodium, potassium, and the linear leakage from the cell, V_m is the total membrane potential, G_i is the maximum conductance for a channel i, and E_i is equilibrium voltage for each channel of sodium, potassium, and linear leakage, m^3 and n^4 fit the non-linear response of the channels to a change in voltage.

Recently this model has been challenged, as it does not exactly match the recorded data from the membrane potential from neurons in the mammalian cortex (Naundorf, Wolf, and Volgushev, 2006). A new model for cortical neuron activation is presented by Naundorf *et al.* (2006) which includes sodium channels with a positive feedback mechanism which is a better fit to the recording for the early depolarisation of the neuronal membrane. This faster depolarisation would allow networks to respond with a wider range of propagation speeds and response times compared to networks built with Hodgkin-Huxley type units. It is unknown, at this stage, what effect this more accurate model will have on network level behaviour. This demonstrates one of the problems encountered when building models that rely heavily on a particularly complex model. When the minute details of the model change, all of the results have to be examined to determine if they are still valid.

The complexity of these unit models makes the analysis of the function of the network much more difficult. As the units' complexity starts to match that of real neurons it becomes very difficult to differentiate what is functionally significant and what is idiosyncratic for the particular model. In this thesis we will use a relatively simplistic unit model based on the *perceptron* presented by Rosenblatt (1958). Networks which use many of these perceptron units in layers are designated as *multi-layer perceptrons* (MLP) networks. In the following sections we will refer to the components of an artificial neural network as units rather than perceptrons. The following sections describe the features of these units.

2.2.1 Unit Input

Each unit receives input which is the sum of the weights connecting that unit with other units.

$$\upsilon_j = \sum_{i=1}^{N} (\psi_i w_{ij}) \tag{2.3}$$

where ψ_i is the activation of the unit i, w_{ij} is the weight of the connection from unit j to unit i, and N is the total number of units connected to unit j.

This sum of weighted inputs ignores the interesting calculations that occur at the synaptic level, for example shunting inhibition (Blomfield, 1974; London and Hausser, 2005; Gasparini and Magee, 2006). However, these interactions can be approximated by inserting more non-linear units to fit more complex calculations.

2.2.2 Unit Activation

To determine the output, or activation, of a unit, the input is passed through an activation function $f()$. There are many ways of applying activation functions. Two categories are linear and non-linear activation functions. It has been shown that the linear activation functions do not provide enough complexity to solve many real world problems such as the exclusive–or (XOR) function (Minsky and Papert, 1969). Thus we will focus on the non-linear activation functions.

In most real neurons the axon hillock mediates the production of an action potential. The neuron either activates and sends an action potential, or remains quiet. This means that the neuron has a non-linear response to input. There are neurons that have a linear response to a stimulus, for example, the rods and cones in the retina produce a response that is, for the majority of activation levels, linear with respect to the number of photons stimulating the cells (Marr and Poggio, 1979). The idea of a threshold as the non-linearity is used in many artificial models of neuronal activation. The simplest model using a threshold T is the "greater than" logical test as described in Equation (2.1).

For the simple threshold model presented Equation (2.1), the activation calculation can be rewritten to refer to the time (t) at which the activation occurs (t-1 is the activation in the previous time step). The equation is:

$$\psi_i^t = f\big(\upsilon_i^{t-1} - T_i\big) \tag{2.4}$$

This can be seen diagrammatically in Figure 2.3.

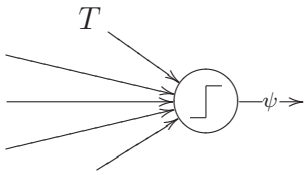


Figure 2.3: Diagrammatic view of a unit receiving input from four inputs and a bias.

The original perceptron model (Rosenblatt, 1958) used the sign function as the activation function:

$$\psi_i = sg(v_i - T_i) \tag{2.5}$$

The most common non-linear activation function used in neural networks is a sigmoid function. The sigmoid function provides a smooth monotonically increasing activation value. Part of the popularity of the sigmoid function is that it is differentiable, giving the ability to estimate how the activation function will change given small changes to the input of the unit. Whereas the threshold function changes its activation suddenly, the sigmoid changes smoothly and continuously. Small changes to the weights of the network make small changes to the output of the sigmoid units creating a smooth output surface. This surface is used by back-propagation and other learning algorithms to systematically approach a set of weights that solve the given problem. The surface can be visualised as being created by the small changes in the activation value of the output unit as the weights are changed. Threshold functions have plateaus where the output does not change for a large number of different weights, whereas the smooth sigmoidal function provides a gradient at all points on the surface. The error of the network can also be visualised as a surface in the space of all possible weights, where the surface is the sum of the error of the network over the given learning tasks. Given an error surface, a solution to a problem is where the error is zero, or if it does not reach zero, the minimum error value on the surface. Given the smooth surface, gradient descent will be able to find at least a local minima, if not the global minimum.

It is possible to simulate the smooth error surface created by a sigmoidal activation function by applying noise[1] to the input of a threshold function. This creates a prob-

[1]Noise and the random numbers used to generate the noise are too important to leave to chance. Therefore we have implemented the algorithm presented by Marsaglia, Zaman, and Tsang (1990) so that our results are suitably random and yet repeatable on various machines by initialization with a

abilistic activation for the unit. Units whose input is close to their threshold will be active with a probability of 50%.

If the noise applied to the unit inputs is generated from a Gaussian distribution then the probabilistic activation of the unit, which is the cumulative distribution function Φ, becomes similar to the sigmoid activation function. This can be seen in Figure 2.4 where:

$$g(x) = \frac{1}{1 + e^{-x}}$$
$$\Phi(x) = \frac{1}{2}\left(1 + \texttt{erf}\left(\frac{x}{\sigma\sqrt{2}}\right)\right)$$

This form of smoothing using noise is applied when performing delta learning (see Section 2.3.2), and can be applied during testing. If this form of stochastic activation is used the network becomes much more like a Boltzmann machine (Ackley, Hinton, and Sejnowski, 1985). For simplicity of analysis and comparison with other research, we have not applied noise during relaxation in this thesis.

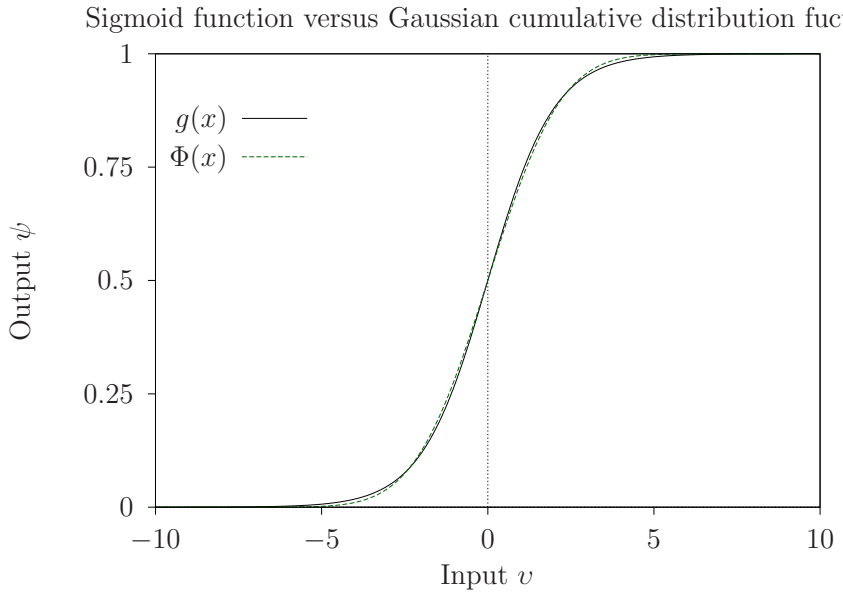


Figure 2.4: Comparison of the sigmoid function with the cumulative distribution function for a Gaussian with $\sigma^2 = 3.0$.

The threshold of each unit defines the position of the *decision surface*. The decision surface lies between the input values that activate or deactivate the unit. Each unit can

known seed. The seed used for every data run is included in the output, enabling replication if required. If the random seed is omitted from the parameter file the current time is used (see Appendix A).

be used to define an axis, with the value on the axis being the input to the unit. The activation function can be used to define a hyper-surface in this hyper-cube. Changing the weights changes the input values and therefore the position of the decision surface. We define that a unit is "close" to the decision surface if small changes to the weights change its activation state. Noise on the input to a unit that is close to the decision surface may result in the unit changing from active to inactive or *vice versa*, while noise added to a unit that has input far from the decision surface will only cause change infrequently.

2.2.3 Updating Connections

The strength of the input to a unit i is determined by the strength of the connections and the number of connected units that are active. Changing the strength of the connections alters the input to the unit, which in turn, may cause the output to change. The function to alter the connections is called the *learning algorithm*.

A connection's strength can be represented as a Boolean value, integer weights, or more commonly as floating point numbers. Both integer and Boolean representation limit the amount of information that can be stored in connections. This limitation can have significant influence on the performance of some learning algorithms. Processes that require very accurate values or gradient descent are often less effective and in some cases will not work at all with limited information per connection.

The connections are usually initialised as either 0 or a small randomised value either side of 0. The learning algorithm makes small positive and negative alterations to these connections. If unit j is active and the connection between unit j and unit i (denoted w_{ji}) increases then the input to unit i will increase by the activation value of unit j times the increase in the connection weight w_{ji}.

In real neural networks there are many mechanisms involved in the alteration of connections between neurons. *Long Term Potentiation* (LTP) describes the increase in connection strength between neurons which is demonstrated by an increase in the likelihood that the post-synaptic neuron will become active when the pre-synaptic neuron sends an action potential. There is still debate about the mechanisms that underpin LTP (Malenka and Nicoll, 1999; Palmer, Isaac, and Collingridge, 2004), but for our purpose it is sufficient that there is a mechanism that changes the connection strength between neurons.

2.3 Learning Algorithms

Learning algorithms can be broken down into three categories: supervised, reinforcement, and unsupervised. The distinction can be made by the amount of information required for each system. Supervised learning requires an external agent or "teacher" to provide the desired output. The output units in the network compare themselves to the desired outputs and estimate the direction, and possibly the distance, between the current state and the desired state. The sum of the distances between the current output and the desired output is often used as the error of the network. The learning algorithm uses this direction and distance to alter the connections to the output units, and other units, in order to decrease the error.

Reinforcement learning only requires a single signal giving a measure of "goodness". A unit will update its connections to other units based on the principle of increasing the likelihood of the events that lead to the reinforcement. This is a somewhat blind mechanism as it has no information about which direction will improve performance. Without a gradient on the error surface, reinforcement learning just amplifies activities that were associated with a "good" signal and depresses associations that resulted in negative feedback.

The last category, unsupervised algorithms, have no notion of the desired or correct answer. The network changes its connections based on patterns in the input data. The only goal that could be attributed to unsupervised learning is the desire to find similarity or correlation between events in the environment. The fact that networks are able to perform any interesting alterations to their behaviour without a predefined goal is remarkable. Unsupervised learning can find structure in input data or reproduce either single events, or sequences.

2.3.1 Hebbian Learning

Hebbian learning is a form of learning that was suggested by Hebb (1949). The key feature of Hebbian learning is that associations between neurons will be enhanced after each learning cycle. When one unit contributes to the activation of another unit, that connection will be strengthened so that there is an increase in the potential for unit A to activate unit B (Hebb, 1949). The biological basis for this learning algorithm was discovered much later in the form of LTP. Hebbian learning is now used to refer to most forms of learning which increase the correlation between the activation of units in a network. This is most commonly seen in an unsupervised context, as it does not

require any goal or direction to focus learning.

When we use this principle in an artificial neural network, it provides a rule for the alteration of the connections between units. The connection strength increases when the units at each end are simultaneously active. Active units are represented by a 1 and inactive by 0. The learning rule is

$$\Delta w_{ij} = \eta \psi_i \psi_j \tag{2.6}$$

where w_{ij} is the strength of the connection from unit j to unit i, η is the learning constant, and ψ_i is the activation of a unit i in the network.

		ψ_i	
		1	0
ψ_j	1	+	0
	0	0	0

Table 2.1: Weight changes using the Hebb rule, with +1/0 activations.

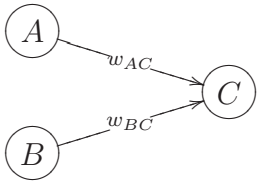


Figure 2.5: Diagram of a simple 3 unit network with one output unit, two input units and two weighted connections.

This can be seen diagrammatically in Figure 2.5. If unit A activates at the same time as unit C and thus contributes to its activation then the connection w_{AC} will increase in strength so that the next time unit A is active, C is more likely to be active.

The extension from associating units that are firing to auto-associative content addressable memory is relatively straightforward. An auto-associative task is where the network has to produce an output which is identical to the input. Auto-associative learning is important for content addressable memory as it can provide a way to learn patterns which can later be found with a partial stimulus. The Hebbian rule suits the task of auto-association as individual units are adjusted to become associated, and this adjustment allows groups of units to become associated. These groups of

associated units form the content of the remembered event. This process is described in Section 2.4.

There is at least one major problem with the Hebb postulate as it has been used in artificial neural networks. The learning function provides no mechanism for decreasing the connection strength between units. The function only describes a mechanism for strengthening the connection. Without some form of decay or inhibition, the network would quickly saturate with activation as all the connections increase in strength. One possible solution to this problem is to leave the association rule unchanged, but adjust the activation values of the units. If the sign function $sg()$ is used as the activation of the units then negative values are introduced to the system. When a negative output is associated with a positive input the resulting change will be negative and thus move the connection strength toward the negative values.

Using the Hebbian learning rule from Equation (2.6) we now have an algorithm that can make both positive and negative changes to the connections in the network. Table 2.2 shows the result of applying Equation (2.6) to activations that are either -1 or $+1$.

		ψ_i	
		1	-1
ψ_j	1	+	−
	-1	−	+

Table 2.2: Weight changes with +1/-1 activations.

It is possible to alter the learning on networks with 0 and +1 as the activation values so that the association rule generates the same weights as the $-1, +1$ space: $\Delta w_{ij} = (2\psi_i - 1)(2\psi_j - 1)$.

There are other solutions to the problem of a lack of inhibition. Simple solutions include having a universal decay term applied once every time step, or to normalise the input to units so that when one connection increases its strength other connections are decreased (Chechik, Meilijson, and Ruppin, 2001), or having separate LTP and LTD learning rules such as the CPCA (Norman and O'Reilly, 2003). Investigating the influence of each of these changes to the learning algorithm is beyond the scope of this thesis, as we are focusing on catastrophic forgetting rather than possible inhibition mechanisms. For simplicity most of the following networks use -1/+1 activation, and when using 1/0 networks we use the $\Delta w_{ij} = (2\psi_i - 1)(2\psi_j - 1)$ update rule to generate

the weight matrix.

2.3.2 Delta Learning

The delta learning algorithm[2] uses the simple approach of altering the connections to a unit so that the output of the unit is more like the desired output. Because of the requirement for a "correct" desired output, delta learning is usually used as a supervised learning algorithm. In our case we will be using delta learning for an auto-associative task, and so the desired output is in fact the input signal. This allows us to use delta learning, which is usually only considered for supervised learning tasks, while retaining the advantages of unsupervised learning and having all the information locally available.

The simplest form of delta learning is to subtract the actual output from the desired output and use the result as an error term. This error term is used to alter the connections to the output unit. Rosenblatt (1958) used a delta learning rule to train the units which he called perceptrons. The formulation is:

$$\Delta w_{ij} = \eta(D_i - \psi_i)\psi_j \tag{2.7}$$

where η is a learning constant, D_i is the desired output and ψ_i is the activation of unit i.

Using this learning algorithm, if the desired D_i output is 1 and the current activation of the unit ψ_i is -1, then there will be a $+\eta$ addition to all the connections from units that had $+1$ activations. Because of the symmetry in the network, this occurs both from unit j to i and also from unit i to j. The total change for the +1,-1 space is therefore 2η. Given the same pattern of activation, the sum of the weighted input to the unit will be increased by:

$$\Delta \upsilon_i = 2\eta \sum_{j=1}^{m} \psi_j \tag{2.8}$$

where m is the number of units connected to unit i.

This learning algorithm has been shown to always converge in a finite time if the desired output is linearly separable (Novikoff, 1962). This means that if a set of weights exist that can solve the linearly separable problem this simple delta learning rule will find a solution. This result makes delta learning a very attractive algorithm for many

[2]Delta learning is sometimes referred to as the "Widrow–Hoff rule" as these authors first introduced this style of learning (Widrow and Hoff, 1960).

A	B	AND	ψ	e
1	1	1	0	1
1	0	0	0	0
0	1	0	0	0
0	0	0	0	0

Table 2.3: The training patterns, inputs, and outputs for an untrained network on the AND problem. ψ is the output of the network and e is the error.

problems. An example of a linearly separable problem is show in Table 2.3, and the diagram of the network in Figure 2.6

However, there are limitations to delta learning, as hinted at by the condition of linear separability. For problems such as XOR, where the outputs of the network cannot be constructed from a linear combination of the inputs, there is no set of weights directly connecting input to the output that can solve the problem. This limits the solution space of the single layer delta learning system. Multilayer systems can solve arbitrary functional mappings, with back-propagation (Rumelhart, Hinton, and Williams, 1986) being the first algorithm that could learn these mappings.

One of the problems with delta learning on threshold functions is that it only makes changes to the weights when there is an error in the output of the unit. Given an activation function such as the sign or threshold function, the only error signal values are $1, 0, -1$ for threshold and $2, 0, -2$ for sign. As the network is using this error to adjust the weights it will stop adjusting them as soon as the outputs are just on the correct side of the decision surface. Figure 2.7 shows a possible decision surface that solves the *AND* problem. This solution is not particularly stable as the surface is close to the two points at $< 1, 1 >$ and $< 0, 1 >$. An alteration to the threshold T could easily move the decision surface so that one of the training points would generate an error. This arises because the learning stops as soon as the decision surface provides the desired output where all four training patterns give the desired output.

This fragility can be alleviated by adding noise to the calculations in the network. The two types of noise that we use are noise on the summed input to a unit, "input noise", and noise on the propagated activation of the units in a learnt pattern which in effect changes the autoassociative learning task into a slightly heteroassociative task, "heteroassociative noise".

Input noise is the application of noise to the input of a unit so that it generates an

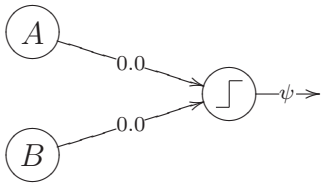


Figure 2.6: Diagram of the untrained, two input, one output network from Table 2.3.

A solution to the *AND* problem.

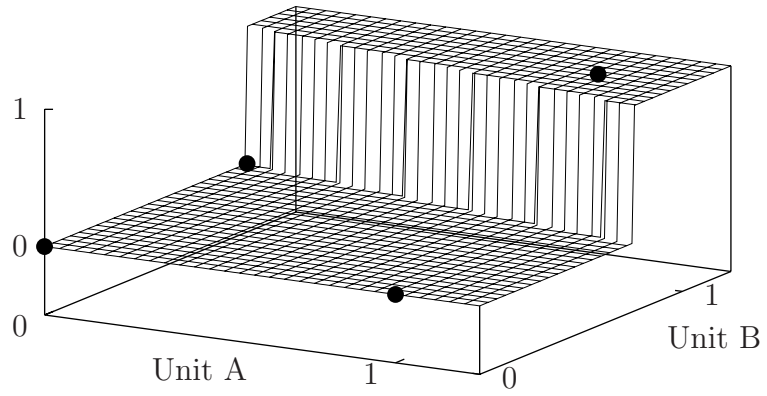


Figure 2.7: Activation surface for the *AND* function learnt by a perceptron network. The decision surface appears as a cliff in the activation surface. The values for the weights are $wA = 0.2,\ wB = 1$ and $T = -1.1$.

error when close to the decision surface. These additional errors move the least stable units further away from the decision surface. This noise can be either absolute or relative. Absolute noise uses a random number generated from a Gaussian distribution with a mean of zero and a standard deviation determined by the learning parameter ν. For the examples in Appendix B the standard deviation $\sigma = 0.5$ (GausAbsoluteRange = 0.5). Absolute input noise can be seen as defining a gradient from the decision surface which units move away from. The larger the σ the more often units will have their input disrupted and therefore generate errors resulting in learning. Relative noise is where the width of the Gaussian (σ) is scaled to match the current mean input to the units. This forces noise to continually be applied during the whole learning process. As the mean input to the units increases so does the amount of noise. This forces the network to continue to learn until the maximum number of epochs are reached as there is no way to converge on an error free network, as the noise scales to ensure there are always errors to correct. Relative noise also makes the inputs to the units more uniform in magnitude. Units whose summed input is lower than the mean are more likely to receive learning and thus increase their summed input. This can be visualised as a force field extending from the decision surface which continually pushes the decision surface away from the defined training points.

The result of this noise can be visualised in the simple network above trying to learn the "AND" function. Input noise visually creates a Gaussian blur around the training points forcing the decision surface to move away. Figure 2.8 shows the result of training with the application of absolute Gaussian noise with a mean of 0 and a deviation $\sigma^2 = 2$ on the input to the output unit of the network learning the "AND" function. The decision surface has moved further away from the training points. The weights of the network have increased to compensate for the noise, and symmetry starts to emerge as the network tries to optimise the distance from the training points.

Heteroassociative noise (ν_h) is where a percentage of the input units are flipped before the error is generated. This alters the summed input to the output units, and therefore the actual outputs used to generate errors. Units which were only just on the correct side of the decision surface may now generate an error and have their weights changed. This is called heteroassociative noise as for a Hopfield network it changes the autoassociative task into a slightly heteroassociative task, as the inputs which have been altered by noise, are no longer identical to the desired output. This input to output mapping is aimed at forming a basin of attraction around a learnt pattern, by forcing patterns that are close to the learnt pattern to relax to the learnt pattern. The

Optimised solution to the *AND* problem.

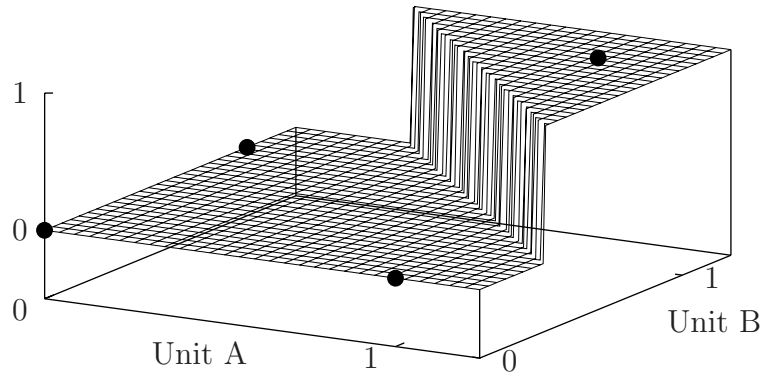


Figure 2.8: Activation surface for the *AND* function learnt by a perceptron network with Gaussian noise. The decision surface appears as a cliff in the activation surface. The values for the weights are $wA = 3.19,\ wB = 3.17$ and $T = -4.97$.

noise level is usually relatively low $\nu_h < 10\%$.

In Hopfield networks delta learning was proposed as a local learning rule by Diederich and Opper (1987). This performs the standard error correction on every unit, using the auto-associative approach in which, during training, the input is the desired output. The procedure is:

- while the total error of the population is above a threshold repeat
  - for each pattern in the base population
    - initialise the network with the pattern to learn
    - if using heteroassociative noise, alter the activation of ν_h of the units.
    - if using input noise, add Gaussian noise ν_i to the input of all output units
    - calculate the outputs for all units
    - generate the error based on the difference between these outputs and the original pattern
    - update the weights based on the error above

This process may take many iterations to converge on weights that make all of the training items, called the base population, stable. The level and the type of noise also contribute to the length of time required to converge on a solution. Noise is important for making the learnt items more stable and creating a basin of attraction around each pattern. The noise makes the weights both larger and more symmetric which combine to make the learnt patterns more robust.

2.3.3 Local Learning

One of the important features of a biologically plausible learning processes is that it has to be *local*. For a learning algorithm to be local it must make changes to the connections between units based *only* on information that relates to the connection or the two units which it connects. Local calculations do not require values acquired from the whole learning population or from the entire network. Hebbian learning is the prototypical local learning algorithm. The pseudo inverse (Hertz, Krough, and Palmer, 1991; Storkey, 1997) learning rule is an example of a non-local learning procedure.

Delta learning is a local learning algorithm as all the information to make an update is available locally, so long as there is an input which represents the desired output. If the goal of the network is auto-associative then the input can act as the desired output. This allows us to explore the use of Delta learning in a local learning context.

2.4 Hopfield Networks

The model of associative memory that will form the basis of this thesis is the Hopfield Network. The essence of this model was first presented by Pastur and Figotin (1977) as a model of magnetic spin glass systems in physics. The model became popular as a model of memory after Hopfield (1982) proposed it as a form of content addressable memory. Hopfield networks have remained relevant to both the artificial intelligence and the physics communities.

2.4.1 Hopfield Network Structure

The network is constructed as a set of N simple units, with each unit having an associated activation value. The set of units can be represented as a vector of activations $\mathbf{X}$, which describe the current state of the network.

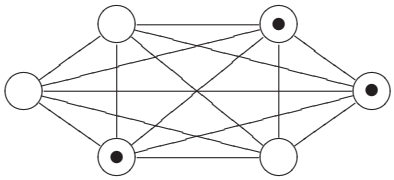


Figure 2.9: Hopfield network of 6 units $\mathcal{H}_{6,0}$.

Figure 2.9 shows a network of six fully connected units with zero as the low value for the activation function, denoted $\mathcal{H}_{6,0}$ ($\mathcal{H}_{6,\pm}$ represents activation of $< 1, -1 >$). To represent the state of the network we can construct a vector from the top left unit and follow around in a clockwise direction. The vector $\mathbf{X}$ represented by the network above would be $< 0, 1, 1, 0, 1, 0 >$. The state of the network can be represented as any one of the 2^6 possible patterns using this vector based approach.

We can represent the connections in the network as a weight matrix $\mathbf{W}$. Each unit's connections are represented by a row in the matrix. The central diagonal is left blank as we are currently prohibiting self connections. The weight matrix is either symmetric, as in Figure 2.9, or asymmetric. For the weight matrix to be symmetric the connection from a unit i to unit j must be the same as the connection from unit j to i ($w_{ij} = w_{ji}$) for all units. Hebbian learning makes symmetric updates to the weights in the network and thus creates a symmetric weight matrix. Symmetry in the weight matrix has the advantage of guaranteeing relaxation to a stable state and will be discussed in the next sections.

In addition to the units in the normal Hopfield network it is possible to add a bias unit, a unit which is always active. For delta learning the connection to the bias unit can be used to store the threshold for a unit. For Hebbian learning the bias unit is usually removed from the network. In this thesis all networks will have a bias unit included unless otherwise stated. Results from the networks have been repeated with bias units removed, and for large networks the results are very similar in either condition. The justification for this is that when the number of units in the network increases, the contribution of a single bias unit becomes less significant.

The units in the network in Figure 2.9 have an activation value of 1 and an inactive value of 0. As mentioned earlier there is also the possibility of having the inactive value of a unit represented by -1, denoted $\mathcal{H}_{6,\pm}$. The different activation values alter some of the characteristics of the network, particularly the dynamics (Horn, Levy, and Ruppin, 1998b; Davey, Adams, and Hunt, 2000), but many other characteristics are the same,

including the total number of possible states, learning algorithms, and connectivity. As the two types of inactive value are not isomorphic, the activation values used will be included as a parameter influencing the results.

2.4.2 Dynamics

The Hopfield network cycles through many configurations before settling on a stable output pattern. When a pattern of inputs is presented to the network as a vector $\mathbf{X}$, the activations of all the units are set to match those values. In the relaxation phase the network is then allowed to calculate the actual output of each unit. This is done by selecting a unit and performing the sum of all the incoming connections, weighted by connection strength, and passing this value through the activation function. As each unit is selected it either changes its activation or remains in the current state. After many cycles where an individual unit may be updated many times the network settles into a stable state where no matter which unit is selected it will remain in the same activation state.

This ability to relax to a stable state is what we describe as the process of recalling a memory. When presented with a vector of input values the network will relax into some state. The relaxation of a Hopfield network that has been trained with Hebbian learning, and therefore has a symmetric weight matrix, can be shown to follow the Lyapunov, or energy, function (Hopfield, 1982; Kühn, Bös, and van Hemmen, 1991):

$$H = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\psi_i\psi_j \tag{2.9}$$

where the sum H is monotonically decreasing after each update cycle. The Lyapunov function guarantees convergence to a stable low energy state in which no unit will change activation as any single unit change would result in a higher energy state. For an introduction to these forms of calculations see Hertz *et al.* (1991). The decrease in energy is caused by the activation function that either leaves the unit the same and thus the energy the same, or changes the activation to match the sum of the inputs, decreasing the energy of the network.

2.4.3 Energy Surface

Instead of focussing on the energy of the entire network, it is possible to look at the energy of individual units. Negative energy values are low energy and thus stable,

whereas positive values are high energy and unstable. The greater the magnitude of the energy of a unit (h) the further the unit is from the decision surface. For the network to be in a stable state, every unit must have a negative energy. For the network in Figure 2.9, if the weights are set to zero then the energy of each unit is zero in every state and the total energy of each state is also zero.

The *energy* function in (2.9) can be written as the sum of the energies of each unit:

$$H^p = \sum_{i=1}^{N} h_i^p \tag{2.10}$$

where h_i^p is the energy of unit i in state p, which in turn is defined as:

$$h_i^p = -(\upsilon_i^p \psi_i^p) \qquad \text{for } \mathcal{H}_{N,\pm} \tag{2.11}$$

$$h_i^p = -\frac{1}{2}(\upsilon_i^p((2\psi_i^p) - 1)) \qquad \text{for } \mathcal{H}_{N,0} \tag{2.12}$$

for the two different types of unit activation +1/-1 ($\mathcal{H}_{N,\pm}$) and +1/0 ($\mathcal{H}_{N,0}$).

These equations give negative values for units where the input (υ) is the same sign as the unit activation (ψ), and positive values when there is a difference between the current input and the current output.

If the pattern displayed in Figure 2.9 is converted to $< -1, +1, +1, -1, +1, -1 >$ for the -1/+1 space, and is learnt using Hebbian learning from the zero network[3], every unit will change its weighted connections. The energy for each unit after learning, where the network is in the learnt state p is:

$$\begin{aligned} \upsilon_i^p &= \sum_{j=1}^{N} (\psi_j^p w_{ji}) \\ h_i^p &= -(\upsilon_i^p \psi_i^p) \\ H^p &= \sum_{i=1}^{N} (h_i^p) \qquad \text{from (2.10)} \end{aligned}$$

Having learnt only the original pattern $p =< -1, +1, +1, -1, +1, -1 >$ every unit has the same energy[4] $h_i^p = -5$, and the network's energy is $H^p = (-5 \times N) = -30$. If unit 1 has its value altered from -1 to $+1$ then the energies of each unit in the new state p' would be $< +5, -3, -3, -3, -3, -3 >$ giving a total network energy of

[3]The zero network is network where all the connections are set to zero

[4]Self connections are zero and all other connections are either 1 or -1

$H^{p'} = -10$. This pattern is unstable as one of the units has an energy value greater than 0.

When a network has only a single pattern learnt there is a great deal of symmetry, as can be seen by the symmetry in the energy of the units. From the example above, all patterns that have only one unit flipped have energy $H^{p'} = -10$. With two units flipped, the energies of the units in these states become $< +3, +3, -1, -1, -1, -1 >$, $H^{p''} = 2$. With three units flipped all the units have energy $h_i^{p'''} = 1$, $H^{p'''} = 6$. This symmetry in unit energy is partly caused by the symmetry of the learning algorithm and partly by the symmetry of the activation values.

The symmetry between -1 and $+1$ activation values means that if all of the units are flipped the energy of the network is the same as the original pattern. What this means for sampling the space of all possible 6 unit patterns is that 50% of randomly created patterns will relax to the original pattern $< -1, +1, +1, -1, +1, -1 >$ and the other 50% to $< +1, -1, -1, +1, -1, +1 >$. Because of this symmetry we will treat relaxation to the inverse of a pattern as equivalent to relaxation to the original learnt pattern (Hopfield, 1982).

One method for breaking the symmetry in these networks is to add a bias unit. The bias unit breaks symmetry as it is always in the active (+1) state. This is only effective in small networks as the change to the basins of attraction for real and inverse patterns are $N/2 + 1$ and $N/2 - 1$ respectively. When N is large this change becomes insignificant.

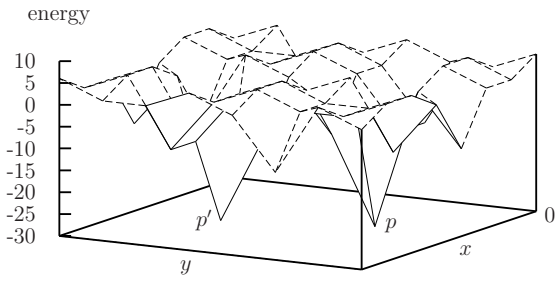


Figure 2.10: Demonstrating the symmetry of the basin of attraction of pattern p and its inverse p', using a 2D representation of the energy surface of a 6 unit network $\mathcal{H}^p_{6,\pm}$ with pattern $p =< -1, +1, +1, -1, +1, -1 >$ learnt.

There are 2^N states in a binary valued Hopfield network, each with an energy value. Learning alters the weight matrix which in turn alters the energy value of each of these states. These states are an example of an N dimensional hypercube, where each state is a vertex of the cube. Changing the activation value of a unit moves the state along that axis to an adjacent vertex of the cube. Relaxation of the Hopfield network is a path through the hypercube where each change moves to an adjacent vertex with lower energy. Thus the energy calculation defines a hyper-surface in the cube. Figure 2.10 shows a three dimensional mapping[5] of the surface generated in the network above having learnt the pattern $< -1, +1, +1, -1, +1, -1 >$. The two low energy areas in the graph correspond to the learnt pattern and its inverse.

There are a large number of learning algorithms that can form these energy surfaces. Some of them generate a surface that guarantees relaxation to a stable state while others may never converge. The variants include Hebbian learning, delta learning, anti–Hebbian learning, pseudo–inverse, and many specialised variants (Amit and Brunel, 1995; Storkey, 1997; Athithan, 2002). The energy surface created by each learning algorithm is distinct, and small changes to algorithms can cause large changes in the energy surfaces created.

2.4.4 Capacity

The simplest definition of the capacity of a Hopfield network (M^{cap}) is the number of learnt patterns that are stable. This simple definition does not consider the percentage of stable patterns in the network that are learnt patterns, nor the percentage of the learning population that are stable, merely the total number. There are various ways to measure the limit of the amount of information that can be stored, and different capacities for the individual learning rules. The original estimation for the capacity of the Hebbian learning rule was $M^{\text{cap}} = 0.14N$ (Amit, Gutfreund, and Sompolinsky, 1987a), where N is the number of units in the network. Rigorous mathematical proofs (McEliece, Posner, Rodemich, and Venkatesh, 1987; Bovier, 1999; Löwe and Vermet, 2005) have shown that the upper bound for perfect recall is actually closer to:

$$M^{\text{cap}} = \frac{N}{\gamma \ln N} \tag{2.13}$$

[5]This figure does not accurately represent the true shape of the basin of attraction of the patterns as the relationships are lost when the hypercube is flattened to 2D, however it does show the symmetry in the energy surface. The flattening from 6D to 2D is performed by dividing the 6 bit representation into two 3 bit numbers which are placed on the x and y axes. In this mapping the energy of state $< -1, +1, -1, -1, +1, +1 >$ is shown at position 2,3 (010, 011).

where $\gamma = 2, 4, 6$ depending on the exact dynamics of the storage required. These results were generated on *independent identically distributed* (i.i.d.) patterns generated by Bernoulli random values of ± 1. The theoretical maximum amount of information that can be stored in the Hopfield network, independent of the learning rule, is $M^{\text{cap}} = N$ patterns for networks with symmetric connections, and $2N$ for asymmetric connections (Gardner, 1988). This result is for uncorrelated patterns where the probability of being active is 50%, $\alpha = P(x_i = +1) = 0.50$. Unless otherwise stated, the simulations in this thesis use these types of patterns. When patterns other than these are used, they will be noted as $X_{0.2}$ when $\alpha = 0.2$, indicating that 20% of the units in the pattern X are active.

There are many different maximum capacity values that can be presented depending on the type of pattern stored (Amit, Gutfreund, and Sompolinsky, 1987b), the learning procedure (Storkey and Valabregue, 1999), connection symmetry (Chen and Amari, 2001), and the definition of successful storage (Löwe and Vermet, 2005). The number of learnt patterns that are stable is only one of the possible measures of the performance of the network. Given that we want a network to be an effective content addressable memory system there must be a large number of content probes that relax to stored patterns and the ratio of learnt patterns to spurious patterns must be high.

Spurious patterns are defined as the stable states in the network that are not part of the learnt set of patterns. In the $\mathcal{H}_{6,\pm}$ example above, with one pattern learnt, the only stable states were the learnt pattern and its inverse. For Hebbian learning if three or more patterns are learnt, spurious states are created. As the number of learnt patterns increases the number of spurious patterns also increases, but at a much faster rate. Once the network has learnt more than the maximum capacity for Hebbian learning ($0.14N$), spurious states start to dominate the networks and the learnt patterns become unstable.

2.4.5 Basins of Attraction

The basin of attraction for a stable pattern is the set of states which relax to that pattern. The larger the cardinality of the set the greater the size of the basin of attraction. In the $\mathcal{H}_{6,\pm}$ network above with one learnt pattern, the two stable states (the pattern and its inverse) have large basins of attraction. As the number of patterns learnt by the system increases the size of the basin of attraction of each individual pattern decreases.

The size of a basin of attraction is very important when using the Hopfield network

as a content addressable memory. If a learnt pattern is to be recalled from a noisy version of the pattern, the noisy probe has to be within the basin of attraction of the learnt pattern. If the basin is too small then the only patterns that will relax to the learnt pattern are those that start with almost identical activation values. When the learnt patterns no longer have basins of attraction the memory system is only useful as a familiarity test rather than a content addressable memory.

2.4.6 Update Timing

There are two categories of update timing for Hopfield networks; synchronous or asynchronous. The process described in Section 2.4.2 is an asynchronous process where each unit is selected at random and that unit is updated. The equivalent synchronous process involves every unit updating at the same time. Synchronous updates can be calculated quickly using matrix multiplication with the weight matrix $\mathbf{W}$ and the current activation of the network $\mathbf{Y}^t$ at time t:

$$\mathbf{Y}^{t+1} = \mathbf{Y}^t \cdot \mathbf{W}^t \tag{2.14}$$

This dot product calculation is $O(N^2)$ for each cycle. This update rule unfortunately breaks the convergence guarantee as it can result in the network entering a *"blinking" state* (van Hemmen, 1997). This is where a set of units continually change from one state to the other and back again. This can easily be visualised in a two unit network connected by a negative weight. Units A and B are initialised as $+1$ at time $t = 0$.
At $t = 1$:

$$\begin{aligned}
\psi_A^1 &= sg(w_{AB} \times \psi_B^0) \\
&= sg(-1 \times 1) \\
&= sg(-1) \\
&= -1
\end{aligned}$$

At $t = 2$:

$$\begin{aligned}
\psi_A^2 &= sg(w_{AB} \times \psi_B^1) \\
&= sg(-1 \times -1) && (B \text{ updated at time } 1) \\
&= sg(+1) \\
&= +1
\end{aligned}$$

This network "blinks" between state $< 1, 1 >$ and $< -1, -1 >$ as both units are updated at the same time. If only unit A is updated the network will stabilise on either the pattern $< -1, 1 >$ or $< 1, -1 >$. These two configurations are equally stable and performing an update on either unit will not change the activation state of the network $\mathbf{Y}$. Given these blinking states, synchronous update is not a behaviourally neutral replacement for asynchronous update. Van Hemmen (1997) argues that you can use these blinking states as indicators that the network has not found a stable learnt pattern. This unfortunately means that the ability to relax to a learnt pattern is severely reduced in the network suggested by van Hemmen.

As relaxation is one of the most common operations performed on the Hopfield network it is important to make the implementation as time efficient as possible. Synchronous updating is fast as there are only $N^2\,c$ updates required, where c is the number of cycles before the network relaxes or finds a blinking state. However, because this relaxation is not equivalent to asynchronous update, we also need a fast implementation of the asynchronous update procedure.

When randomly selecting units to update it is common to select a stable unit, which will not change its activation state. In its raw form the update rule is an example of the classic "coupon problem" of collecting items when each item is selected randomly with replacement (Durrett, 1996). Simple random selection is extremely slow for large networks. The first simple approach to increasing the speed of relaxation is to create a random permutation of the units and update each unit once per cycle. This is $O(N^2)$ for each update cycle. The difference between this and synchronous updating is that each unit is updated and its *new value* is used for all of the subsequent calculations in that cycle. Permuted order selection has similar dynamics to the asynchronous update but with a much faster relaxation time. There are however, sequences of updates which are possible with the raw form that cannot be generated by randomly permuting the update order, for example any sequence of updates that include any unit twice between updates of another unit. These are common, and so although the performance is similar it is not equivalent.

We propose a faster and operationally equivalent way to perform a fully asynchronous update, by changing the way the units are selected for updating. When randomly selecting units, only unstable units will change the state of the network. The state changes are equivalent to the state changes caused by the simpler sequence that only includes the units that will change. Randomly selecting from all units and then discarding selections which do not change state is equivalent to randomly selecting

from only the unstable units. This can reduce number of operations significantly. To achieve this we need a list of units that are unstable, which is updated after every unit change. Creating this list is $O(N^2)$ for the first iteration, but to maintain the list requires only N calculations per unit update. When a unit is updated it signals all the other units and changes the recorded sum of their current input. Thus if a unit A is updated, then all the other units will alter their input by the change in A multiplied by the weight of the connection to A:

$$v_B = v_B + (\psi_A^t - \psi_A^{t-1}) w_{BA} \tag{2.15}$$

where v_B is the unit input and $(\psi_A^t - \psi_A^{t-1})$ is the change in activation for unit A. Thus each unit update is $O(N)$. For large networks the comparison between the $O(N^2)$ for each cycle and $O(N)$ for each unit update makes this form of updating more than an order of magnitude faster. For a 1000 unit network $\mathcal{H}_{1000,\pm}$ with 20 patterns learnt, the comparison between our approach of keeping a list of units that are unstable (List Asynch) and the standard approach of processing a random permutation of the units (Perm Asynch) is shown in Table 2.4. The numbers represent the averaged number of units that were updated during relaxation. The 6000 in "Spurious – Least / Perm Asynch" indicates that there were six cycles through the 1000 unit network, thus 6000 unit updates. By comparison only 552 units were updated for the same condition for "List Asynch". List Asynch requires about 10% of the processing used for Perm Asynch when relaxing to learnt patterns and only 5% for spurious patterns (for the distinction between spurious and learnt patterns see Section 2.4.5).

		List Asynch	Perm Asynch	Comparison L/P
Learnt	Least	693	5,245	13.2%
	Avg	808	8,359	9.7%
	Most	886	10,856	8.2%
Spurious	Least	552	6,000	9.2%
	Avg	779	14,749	5.3%
	Most	1095	32,000	3.4%

Table 2.4: The number of unit updates during relaxation for list and permutation asynchronous updating. Values are averaged for the 20 learnt patterns and the first 100 spurious patterns found in an $\mathcal{H}_{1000,\pm}$ network found using 5000 random probes.

The difference between spurious and learnt patterns indicates that the shape of the

basin of attraction for each is different. The learnt patterns tend to have fewer unit updates before reaching a stable state. This could indicate that the basin of attraction for a learnt pattern is more symmetric, as for most units selected the update moves toward the stable learnt pattern.

2.5 Performance

There are a number of different ways to measure the performance of a content addressable memory system. The relevance of a metric needs to be weighted by the goals of the network. Different authors use very different measurements which seem to be chosen to best differentiate their network from other solutions (for example compare Christos (1996) and van Hemmen (1997)).

2.5.1 Capacity

The standard measure of performance is the stability of the patterns that have been stored, the capacity (Hopfield, 1982). If all the desired patterns are stable then the network is considered to be working very well. When comparing two learning procedures, the one that can make more patterns stable is usually considered to be the best. However, simply measuring the number of stable states is not necessarily indicative of an effective learning algorithm. For example, given the constraint of all learnt patterns having exactly 50% of their units active, we can construct a network where all the weights are negative one. Every state that has exactly 50% of its units active will be a stable state in this network. We can show this by looking at one of these 50% active states. Selecting an active unit at random the unit input is:

$$
\begin{aligned}
\upsilon_i &= \sum_{j,j\neq i}^{N} w_{ij}\psi_j \\
&= \sum_{j}^{(N/2)-1} (w_{ij} \times 1) + \sum_{k}^{N/2} (w_{ik} \times -1) \qquad \text{where } j \in [\psi^p = 1], k \in [\psi^p = -1] \\
&= (\frac{N}{2} - 1)(-1 \times 1) + \frac{N}{2}(-1 \times -1) \\
&= -(\frac{N}{2} - 1) + \frac{N}{2} \\
&= +1
\end{aligned}
$$

thus all active units remain active and inactive units remain inactive. The $N/2 - 1$ comes from the removal of self connections. There are ${}_NC_{(N/2)} = N!/((N/2)!(N/2)!)$

which is approximately $2^N\sqrt{2/(\pi N)}$ of the 2^N patterns that are stable in this network. This grows exponentially when compared to the linear growth of $0.14N$. For example, in a 100 unit network there are 10^{29} stable states, but these states make no distinction between learnt patterns and spurious patterns. This problem demonstrates the inadequacy of capacity, without discrimination between types of patterns, as the only measure of performance.

We will refer to the number of learnt patterns that are stable as the recognition capacity. A pattern that is only just stable, may be recognised but not recalled from similar content. The example above shows that the number of stable patterns may not be correlated with the performance of the network as a content addressable memory. Another measure of capacity is the number of learnt patterns that are *recalled*. The recall capacity is the number of learnt patterns that are found when probing the network. This is related to the basin of attraction of the stable learnt patterns. A large basin allows the content in the "content addressing" to have a larger variation from the original pattern.

2.5.2 Basin Size

Content addressable memory requires that a learnt pattern not only be stable, but that it must be able to be recalled from a partial pattern. Thus the size of the basin of attraction is an important measure of the performance of a content addressable memory. There are several ways of testing the basin of attraction for a pattern. The possible inputs are:

1. The exact learnt pattern.
2. A percentage permutation from learnt pattern.
3. A randomly generated pattern.

The possible results of relaxation are:

1. The exact learnt pattern.
2. A pattern within some percentage permutation of a learnt pattern.
3. A spurious pattern.
4. The prototype of a large number of similar items.

Combinations of the above input/output pairs test for different properties of the basin of attraction. Some of the standard pairings are shown in Table 2.5.

All of these combinations measure something interesting about the Hopfield network. The row of results related to inputs that are "close to learnt" can be used to

	Exact Learnt	Spurious
Exact Learnt	stable learnt pattern	unstable learnt pattern
Close to Learnt	size of learnt basin	size of learnt basin
Random Probe	% of pattern space	number & size of spurious

Table 2.5: Different types of information that can be gathered by combinations of different probes (rows) and the stable patterns found (columns).

discover the size of the basin of attraction of a learnt pattern, with relaxation to the pattern indicating that the basin includes the point probed. The use of "Exact Learnt" can be extended to include patterns that are close to the learnt patterns. It may be acceptable in certain situations to accept patterns that are close to a learnt pattern, say less than 1% different, to be an example of the learnt pattern. This flexibility can be used to compare results where the number of learnt patterns that are exactly recalled is low, but there are many learnt patterns which have a stable pattern very close by.

When investigating the basin size of a learnt pattern, it is possible to give an estimate of the average size of basins of attraction in an attractor network. Gardner (1988) calculated the capacity and basins of attractions of patterns given i.i.d. random patterns. This work has been extended with more accurate predictions of the average size of basin by Koyama, Fujie, and Fujiwara (1996) and Okada (1996). These calculations rely on the types of patterns being stored and describe theoretical limits to capacity with an idealised learning algorithm as well as estimates for actual learning algorithms like Hebbian learning. Changes to the type of patterns stored and the learning algorithms would require developing new estimations which may or may not be possible. Without formal theoretical average basin sizes it is necessary to find other ways of estimating the size of the basins of attraction.

Storkey and Valabregue (1999) use two measures for the size of basins of attraction; the direct size, where every pattern within radius k relaxes to the central pattern; and the indirect size, a radius within which some percentage β of the k permuted patterns relax to the centre of the basin. A third measure is the number of random probes of the network that relax to a particular stable pattern. This gives an indication of the size of the basin as a percentage of the pattern space. The percentage of total pattern space is a useful comparison between states with very large basins.

The number of calculations required to exhaustively calculate the basins of attrac-

tion in an $\mathcal{H}_N$ are extremely large. Each point in the 2^N space of patterns requires at least an $N^2 + Nm$ relaxation process, where m is the number of units required to update during relaxation. The relaxation process is non-deterministic as it selects a unit at random to update. Therefore for each initial pattern it would be necessary to map every possible selection order and trace all the possible paths from the initial pattern to a stable pattern. This sampling process could be performed for states with one or two activation values different to a stable pattern, but, as the number of alterations increase the number of patterns to check increases with ${}_NC_k$ where k is the number of alterations, equivalent to the radius of the basin that is being tested.

The radius k can also be described as the overlap[6] (m_f) between the original learnt pattern and the test pattern ($m_f = k/N$). For a 100 unit network, a radius of $k = 10$ is the same as an overlap of $m_f = 0.9$. The number of states within this radius is approximately 10^{12}. Testing these patterns to see if they relax to the initial stable pattern requires a relaxation with at least 10^4 operations, giving a lower bound of 10^{16} calculations per stable pattern. This lower bound does not include the situations where the path of the relaxation does not directly converge on a stable pattern. This incredibly large number of calculations is impractical on current hardware[7].

To estimate the size of the basin of attraction we try to find an overlap value such that 50% of the randomly perturbed patterns will relax to the original base pattern (Chengxiang, Dasgupta, and Singh, 2000). To find this overlap value suggest the use of an iterative method which slowly decreases or increases the overlap using simulated annealing. This process is:

- initialise the overlap to $m = 0.75$ (75% of units identical to the state at the centre of the basin)
- repeat for l cycles
    - initialise the network with a pattern that has an overlap of m with the learnt pattern
    - relax the network to a stable state
    - if the stable state is the desired pattern decrease the overlap by a branching factor (therefore testing a larger basin), otherwise increase the overlap by the same amount, $m = m \pm T$
    - use simulated annealing to decrease the branching factor

[6] We use overlap as it is independent of the size of the network, while radius k is dependant on N.
[7] On a 1Ghz processor this is $10^{16}/10^9 = 10^7$seconds, or about 4 months per state to test.

When the network successfully relaxes to the centre of the basin the overlap is decreased, meaning that the estimated size of the basin of attraction is larger. To estimate the point at which 50% of probes relax to the stable state, we take the average estimated overlap m of the last 100 probes. This measure can only be a rough estimate of the actual size of a basin of attraction, but it does provide consistent ordering of patterns based on estimated basin size which are consistent with more time consuming tests of basin size.

Table 2.6 shows the estimates for a 100 unit network which has used Hebbian learning to store $0.06N$ or $0.13N$ patterns. The "Ave. Relax" numbers show that the estimate is performing well with nearly 50% of the altered patterns successfully relaxing to the stable learnt pattern, with standard deviation of about 4.7. The average basin size for patterns in a network with $0.06N$ learnt patterns is much larger than the average for patterns in the $0.13N$ network. A basin with overlap of 0.870 is $\frac{1}{36}^{th}$ of the size of a basin with an overlap of 0.850. For a more detailed analysis of relaxation using different levels of overlap see Chengxiang *et al.* (2000).

Learnt	Est. m_f	Ave. Relax	St. Dev
0.06N	0.853	49.31	4.75
0.13N	0.870	49.24	4.69

Table 2.6: Basin estimate values.

Using this measure the largest possible basin of attraction has an overlap of $m_f = 0.5$, while the smallest has an overlap of $m_f = 1.0(k = 0)$. A basin of $m_f = 0.5$ consumes the whole space of the network as when there is less than 50% overlap with the original pattern it will relax to the inverse of the pattern. The radius of the basin of attraction of a pattern determines how much of the state space is consumed by the basin. In a 100 unit network a pattern with a basin of $m_f = 0.85, k = 15$ has 3×10^{17} patterns that relax to it, but this is only 0.00000000002% of the total state space. A smaller basin with an overlap of $m_f = 0.87$ has only $\frac{1}{36}^{\text{th}}$ of the patterns in its basin as compared to an overlap of $m_f = 0.85$. Table 2.7 shows an example of this basin estimate versus the actual number of probes relaxing to particular patterns in a 100 unit network.

Figure 2.11 shows the relationship between the number of probes that relax to a pattern and the estimated basin of attraction. The empty area in the bottom left shows that patterns do not have both a large basin of attraction (a smaller required overlap

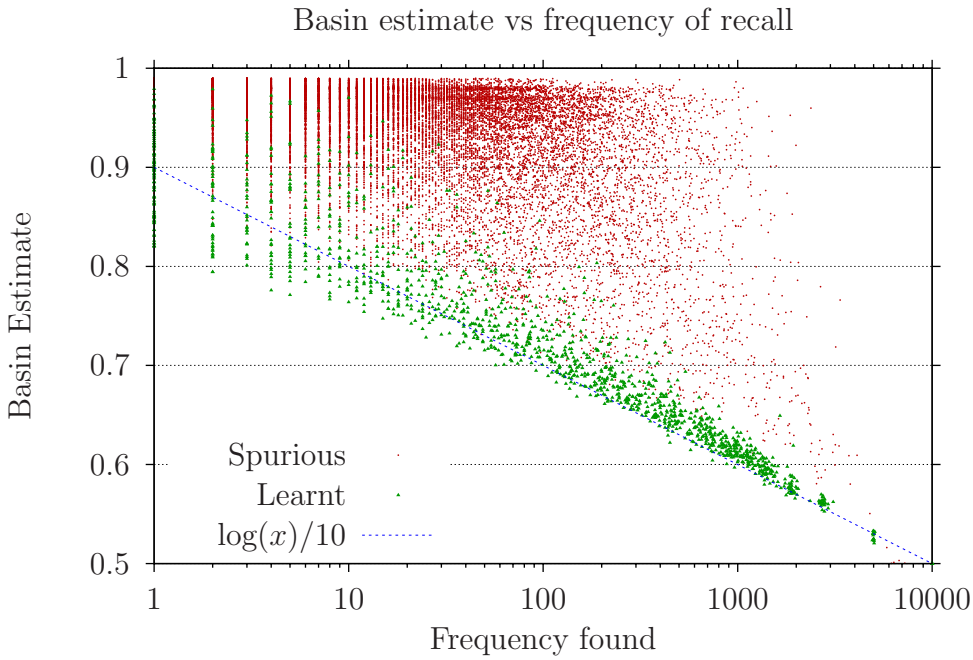


Figure 2.11: Estimated basin size against the frequency at which patterns were found with 5000 random probes in an $\mathcal{H}_{100,\pm}$ network. The learnt patterns mostly lie on the diagonal. (1707 learnt patterns, 22917 spurious patterns, from 10 trials learning 30 patterns using Hebbian learning).

Patt	Actual frequency	Estimated overlap m_f	Actual % state space	Estimated % state space
0	678	0.57986	13.66%	13.50%
1	940	0.57042	18.80%	19.08%
2	913	0.57146	18.26%	18.46%
3	915	0.57057	18.30%	19.00%
Spurious	1554	-	-	-

Table 2.7: Example of sampling the basins of attraction but random sampling (Actual frequency) and by using our estimation of the basin size (Estimated overlap), showing that the percentage of state space predicted by the basin size (Estimated % state space) is similar to the the value found with random proves (Actual %state space) in an $\mathcal{H}_{100,\pm}$ with 4 patterns learnt and using 5000 probes.

to relax to the pattern) and low sampling frequency. The learnt patterns mostly lie near the diagonal $0.9 - log(x)/10$, as they have basins that are relatively uniform. The estimation process which expands the basin until 50% of probes are not relaxing to the pattern, works well with these uniform patterns as they are equally stable in most dimensions. The spurious patterns, on the other hand, are spread out in the graph as they have non–uniform basins of attraction. A non–uniform basin can still attract a large number of probes from sections of the space which are not evenly distributed. This can be visualised in 2D in Figure 2.12. The "expanding basin estimate" will give a value much lower than the frequency for these non-uniform patterns. The accuracy of this measurement also decreases as the basins get smaller (m_f approaches 1) because of the infequency of probes relaxing to these small patterns.

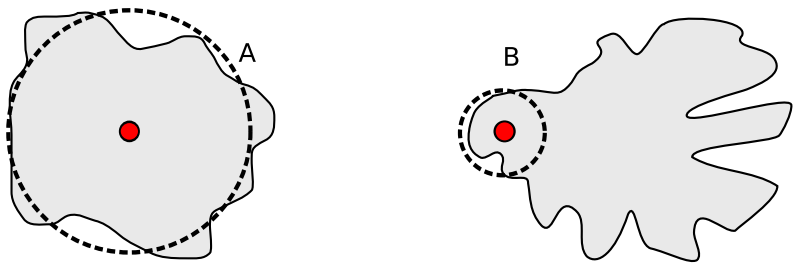


Figure 2.12: Visualisation in 2D of the expanding basin estimate for a uniform learnt pattern A and a not–uniform spurious pattern B.

2.5.3 Hamming Distance

The Hamming distance between two Boolean arrays is the number of indices at which they differ. A learnt pattern that is stable will have a Hamming distance between the input and output states of 0, as the pattern will not change during relaxation. If the requirement for perfect recall of a pattern is relaxed slightly, and a pattern close to a learnt pattern is considered a successful recovery, the distance between the new centre of the basin of attraction and the learnt pattern that is close to it can be used as a measure of the performance of the network. We have used this measure of performance in this thesis, and in the published works that are part of this thesis Robins and McCallum (1998, 1999, 2004). If the Hamming distance between a learnt pattern and the stable pattern it relaxes to is small the assumption is that the centre of the basin of attraction has moved a short distance. This however is not always the case. In a network with a very large number of spurious states the distance that any probe moves during relaxation is small. This is not because the network has moved the centre of a learnt pattern to the new stable state, but because of the increased likelihood of finding one of the plentiful stable spurious configurations.

The Hamming distance between an unstable learnt pattern and the stable state it relaxes to can be used in conjunction with an estimate of the size of the stable state's basin of attraction. If the stable state has a large basin of attraction then it might be possible to claim that the learnt pattern has moved and is "mostly" remembered. To say the pattern has moved, the Hamming distance between the unstable learnt pattern and the associated stable state should be much smaller than the basin of attraction, and the "basin should only contain a single unstable learnt pattern". If the basin contains two or more learnt patterns, then the spurious state might be either a prototype for those patterns, or have a strong correlation to them. In either case it would be difficult to claim that an individual pattern had been successfully "remembered" if it was just one of many unstable learnt patterns that relax to a large spurious pattern.

2.6 Conclusion

In this chapter we introduced a unit model for a neuron. These units are used to construct Hopfield networks which are the focus of this thesis. We present two contrasting learning algorithms for adjusting the weights in the network: Hebbian learning with its strong biological basis and simple update rule; and delta learning, a powerful error correcting procedure that guarantees convergence. Both of these procedures alter the

weighted connections in the Hopfield network to learn some set of desired patterns. By learning we are referring to making the patterns the content addressable memories in a Hopfield network.

The concept of energy was introduced. Each unit calculates an energy value. Positive values indicate that the unit is unstable and will change activation in the next time step, while negative values indicate stable units. The sum of the individual energies of the units is the total energy of a particular pattern in the network. If every unit in the pattern has a negative energy then that pattern is a stable state of the network. Thus, to learn a pattern, the energy of every unit in the network must be altered until they are all negative for that pattern.

The energy surface of a Hopfield network is defined as the total energy of the network in each possible activation pattern. Basins of attraction in the energy surface are like "depressions" in the surface. These low energy local minima form the basis of the content addressability of the network. Probe patterns that are close to a learnt pattern, within the basin of attraction, will relax to the learnt pattern and thus will have addressed the memory by presenting most of the content. A large basin of attraction means that more memory probes will relax to the pattern at the centre of the basin. In terms of content addressability, less of the original content is required for a probe to find the learnt pattern. Stable patterns without a basin of attraction do not fit with the concept of a content addressable memory as they can only be used as a recognition system, because any change of content results in relaxation to a different pattern.

There are many performance measures for Hopfield networks. The most often discussed is the capacity of the network, measured by the number of desired patterns that can be learnt as stable states in the network. With Hebbian learning the capacity of the network is approximately $0.14N$. Relying on the capacity as the only measure of performance can be misleading, particularly when the goal is to create a content addressable memory. Delta learning is able to make as many as $2N$ patterns stable, but these patterns have almost no basin of attraction. It is also possible to make every pattern with the same set level of activation stable. This allows ${}_NC_{(N/2)} = N!/((N/2)!(N/2)!)$ pattern to be stable. This network is not a good content addressable memory even though it has a massive capacity. Thus, when a large capacity is achieved, the basins of attraction of the stable patterns must also be tested. A well performing content addressable memory system would have a large number of stable learnt patterns with reasonable basins of attraction.

Chapter 3

Catastrophic Forgetting in MLP Networks

Catastrophic forgetting (CF) is the sudden and total loss of a memory system's ability to recognise or recover stored memories. It has been identified as a problem for multi-layer perceptron (MLP) type architectures (McCloskey and Cohen, 1989; French, 1992; Robins, 1995). In this chapter we briefly review possible causes and solutions in MLP networks, including reducing representational overlap, rehearsal, and pseudorehearsal. In the next chapter we discuss the possible causes of catastrophic forgetting for an auto-associative task in Hopfield networks and compare possible solutions for this problem.

Catastrophic forgetting occurs in learning environments that require many items to be learnt in succession. First a set of items are learnt until the system remembers them, and then more items are added. This continual addition of new information is more like the real environment in which biological neural networks have to operate, than static tasks where all the information is present at one time. If biological networks suffered from CF then new information would obliterate recall for all previous events, and as a result animals would only ever have short term memory. This task of learning many items over an extended time, where only the current information is present, highlights the "stability/plasticity dilemma" (Grossberg, 1987). The problem is how to keep the current behaviour stable while allowing the flexibility and plasticity necessary to incorporate new information.

Ideally when a learning system finds a solution to a problem (in the case of an ANN finds a set of weights that map inputs to outputs) the solution should be robust enough to cope with learning new solutions to different problems. As the problems presented change over time, important solutions (ones which are critical to survival) would be

retained even when solutions have to be found to new problems. Most ANNs err on the side of excessive plasticity. These learning systems are designed to use all the available resources to solve the current problem as fast as possible. By utilising every available resource they disrupt the solutions to previously presented problems. For these ANNs if a connection is involved in the mapping from input to output for a new problem, its weight will be adjusted. If the weights had been tuned to solve a previous set of problems, then by wrenching them towards new values the balance that was important for the previous learning is destroyed. Grossberg suggests the analogy of a human trained to recognise the word "cat", and subsequently to recognise the word "table", being then unable to recognise "cat". This is what we will call catastrophic plasticity, where unmanaged plasticity, the ability to change the weights in the network, is the primary cause of catastrophic forgetting.

3.1 Catastrophic Plasticity

While stability/plasticity issues are very general, the term "catastrophic forgetting" has tended to be associated with a subset of learning environments, that of structurally static networks trained using supervised learning. A number of studies have used MLP type networks (typically using back-propagation for training) to highlight the problem of CF and explore various issues (McCloskey and Cohen, 1989; Hetherington and Seidenberg, 1989; Ratcliff, 1990; Lewandowsky, 1991; Murre, 1992; French, 1992, 1994, 1997; McRae and Hetherington, 1993; Lewandowsky and Li, 1994; Sharkey and Sharkey, 1995; Robins, 1995, 1996; Frean and Robins, 1999). The standard demonstration of the problem uses a network that has been trained on a "base" population, and then tests performance once a new item or group of items has been learnt. The effect of this new learning on the old items (base population) can be illustrated by re-testing the base population. If the performance falls off very sharply it is called "catastrophic forgetting". This sudden and catastrophic loss can occur with only a single new item learnt.

Figure 3.1 shows the performance of an MLP network trained on a base population and then tested after learning a number of new items. The primary cause for this type of forgetting is the plasticity of the network rather than capacity of the network.

In Robins and McCallum (1999) we categorised the methods that reduce CF in MLP type networks into three general categories; reducing overlap, rehearsal and pseudo-rehearsal.

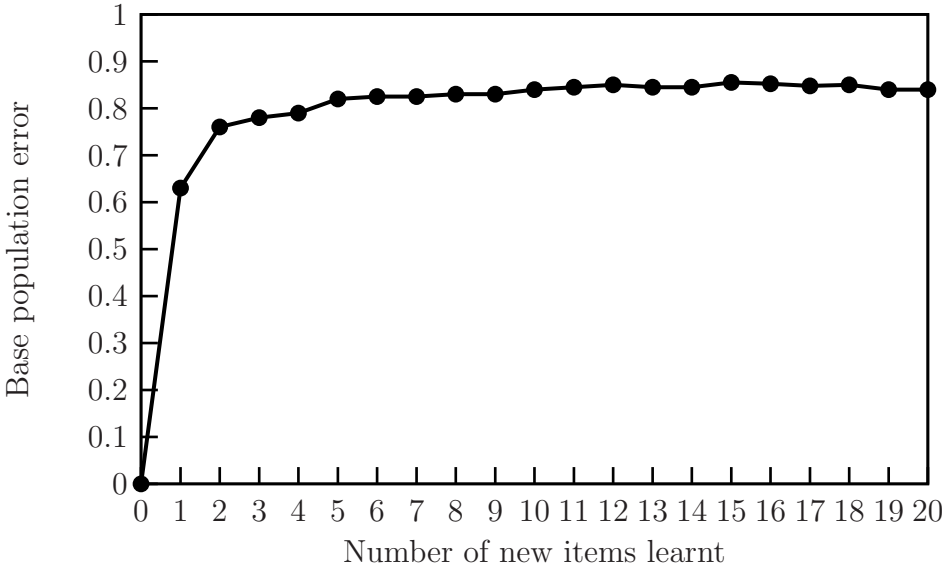


Figure 3.1: Catastrophic forgetting in an MLP network caused by plasticity (adapted from Robins (1995)).

3.2 Reducing Overlap in MLP Networks

French (1992) suggests that the extent to which catastrophic forgetting occurs is largely a consequence of the overlap of distributed representations, and that the effect can be reduced by reducing this overlap. Rather than plasticity this identifies correlation between patterns as the root of the problems. Correlated patterns are forced to share representations in the network. If this overlap is removed the patterns can be learnt without disturbing the currently stored memories. There are various approaches for reducing overlap in the representation of patterns. The novelty rule (Kortge, 1990), activation sharpening (French, 1992), context biasing (French, 1994), and the techniques developed by Murre (1992), and McRae and Hetherington (1993) all fall within this general framework. The goal of these methods is to decrease the overlap of the internal representation (the hidden units used to represent the mapping from input to output) of patterns that have been learnt. This has two benefits – it allows new items to be learnt using units that have not been involved in storing other patterns, and it increases the strength of the units that are involved in learning a pattern making the network less susceptible to interference.

This increased orthogonality of hidden unit representations does not, in general, prevent CF from occurring, although it does ameliorate its effects so that retraining of the base population is much faster.

The novelty rule of Kortge (1990) has been shown to prevent CF, but can only be used with autoencoder (autoassociative) networks. Given that one of the causes of CF in MLP networks is moving all the weights to solve the new problem, it is also possible to ease the problem by pre-training the network on a population which is representative of the both the base population and a selection of possible new patterns. This could be seen as simulating prior knowledge of a task domain as explored by Sharkey and Sharkey (1995); McRae and Hetherington (1993). In McRae and Hetheringtons simulations this pre-training reduced the overlap of hidden unit representations in subsequent learning, as the units had already moved their weights to cover the entire space of possible problem to solution mappings. This unfortunately does not prevent CF as new patterns that were not foreseen in the selection of representative patterns will still cause CF. This solution also sacrifices some of the other advantages of MLPs such as the ability to generalise to new parts of the pattern space.

3.3 Rehearsal and Pseudorehearsal in MLP Networks

A second general approach to preventing catastrophic forgetting involves "rehearsing" the base population by incorporating some base population items in to the new learning population. This process deals with the catastrophic plasticity, by allocating some of the plasticity to relearning the base population items. Assuming that we learn new items one by one in a sequence:

- with access to the entire training population
- for each new item to learn
    - construct a new training population with the new item and items selected from the base population
    - train with the new population using the same learning algorithm as the base population learning

The number of items selected from the base population can vary from no items, resulting in CF, through to all the items which results in the same performance as non-sequential learning where all items to learn are part of the base population (McClelland, McNaughton, and O'Reilly, 1995). Significant performance improvements can be achieved with a relatively small number of rehearsed items (Robins, 1995).

Rehearsal requires access to the original base population so that items can be incorporated throughout later learning. Having access to the base population items from some other store would make the MLP memory system redundant as there is already an accurate recording of all the patterns. Robins (1995) introduced the idea of a pseudoitem population that could be extracted from the network by probing its current behaviour. By sampling the underlying function of the network the functional mapping could be retained by careful rehearsal of the associated input-output pairs. This does not require access to the original items, but uses items generated from the network itself.

With these new pseudoitems the learning procedure would be:

- for each new item to be learnt
    - construct a population of pseudoitems by creating random inputs and passing them forward through the network to generate their associated outputs
    - construct a new training population consisting of the new item and some items selected from the population of pseudoitems
    - train with the new population using the same learning algorithm as the base population learning

This system uses the functional behaviour of the network to protect the performance.

The use of pseudorehearsal has been shown to be effective for a number of different populations: autoassociative and heteroassociative randomly constructed data sets by Robins (1995), and by Ans and Rousset (1997); autoassociative learning of the Iris data set by Robins (1996); a classification task using the Mushroom data set by French (1997); and a structured "task domain" by Silver and Mercer (1996). Pseudorehearsal performs well on a number of different problems with different underlying learning algorithms.

There are limitations to the types of learning algorithms and population sets that pseudorehearsal can improve. The Requirements are:

1. A network capable of learning both the original base population and the new items.
2. A learning algorithm where rehearsal is an effective method for improving performance.
3. A means to extracts items which represent the function of the network.

Requirement 1 results from the need to have the capacity within the network to learn all the patterns presented. The performance of the system is limited by the capacity,

so if the learning system is unable to learn all the patterns, pseudorehearsal will not improve performance. Pseudorehearsal is only effective in networks where rehearsal is effective and Requirement 2 follows from the tight relationship between rehearsal and pseudorehearsal. Requirement 3 is the most restrictive for the pseudorehearsal approach. Finding a mechanism that provides useful pseudoitems is the most difficult aspect of appending pseudorehearsal to a new learning algorithm or network type. The mechanism is different for different network structures. As pseudorehearsal preserves the "function" of the network there needs to be a way to extract this function in order to preserve it.

In an MLP network the function of the network can be extracted by sampling with a random input pattern and passing that input through the network to generate an output. This pairing of a random input and output accurately represents the function of the network as shown in Robins (1995) and Frean and Robins (1999). How well the generated pseudoitems represent the function is important to the preservation of that function. Too few pseudoitems will not represent the function well and too many will prevent the function from adapting to learn the new items. The type of pseudoitem generated is also important. For networks where input–output mapping is not the primary function, such as Hopfield networks, these type of pseudoitems will not help preserve the function of the network (see Section 4.5 for details of pseudoitem generation in Hopfield networks).

Pseudorehearsal focusses on the function of the network rather than the current weight configuration. As there are many combinations of weights that will encode a particular base population set, the current set of weights is not preserved. The training of the network will alter the weights and the behaviour of the network, but if the pseudoitems are an accurate representation of the function these changes will be restricted to the region of the new item. This localisation of changes to the function allows pseudorehearsal to preserve performance in MLP type networks (Robins, 1995; Frean and Robins, 1999). Frean and Robins (1999) provides an introduction to the formal analysis of pseudorehearsal in linear MLP networks.

In the solutions so far it is not the capacity of the network that is causing CF, as each of the networks has satisfied Requirement 1 of pseudorehearsal preconditions (page 49), and could store all the items if presented simultaneously. These solutions target the other causes of CF including excessive plasticity in the connections and overlapping representations in hidden layers. The capacity and the complexity of the function that can be learnt by an MLP network is limited by the network architecture

— the number of units, the number of weights, and the complexity of the activation function. A network with one unit connecting the input layer to the output layer is very limited in the mappings it can produce. All networks and learning algorithms have a maximum capacity, and the behaviour near this limit can either be stable or catastrophic.

3.4 Catastrophic Capacity

Catastrophic forgetting caused by an MLP network reaching capacity can be easily shown in networks with very limited resources, but the problem does scale to large networks. The most obvious example of catastrophic loss of performance caused by reaching capacity is when there is no solution to a particular set of input–output pairs within the limited network architecture.

The encoder network is a simple MLP network with N inputs, k hidden units, and N output units. The input units are only connected to the hidden units and the hidden units connect to the output units. Thus all the information must pass through the small number of hidden units in the centre of the network. The pairs that form the learning population are unit vectors where only one unit is active, (e.g. $< 1,0,0,0,0,0,0,0 >$ $\rightarrow < 1,0,0,0,0,0,0,0 >$). For a network with 8 inputs and 8 outputs there are 8 of these pairs. For any learning system to be able to solve this mapping the number of hidden units k must be larger than $\log_2(N)$. If the number of hidden units is less than this the network will be unable to learn the mapping as there is not enough complexity in the structure of the network.

The thermal delta learning rule presented by Frean (1990) is an attempt to ameliorate this problem. The principle is that over time the learning rule changes its response to errors. Early in the learning process all errors are treated as equally important, but as learning proceeds the weighting for correcting small errors is increased and large errors are reduced. The principle is that if the network is unable to solve competing tasks it should give up on some of the problem set to focus of solving those which are close to a solution. The affect of this is that the network will start to ignore the training items with large errors. A temperature variable is used to weight the importance of the errors. The temperature is reduced during the learning epoch, with the cooler temperatures focussing on small errors. By the end of the allocated epochs the network is only making small changes to perfect the set of patterns that are possible to learn.

In a Hopfield network this allows the network to ignore patterns that are difficult

to learnt. The thermal rule will learn a subset of the random population which may share a number of similarities. With this selection of a group of patterns that are easy to learn together, the maximum number of patterns that can be learnt increases. This does not mean that there is now more information stored in the network than the theoretical maximum.

The theoretical information capacity of a network is a measure of the amount of information that can be stored in the network. This can be calculated independently of the learning algorithm as it relies on the number of individually adjustable elements, usually the weighted connections, in the network.

The amount of information in a set of patterns is related to how similar they are. The first pattern in a population contains a lot of information as the network has not learnt anything about the population. If the following patterns are almost identical to this first pattern, each new pattern only adds a small amount of information to the total information complexity of the learning population. Measures of the capacity of a network given by the number of patterns that can be learnt, are given using random patterns. A set of 10 randomly generated patterns contain almost twice the amount of information of a set of 5 generated patterns. With non independent pattern the amount of information in the first 5 pattern is likely to be much higher than in the next 5 patterns as some of the dependencies will already have been learnt by the network. For an in depth analysis of the information capacity of Hopfield networks see McEliece *et al.* (1987); Löwe (1999); Löwe and Vermet (2005)

The most common solution to a structurally limited network is to increase the resources available without increasing the complexity of the learning problem. For MLP networks this can be done by increasing the number of hidden units. Finding the right number of hidden units for a task is still considered a trial and error process of estimating a reasonable number and evaluating the results. Unfortunately for the Hopfield network there are no hidden units, and so the complexity of the internal structure of the network cannot be increased without changing the network into a generic recurrent network which is no longer directly comparable with the original network.

The other resource limit applied to MLP networks is the number of epochs that are allocated to solving the current problem. If the number of epochs is severely limited then the learning algorithm may not have time to converge on a solution. The number of epochs allocated to learning is usually limited, as it is often unknown if the current network is guaranteed to converge on a solution for a particular problem. There are

various approaches to estimating an appropriate number of epochs to allocate to a particular task, some of which involve monitoring the performance of the network to see if the error is continuing to decrease, or if the network has plateaued, or started to diverge.

3.5 Summary

The main feature of catastrophic forgetting is the sudden and dramatic loss of performance of a neural network. In MLP networks this can be caused by unmanaged plasticity (catastrophic plasticity), or limited theoretical capacity (catastrophic capacity).

The solutions to catastrophic plasticity include rehearsal, pseudorehearsal and overlap reduction. Rehearsal is very effective at removing CF but requires a secondary memory store for the base population which would render the MLP irrelevant. Pseudorehearsal is able to improve the performance of the MLP to almost the same level as rehearsal without having to continually refer back to the base population. Pseudorehearsal provides a mechanism that works within the resources of the MLP. In the next chapter we will investigate the application of pseudorehearsal to an entirely different type of network, the Hopfield network.

Limited capacity is also a potential problem for MLP networks. This is a problem for any system that is dealing with complex learning tasks with limited resources. There are two main limitations, structural and time. If the network's structure is not capable of learning the task then no matter how long the network is given it will never converge on a solution for the whole training set. To solve the problem of limited capacity more resources can be added to the network, such as additional hidden units. Other methods involve altering the learning algorithm so that over time it gives up trying to learn items with large errors and focuses on improving items that are close to a solution.

In the next chapter we investigate the causes of catastrophic forgetting in Hopfield networks and evaluate several potential solutions.

Chapter 4

Catastrophic Forgetting in Hopfield Networks

Although usually explored using MLP networks, catastrophic forgetting (CF) can be just as easily demonstrated in Hopfield type networks, as is the case in Robins and McCallum (1998). However there are a greater variety of causes for catastrophic forgetting in Hopfield networks depending on the learning algorithm and the types of patterns that are learnt.

Figure 4.1 summarises data which illustrates CF as described in Robins and McCallum (1999, Fig 1.). The network is trained on a base population of 44 items so that every item is stable. If the network is then trained in the same way on a single new item, the number of unstable patterns in the base population increases dramatically. This is illustrated in the "None" condition of Figure 4.1 (with no intervention to manage CF). The error continues to increase rapidly as further new items are learnt, and eventually the base population is effectively wiped out. Similar issues have been explored using Hopfield networks by Nadal, Toulouse, Changeux, and Dehaene (1986) and Burgess, Shapiro, and Moore (1991), and in the context of the "unlearning" mechanisms detailed in Section 4.4.4.

When considering the causes of CF in Hopfield networks the capacity of the network is often the first and most obvious issue. The Hebbian learning algorithm generates CF near the capacity of the network, at which point, if more patterns are learnt, all information both old and new is lost. We will discuss the capacity, both in terms of the theoretical information capacity and the capacity of particular learning algorithms (Gardner, 1988).

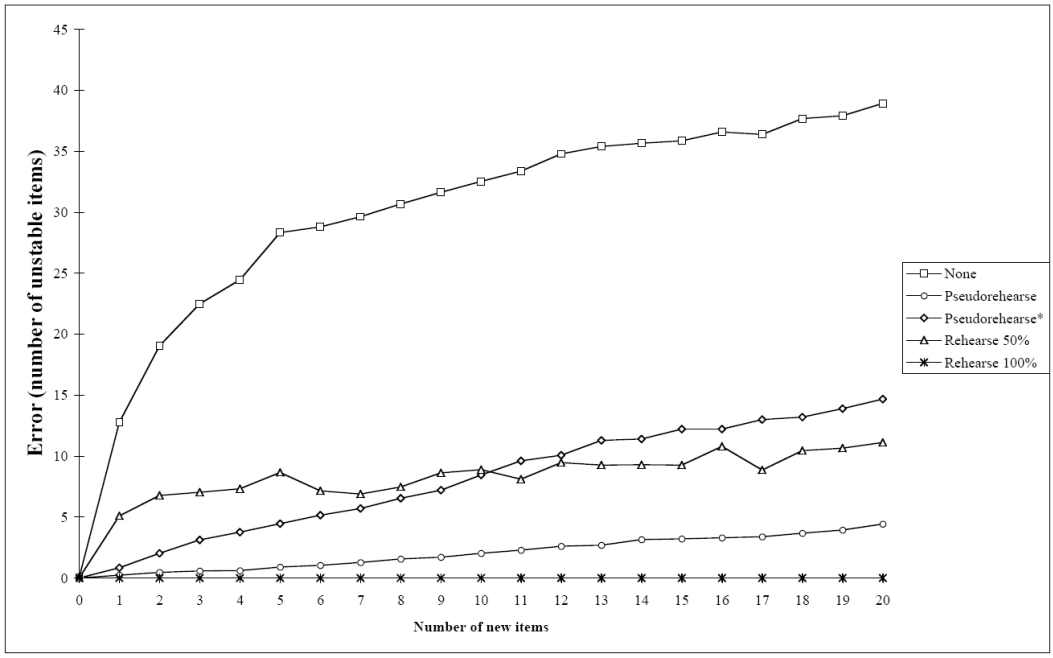


Figure 4.1: Catastrophic forgetting in an $\mathcal{H}_{64,\pm}$ network caused by delta learning's excessive plasticity. Base population of 44 items with 20 new items learnt sequentially (adapted from Robins and McCallum (1999) Fig 1.).

4.1 Catastrophic Capacity

Hopfield networks have a relatively low capacity compared to the MLP networks presented earlier, partly because they do not have the additional internal machinery of hidden units. Adding units to a Hopfield network adds complexity to the problem as *all* units are both inputs and outputs. Without hidden units it is not possible to just increase the resources available to the learning algorithm by adding units. Thus the capacity of a Hopfield network is directly proportional to the number of units in the network.

The number of units in the network defines its information storage capacity. The number of patterns that can be stored depends on the amount of information in the set of patterns. Each set of patterns can contain differing amounts of information. Patterns that have only a single unit different to a prototypical pattern, for example a single inactive unit with the prototype of an entirely active network, contain less information than a set of patterns where the activations have a 50% chance of being different to some prototype. Randomly generated patterns contain a large amount of information as estimated using either Kolmogorov complexity[1] or Shannon information therory (for in depth coverage of information therory see Cover and Thomas (1991)). Information based on probability theory in this context can be described as how surprising an activation of a unit is given the previous set of patterns. With only a single unit change in each pattern from the average, there are $N-2$ units which remain in exactly the same configuration as the previous pattern, and so there is little surprise and thus little information.

For binary units the maximum amount of information is where each pattern has 50% of its units active. In this thesis all patterns are randomly generated with each unit have a 50% chance of being active. Some patterns have a coding ratio (ratio of active unit to inactive units) of above or below 50%. This type of pattern is used for experiments in the rest of this thesis, unless otherwise stated.

Section 2.4.4 discussed the capacity of a Hopfield network and presented the estimate of $0.14N$ as a rough upper bound for Hebbian learning. Above this value Hebbian learning shows a particularly severe form of CF where rather than losing just the old material the network can spontaneously lose all stored patterns including the new items. This can be caused by a single large spurious attractor taking over the whole space with a massive basin of attraction, or by the generation of thousands of small

[1]Given that the algorithm used to generate the pseudo-random numbers is relatively complex

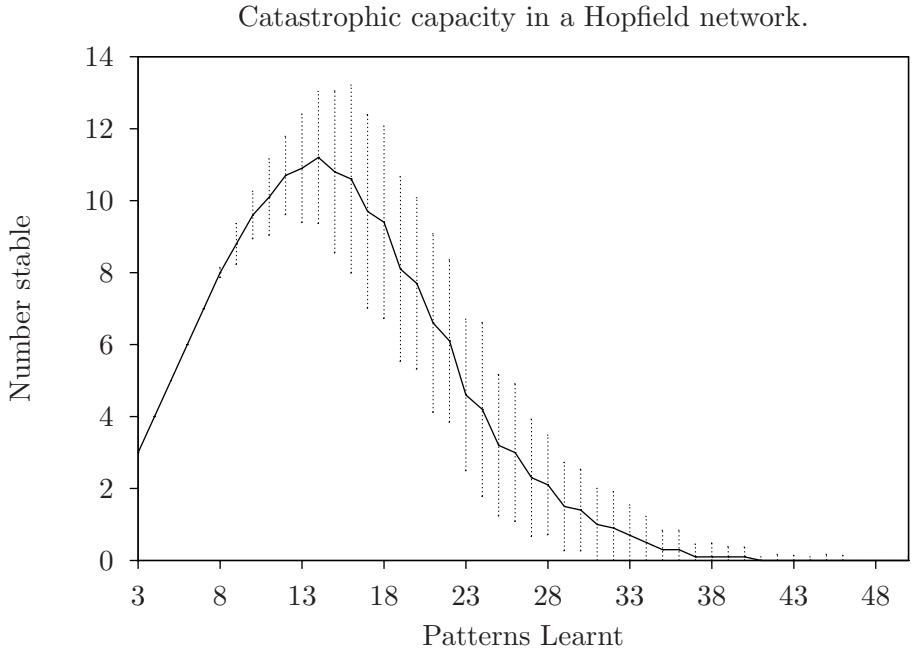


Figure 4.2: Number of learnt patterns stable in a 100 unit Hopfield network $\mathcal{H}_{100,\pm}$ with Hebbian learning showing CF (averaged over 100 repetitions, 50% random patterns).

attractors (stable states with a basin of attraction). When developing solutions to CF in Hopfield networks it is important to evaluate them in networks which are far from capacity as well as those which are close to their theoretical limits.

The catastrophic loss of pattern stability is shown in Figure 4.2 for a network of 100 units. The line shows the average over 100 repetitions with error bars extending to one standard deviation. This network shows classic CF under the pressure of capacity. The storage of this 100 unit network $\mathcal{H}_{100,\pm}$ from equation 2.13 is approximately 14 patterns. Note that by the time 18 patterns have been presented the likelihood of a learnt pattern being stable is about 50% (9 patterns are stable on average). As more patterns are presented fewer and fewer of them remain stable in the network.

4.2 Catastrophic Correlation

Most of the capacity measures for Hopfield networks involve the analysis of uncorrelated random patterns (independent identically distributed random variables – i.i.d.). Assuming that all the patterns that need to be learnt will come from these sorts of clean distributions is unrealistic. Patterns from real data sets are likely to have corre-

lations, and in fact it is likely that it is the correlations between events that makes it possible to extract causal relationships. Correlations alter the analysis of capacity: 1) the maximum theoretical capacity increases in relation to the amount of overlap and the nature of the correlation (Gardner, 1988; Löwe, 1998; Gandolfo, Laanait, Messager, and Ruiz, 1999; Kimoto and Okada, 2002; Bogacz and Brown, 2003), and 2) Hebbian learning performs poorly with correlated patterns as the crosstalk term becomes very large and the pattern which is the correlation between inputs becomes the only stable attractor.

To understand why correlation increases the number of patterns that can be stored, the amount of information in each pattern must be analysed. In the extreme case of only one unit being active in each pattern it is easy to see how to store any of the N "one unit active" patterns. To store these patterns the connections between units are set to a negative value so that active units tend to deactivate other units, and the unit that is active in a pattern has a positive connection from the bias unit so that if it is active it remains active. This easy solution for storing N patterns is a result of each pattern only having a small amount of information. Because of the symmetry in binary activation, patterns with all but one unit active contain the same amount of information as patterns with only a single unit active.

Amit *et al.* (1987b) argue that Hebbian learning can be altered to accommodate correlated patterns where the correlation is caused by a low coding ratio (only a small percentage of the units are active). The learning rule can be changed to:

$$\delta w_{ij} = (\psi_i - \alpha)(\psi_j - \alpha) \tag{4.1}$$

where α is the sparseness of the coding. This concept of correlation caused by a uniform decrease in the number of units that are active is useful only when the patterns have a known activity level and that activity level is the *only* correlation between the learnt patterns. The patterns themselves need to be uncorrelated in terms of which units are active for this approach to work.

Correlation does not come from just a change in the coding ratio for the patterns. One interpretation for correlated patterns is the concept of a prototype. The prototypical car will have many features in common with individual cars. A learning set that contains groups of similar objects will have correlated patterns and the combination of these correlations may be the prototype for the group.

Hopfield networks with Hebbian learning cannot make highly correlated pattern stable. Instead a pattern which is the combination of the learnt patterns will become

a stable state with a large basin of attraction. There are two types of correlation, one caused by a low coding ratio where the patterns are still independent, and correlation caused by patterns which share a large number of features. The difference can be most clearly demonstrated with an example.

In this example the learning population is generated by making small perturbations of a checkerboard pattern (Figure 4.3). These patterns have a coding ratio of about 50% but are also extremely highly correlated. This form of correlation immediately destroys the basin of attraction of the individual patterns and the only attractor is the correlated prototype, in this case a checkerboard pattern. To show this, consider three patterns that are based on a checkerboard pattern with k units altered ($k << N$) with alteration ratio $(\alpha_a) = k/N$. Having learnt three patterns there are four catagories of units representing the different types of overlap of the perturbed checkerboard patterns. The average number of units in each set (cardinality C), can be calcuated from the alteration ration α_a

1. The set of units altered in all three patterns.
   $\Phi_3^3 = \phi_{1\wedge 2\wedge 3}$,
   $C = \alpha_a^3$
2. Units altered in two of the three patterns.
   $\Phi_2^3 = \phi_{1\wedge 2} + \phi_{2\wedge 3} + \phi_{1\wedge 3} - \phi_{1\wedge 2\wedge 3}$,
   $C = 3(\alpha_a^2 \times (1 - \alpha_a))$
3. The set of units altered in just one pattern.
   $\Phi_1^3 = (\phi_1 + \phi_2 + \phi_3) - (\phi_{1\wedge 2} + \phi_{2\wedge 3} + \phi_{1\wedge 3} + \phi_{1\wedge 2\wedge 3})$,
   $C = 3(\alpha_a \times (1 - \alpha_a)^2)$
4. The set of units that conform to the checker board in every learnt pattern.
   $\Phi_0^3 = N - \phi_{1\vee 2\vee 3}$,
   $C = (1 - \alpha_a)^3$

where $\phi_{1\wedge 2\wedge 3}$ is the set of units whose activation was altered in all the patterns 1, 2 and 3; $\phi_{1\wedge 2}$ is the set of units altered in patterns 1 and 2; ϕ_1 is the set of units altered in patterns 1; and $\phi_{1\vee 2\vee 3}$ is the set of units altered in pattern 1, 2 or 3.

If the altered units are selected from an i.d.d. at random then Φ_3^3 would have cardinality $\approx \alpha_a^3 N$. Given small k, the number of units altered in more than one pattern is very small. A unit i selected at random from the set of units in Φ_1^3 will have connections to units j in Φ_0^3, each with a weight $w_{ij} = 1$ giving a net input to the unit of $N((1 - \alpha_a)^3)$. Given that the units in the other categories have at most $w_{ij} = \pm 3$ so long as $\alpha_a < 0.14$ for three patterns and $\alpha_a < 0.22$ for four patterns, the unit will

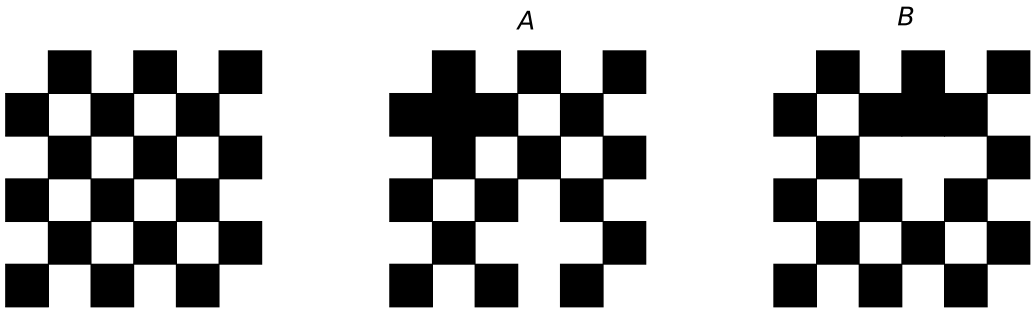


Figure 4.3: Checker board with two possible perturbations A and B.

change state to match the checker board. The more patterns which are learnt from the set of patterns with minor alterations to the checker board pattern, the larger the prototype's basin of attraction becomes.

If the units in the pattern are slowly "fatigued", where their input is lowered by a set amount, after several iterations individual patterns emerge from the prototype. This is because the units that were altered in the learnt patterns are closer to threshold than the units that are only in the checker board pattern. The major problem with this is that the patterns generated by this process of fatiguing the units are not just the learnt patterns, but include many spurious patterns.

4.3 Catastrophic Plasticity

Catastrophic forgetting in Hopfield networks with Hebbian learning is mainly attributable to capacity and correlations. However by using delta learning both the catastrophic correlation and catastrophic capacity problems are reduced. Delta learning deals with correlation by only making changes that improve performance, and the capacity is increased dramatically to close to the theoretical information capacity of the network.

However delta learning in Hopfield networks also suffers from excessive plasticity. Delta learning is able to learn a larger number of patterns by slowly adjusting the weights until they make the current set of patterns stable. Thus when learning new patterns, all the weights are changed to match the new patterns. This plasticity, the key to the power of the learning algorithm, can cause catastrophic forgetting.

As discussed in Robins and McCallum (1998) there is a great deal of similarity between the CF caused by excessive plasticity in MLP networks and in Hopfield networks using delta learning. Figure 4.4 shows the number of base population patterns that

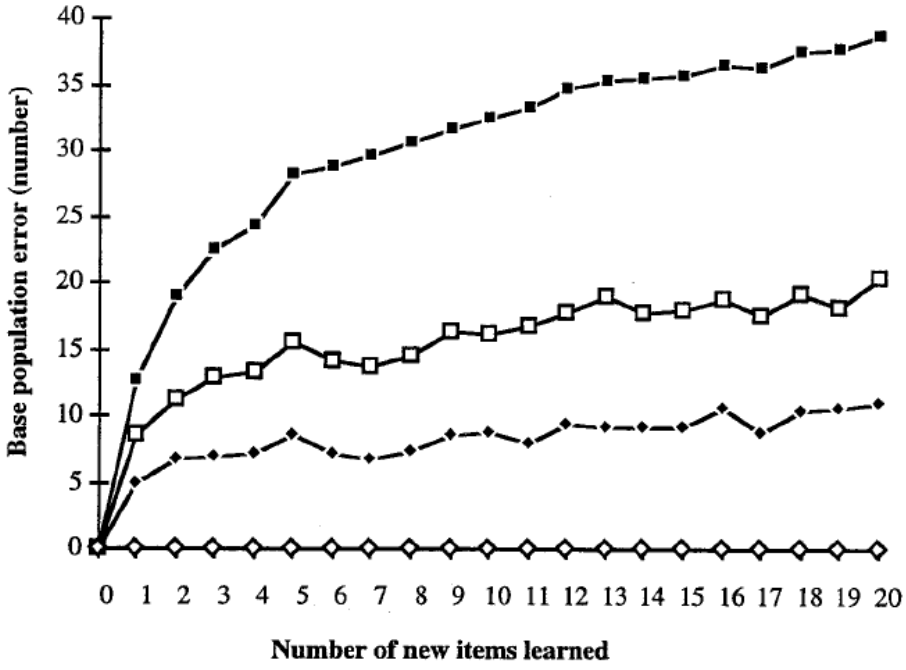


Figure 4.4: Catastrophic forgetting in a Hopfield network caused by plasticity (adapted from Robins and McCallum (1998) Figure 4).

become unstable as new patterns are learnt. It is this type of CF that is the target of the pseudorehearsal solution presented below in Section 4.5.

4.4 Existing Solutions

Catastrophic forgetting does not have a single cause, and although the various solutions tend to focus on a particular cause, they usually have an affect on all of the causes. For example, the unlearning solution (see Section 4.4.4 was initially designed to solve the problem of the limited capacity of Hopfield networks, and was later analysed to show that it was also affecting the problem of catastrophic correlation (Christos, 1996). The complexity in the interactions between the solutions and the various types of problems makes it difficult to directly compare solutions. Each solution makes trade-offs between plasticity and stability, complexity and robustness, and biological plausibility.

One method for dissecting these solutions is to organise them by the level at which they operate. The levels that we define are:

- Synaptic Level
- Neuron Level

- System Level

In the rest of this chapter we review the existing solutions to CF and place them into this taxonomy, as well as presenting our own pseudorehearsal solution. The goal of breaking the solutions into these categories is to establish a comparison matrix so that solutions can be compared by complexity and level rather than just raw performance. A system that requires a much more elaborate control mechanism must perform significantly better than the simplest solution to be considered an improvement. The performance of some system level solutions can be mainly attributed to the effect of an associated synaptic level alteration. By analysing the level at which a solution operates we can use a combination of performance and complexity to compare results.

4.4.1 Synapse Level

The synaptic level solutions focus on changing the way in which synaptic updates are performed given local information. In this section we include systems that either alter the way in which the local update rule is applied to the synapses, or use a post–processing phase to alter the weight matrix after a pattern has been learnt.

Weight decay

Weight decay is one of the simplest post–processing operations to implement. After each item, or set of items, are learnt every weight in the network is multiplied by a value just less than 1. This reduces the magnitude of each weight, moving it closer to 0. As Hebbian learning alters the weights by a fixed amount, it is easier to learn a new pattern when the weights are small. There is also biological evidence of a decay mechanism which reduces and even eliminates connections over time. There appears to be two types of LTP(long term potentiation), decremental and persistent (Abraham, 2003). Hebbian learning with weight decay would be equivalent to decremental LTP.

This can be considered a form of managed forgetting, where the forgetting of previous information is included in the learning algorithm. Using weight decay it is possible to calculate the influence on each weight that an old learnt pattern will have, directly from the number of patterns that have been learnt since it was presented. With weight decay of $d = 0.1$ after 10 additional patterns have been learnt, the contribution of the original pattern will be $0.9^{10} = 0.35$ of that pattern's original contribution. This controlled forgetting allows weight decay to avoid CF.

Christos (1996) showed that weight decay is able to reduce the CF caused by over-

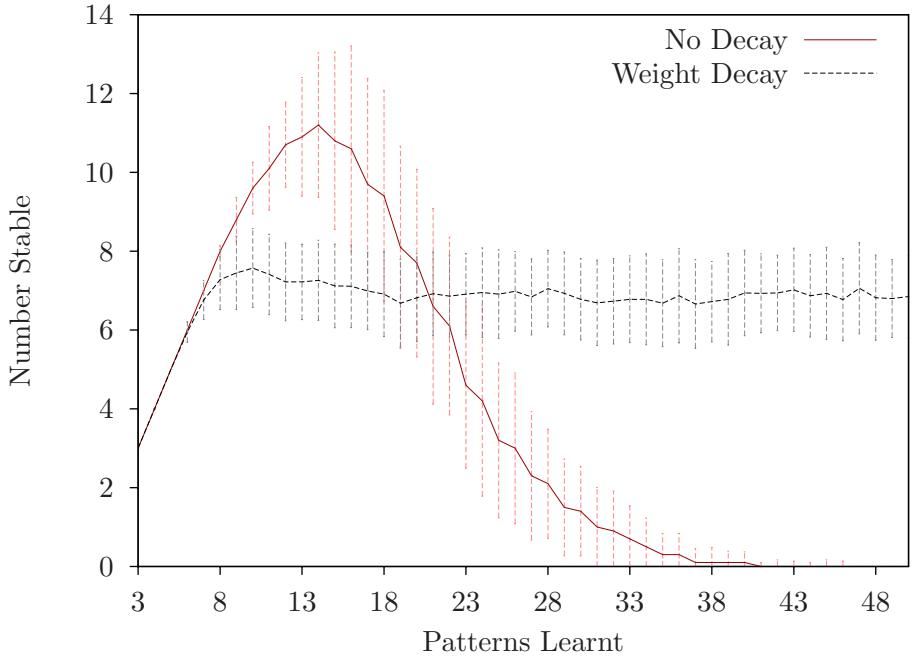


Figure 4.5: Number of learnt patterns stable in a 100 unit Hopfield network $\mathcal{H}_{100,\pm}$ with Hebbian learning and weight decay $d = 0.1$ per item (averaged over 100 repetitions, with error–bars showing standard deviation).

loading the network, at the cost of lowering the effective capacity to $0.05N$. Figure 4.5 shows our comparison between normal Hebbian learning and learning with a weight decay of $d = 0.1$. The weights are each new pattern are altered so that $(w_{ij} = w_{ij}(1-d))$ (see Appendix B.1 for other parameters used for this simulation). The CF of the standard learning system can be clearly seen when the network learns more than $\approx 0.14N$. However, using weight decay the system is able to retain about $0.07N$, slightly above the estimate of $0.05N$ by Nadal *et al.* (1986). The patterns that are stable in this network are the most recently stored patterns. Although stable, the basin of attraction of the oldest stable pattern is very small. Most of the random probes of the network fall within the basin of just the most recently learnt pattern. Figure 4.6 shows the probability with which a pattern is stable after learning multiple patterns using Hebbian learning and weight decay of $d = 0.1$. This graph shows that after learning 20 patterns, the 15th pattern that was learnt was stable in about 70% of the 100 trials. The fall off from 100% stable for the last two or three patterns learnt to never stable is similar for each of the three conditions shown.

As noted by Christos (1996) it may seem better to learn $0.14N$ patterns and then

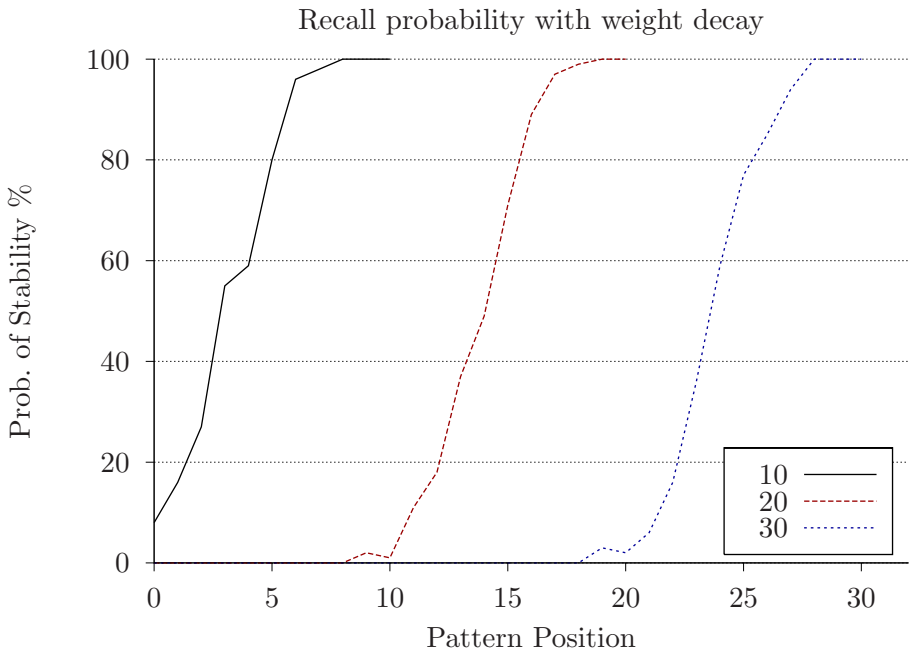


Figure 4.6: Probability of a learnt pattern being stable based on when it was learnt. Probabilities are shown after 10, 20, and 30 patterns have been learnt in an $\mathcal{H}_{100,\pm}$ network with Hebbian learning and weight decay of $d = 0.1$ per item (100 repetitions).

wipe the network clean and learn the next set of $0.14N$ patterns. This "manages" the forgetting by removing all the patterns and working with a clean network to learn new information. This does allow new patterns to be learnt continually over time, but at the cost of catastrophic removal of all previously learnt information every $0.14N$ patterns. Christos suggests that the reason that this is not a viable solution is the lack of a biological analogy, but there is also the problem that when averaged after each pattern is learnt the number of patterns stored is only $0.07N$.

Weight decay provides a simple and successful solution to the problem of CF. This success however results in a network with a very low *recognition capacity*, and an even smaller *recall capacity*. When probed, the network almost always returns only the most recent pattern, and the older patterns fade very quickly. However weight decay will be important in analysing the behaviour of other solutions as it provides not only a base performance level, but also appears be the main cause for the improved performance of other systems such as unlearning, a possibility discussed later in this chapter.

Weight capping

Willshaw, Buneman, and Longuet-Higgins (1969) showed that perceptron networks with only three possible connection values; -1, 0, +1 were able to solve a variety of problems. Surprisingly when a Hopfield network has the weights capped to a low value, it ameliorates some of the problems of CF. Figure 4.7 shows the result of five different capping values. With a learning constant of $\eta = 1$ the condition labelled "1" matches the Willshaw model capped at 1 and -1. In an $\mathcal{H}_{100,\pm}$ the best results are achieved with a capping value of 3 (7 different weight values). The performance of weight capping is only slightly below that of weight decay. As with weight decay, weight capping can be justified from a biological perspective. The cap represents a constraint on the maximum efficacy of a synapse between two neurons.

The similarity between weight decay and weight capping may not be obvious at first, but when inspecting the weight changes it becomes clearer. The effect of weight decay is to bring the weights closer to zero which allows the Hebbian learning rule to make a large enough change to the network to make the new pattern stable. Using a very low weight cap the weights can never move far from zero, and so the large correlations that normally cause CF are removed. The combination of weight capping and decay does not provide a cumulative performance improvement as the two approaches solve the same problem in similar ways. Both of these systems perform well, and in combination with their biological plausibility, they provide a solid lower bound to assess the more

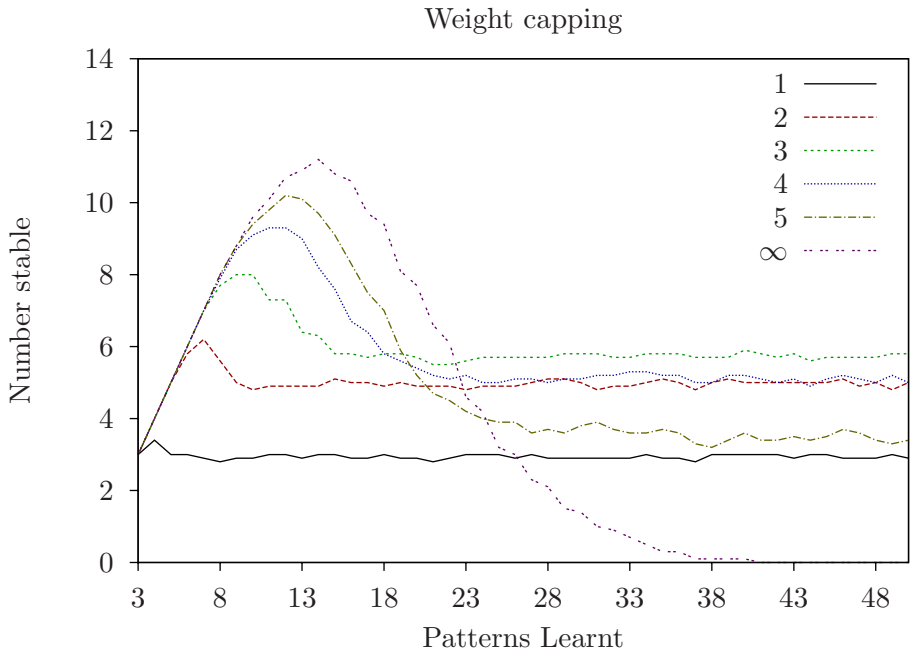


Figure 4.7: Number of learnt patterns stable in a 100 unit Hopfield network $\mathcal{H}_{100,\pm}$ with Hebbian learning and weight capping values 1–5 (100 repetitions, N.B. error bars removed for readability).

complex solutions.

Learning rule alteration

There are many alterations to the basic Hebbian learning rule to try to make it either more biologically plausible, or more effective. Dayan and Willshaw (1991) showed that for catastrophic correlation caused by a low coding ratio the best learning algorithm was the one suggested by Amit *et al.* (1987b) (see Equation: (4.1)). A similar result was achieved by Palm and Sommer (1996) by changing the activation values of the units so that instead of $(1, -1)$ or $(1, 0)$ the output was $(1, -a)$ where $a = \alpha/(1 - \alpha)$ (α is the coding ratio). Thus, with 50% of the units active, as is the case with most of the examples in this thesis, the activation values are $a = 0.5/(1 - 0.5) \rightarrow (1, -1)$. Both of these solutions solve the problem of catastrophic correlation for patterns with low coding ratio. They do not however solve CF with still limits Hebbian learning.

4.4.2 Neuron Level

In this section we present solutions that alter the network based on the behaviour of each unit. Methods at this level use the inputs or outputs of a single unit as the focus of manipulation. These approaches use information that is entirely local to the unit and so can be processed without additional inputs from other units. This allows these unit level approaches to remain very plausible as an explanation for the high level of memory performance found in biological networks.

Normalisation of unit input

Chechik *et al.* (2001) describe a mechanism for normalising the inputs to each unit so that the sum of all the connections is zero. The problem that Chechik *et al.* (2001) were focused on was the issue of catastrophic correlation when learning patterns with low coding ratios. The large number of inactive units tends to overwhelm any individual active units and the patterns become difficult to learn.

The contribution of Chechik *et al.* (2001) was to show that a local neuron level alteration could achieve a similar result to the learning rule change suggest by Amit *et al.* (1987b)(see Equation:(4.1)) without having to pre-calculate the coding ratio. This normalisation process equalises the weight changes so that the simple Hebbian learning rule is still able to learn patterns which have a low coding ratio. This form of zero sum normalisation does not alter the performance of Hebbian learning in the $\alpha = 50\%$ space and so CF caused by capacity is still a problem, see Figure 4.8. With the low α patterns this zero sum normalisation returns the performance to that of the learning rule in Equation (4.1).

We have used the idea of unit input normalisation and altered it slightly to prevent CF caused by learning more than the normal $0.14N$ patterns. Instead of forcing a zero sum on the inputs, we use the absolute value of the input weights and constrain the total input to the unit. This is equivalent to having a limited pool of resources to allocate to either positive or negative connections. This allows units to have more negative or positive connections, but constrains the absolute magnitude of the input. Figure 4.9 shows the performance of this resource normalisation process on a network as it learns items one after another.

The process for unit resource normalisation is:

- for each item to learn
    - set the network to the input pattern

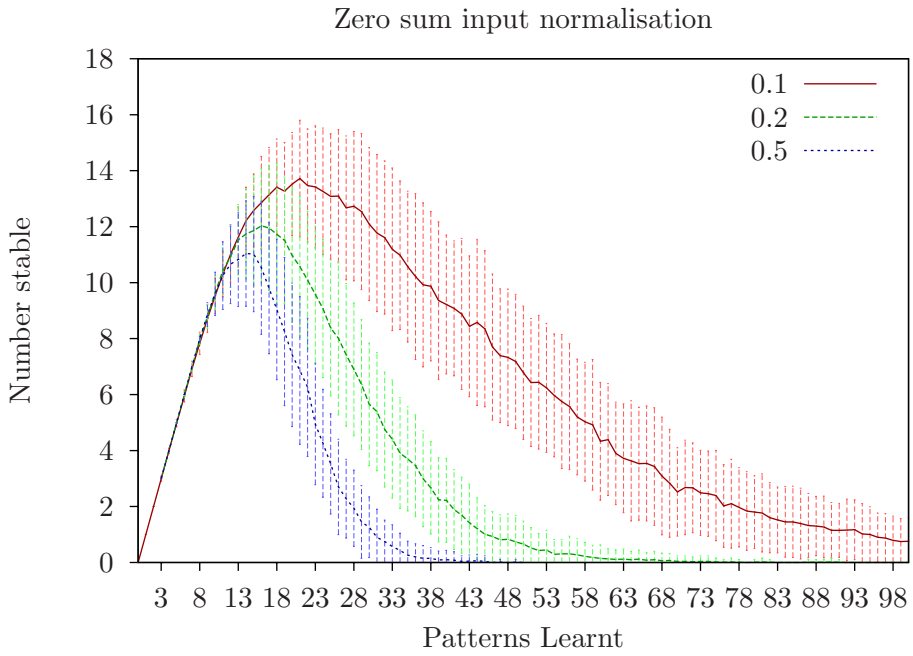


Figure 4.8: Number of learnt patterns stable in a 100 unit Hopfield network $\mathcal{H}_{100,\pm}$ with Hebbian learning and zero sum normalisation for coding ratios of $\alpha = 0.1, 0.2, 0.5$ (100 repetitions).

- apply Hebbian update to all of the connections in the network
- for each unit
    - calculate the total absolute value of the weights connected to the unit
    - if the total is greater than the resource limit, multiply the weights by the ((resource limit)/sum) so that the resources used remains at or below the resource limit

This normalisation process is able to alleviate forgetting caused by catastrophic capacity, but the performance is again close to the $0.07N$ performance of weight decay and weight capping. Part of the reason for the similarity is that weights cannot continue to grow unconstrained. Although individual weights can grow much larger than in the weight decay or weight capping situation, the size of the weights is limited by the normalisation process. If the patterns have relatively low correlation the normalisation process will work in a similar way to weight decay, as weights will be lowered by about the same amount and in a similar way to weight decay. This normalisation process was tested with various resource limits with the best performance in the 100 unit network at around $2N$. This value may alter slightly with larger networks ($2.5N$ was better in

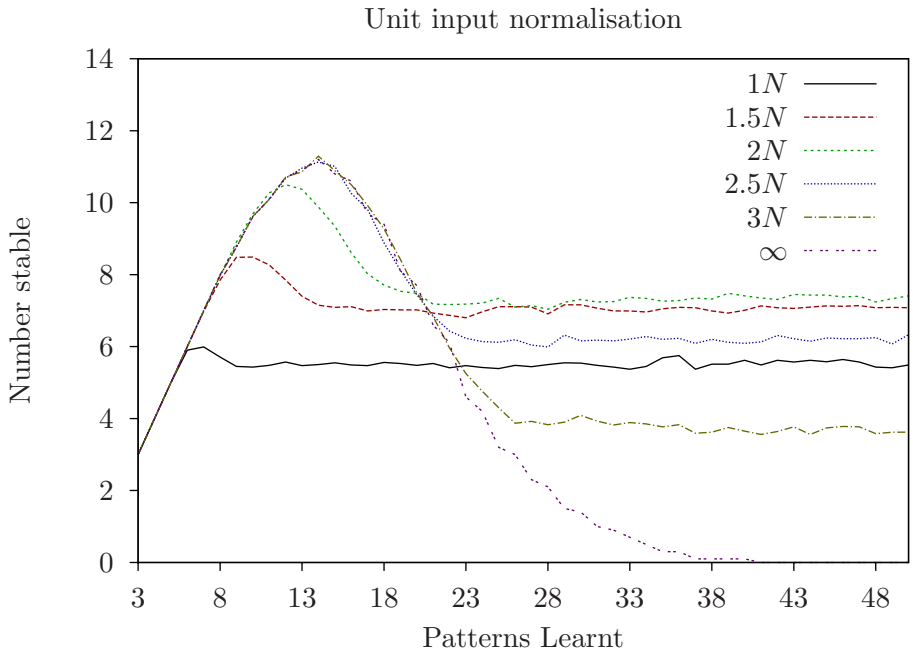


Figure 4.9: Number of learnt patterns stable in a 100 unit Hopfield network $\mathcal{H}_{100,\pm}$ with Hebbian learning and unit resource normalisation of 1.5N–3N (100 repetitions, N.B. error bars removed for readability).

a 500 unit network) but the importance of this result is that the performance of the network is similar to the $0.07N$ in line with weight capping and decay.

Although the normalisation process performs at a similar level to weight decay, the alteration means that the normalisation process no longer solves the catastrophic correlation problem associated with sparsely coded patterns (patterns with a low coding ratio). Figure 4.10 shows the performance of resource normalisation ($2N$) with varying levels of sparsely coded patterns. Note that with 0.1 (10%) the network is unable to learn even three patterns.

The combination of the two types of normalisation, zero sum and resource limiting, solves both problems as shown in Figure 4.11. The figure also shows the resource normalisation alone "RN" cannot learn more that two patterns with this low coding ratio. The reason for this is the Hebbian learning cannot normally learn more that two of these highly correlated patterns. Resource normalisation only affects the weights when they become large, and so does not prevent catastrophic correlation. The network is able to perform at about the $0.07N$ level. Thus, although the algorithm does not perform significantly better than either of the two synaptic level solutions, it provides another point of comparison and is now able to work with various coding ratios.

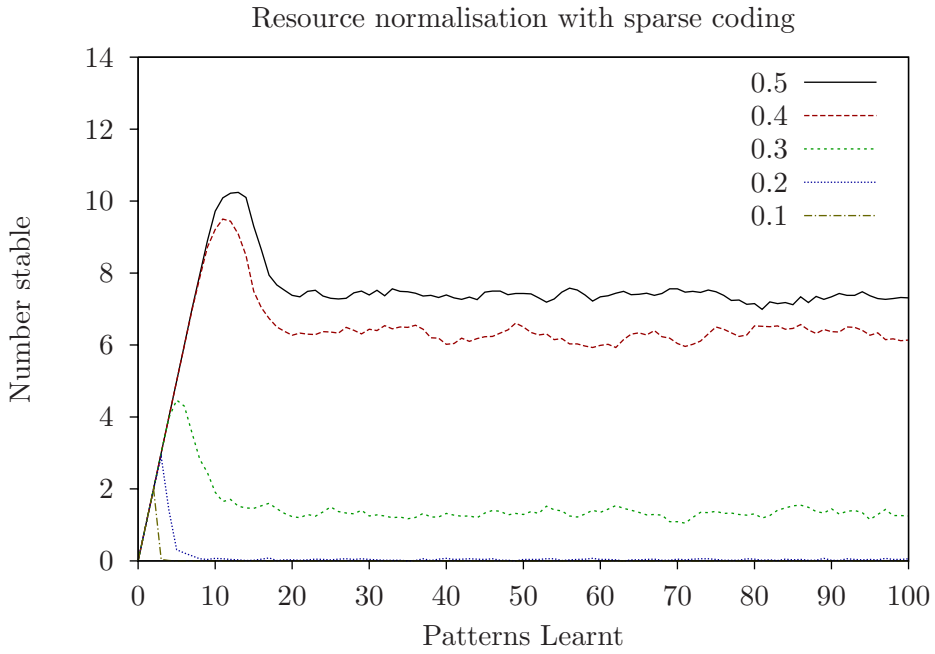


Figure 4.10: Number of learnt patterns stable in a 100 unit Hopfield network $\mathcal{H}_{100,\pm}$ with Hebbian learning on patterns with various coding ratios 0.1 – 0.5 (100 repetitions, N.B. error bars removed for readability).

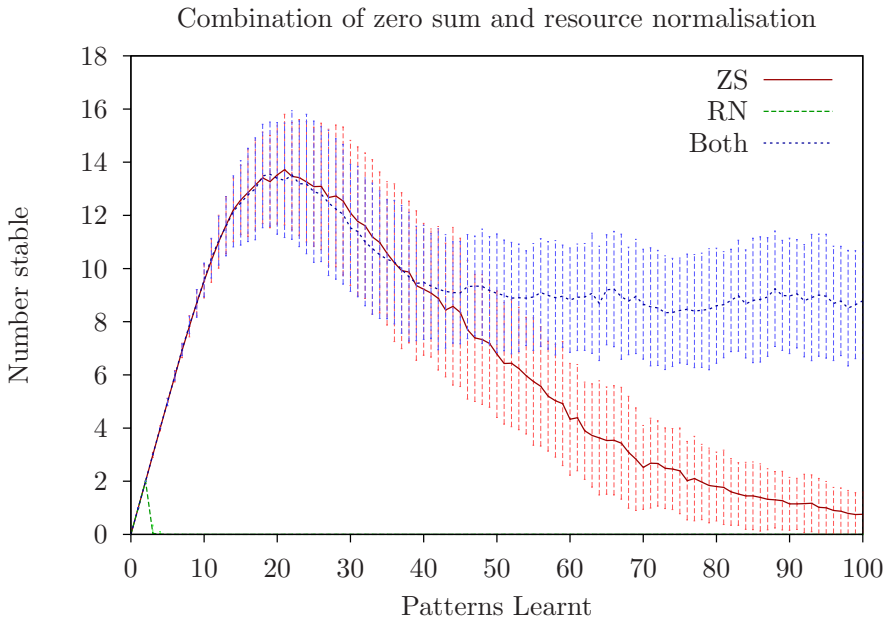


Figure 4.11: Number of learnt patterns stable in a 100 unit Hopfield network $\mathcal{H}_{100,\pm}$ with Hebbian learning on patterns with a coding ratio of 0.1. The conditions are: zero sum normalisation "ZS", resource normalisation "RN" (normalisation value of $2N$ only visible in the transition from two to three patterns learnt), or both unit resource normalisation and zero sum normalisation "Both" (100 repetitions).

We have also shown that regulation of the weights can occur at either the unit level, as above, or over the entire network. Regulation over the whole network is similar to the above process except that the resource limiting is applied to the total of all of the weighted connections in the entire network, and applied to all weights in one step:

- for each item to learn
    - set the network to the input pattern
    - apply Hebbian update to all of the connections in the network
    - calculate the total absolute value of all weights in the network
    - if the total is greater than the resource limit, multiply all the weights by the (resource limit/sum) so that the resources used remains at or below the resource limit

This normalisation forces the weights in the network to compete for resources, reallocating resources to the new patterns. Each time a new pattern is learnt, the weighted connections associated with that pattern are updated and if the network requires more resources for that, it takes it from all the weights equally. This decreases the influence of old patterns on the current weight configuration allowing new patterns to continue to be learnt over time. Figure 4.12 shows the performance of normalisation applied to the total absolute value of the weights within the network.

The results are remarkably similar to the normalisation process applied to each individual unit. Thus, this process is not significantly better than the more local unit by unit normalisation, and indeed the performance of the best resource limit of $2N$ is similar to the very simple process of weight decay.

Neuronal regulation

Horn, Levy, and Ruppin (1998a) developed a system to regulate the connections to units in terms of the unit's activation in the learnt patterns. Like the normalisation process above, this system is designed to improve the performance with sparsely coded patterns. The principle is to change the weights that contribute the most strongly to a set upper bound, and to set the weights below a threshold to zero. This simplifies the possible values of weighted connections, and removes small amounts of noise. In our simulations this process improved performance on sparsely coded networks but not significantly more than other methods. The main justification for using this model is the biological plausibility of the modelling and the improvement in the basins of attraction of the learnt patterns.

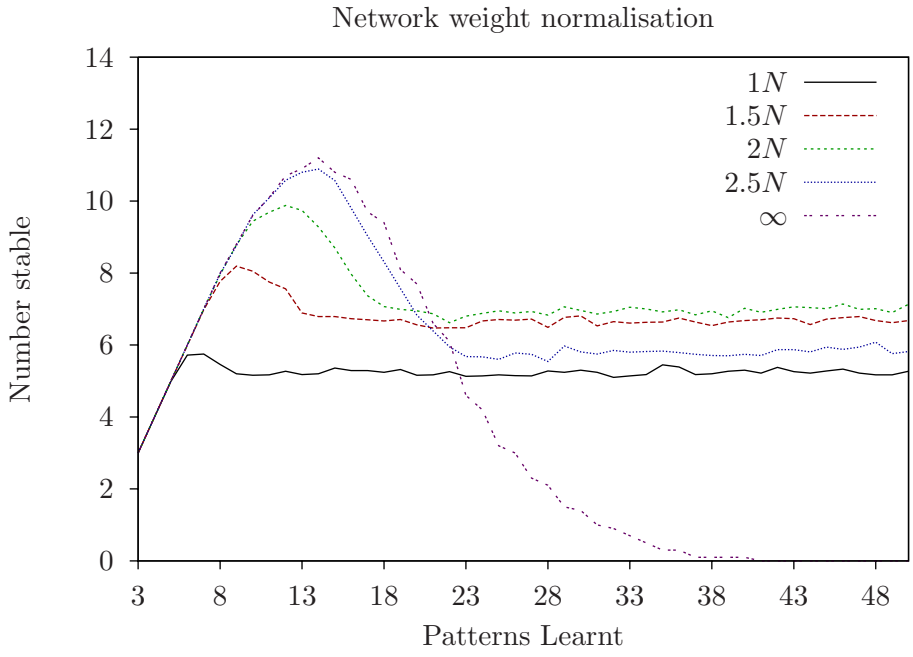


Figure 4.12: Number of learnt patterns stable in a 100 unit Hopfield network $\mathcal{H}_{100,\pm}$ with Hebbian learning and weight normalisation of the network from 1N–2.5N (100 repetitions, N.B. error bars removed for readability).

4.4.3 System Level

The solutions to the problems of CF presented so far have been focused on small parts of the network. These solutions provide a baseline for the performance of more elaborate solutions. The two system level solutions which we investigate are unlearning (Crick and Mitchison, 1983; Hopfield *et al.*, 1983; van Hemmen, 1997; Christos, 1996) and Pseudorehearsal (Robins and McCallum, 1998, 1999; French, 1999; Ans, Rousset, French, and Musca, 2004).

4.4.4 Unlearning

The unlearning solution for CF was presented by Hopfield *et al.* (1983) within a year of his presentation of the network architecture. This solution to CF sparked interest in the general memory community including Crick and Mitchison (1983). One of the features of Hopfield networks that was identified early on was that the state space becomes dominated by spurious patterns, and occasionally a single large spurious pattern. The overwhelming almost cancerous nature of these spurious patterns leads to the idea that by removing them the network may be able to recover some of its original performance. Hopfield *et al.* (1983) first described unlearning as applied to a static learning problem. The performance of the network did indeed improve greatly when the large spurious patterns were removed.

Hopfield *et al.* (1983) explained the action of unlearning in terms of its effect on the energy surface of the network. When a pattern is learnt it creates a well, or basin of attraction, in the energy surface. When two basins are close to one another the overlap forms a new, larger basin of attraction between the original patterns. The information about which patterns were stored is still buried in the network, but it is now masked by the large spurious basin. To recover the original patterns, this large basin must be reduced to the point where the stored patterns become stable and have their own basins.

The procedure for unlearning is:

- learn patterns as normal with Hebbian learning
- for a set number of runs
    - probe the network by setting the state of the network to a random configuration and letting it relax to the nearest stable state
    - apply Hebbian learning with a small negative constant $-\eta$. For each connection $\delta w_{ij} = -\eta \psi_i \psi_j$

- test the network on the original population

This process is able to recover patterns which had become unstable in the network, and it also increases the basin of attraction of stable learnt patterns, so long as they were not the patterns that were found by random probing.

Crick and Mitchison (1983) suggested that the random probing and unlearning process might be connected to the random pulses that cascade through the brain during R.E.M. sleep. This suggestion raised a great deal of interest in this model not only as a model of associative memory, but also as a model that might explain the mystery of dreams. The model presented suggested that during the day items were learnt, and then at night the spurious patterns were found by random probing, equated with dreams, and then removed, allowing the real memories to be available the next day. This was a very appealing model as it explained why dreams can seem so illogical and made up of strange combinations of daily events.

The model presented by Hopfield was effective in the static case where a set of patterns was learnt and then the unlearning was applied. But as was pointed out when discussing pseudorehearsal in MLP networks, the world does not present all the information to be learnt in a single coherent package. Learning occurs over time. The unlearning account was extended to cover sequential learning by Wimbauer, Klemmer, and van Hemmen (1994), Christos (1996), and van Hemmen (1997).

When applying the unlearning process to a network there are many parameters to consider. Two of the most important are the number of unlearning cycles to apply, and when to apply them. If the amount of unlearning performed in a cycle is too great, the network will decrease the connection weights until it no longer has any stored information. The original proposal of unlearning in the static model of Hopfield *et al.* (1983) was to apply unlearning at the end of the learning cycle. We are interested in the performance of the network after many learning/unlearning cycles with new patterns presented in each learning cycle. The two possibilities for applying unlearning are to apply unlearning after each new item is learnt (Christos, 1996), or as a restorative process after a block of patterns have been learnt, where each block is considered one learning cycle. (van Hemmen, 1997).

4.4.5 Unlearning After Each Pattern

Christos (1996) proposed an unlearning system for the sequential learning problem in which the unlearning is applied after each learnt pattern. The process is:

- learn a small base population of items
- perform unlearning of (U base population) items, each of which require
    - probing the network with a random pattern and relaxing to find a stable state
    - performing unlearning on the pattern by applying Hebbian learning with a learning constant of $-\eta$
- for each new pattern to learn
    - learn the new item using Hebbian learning
    - perform unlearning of U new probes with unlearning constant $-\eta$

where U is the number of probes to unlearn after each learnt pattern is presented. The number of unlearning probes is larger after the base population is learnt so that there is still the same number of unlearning items per learnt pattern. U and $-\eta$ are usually altered together so that the amount of unlearning per pattern is close to the amount of learning performed. With the learning constant $\eta = 1$ and $U = 50$, the unlearning constant is set to $-\eta = -0.01$.

This process is able to alleviate the problems of CF when the network is presented with more than $0.14N$ patterns, as can be seen by comparing the results from our replications in Table 4.1 and Table 4.2. After learning, the network is probed with 2000 random inputs, each of which are relaxed to a stable state. This measure tests the size of the basins of attractions of the learnt patterns as a percentage of the network state space. The percentage of the space that relaxes to a particular learnt pattern (columns) versus an non–learnt/spurious state (the bottom row) gives a measure of the recall capacity, the performance of the network as a memory recall system. Without unlearning, the basins of attraction of the learnt patterns decrease quickly as more and more patterns are added. After about $0.2N$ patterns, the network will almost always relax to spurious states. With unlearning, the percentage of learnt patterns recalled is much higher, and the percentage of spurious states found correspondingly lower. The values 0^* indicate learnt states that are stable when probed with exact duplicates of the original pattern, but were not found by the probing process. This is useful as it shows that within 2000 probes most of the stable patterns have been found.

Figure 4.13 shows the number of patterns that are stable over time, both before and after unlearning is applied. The separation is caused by the unlearning process restoring some of the patterns that had become unstable. Figure 4.14 shows the direct comparison between Christos' unlearning and weight decay. Unlearning performs slightly

	Total number of stored patterns																				
Pattern	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	28.10	20.50	17.25	11.05	5.35	4.15	5.80	2.85												
2	29.05	18.45	11.45	9.90	7.20	4.90	2.30	1.75	1.85	0.80	0.45	0*	0.20	0.05	0*	0*				
3	25.60	18.85	15.45	11.35	8.00	5.60	8.65	6.45	5.40	3.45										
4		18.45	12.85	9.05	7.30	6.30	5.50	3.40	2.80	1.85	1.15	0.80	1.20	0.45	0.95		0.15			
5			14.45	12.80	10.60	8.25	7.80	5.00	4.50	2.80	1.85	0.55								
6				8.10	7.75	4.30	2.70	2.20	2.00	1.85	0.85	0.35	0.30	0.50	0.45	0.55	0.05	0.05	0*	
7					7.25	4.10	2.80	1.70	0.90	0.60	0.25	0.10	0*	0*	0*	0*	0*	0*		
8						9.95	6.05	4.25	3.90	3.30	1.20	1.30	0.55	0.70	0.20	0.20	0.05	0.10	0.05	
9							8.45	6.45	3.95	2.85	2.05	1.60	0.60	0.90	0.25	0.75				
10								1.95	1.00	0.65	0.35	0.05								
(A) 11									5.80	3.90	2.50	1.75	0.65	0.60	0.15	0.10	0.10			
12										1.30		0.05								
13																				
14												1.55	0.60	0*	0.15	0.05	0*	0*		
15																				
16															0.25	0.30		0.10	0.15	0*
17															0.25		0*	0*		
18																0*	0*	0*		
19																	0*	0*		
20																				
21																				
22																				
Spurious	17.25	23.75	28.55	37.75	46.55	52.45	49.95	64.00	67.90	76.65	89.35	91.90	95.90	96.80	97.35	98.05	99.65	99.75	99.80	100

Table 4.1: Percentage retrieval for individual patterns in a $\mathcal{H}_{100,\pm}$ network having learnt patterns until it is overloaded, with no unlearning. 2000 probes used to generate percentages with 0^* indicating patterns that were stable but were not found during sampling.

		Total number of stored patterns																			
	Pattern	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	1	40.70	13.20	4.25	2.80	1.20	0.95	0.15	0.15	0*	0.05										
	2	28.85	9.95	3.15	0.75	0.45	0.15										0*				
	3	24.60	9.75	3.80	1.75	1.15	0.40	0.15													
	4		52.75	28.70	13.90	8.90	8.35	4.95	3.00	2.95	1.60	1.40	0.95	0.75	0.20	0.15	0.05	0.10	0.10		
	5			48.70	12.70	7.40	6.55	1.60	0.50	0.15	0.05	0*	0*								
	6				58.40	22.35	4.40	1.05	1.35	1.65	0.45	0.10	0.10	0.05							
	7					42.90	27.40	14.25	8.05	8.20	1.65	2.25	1.05								
	8						34.30	12.15	8.65	4.90	1.40	1.65	0.70	0.10					0*	0*	0*
	9							60.15	18.50	7.00	2.45	0.80	0.50	0.35	0.10	0.10	0*	0*		0*	
	10								38.45	11.45	4.40	2.90	2.00	1.35	0.75	0.35	0*	0*		0*	0.05
(B)	11									42.00	10.10	8.15	3.25	2.55	1.50	1.10	1.15	0.75	0.05	0.05	0.05
	12										4.65	6.35	2.80	0.40	1.85	0.55	0*	0*	0.30	0.15	0.05
	13											33.75	11.30	10.00	5.90	3.70	1.10	1.00	0.45	0.50	
	14												24.85	8.90	3.75	3.85	2.25	1.75	0.50	0.20	0.25
	15													19.20	4.85	0.15				0*	0.05
	16														25.85	6.30		3.65	1.95	1.20	0.60
	17															32.65	11.90	5.45	3.20	3.60	1.70
	18																18.40	10.25	4.65	2.25	1.15
	19																	28.45	13.50	5.05	2.45
	20																		25.70	9.00	0.80
	21																			19.95	7.05
	22																				16.75
	Spurious	5.85	14.35	11.40	9.70	15.65	17.50	5.55	21.35	21.70	73.20	42.65	52.50	56.35	55.25	51.10	65.15	48.60	49.60	58.05	69.05

Table 4.2: Percentage retrieval for individual patterns in a $\mathcal{H}_{100,\pm}$ network having learnt patterns until it is overloaded, with unlearning applied after each new item (unlearning 50 probes with an unlearning constant of $-\eta = -0.1$). 2000 probes used to generate percentages with 0^* patterns that were stable but were not found during sampling.

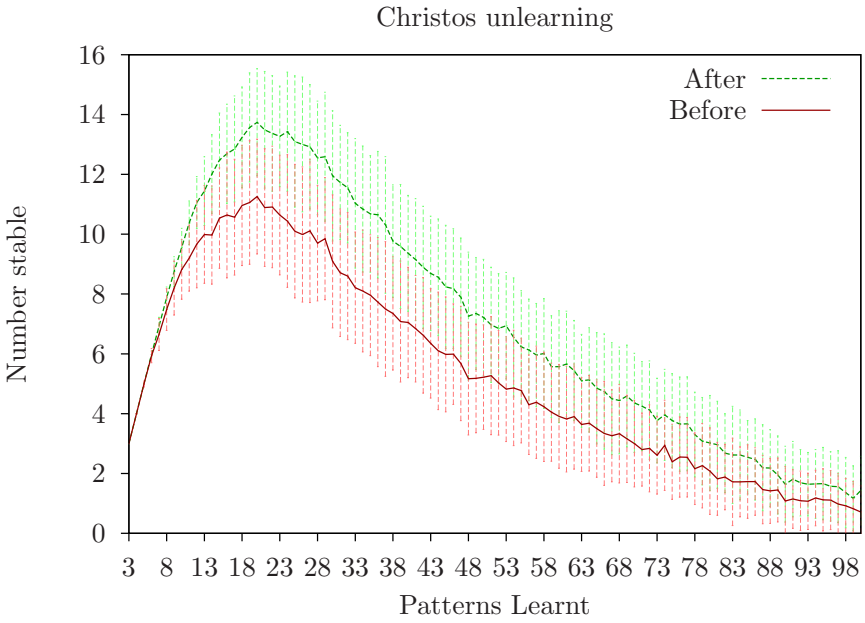


Figure 4.13: Number of learnt patterns stable in a 100 unit Hopfield network $\mathcal{H}_{100,\pm}$, with Christos unlearning applied in bursts of 50 patterns with unlearning constant $-\eta = -0.01$. 'Before' represents the number stable before unlearning occurs (100 repetitions).

better than weight decay initially, but the performance decreases as more patterns are added. Christos (1996) identifies that this form of unlearning is not significantly better than weight decay, but does not elaborate on the long term performance.

This form of unlearning has a similar effect on the weights as weight decay. With randomly distributed patterns, large weights are more likely to form part of a stable pattern than small weights. Thus, over many runs, the large weights receive a higher proportion of unlearning. This effectively decays the large weights until they are only participating at a similar level to all the other weights in the network.

A second effect which occurs in the serial learning environment is that the most recent learnt pattern has a large basin of attraction, which draws in most of the probes, and therefore, most of the unlearning. Christos pointed out that most of the unlearning effort goes into weakening the pattern that has just been learnt.

The trick with unlearning is to find the right amount of unlearning so that the spurious stable states are removed without destroying the learnt patterns. Too little and the weights grow over time, and the performance deteriorates until the network cannot store any patterns. Too much and all the learnt patterns are removed. Even when a balance is found, the long term (more than 50 patterns) performance of this

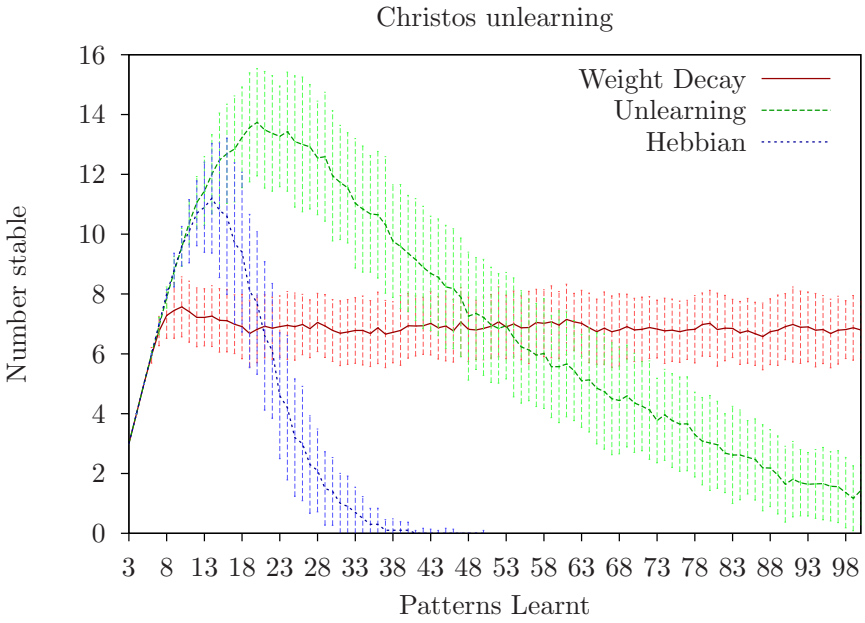


Figure 4.14: Number of learnt patterns stable in a 100 unit Hopfield network $\mathcal{H}_{100,\pm}$ with Christos unlearning applied in bursts of 50 patterns with unlearning constant $-\eta = -0.01$, compared with weight decay of 0.1 (100 repetitions).

form of unlearning is worse than simply applying weight decay.

4.4.6 Unlearning After Overloading

One of the problems with unlearning after each new pattern is that most of the unlearning is being directed at the most recent pattern. This problem is avoided when the network is overloaded with patterns, as the learnt patterns are no longer able to be found by random sampling as none of them are stable. Thus, applying unlearning after the network has been overloaded with many more patterns than the capacity of the network (learnt patterns $m >> 0.14N$) results in only spurious patterns being unlearnt. Van Hemmen(1997) proposed a system where $0.4N$ patterns are learnt and then unlearning is applied:

- break the learning population into groups of q patterns
- for each block of patterns
    - learn the q items with Hebbian learning. Where $q > 0.14N$ (e.g. $q = 0.4N$)
    - perform unlearning of U patterns at $-\eta$ so that $U = q/-\eta$. $(q = 40, -\eta = 0.01, U = 4000)$

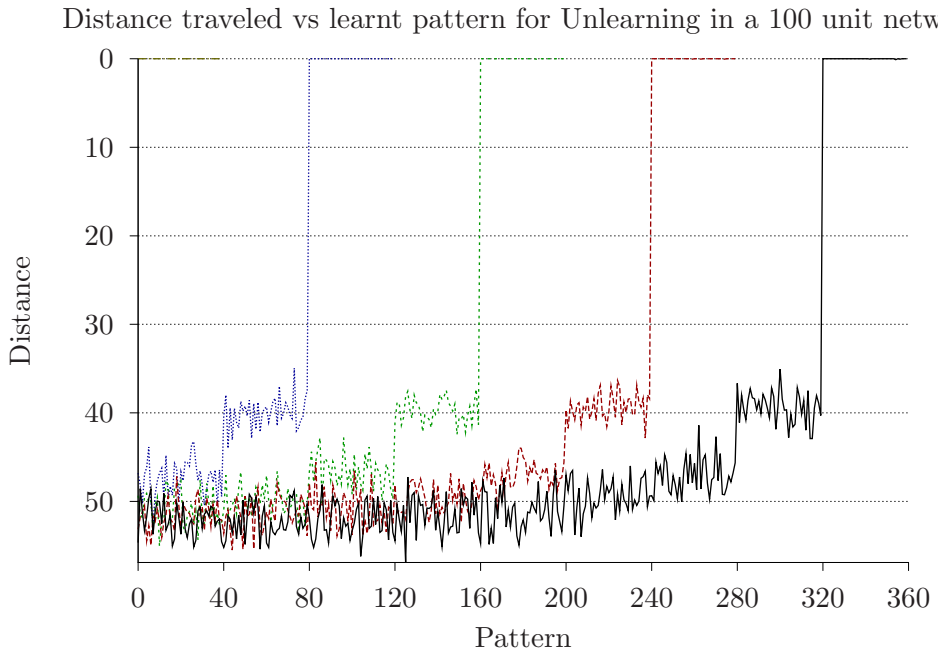


Figure 4.15: Van Hemmen unlearning showing the performance after 40, 120, 200, 280, and 360 patterns have been learnt in blocks of 40 patterns learnt, followed by unlearning of 500 probes at $-\eta$=-0.02 (100 repetitions).

This system was inspired by the sleep–wake cycle where learning occurs during the day and then a post–processing phase appears to occur during sleep.

The results of our simulation of this system are presented in Figure 4.15. The y axis is the distance a pattern travels before it finds a stable state. Van Hemmen uses this measure as he is not interested in developing a system where random probes result in learnt patterns, but in a system where probes that are very close to a learnt pattern relax to the pattern. This is a recognition rather than a recall system. The measure used is the Hamming distance (H) between the initialised learnt pattern and the state of the network after relaxation. If the distance is 0, then the learnt pattern is stable. Van Hemmen uses overlap m_f rather than Hamming distance, but they are equivalent as overlap $m_f = 1 - (H/N)$. In Figure 4.15 the last $0.4N$ patterns are stable while the previous $0.4N$ patterns, learnt in the previous "wake–dreaming" cycle, have a Hamming distance of $H \approx 0.4N = 40$ (overlap of $m_f = 0.6$).

Van Hemmen makes the argument that the higher the overlap the "better" a pattern has been remembered. As the number of patterns presented (n) increases he states that:

"... as n proceeds, not only the most recently stored Δq patterns but the

one but last bundle is remembered better and better."

This claim of improved performance based on a decrease in the distance between the nearest stable pattern and a learnt pattern does not follow. The distance travelled by the network as it relaxes is determined by the density of spurious stable states, not by how well remembered the pattern is.

To claim improved performance there must be something about the shorter distance that could be used to make a statement about the learnt patterns. The decreased distance is not so small that it can be used as a recognition test, as the numbers presented are an average and individual patterns vary greatly in the distances they travel.

The basins of attraction of the stable states generated using this form of unlearning are relatively small. Van Hemmen states that during the random probing of the network, the stable states (the attractors) found will almost always be spurious states. Table 4.3 shows some of the basin sizes for learnt patterns in an $\mathcal{H}_{100\pm}$ with van Hemmen unlearning applied. The minimum, mean and maximum are given for the patterns between 160–199 after the fifth cycle of learning $0.4N$ patterns and unlearning 500 patterns with unlearning $-\eta = -0.002$. Our simulations confirm that the learnt patterns are almost never found through random sampling of the network. Out of 1,000,000 random probes only 132 ($< 0.02\%$) relaxed to learnt patterns.

	Est. m_f
minimum	0.852
mean	0.869
maximum	0.883

Table 4.3: Learnt patterns 160–199 in a 100 unit network $\mathcal{H}_{100\pm}$ with van Hemmen unlearning.

Van Hemmen also provides an estimate of the amount of unlearning to apply in each unlearning (dream) cycle based on the amount of correlation in the patterns that have been learnt. If too much unlearning is applied then the whole system collapses. Too little and the performance degrades quickly after performing only a few cycles of learning and unlearning. To find the right level of unlearning the amount of correlation between the patterns needs to be known. Unlearning removes the spurious states generated by correlation, so the amount to remove is critically dependent on the

correlation. Van Hemmen's measure of pattern correlations is:

$$\mathcal{C}_{pq} = \frac{1}{N} \sum_{i=1}^{N} \psi_i^p \psi_i^q \tag{4.2}$$

where p and q are patterns in the learning set and ψ_i^p is the activation of unit i in pattern p. Van Hemmen uses an estimate of the correlation based on activation level assuming that the only correlation comes from the large number of inactive units. This does not actually solve the generalised problem of correlation for patterns that have correlations caused in other ways, for example the checker–board pattern in Section 4.2. When unknown amounts of correlation are present in the learning population it is not possible to use this estimate.

Unfortunately to calculate the actual amount of correlation $\mathcal{C}_{pq}$ involves every unit in all of the learnt patterns. This enormous amount of pre–processing is difficult to reconcile with van Hemmen's desire to create a biologically plausible model of memory. It also presents a problem in that the whole population of patterns has to be present so that the correlation between the patterns can be evaluated, making the presentation of the patterns in blocks of $0.4N$ redundant as all the patterns are available at one time.

Another problem with this process is that it forces the system to store many more patterns than the normal capacity of a Hebbian based Hopfield network. This massive overloading of the network means that for the majority of the time the network is running it is unable to be used as an auto-associative network. There are no useful stable patterns until after unlearning. This would be the human equivalent of being able to remember the previous days events for a few minutes in the morning and then, having added a few more items, being unable to remember anything until the next morning.

The overload and unlearn process removes the large spurious patterns, but during the process an enormous number of spurious patterns with very small basins of attraction are generated. The energy surface caused by unlearning is densely pitted with many stable spurious patterns, and on this surface there are a few learnt patterns which also have very small basins of attraction, but are still stable. This very pitted surface explains the short distance that a previously learnt, but unstable pattern, travels when relaxed. The network quickly finds a nearby spurious pattern that is stable as there are so many small stable spurious patterns on the surface.

In summary, the overload and unlearn process works moderately well as a recognition system for the patterns that have been learnt in the most recent block. As van Hemmen states, the learnt patterns are almost never found during probing of the

network so this system is very limited as a content addressable memory. The basin estimates in Table 4.3 show that for a pattern to be recovered, nearly 90% of the pattern has to be intact in the probe. Given all the constraints on this learning system, and the parameter relating to the global correlation between patterns, it is difficult to see how this could be effectively used to preserve the Hopfield network as a content addressible memory.

4.4.7 Delta Learning

Delta learning in Hopfield networks does not suffer from catastrophic forgetting in the same way as unmodified Hebbian learning. Figure 4.16 shows the performance of delta learning against Hebbian learning with weight decay. The performance of delta learning is slightly better than that of weight decay, and there is not the total loss of functionality that results from unmodified Hebbian learning. In the network that generated the data for Figure 4.16, new patterns were learnt one at a time for a maximum of 5000 epochs or until the error was less than 0.001. The network required, on average, only 711 ($stddev = 34.94$) epochs to learn each new pattern (100 items for 100 repetitions, giving 10,000 data points for this average). The maximum time taken was only 1489 epochs, so at no stage did the learning stop because of the epoch limit. The patterns were learnt with absolute Gaussian noise[2] (ν_i) of 0.5 applied to each unit in each epoch of learning. This noise helps to make the learnt patterns more robust and to create a basin of attraction around the new pattern.

Adding weight decay to delta learning does not improve performance. Weight decay increases the speed at which older patterns are forgotten, and so although it does very slightly decrease the number of epochs required to learn new patterns it is not beneficial to the overall capacity of the network.

As discussed in Section 2.3.2 delta learning in MLP networks introduces its own type of catastrophic forgetting – catastrophic plasticity, caused by excessive change of the weights to accommodate a new pattern. This also occurs in Hopfield networks. The forgetting of the base population is seen in Hebbian learning with weight decay, but the forgetting is gradual and continuous rather than catastrophic. The difference between the two is partly due to the low capacity of Hebbian learning. With delta learning, the number of patterns that can be stored in the network is much larger than with Hebbian learning, so when new patterns are learnt there is more to lose. The

[2]Gaussian noise as described in Section 2.3.

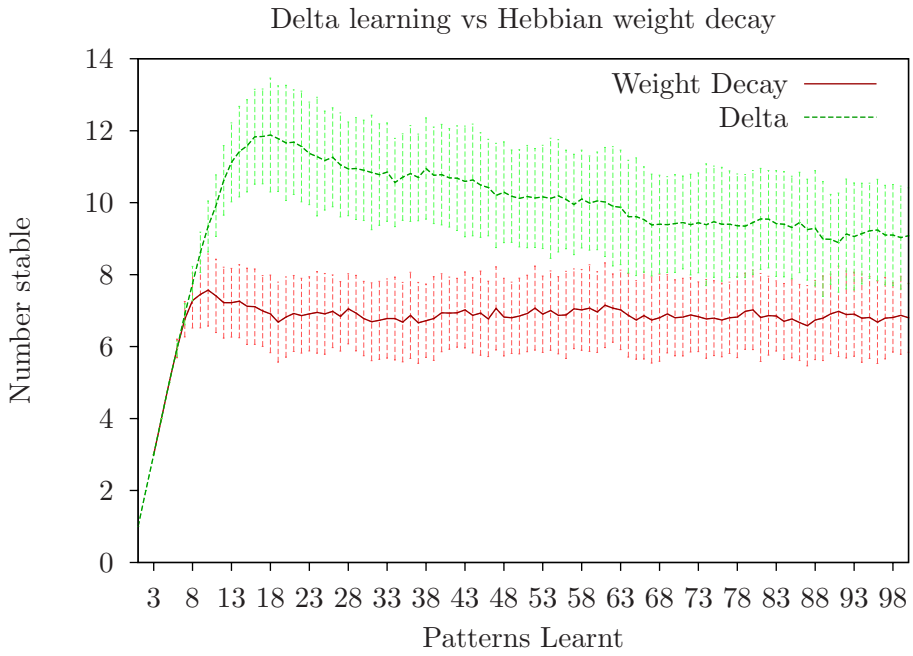


Figure 4.16: Number of learnt patterns stable in a 100 unit Hopfield network $\mathcal{H}_{100,\pm}$ with delta learning versus Hebbian learning with weight decay $d = 0.1$ per item (100 repetitions).

ongoing performance of delta learning and Hebbian learning in Figure 4.16 is similar. However, delta learning has a much higher possible capacity than Hebbian learning, and it is when patterns are learnt in blocks that both the higher capacity and catastrophic plasticity become evident.

Figure 4.17 shows CF caused by the plasticity of the delta rule. The gradient of the forgetting caused by learning new patterns is related to the number of patterns in the base population. The larger the number of patterns learnt, the more fragile the learning. This is a result of the balancing that delta learning does to make multiple patterns stable. Delta learning is able to make $0.9N$ patterns stable, but after just ten more patterns are added its performance is down to only $0.05N$ patterns. This massive loss of stored information is the CF that we will try to prevent when using this algorithm.

It is possible to combine delta learning with weight decay, weight capping, and neural regulation. Both weight decay and weight capping degrade the performance of delta learning. Weight decay just degrades previously learnt patterns faster, and weight capping forces delta learning to spread its changes across more weights and thus degrade previous patterns. Neural regulation also decreases the weights independently of delta

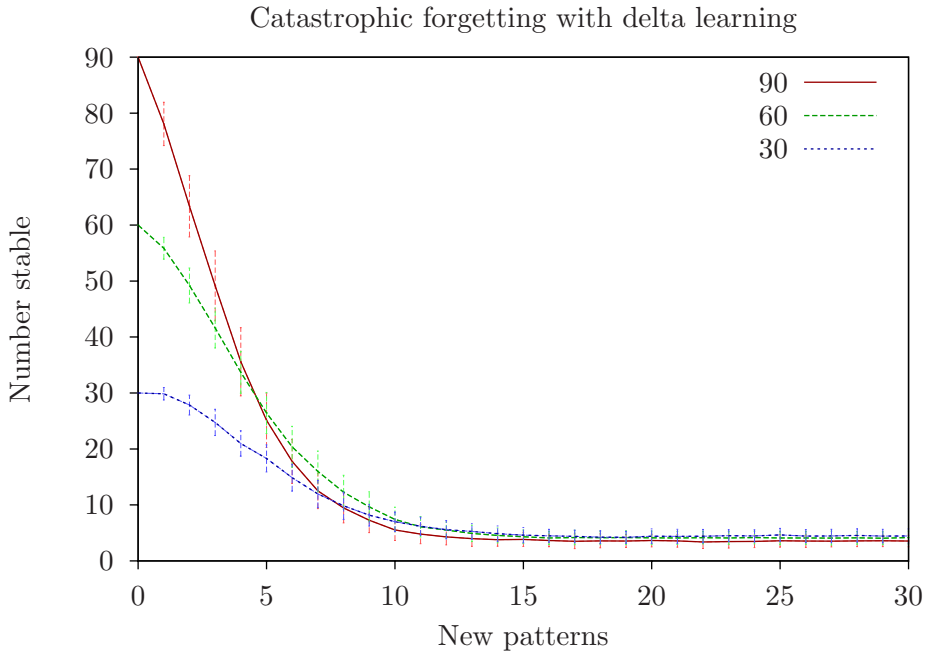


Figure 4.17: The number of patterns stable in a 100 unit Hopfield network $\mathcal{H}_{100,\pm}$ with delta learning of a base population of 30 ($0.3N$), 60 ($0.6N$), and 90 ($0.9N$) patterns followed by learning of new patterns (100 repetitions).

learning. This decrease of weights again degrades the stability of previously learnt patterns. Unlike Hebbian learning, delta learning does not need an explicit process to remove old patterns, as it will continue to learn until the new pattern overrides the previous weights. These systems that control the forgetting of old patterns, do not improve the capacity of delta learning.

4.5 Rehearsal and Pseudorehearsal in Hopfield Networks

The maximum capacity of the Hopfield network using delta learning and randomly generated patterns is N for symmetrically connected networks and $2N$ for asymmetrically connected networks (Gardner, 1987; Gardner, Stroud, and Wallace, 1989). When learning patterns iteratively, delta learning quickly falls to a level much lower than this maximum capacity. Two approaches that proved to be very effective at improving the capacity of MLP networks were rehearsal and pseudorehearsal. In this section we investigate the use of these rehearsal mechanisms in Hopfield networks.

4.5.1 Full Rehearsal

The upper bound for rehearsal in the Hopfield network is 100% of the patterns presented are learnt until the network approaches the capacity of N or $2N$. To test the effectiveness of rehearsal for delta learning in the iterative case we include the entire learnt population with every block of new patterns. Figure 4.18 shows the performance of full rehearsal in an $\mathcal{H}_{100\pm}$ network with asymetric connections, having learnt random patterns with a coding ratio of 50%. Delta learning was given 4000 epochs per block of 5 new patterns to relearn the old population and incorporate the new group of patterns. To achieve this level of stablility the network is trained without noise. The maximum capacity refers only to the stability of a pattern, not the basins of attraction. As the network learns more patterns the number of epochs required increases. This steady increase skyrockets when the network nears capacity. In Figure 4.18 the right hand axis shows the number of epochs used on a log scale. At about 160 ($1.6N$) patterns the number of epochs shoots up to the maximum of 4000 epochs, and the network starts to suffer from the problem of catastrophic capacity.

Making patterns barely stable is not the main goal of a content addressable memory. The patterns learnt without noise have almost no basin of attraction, and any alteration

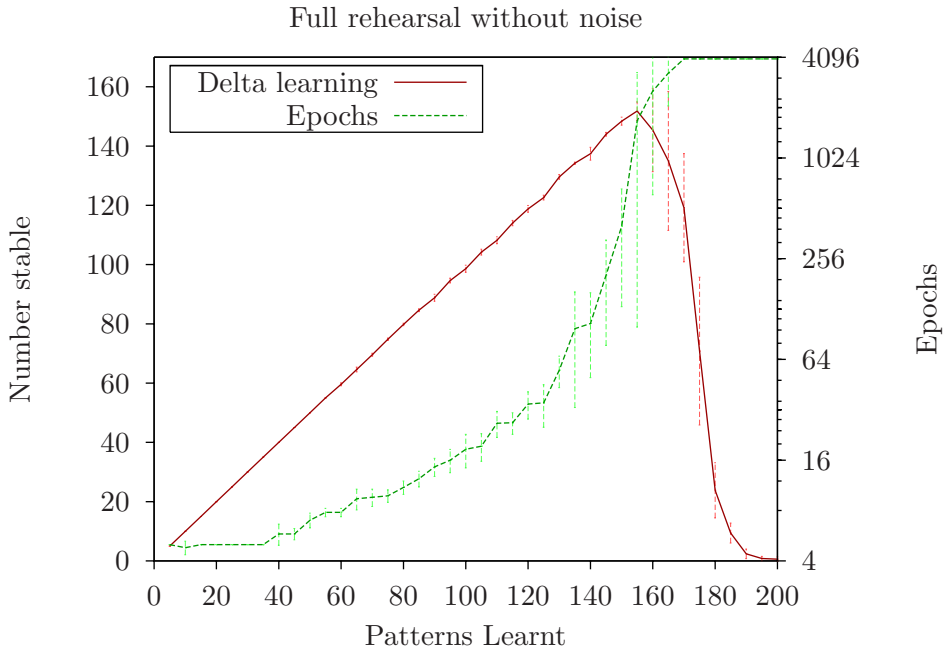


Figure 4.18: Full rehearsal of the entire learning population with blocks of 5 new patterns at a time. $\mathcal{H}_{100,\pm}$ network with learning constant $\eta = 0.1$ with no noise. Epochs have a maximum of 4000 and an error criterion $\delta_l = 0.001$ (10 repetitions).

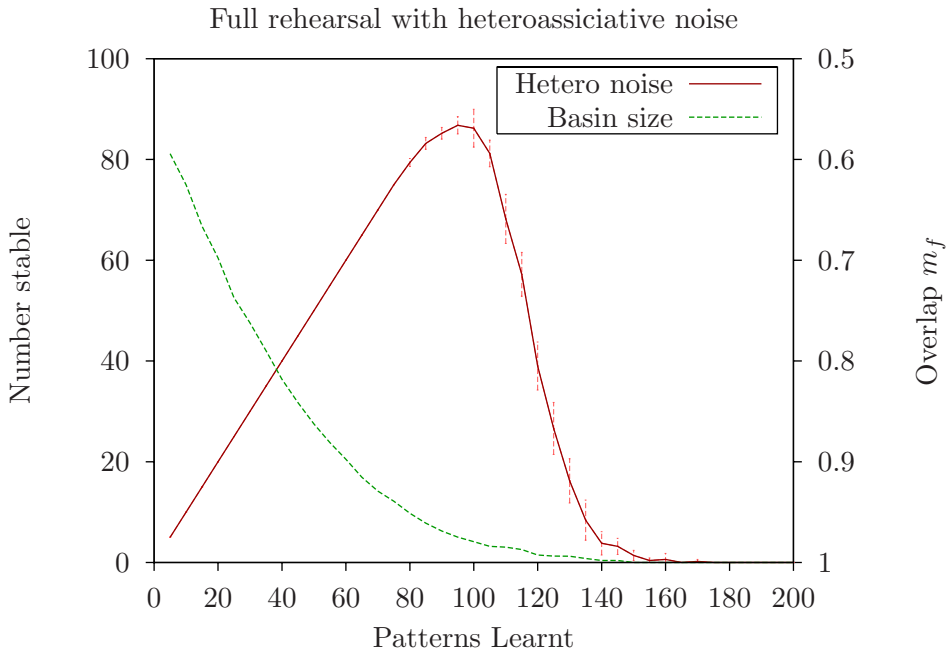


Figure 4.19: Full rehearsal of the entire learning population with blocks of 5 new patterns at a time. $\mathcal{H}_{100,\pm}$ network with 4000 epoch limit, learning constant $\eta = 0.1$ and heteroassociative noise $\nu_h = 5\%$ (10 Repetitions).

to the stored pattern would result in relaxation to a spurious pattern. By learning with noise, for example heteroassociative noise ($\nu_h = 5\%$), the patterns are forced to have a basin of attraction. Unfortunately by adding noise the capacity is decreased because the network is no longer just trying to make patterns stable, but trying to form basins of attraction around those stable patterns. With noise included in training the number of epochs used increases and often reaches the epoch limit before the errors have disappeared. There are therefore two possible reasons for the network failing to learn the patterns; 1) the absolute capacity of the network is too low (a solution does not exist), or 2) the network has not been given enough time to find the solution.

Figure 4.19 also includes the average size of the basins of attraction of the learnt patterns. This has overlap $m_f = 0.5$ at the top of the graph as this is the largest possible basin of attraction for stable patterns. The basin size continues to decrease (an increase in the amount of overlap required to be inside the basin of attraction) as more patterns are learnt. As the network reaches capacity the basins of attraction of the learnt patterns have almost disappeared.

One of the problems presented by iterative learning of a large population is that

patterns presented early get a much larger amount of learning and already have large stable basins of attraction before new patterns are added. This makes the task of balancing the weights harder as the network already has large weights which may need to be lowered to allow for the new patterns. Thus the type of noise is critical to how successful the continual learning process is. Relative Gaussian input noise forces the units to continually make weight adjustments every epoch. This continual error correction means that none of the patterns reach the criterion and stop learning. As every pattern is continually learning the patterns presented early have a head start over new patterns and begin to dominate as they receive the same number of updates during learning as the new patterns.

The network experiences CF as it reaches the limit of its resources. The network is unable to converge on a solution in the 4000 epoch and this leaves all the patterns in an unstable state. This is a bit like trying to catch two balls with one hand, a system designed to converge on a single solution will miss both balls as it moves back and forth around the centre of them. This is a clear example of CF caused by catastrophic capacity.

Unlike "full rehearsal" in an MLP network, where the amount of information presented was not close to capacity, Hopfield networks quickly reach capacity even for delta learning, and without some management of the forgetting process catastrophically forget the information that has been learnt, as well as being unable to learn new information. One way to restore performance would be to limit the rehearsal to just the last $0.5N$ patterns. This will provide a network which quickly forgets patterns as they drop out of the rehearsal set. This also implies that there is another storage medium that is able to store all of the patterns so that they can be rehearsed by the Hopfield network. If that were the case then the Hopfield network would be redundant.

Full rehearsal is impractical and suffers from catastrophic capacity, but the pseudorehearsal method that works in MLP networks can be adjusted to work with Hopfield networks, as we were able to show in Robins and McCallum (1998).

4.5.2 Pseudorehearsal in Hopfield Networks

The goal of pseudorehearsal is to protect the functionality of the network while new patterns are learnt. Section 3.3 demonstrated that pseudorehearsal can be used to prevent loss of information in an MLP network. The algorithm preserves the input-output mapping of the MLP network because the pseudoitems accurately capture its important behaviour, and the rehearsal of these items preserves that behaviour.

As described in Section 3.3, there are three Requirements for pseudorehearsal to be effective: 1) enough capacity to successfully learn; 2) rehearsal is effective; and 3) representative pseudoitems. Hebbian learning fails both 1 and 2. The capacity of unmodified Hebbian learning is very low, and there is not much room for improvement between capacity with simple weight decay $0.05N$ and the maximum capacity of $0.14N$. Hebbian learning also does not benefit from rehearsal. When a population of patterns are learnt a second time, the performance is identical with the only change being that all the connection strengths are doubled. Hebbian learning does not take into consideration the current state of the network, so does not slowly converge on a solution but must find it in one pass or not at all.

In order to test pseudorehearsal in Hopfield networks we need an algorithm that satisfies these three Requirements. Delta learning is the simplest candidate and retains many of the benefits of Hebbian learning. Using the input as the desired output results in all the information required for updating weights still being available locally. As shown above delta learning with full rehearsal is able to alleviate CF. Thus delta learning satisfies Requirements 1 and 2 allowing us to focus on the type of pseudoitems required to extract the behaviour of the Hopfield network.

A naïve implementation of pseudorehearsal in a Hopfield network is to sample the input–output mapping using the same random sampling technique as an MLP network (i.e. "feed forward" for just one cycle). This generates a population of input–output pairs as the pseudoitem population. When these items are used during pseudorehearsal they actually degrade the performance of the network faster than doing nothing at all. The reason for this poor performance is that the mapping from the current state of the network to the next state of the network does not reflect the important behaviour of relaxing to attractors. Hopfield networks rely on the existence of attractors in a state space, which are found by an iterative relaxation procedure. Using input–output pairs preserves the slope of the energy function at a point on the energy surface. The preservation of this slope does *not* preserve the desired behaviour of the network, which is related to the attractors at the bottom of the slope rather than the transition caused by the slope.

The intent of pseudorehearsal is to preserve the behaviour of the network, so for an associative memory system this requires the preservation of the attractors. The items to rehearse must reflect this functionality. Random probing of the network generates many stable states, some learnt and some spurious. By using these stable states (which include both learnt and spurious patterns) as the pseudoitems, pseudorehearsal may

be able to preserve the function of the network. This process is now slightly different to pseudorehearsal in an MLP as many of the rehearsed items are learnt items rather than just items similar to the base population.

The algorithm for pseudorehearsal in a Hopfield network is:

1. learn a small part of the base population of patterns in the standard way
2. generate a pseudopopulation by
   (a) generating a random pattern to probe the network
   (b) relaxing the pattern to a stable state
   (c) storing the stable state for use in the pseudoitem population
3. generate a new training population by interlacing the new patterns to learn with the pseudopopulation
4. train using this population until some error criterion has been reached
5. repeat from step 2 for each set of new patterns

There are many parameters that can be tuned when using this algorithm. Almost every step has one or more parameters associated with it, some of these are listed in Table 4.4. In our simulations all of these parameters and more can be adjusted. Appendix A discusses each of the parameters and Appendix B.2 gives an example of the parameters used for simple pseudorehearsal. There are too many possible combinations to exhaustively test them all, so the values must be estimated from testing the performance of the system in various configurations. Some settings are chosen by their psychological plausibility, some for ease of calculation, and others by trial and error.

The default values for a pseudorehearsal run in an $\mathcal{H}_{100,\pm}$ are:

Coding ratio	$\alpha = 0.5$
Base population	$BP = 5$
Learning constant	$\eta = 0.1$
Epochs	$Ep = 500$
Error criterion	$\delta_l = 0.001$
Noise during training	$\nu_h = 5\%$
Relaxation cycles	$r = 4\mathrm{N}$
Pseudorehearsal probes	$\mathcal{P}p = 2000$
Pseudorehearsal items	$\mathcal{P}i = 300$

Given the above parameters any and all stable patterns found by probing will be rehearsed. This will therefore protect both learnt and spurious patterns.

Step		Parameters
	generate a learning population	type of patterns coding ratio number of patterns
1.	learn a small part of the base population of patterns in the standard way	learning constant ηinitial learning error δ_l type of noise during training level of noise during training ν_h, ν_i
2.	generate a pseudopopulation by (a) generate a random pattern to probe the network	number of probes to find pseudoitems $\mathcal{P}p$ probability of activation $\mathcal{S}_p^n$ orthogonality noise during relaxation
	(b) relax the pattern to a stable state	record distance travelled maximum cycles r
	(c) store the stable patterns for use in the pseudopopulation	which patterns to store number to store $\mathcal{P}i$ remove duplicates remove learnt items
3.	generate a new training population by interlacing the new patterns with the pseudopopulation	interlacing schedule rotation of pseudoitems
4.	train using this population until some error criterion has been reached	error on all patterns or just new patterns, error value noise on new patterns noise on all patterns noise ratio for pseudoitems learning ratio for pseudoitems
5.	repeat from step 2 for each set of new patterns	

Table 4.4: Some of the parameters that influence the behaviour of pseudorehearsal.

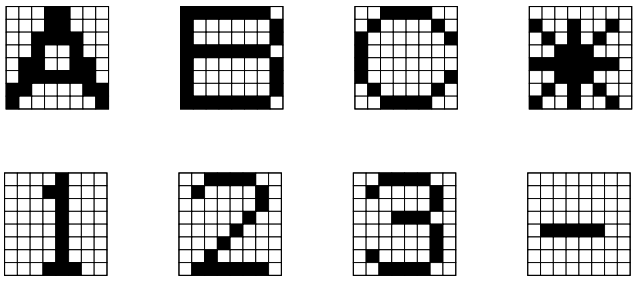


Figure 4.20: Eight example patterns from the population of 64 alpha-numeric patterns.

4.5.3 Pseudorehearsal of Stable Patterns

In our 1998 paper (Robins and McCallum, 1998) we used an $\mathcal{H}_{64,\pm}$ network and stored artificially generated alpha-numeric character set. The characters are represented in an 8×8 grid, examples of which can been seen in Figure 4.20. This set consists of 64 items which allows us to store up to $1N$ patterns. Populations are trained over 500 epochs (presentations of the whole population) using delta learning. In the 1998 paper we used the thermal perceptron algorithm described in Section 3.4. We have simplified the learning algorithm to standard delta, to decrease the complexity of the interactions. New patterns are learnt with heteroassociative noise of $\nu_h = 5\%$.

To test pseudorehearsal, a base population of 44 randomly selected patterns is learnt using delta learning. Then 20 more patterns are learnt (the rest of the population) one at a time with heteroassociative noise applied to each new pattern. This is the same as the delta learning test described in Section 4.4.7. Figure 4.21 shows the performance of the network with no rehearsal on both the current total learnt population[3] "None" and the base population "None BP" with no rehearsal. This graph is different to the figures presented in Robins and McCallum (1998) as those graphs present the errors rather than the success, and in that paper we were only interested in the base population.

Figure 4.21 also shows the results for pseudorehearsal "Pr256" of the first 256 patterns found by random probing of the network. The pseudopopulation is generated after each new pattern is added, in line with the process used in Robins and McCallum (1998, Figure 7). This graph shows that the pseudorehearsal is able to protect against the massive and sudden loss of information that originally occurred after the first new patterns are learnt. Pseudorehearsal is also able to protect many more patterns than standard delta learning. By the time the 64^{th} pattern is learnt the number of

[3]The combination of both the base population and the new items learnt. After 10 items learnt this combined population is 54 patterns.

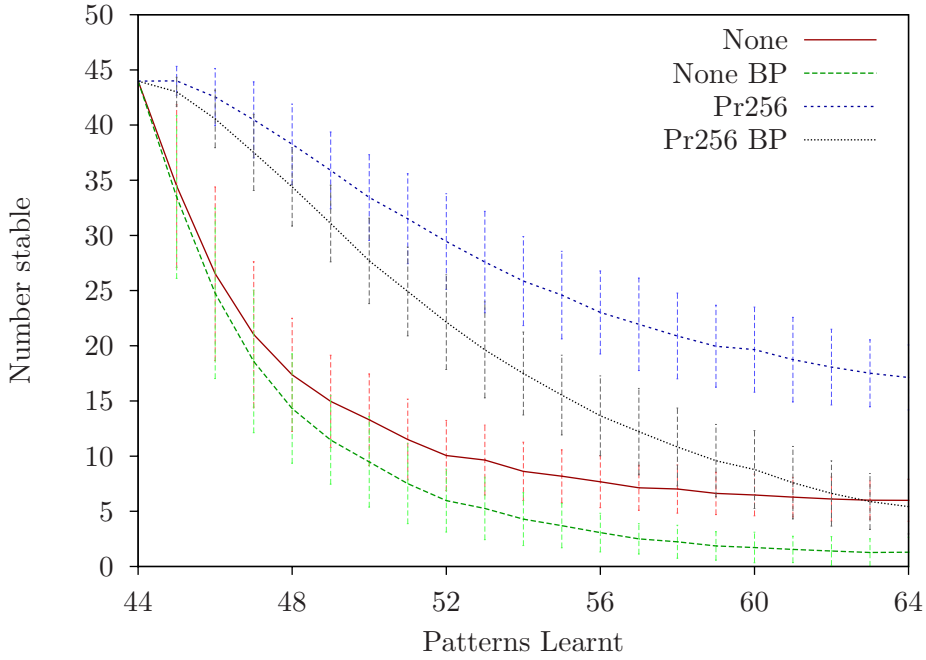


Figure 4.21: The number of stable patterns, both base population and whole population, in an $\mathcal{H}_{64,\pm}$ with delta learning of a base population of 44 ($0.69N$), with 20 new patterns learnt in succession. "None" is simple delta learning, and "Pr256" is pseudorehearsal of 256 random probes of the network generated after each new item is learnt. The "BP" conditions show the number of base population items that are stable for each type of learning (100 repetitions).

base population items "Pr256 BP" that are stable has dropped to about 5 patterns. There are about 15 of the intervening 20 patterns stable, which combined with the 5 base population items gives the 20 patterns stable for the "Pr256" condition after 64 patterns have been learnt.

A population of pseudoitems generated by random probing will naturally contain duplicates of the learnt patterns, as these are the main attractors in the pattern space. With 256 random pseudoitems about 50% of the stable learnt patterns are found and included in the pseudoitem population. If we assume that there is no access to the base population for rehearsal, then there is no need and no principled way, of removing these patterns. However, with all the learnt patterns removed the "PR256*" condition in Figure 4.22 shows that pseudorehearsal does not depend on duplicating the learnt items for its effectiveness. This form of pseudorehearsal is only using spurious pattern information to protect the stability of the learnt patterns. This counter intuitive result

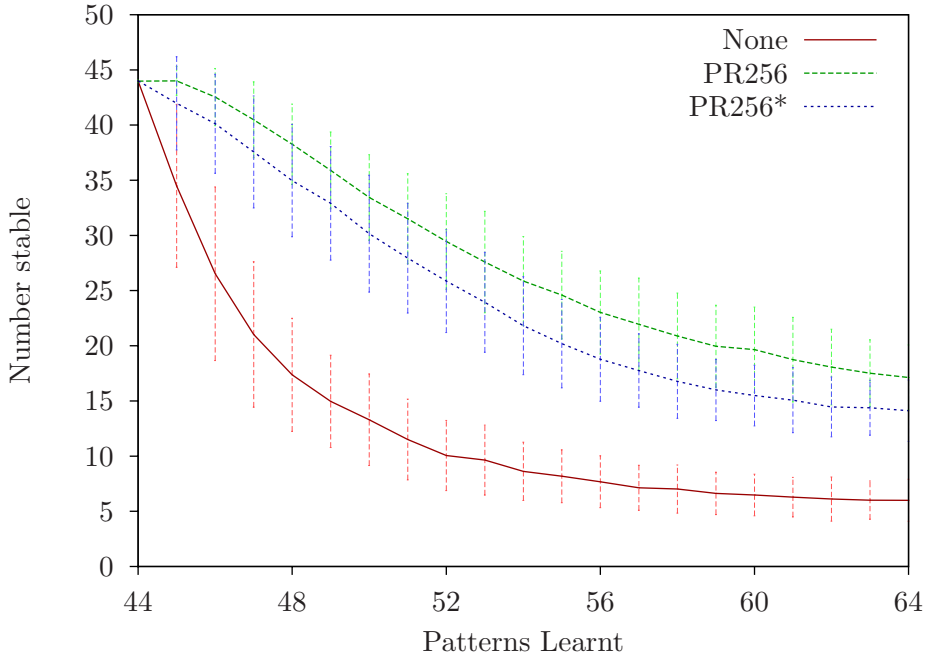


Figure 4.22: The number of patterns stable in an $\mathcal{H}_{64,\pm}$ with delta learning of a base population of 44 ($0.69N$), with 20 new patterns learnt in succession (100 repetitions). "PR256*" is pseudorehearsal of 256 random probes of the network with any learnt patterns explicitly removed from the pseudoitem population.

is discussed later in Section 4.5.4.

To directly compare pseudorehearsal to the other solutions to CF we must test them in the same environment. In the $\mathcal{H}_{100,\pm}$ network with random patterns the results are also clearly improved. Figure 4.23 shows the comparative performance of pseudorehearsal, unmodified delta learning, Hebbian learning with weight decay, and Christos unlearning. Once the initial phase of learning has passed pseudorehearsal is clearly better than any of the other methods presented. In the initial phase unlearning is performing very well, but this performance degrades over time as the correlations between patterns builds and the unlearning is unable to correct for the correlation.

Pseudorehearsal has an interesting peak at about 13 patterns learnt. The number of patterns that are stable grows until this point, and then there is a dip where fewer patterns are stable, before the number starts to increase again at about 16 patterns. The reason for this decline is that after $0.14N$ patterns the learnt patterns start to become unstable simultaneously, and drop out of the rehearsal population. The number of stable learnt patterns decreases in much the same way as delta learning, but rather

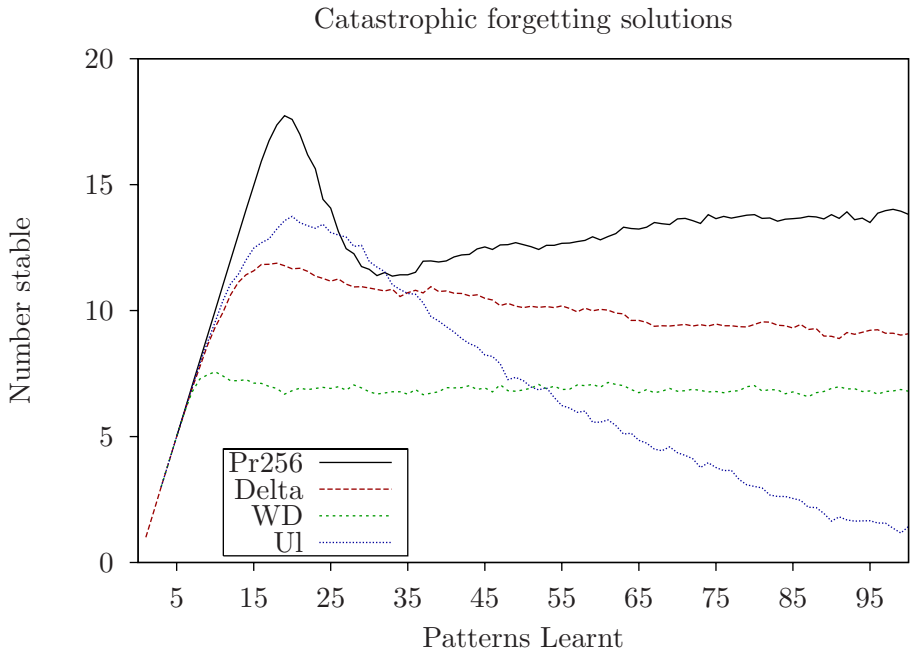


Figure 4.23: The number of patterns stable in an $\mathcal{H}_{100,\pm}$ with pseudo-rehearsal of 300 patterns "Pr256", simple delta learning "Delta", Hebbian learning with weight decay "WD" $d = 0.1$, and Christos unlearning "Ul" ($U = 50, -\eta = 0.1$) (100 repetitions, N.B. error bars removed for readability).

than staying at the lower level the rehearsal starts to stabilise the new patterns that are being learnt. The number of stable patterns slowly increases as some of the old patterns become unstable making way for new patterns. The balance between retaining old patterns and learning new patterns is now part of the parameters of the pseudorehearsal algorithm.

There are many parameters that can be adjusted to manipulate the performance of pseudorehearsal. The first of these is the number of pseudoitems that are used during rehearsal. Figure 4.24 shows the results for pseudorehearsal with a varying number of rehearsal items.

4.5.4 Rehearsing Spurious Items

The rehearsal of spurious patterns to preserve learnt one seems counter intuitive. This intuition is based on the assumption that spurious patterns are just random noise in the network. This is not actually the case. The spurious patterns are combinations (often

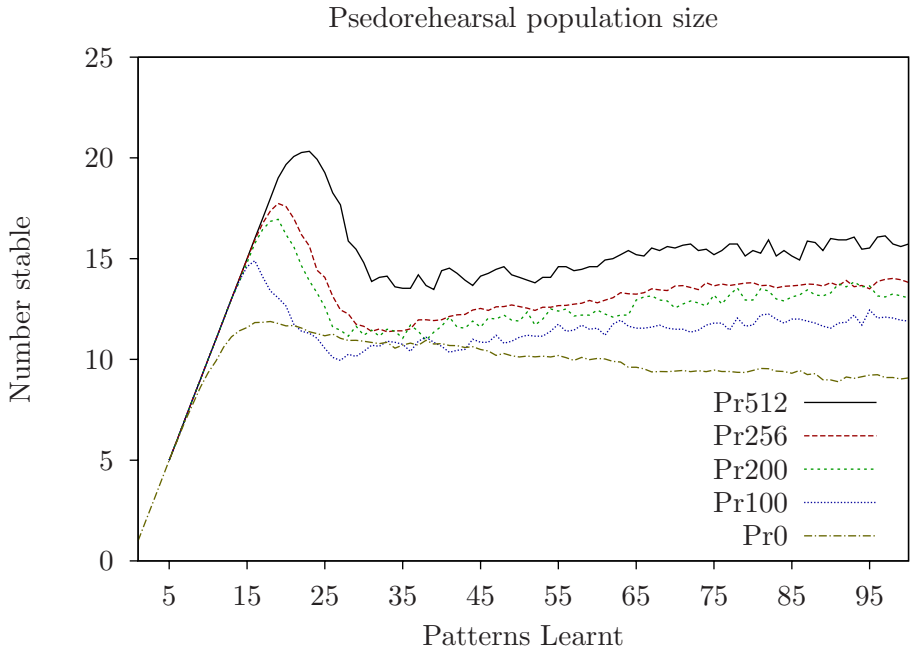


Figure 4.24: The number of patterns stable with varying numbers of pseudoitems in the rehearsal population for an $\mathcal{H}_{100,\pm}$ (512, 200, 100 have 20 Repetitions 0 and 256 have 100 repetitions, N.B. error bars removed for readability).

linear) of stored patterns. The noise is caused by the correlations in the learnt patterns and competition for the state space. Many of the spurious states can be thought of as something like "ripples" caused by the learnt patterns. It would seem that the particular form of ripple caused in the energy surface by a learning stable attractors, may when rehearsed form a reciprocate ripple that preserves a learnt pattern. The fact that some spurious patterns are linear combinations of learnt patterns is best shown by an example.

For this example we are using an $\mathcal{H}_{16,\pm}$ with the three patterns in Table 4.5 learnt using Hebbian learning and a learning constant of $\eta = 1$. Having learnt these patterns there are already a number of stable spurious patterns. To understand the relationship between the spurious patterns and the learnt patterns it is useful to reorder the units so that correlations become more obvious. This can be done as there is no topological significance to the viewing order. If we reorder the units to view them by their activity in each pattern we can see that they are grouped into 2^p categories, in this case 8 groups as there are three patterns $p = 3$. Each unit within a group will have the same weighted connections to all other units. Table 4.6 shows the reordering of the patterns.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Pat A	+	+	−	−	+	−	+	+	−	−	−	−	+	+	+	−
Pat B	−	−	+	−	+	−	+	+	+	−	+	−	−	+	−	+
Pat C	+	−	+	+	+	−	+	−	−	+	+	−	+	−	−	−

Table 4.5: Three patterns to be learnt in an $\mathcal{H}_{16,\pm}$, randomly generated with exactly 50% of the units in the active state.

	$G1$		$G2$		$G3$		$G4$		$G5$		$G6$		$G7$		$G8$	
	5	7	14	8	1	13	15	2	3	11	9	16	4	10	6	12
Pat A	+	+	+	+	+	+	+	+	−	−	−	−	−	−	−	−
Pat B	+	+	+	+	−	−	−	−	+	+	+	+	−	−	−	−
Pat C	+	+	−	−	+	+	−	−	+	+	−	−	+	+	−	−

Table 4.6: Patterns A, B, and C reordered by sorting on the activation in each pattern.

The eight groups created by three patterns are: $(+++), (++-), (+-+), (+--), (-++), (-+-), (--+), (---)$. After learning three patterns all units in $G1$ have a -3 weight connections to units in group $G8$. This sets up a relationship between the units in the two groups. When one group is active there is a strong negative input to each unit in

the associated group. In larger networks there are more units in each group, but the number of groups is entirely determined by the number of patterns. With only three patterns stored in a large network the internal connections within the group will force all units within a group to have the same activation. The number of units in each group could vary, but with a large N and coding ratio centred around 0.5 each group will have approximately $n = \frac{1}{2^p}N$ units in it, in this case 2 units in each.

The original set with groups of active units represented by $+^*$ is:

$$(+^*, +^*, +^*, +^*, -^*, -^*, -^*, -^*)$$
$$(+^*, +^*, -^*, -^*, +^*, +^*, -^*, -^*)$$
$$(+^*, -^*, +^*, -^*, +^*, -^*, +^*, -^*)$$

With three patterns learnt there are always exactly four stable spurious patterns with any network this size or larger. These are:

$$(+^*, +^*, +^*, -^*, +^*, -^*, -^*, -^*)$$
$$(+^*, +^*, -^*, +^*, -^*, +^*, -^*, -^*)$$
$$(+^*, -^*, +^*, +^*, -^*, -^*, +^*, -^*)$$
$$(-^*, +^*, +^*, +^*, -^*, -^*, -^*, +^*)$$

A network with 10000 units will have four spurious patterns that are structurally identical to those above, just with more units in each group. Every simulation run with Hebbian learning confirmed this result. If the first three of these spurious patterns are learnt in a clean network the original three patterns will be three of the four stable spurious states in the new network. This can be seen by resorting the units based on the first three spurious patterns. The learnt patterns are now equivalent to the spurious patterns. This could also be seen as swapping $G3$ and $G4$. This works both in principle and experimentally.

The symmetry in the network can been seen in the weights that connect the units in the groups. The weight matrix for any network with three patterns learnt using Hebbian learning is:

	$G1$	$G2$	$G3$	$G4$	$G5$	$G6$	$G7$	$G8$
$G1$	+3	+1	+1	-1	+1	-1	-1	-3
$G2$	+1	+3	-1	+1	-1	+1	-3	-1
$G3$	+1	-1	+3	+1	-1	-3	+1	-1
$G4$	-1	+1	+1	+3	-3	-1	-1	+1
$G5$	+1	-1	-1	-3	+3	+1	+1	-1
$G6$	-1	+1	-3	-1	+1	+3	-1	+1
$G7$	-1	-3	+1	-1	+1	-1	+3	+1
$G8$	-3	-1	-1	+1	-1	+1	+1	+3

Although this example is limited to networks with only three learnt patterns, it does give an example of how rehearsing the spurious patterns found in the network can preserve some of the information about the original patterns, and in this example make the units stable.

When training with pseudorehearsal the pseudoitems are rehearsed without noise. This difference is very important when rehearsing these random patterns. If the pseudoitems are trained with noise they become the new attractors in the network and override both the previously learnt patterns and the new patterns. Without some way of knowing the type of pattern that is being rehearsed, it is not beneficial to make patterns found by random probing the new attractors in the network.

4.6 Discussion

The solutions to catastrophic forgetting in Hopfield networks can be divided into three levels and two groups. Synaptic level solutions work at the lowest level of a neural network. Both weight decay and weight capping are able to continue to learn new items while gradually forgetting previously learnt patterns. However the capacity of 0.05 patterns is so low that it is not an adequate solution to catastrophic forgetting. These very simple solutions do provide a solid lower bound for assessing the performance of the more complex systems. Weight decay can be seen as a form of managed forgetting. It actively lowers the contribution of the old patterns as new patterns are added. This active forgetting cleans up the network and allows learning to continue indefinitely without the concern that over long periods of time, small errors and crosstalk noise will eventually cause the network to collapse, and no longer function as an content addressable memory. However it does mean that the system will only every remember the most recent learnt patterns. There is no preferential treatment given to significant

memories, for example those that are needed for survival.

The neuronal and system level solutions do not perform significantly better than simple weight decay. Neural regulation is interesting in that the two different types or regulation solve different problems – zero mean solves correlation, and resource limiting solves capacity. This resource management approach can also work in conjunction with weight decay and provides redundancy for the learning system, a feature that is evident in most real world biological systems (Izui and Pentland, 1990).

The unlearning approach on the other hand seems to work in almost the same way as weight decay in networks with i.d.d. patterns. With correlated patterns unlearning is able to remove some of the correlation. Unfortunately, the amount of unlearning is dependent on the amount of correlation, which is impossible to know *a priori* except in the limited case where the correlation is only caused by a low coding ratio. Unlearning received a lot of attention when it was first proposed, as it gave a possible explanation for one of human beings most fascinating behaviours, dreaming. The performance of unlearning in a situation where new patterns are presented continually over time does not support the early enthusiasm for the system. Performing unlearning immediately after a new item is learnt, improves performance initially, but over time the noise and correlations between patterns destroy performance. Van Hemmen's overload and unlearn approach seems to work for larger numbers of patterns, but has several problems. Firstly, the memory system no longer works as a content addressable memory as the basins of attraction of the learnt patterns become minuscule. Secondly, the network is only ever functional after a large block of patterns has been presented and then unlearnt, and gives no results at any other time.

The second group of solutions move away from Hebbian learning and instead use an error correcting delta learning approach. Delta learning does not suffer from catastrophic correlation found with Hebbian learning. As new patterns are learnt they overwrite the changes made by old patterns, and those patterns are gradually forgotten. Thus the delta learning algorithm has an inbuilt forgetting system. It can also be combined with weight decay, weight capping, or normalisation to ensure that the weights in the network do not begin to diverge, leading to a sudden loss of performance after very large numbers of patterns are learnt.

The most successful of the approaches presented is pseudorehearsal. It is able to retain almost twice as many patterns as unmodified delta learning, or Hebbian learning with weight decay. Surprisingly it is still effective when the learnt patterns are removed from the pseudorehearsal population. The fact that relearning spurious patterns is able

to protect the original base population seems counter–intuitive, but can be shown to be theoretically possible with small numbers of patterns, and is backed by experimental data. Pseudorehearsal was effective in both a network with a large base population and correlated patterns, and a network with random patterns and a small base population. The robustness of the results for Hopfield networks, combined with the effectiveness in MLP networks, indicate that the pseudorehearsal may be effective in many other neural network structures.

The promising results from pseudorehearsal warrant further investigation. Pseudorehearsal is not performing as well as full rehearsal and so, perhaps there are ways to improve the approach. One of the major limitations for the pseudorehearsal approach is the inability to differentiate the learnt patterns from the spurious patterns. In the next chapter we investigate various approaches to differentiating the learnt patterns from spurious patterns, including our proposal of an energy ratio measure. In Chapter 6 we show how this knowledge can be used to improve the solutions to catastrophic forgetting. It also includes a summary of the experiments conducted and how they relate to each other. Discussion of the links to sleep and dreaming in biological systems is covered in Chapter 7.

Chapter 5

Discriminating between Learnt and Spurious Patterns

One of the unwelcome features of Hopfield networks is the large number of stable spurious patterns. In a moderately loaded network there are thousands of spurious patterns for every stable learnt pattern. Random probes of the network will always relax to a stable pattern, but there is no indication as to whether the pattern was learnt or not. The ability to determine the stable pattern's type would allow the network to return not only the stable pattern found, but also an estimate of whether the pattern had been learnt.

The pseudorehearsal algorithm presented in Chapter 4 rehearses all the patterns that are found by probing. With the ability to distinguish learnt patterns from spurious patterns, rehearsal could be focused on preserving just the learnt patterns. For this discrimination to be feasible the network should not have access to the real learnt patterns, but instead must calculate the familiarity of a pattern from information available within the network. This chapter will explore various methods for determining familiarity in Hopfield networks, and Chapter 6 will investigate how this knowledge can be applied to improving the solutions to catastrophic forgetting.

5.1 Defining Familiarity

The task of discriminating between learnt (familiar) and spurious (novel) patterns is called "recognition", familiarity detection, novelty detection, or pattern discrimination. Familiarity detector networks are designed to give an indication as to whether a pattern has previously been presented to the network (For a recent review of this area see

Markou and Singh (2003)). The general process is to present a pattern to the network and have a single output node, where the activation of the output node indicates the familiarity of the pattern. In the human brain this function seems to be centred in the perirhinal cortex (Bogacz, 2001) which is located immediately prior to the hippocampus in the processing stream from sensation to action.

The phenomenon of *déjà vu* has lead to a great deal of research in the area of familiarity discrimination for memory systems. Humans only occasionally experience the sensation of being familiar with something that they have never seen before. FMRI and lesion studies indicate that the perirhinal cortex (Brown and Xiang, 1998; Squire, Stark, and Clark, 2004) in the human brain participates in detecting familiar objects and faces. The existence of an obvious neurological response to familiar items suggests that it is at least possible to perform this type of discrimination in biological systems.

Any form of binary detection has four possible combinations of presentation and identification. These are true positives, true negatives, false positives and false negatives. In the popular language of memory false positives are called "*déjà vu*", and false negatives "forgetting". To analyse the performance of a detector these four values are converted into four ratios, Positive Predictive Value (PPV), Negative Predictive Value (NPV), True Positive Rate (TPR) and True Negative Rate (TNR) [1]. Table 5.1 shows the relationship between these measures. The importance of a particular ratio depends on the particular task. For example, in the field of medicine it is very important to decrease the number of false negatives, where a potentially fatal illness is falsely ruled out by a diagnostic test. In this situation a low false negative rate is the primary concern. When comparing different situations it is important to decide the weighting of these ratios.

For the pattern based memory stored in Hopfield networks, the PPV measures the probability that a pattern identified as learnt (familiar) is actually part of the learnt pattern set. PPV is computed by dividing the number of accurately remembered patterns by the sum of all positive responses (TP/TP+FP). This is the rate at which patterns identified as learnt are actually learnt patterns, i.e. the percentage of memory that is not *déjà vu*. NPV is the ratio of true negatives to all negative responses, i.e. actual novel patterns in relation to all patterns classified as novel. The TPR measures the probability that a pattern will be recognized as learnt given that it has actually been presented earlier, and a high TNR indicates that if a pattern is novel, it will successfully be identified as such.

[1] TPR and TNR are also referred to as the Sensitivity and Specificity of the detector.

		Presentation		
		Familiar	Novel	
Response	Familiar	True Positive (TP)	False Positive (FP)	Positive Predictive Value (PPV) = TP/(TP+FP)
	Novel	False Negative (FN)	True Negative (TN)	Negative Predictive Value (NPV) = TN/(TN+FN)
		True Positive Ratio (TPR) = TP/(TP+FN)	True Negative Ratio (TNR) = TN /(TN+FP)	

Table 5.1: Relationship between the various performance measures of a detector system.

A high PPV of over 99% is considered to be a very good result. In a Hopfield network this would mean that out of 100 probes of the network which were classified as learnt, only one of these would be spurious. The way in which the probes are generated influences how reliable this ratio is. If the 100 probes are generated as copies of learnt patterns, then they are not really a random selection of input but a set selected for a particular attribute. This is selection bias, and it limits the claims that can be made based on the predictive values.

We will use PPV and NPV in terms of our particular problem domain of distinguishing stable spurious patterns from the stable learnt patterns in a Hopfield network. We review a number of the suggested mechanisms for distinguishing learnt from spurious patterns, and discuss their performance on this particularly difficult task.

5.2 Familiarity in Hopfield Networks

In Hopfield networks there are two types of familiarity discrimination, distinguishing learnt patterns from any other pattern, and stable learnt patterns from other stable patterns (spurious patterns). As we are interested in the network as a content addressable memory in this thesis, we are interested in the second type of discrimination. In a Hopfield network only a very small percentage of all the possible patterns are stable. Unfortunately as the network learns the majority of these stable patterns are spurious rather than learnt patterns. Thus the task is to distinguish a small number of positive examples from a very large number of negative possibilities.

5.2.1 Pattern Energy

One of the first methods used to detect spurious patterns in Hopfield networks was to measure the energy[2] of the network given an input pattern. Equation (5.1) shows the energy value H^p of a pattern p

$$H^p = -\sum_{j=1}^{N}(v_j^p \psi_j^p) \tag{5.1}$$

This measure follows directly from the requirement that stable patterns have low energy. Both Hebbian learning and delta learning lower the energy of learnt patterns in an attempt to make them stable. Hebbian learning lowers the energy of each unit in the network by $-(N-1)$, and thus the total energy by $-(N-1)N$. The Lyapunov function (see Section 2.4.2) of monotonically decreasing energy requires that all stable states have low energy. For random patterns that have not been allowed to relax, the energy is close to zero as there are equal numbers of units with negative and positive energy.

It is possible to distinguish learnt patterns from spurious patterns using the total energy, as shown by Bogacz, Brown, and Giraud-Carrier (2001) and Crook, Marsland, Hayes, and Nehmzow (2002). Figure 5.1 shows that the learnt patterns have energy that is clearly lower than the spurious patterns. The graphs in this chapter mostly come from randomly selected single runs of the network. This was done to show the individual variation within an example rather than averages, which can be misleading. Later graphs which are averaged, indicate the number of repetitions used. In Figure 5.1 with a small number of patterns stored, the noise created by crosstalk in the patterns has not significantly degraded the total energy of the learnt patterns. The difference between the total energy of learnt and spurious patterns allows the use of a threshold value (labelled "Novelty Value" in the figure) as a simple novelty check. As there are no learnt patterns above this threshold, and no spurious patterns below it, all of the performance measures are perfect. PPV = 100%, NPV = 100%, Sensitivity = 100% and Selectivity = 100%.

Unfortunately this method fails when more patterns are learnt. Figure 5.2 shows the total energy of the stable patterns in a network with 14 patterns stored ($0.14N$). The total energy of most of the spurious patterns is now lower[3] than the learnt patterns.

[2]See Section 2.4.3 for a discussion of the energy of the network.

[3]Low energy indicates a stable state, and so large negative numbers are less than small negative numbers, indicating a pattern is more stable

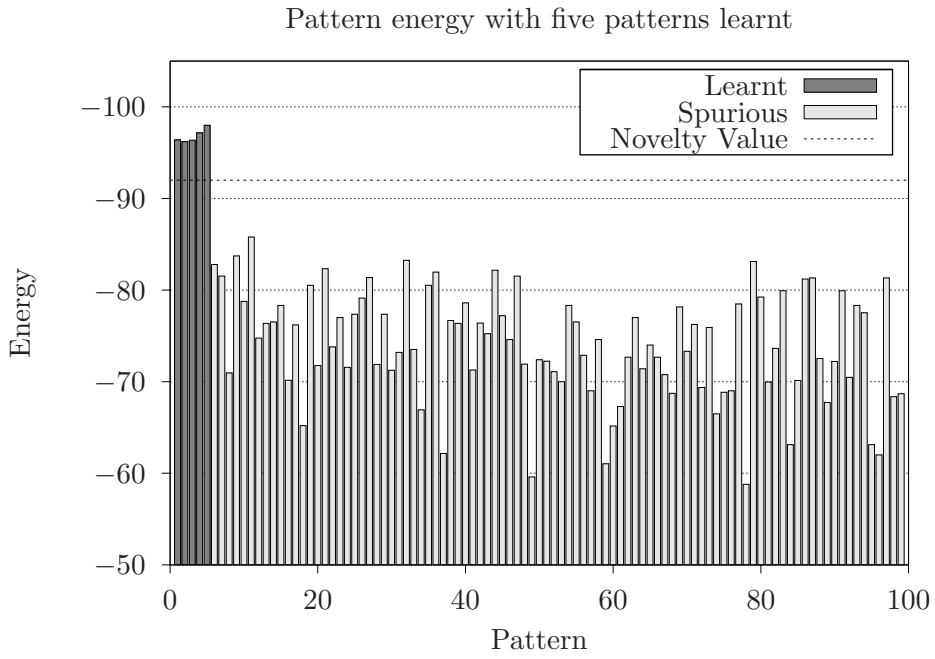


Figure 5.1: The energy of stable learnt and spurious patterns in an $\mathcal{H}_{100,\pm}$ Hopfield network, trained with Hebbian learning. Five patterns have been stored and all are stable. Novelty value of -92. There are 93 spurious patterns found with 2000 probes (Randomly selected single run).

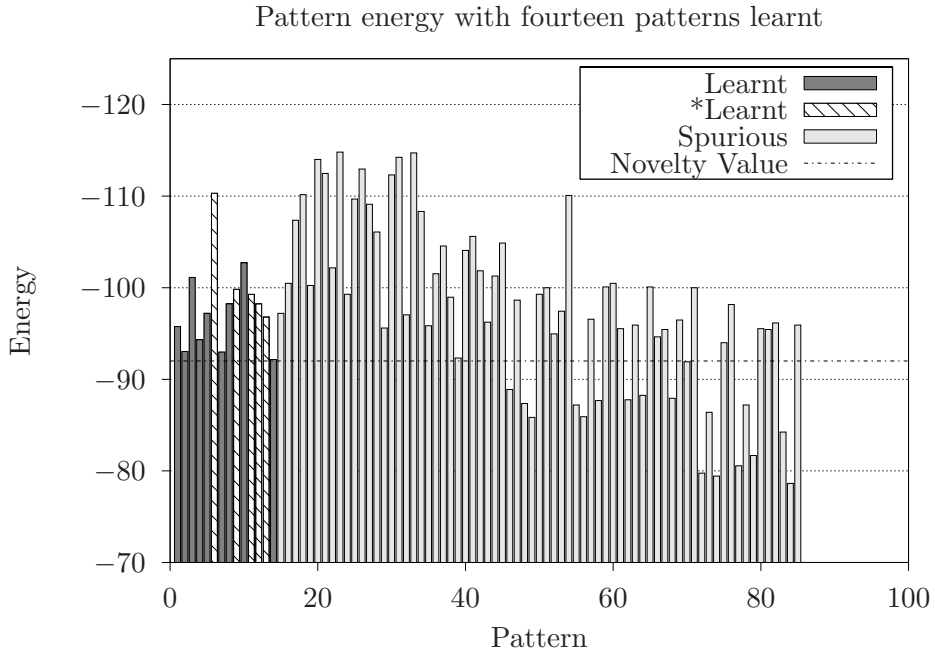


Figure 5.2: The energy of learnt and spurious patterns in an $\mathcal{H}_{100,\pm}$ Hopfield network, trained with Hebbian learning. Fourteen patterns have been learnt but only twelve remain stable. "*Learnt" are learnt patterns that are unstable. There are 70 spurious patterns found with 2000 probes (Randomly selected single run).

There is no way to place a threshold that clearly distinguishes stable learnt patterns from stable spurious patterns. In this situation it is not possible to select a threshold that maintains both a high PPV and NPV. These low energy spurious states render this measure ineffective for moderate to heavily loaded networks.

Bogacz (2001) claims that the pattern energy can still have a good PPV if you consider the space of all possible patterns, not just stable patterns. This statement is not in keeping with the standard definition of PPV. For the PPV to be valid, it must be calculated between samples using the same selection basis. The samples must come from either the whole pattern space or consistently from a selected group of patterns. When selecting patterns to check from the 2^N dimensional space, the positive and negative examples of patterns must be found using the *same* selection mechanism. The chance of randomly selecting one of the learnt patterns without relaxation is $L : 2^N$ (for an $\mathcal{H}_{64}$ with $P = 5$ patterns stored this is $5 : 18\ 446\ 744\ 073\ 709\ 551\ 616$). This means that when probing the space the learnt patterns are extremely unlikely to be encountered, giving a PPV of ≈ 0. As this measure also misclassifies any stable pattern

as a learnt pattern the real PPV is extremely low.

However, Bogacz does use the total energy to distinguish learnt patterns from randomly selected, un–relaxed patterns. He compares the total energy for learnt patterns, which may or may not be stable, against a threshold. Learnt patterns have a lower energy than almost all randomly selected un–relaxed patterns, and so can be classified as familiar. There are millions of spurious patterns that would be misclassified, but as a ratio of all the possible patterns the spurious patterns that are misclassified is quite low. The patterns that this simple energy measure misclassifies are exactly those which are *stable* spurious patterns. The positive predictive value is high when comparing between the learnt subgroup and all possible patterns, but low when the sample space is restricted to just stable patterns. Thus the total energy does not help with the current task of distinguishing stable learnt patterns from stable spurious patterns.

5.2.2 Relaxation Time

Hopfield (1982) proposed that the rate at which units change during relaxation could be used to identify spurious states. The underlying assumption was that the basins of attraction of learnt patterns are more fully formed than spurious patterns, and that a network that relaxes to a spurious state will do so down a long wandering path, while patterns that relax to a learnt pattern will do so quickly from within the basin of attraction.

Following this suggestion Chengxiang *et al.* (2000) proposed a measure based on the length of the relaxation cycle required to reach a stable pattern. As the spurious patterns generally have smaller basins of attraction than learnt patterns, (see Section 5.2.3) the network state tends to "wander around" for longer before settling in one of the spurious states. This is true for a space that is sparsely populated with stable states, most of which are stored patterns.

As the network becomes heavily loaded and spurious patterns become common, the ability to distinguish based on the length of the path breaks down. The basin sizes for the learnt patterns decrease in size and the length of time taken for a probe to find these smaller basins becomes indistinguishable from the length of the paths to find spurious patterns.

In our simulations, the length of the path from the initial random pattern to the final stable state is not correlated with either spurious or learnt patterns. Figure 5.3 shows the average path length for random probes relaxing to spurious and learnt patterns during the serial learning task.

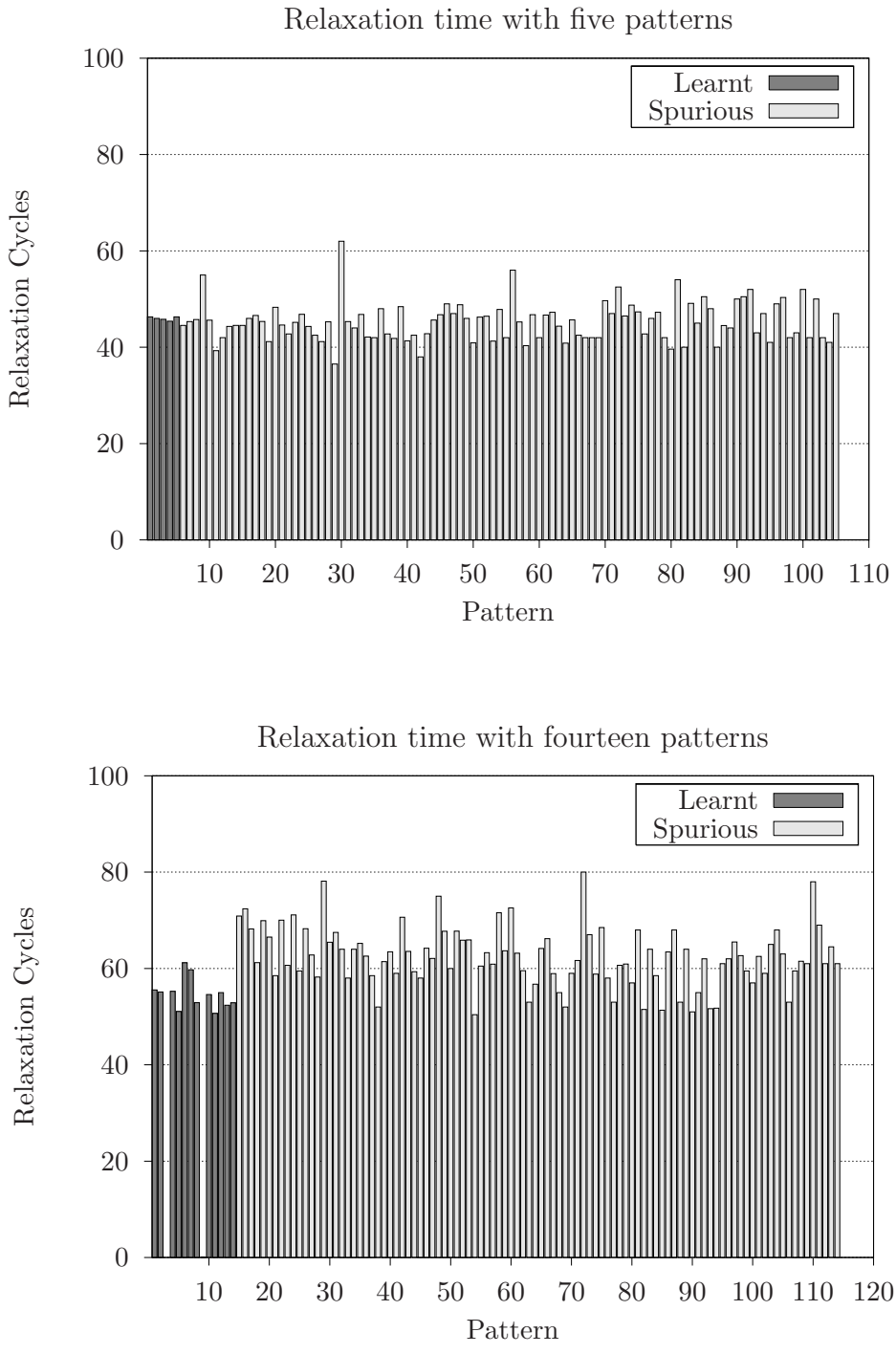


Figure 5.3: The relaxation time for stable learnt and spurious patterns in an $\mathcal{H}_{100,\pm}$ Hopfield network, trained with Hebbian learning. Five patterns with all stable (top) and fourteen patterns where only twelve were found during relaxation (bottom). First 100 spurious patterns found with 2000 probes (Randomly selected single run).

When a Hopfield network is overloaded, one of the mechanisms to restore the stability of the learnt patterns is to apply unlearning. The relaxation time to patterns changes after the application of unlearning. Horas and Bea (2002) show that it is possible to distinguish between learnt patterns and spurious patterns after unlearning. The network they are working with has asymmetric connections and stores patterns in cycles rather than just stable states. The system is only demonstrated for the static case where all patterns are learnt in one phase and unlearning is then applied before testing. For this implementation, the learnt patterns relax faster than the spurious patterns. This form of testing does not use random probing, but initialisation to a pattern that is close to the learnt patterns. Horas and Bea do not discuss whether this decrease may be caused by an increase in the number of spurious patterns close to learnt patterns, but they do use a fairly broad definition of successful retrieval, using an overlap $m_f = 0.9$. Thus, if 90% of the units are the same as a learnt pattern, then the current stable pattern is considered to be an accurate recovery of the original pattern. Our results for the distance traveled during relaxation do not provide a clear distinction between the types of stable patterns. This may be due to the differences in the way the patterns are stored, or that Horas and Bea (2002) use limit cycles rather than static stable states. A limit cycle occurs when the network cycles between a small set of states. Although there is no single stable pattern, the cycle repeats indefinitely and so can be considered stable.

O'Reilly, Norman, and McClelland (1998) use the distance that a probe moves as an estimate of whether the probe was a learnt pattern or a "lure", a probe that has some real world similarity to the learnt items. The specific measure used is the number of units that alter their activation state between presentation and relaxation. If a learnt pattern is still stable then none of the units will alter state during relaxation. O'Reilly *et al.* use patterns with a low coding ratio, which results in the pattern with no units active becoming a large spurious attractor. Almost all of the lure states will relax to the inactive pattern. This is yet another example where different samples are used when comparing results. The "lures" are not stable spurious patterns, but instead patterns generated by the researchers. The distinction between learnt patterns and most other unstable patterns can be found by simply measuring the energy of the pattern. Although this approach works well for lightly loaded networks, when the task is to differentiate known learnt patterns from artificially generated patterns, it is not sufficient for the current task as we require the differentiation of stable spurious patterns from stable learnt patterns. These are again exactly the patterns that this

discrimination method misclassifies.

5.2.3 Basin of Attraction Size

One of the most complex and involved measures of a stable pattern in a Hopfield network is the size of its basin of attraction. As discussed in Section 2.4.5, a basin of attraction is difficult, if not impossible, to calculate analytically from the weight matrix of the network. Without an analytical basis the only methods left involve sampling the network or using an estimate. By sampling the network with thousands of probes a percentage of pattern space that relaxes to each pattern can be found. The estimate of basin size used in this thesis, as described in Section 2.5.2, gives reasonable results while the basin is of reasonable size and relatively uniform.

The principle behind using the basin of attraction size as a differentiator is that learnt patterns should have larger basins of attraction than the spurious patterns. This is obviously true of lightly loaded networks without correlated patterns, where the learnt patterns are the large, stable attractors. Figure 5.4 shows the size of the basins of attraction (as found by sampling, and given by estimate) for learnt patterns and spurious patterns in an $\mathcal{H}_{100,\pm}$ network with Hebbian learning. In the lightly loaded network the basin size is an effective metric, but once the network is heavily loaded the basin of attraction size misclassifies many of the patterns. Although basin of attraction size is important as a measure of performance, it is not a good differentiator.

This measure cannot be used with some of the learning systems described in Chapter 4, such as the unlearning suggested by van Hemmen (1997), as the learnt patterns have virtually no basins of attraction.

5.2.4 Activation Saturation

For a pattern to be stable the input to each unit must be positively correlated with its output. A unit that has a high positive input and a positive output could be considered to have saturated its activation, as its output would still be positive with a lower, but still positive, input. The same is true for an inactive unit with a large negative input. The saturation is analogous to how far a unit is from the decision surface (described in Section 2.3.2). We can visualise the saturation profile of the units by sorting them on the summed input. The top graph in Figure 5.5 compares the saturation profile of the five learnt patterns versus the first ten spurious patterns found in an $\mathcal{H}_{100,\pm}$ network with Hebbian learning. The key observation is the difference in the way in

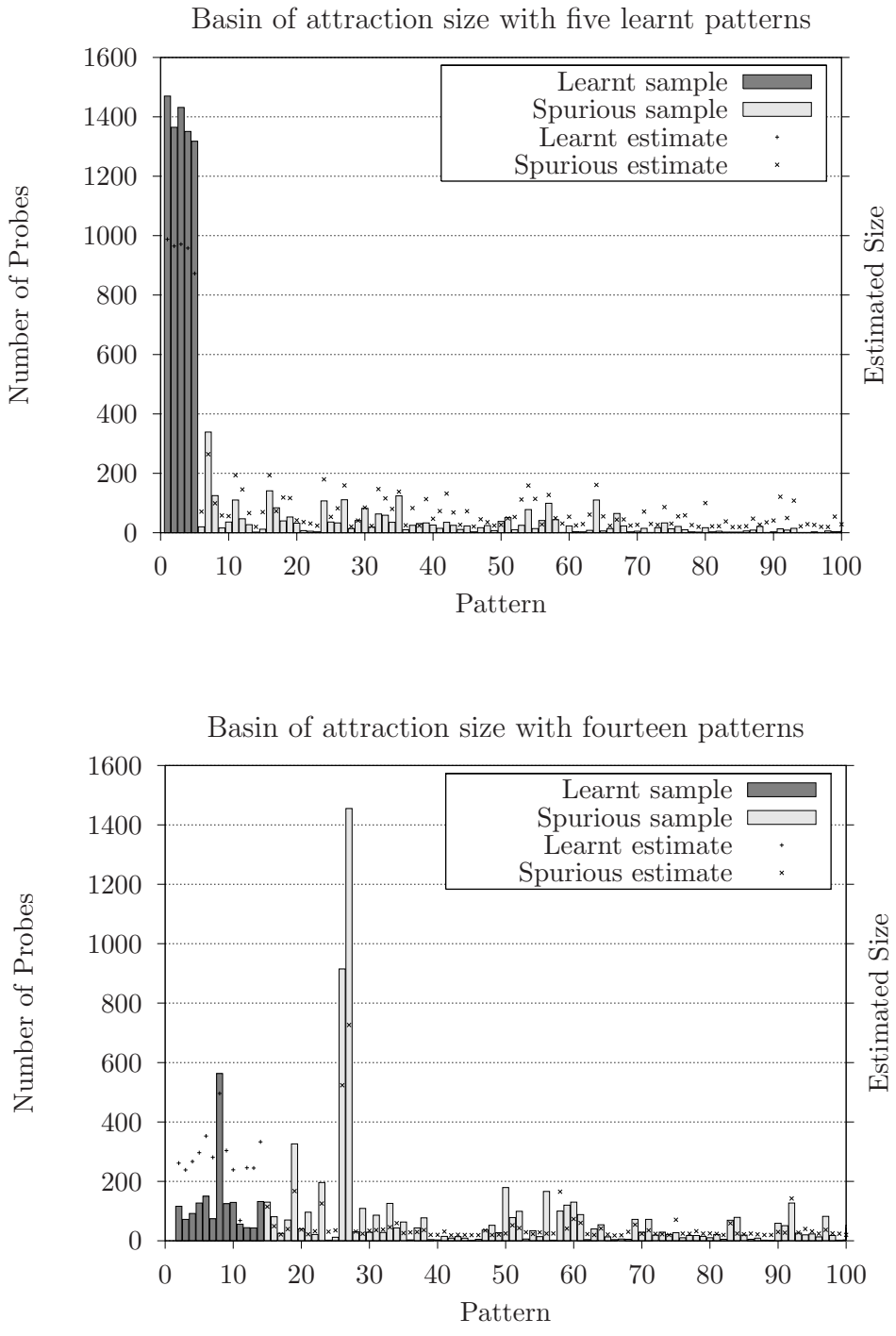


Figure 5.4: Basin size of stable patterns in an $\mathcal{H}_{100,\pm}$, both sampled and estimated. Hebbian learning of five patterns learnt (top) and fourteen patterns learnt (bottom) where only thirteen were stable. Spurious patterns are the first 100 patterns found with 2000 probes (Randomly selected single run).

which the profiles cross the zero axis. The stable patterns have a sharp transition, while spurious patterns have a smoother slope. It is this observation that suggests it might be possible to use the activation saturation as a means to differentiate the learnt and spurious patterns.

To turn the visual step into a metric for comparison we need to create a single number that represents the significance of the step. The step is the transition from negative activation to positive activation, and so the logical choice is to take the minimum positive input and subtract the minimum negative input. This gives a clear distinction in lightly loaded networks, like most other metrics, but unfortunately the distinction degrades as the number of learnt patterns increases. The bottom graph in Figure 5.5 shows the stable learnt patterns in an $\mathcal{H}_{100\pm}$ network with 14 learnt patterns versus the first 10 spurious patterns found. The saturation transition from negative to positive of some of the patterns has become similar to the spurious patterns. Figure 5.6 shows the transition as the "Step Size" for both the learnt patterns and the spurious patterns in an $\mathcal{H}_{100\pm}$ network. In the lightly loaded network the distinction is clear, however, as the network becomes heavily loaded the distinction becomes less clear.

The performance of this measure with delta learning is better than with Hebbian learning, both with only a few patterns learnt and as the number increases. This measure works well with delta learning if noise has been applied to the learning. The two types of noise that we have investigated as additions to delta learning are input noise and heteroassociative noise. These are described in Section 2.3.2.

Both of these types of noise force the units away from the decision surface creating the sharp transition seen in Figure 5.7. Spurious patterns do not have any pressure on units close to the decision surface and so the transition is much smoother.

The saturation profile of a pattern is able to differentiate patterns very effectively when the network is lightly loaded. However, as the number of stored patterns increases, the least stable units in the learnt patterns move closer and closer to the decision surface. The step size requires that the least stable active unit and the least stable inactive unit are far from the decision surface. As can be seen in Figure 5.7 there is a second visible distinction between the spurious and learnt pattern. The very low input and high input units in spurious patterns are much further from the decision surface that the equivalent units in the learnt patterns.

Units which are active with a high positive input have very low energy. Likewise units that are inactive with a large negative input also have very low energy. If we change from viewing the units by the total input, and instead order the units by their

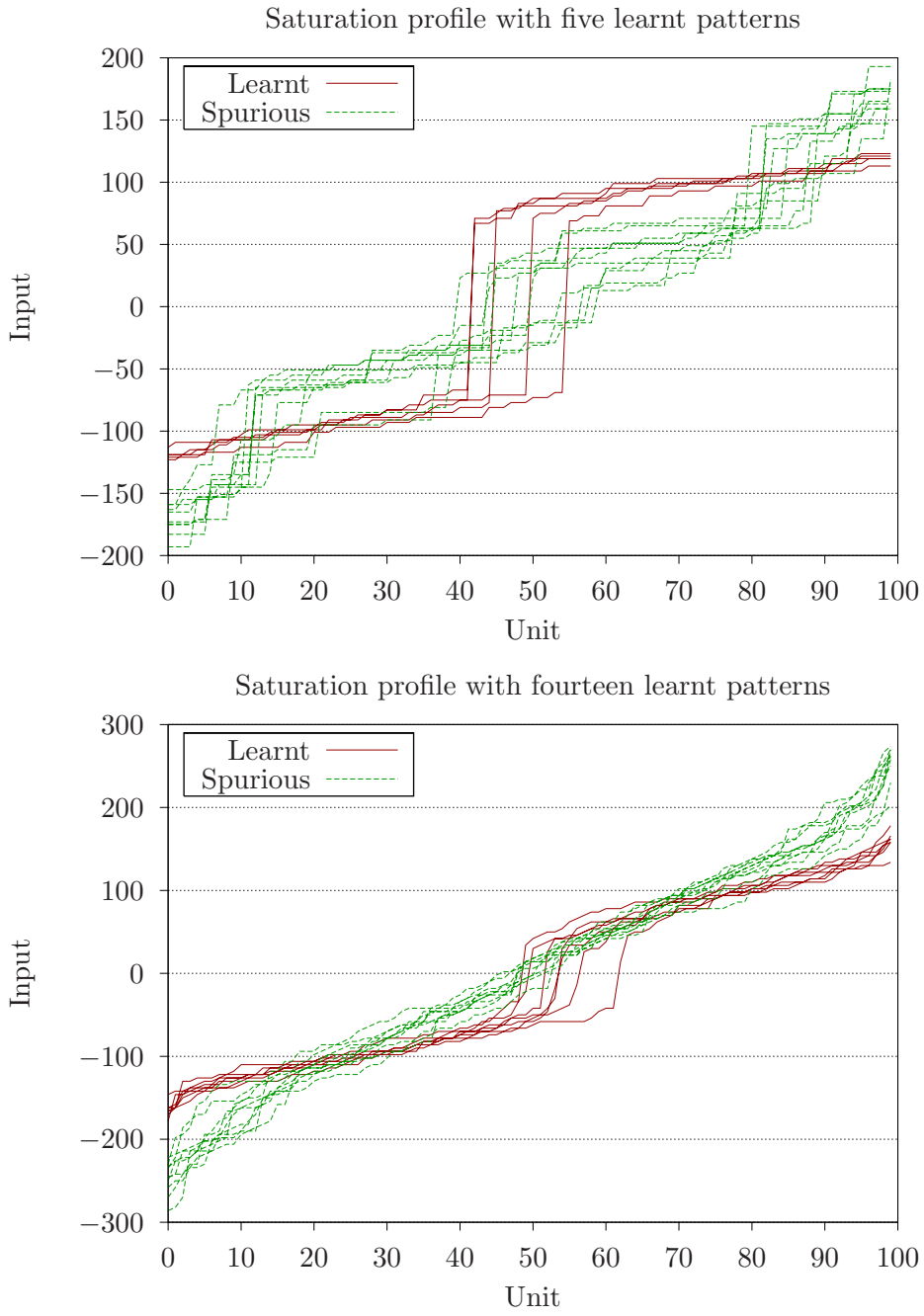


Figure 5.5: The saturation profile for the five learnt patterns (top) and the seven stable patterns out of fourteen learnt patterns (bottom) with Hebbian learning in an $\mathcal{H}_{100,\pm}$ network. The spurious patterns for both are the first ten spurious patterns found with 2000 probes. The unit numbers on the x–axis refer to the rank order, based on input value of the units in the individually sorted patterns.

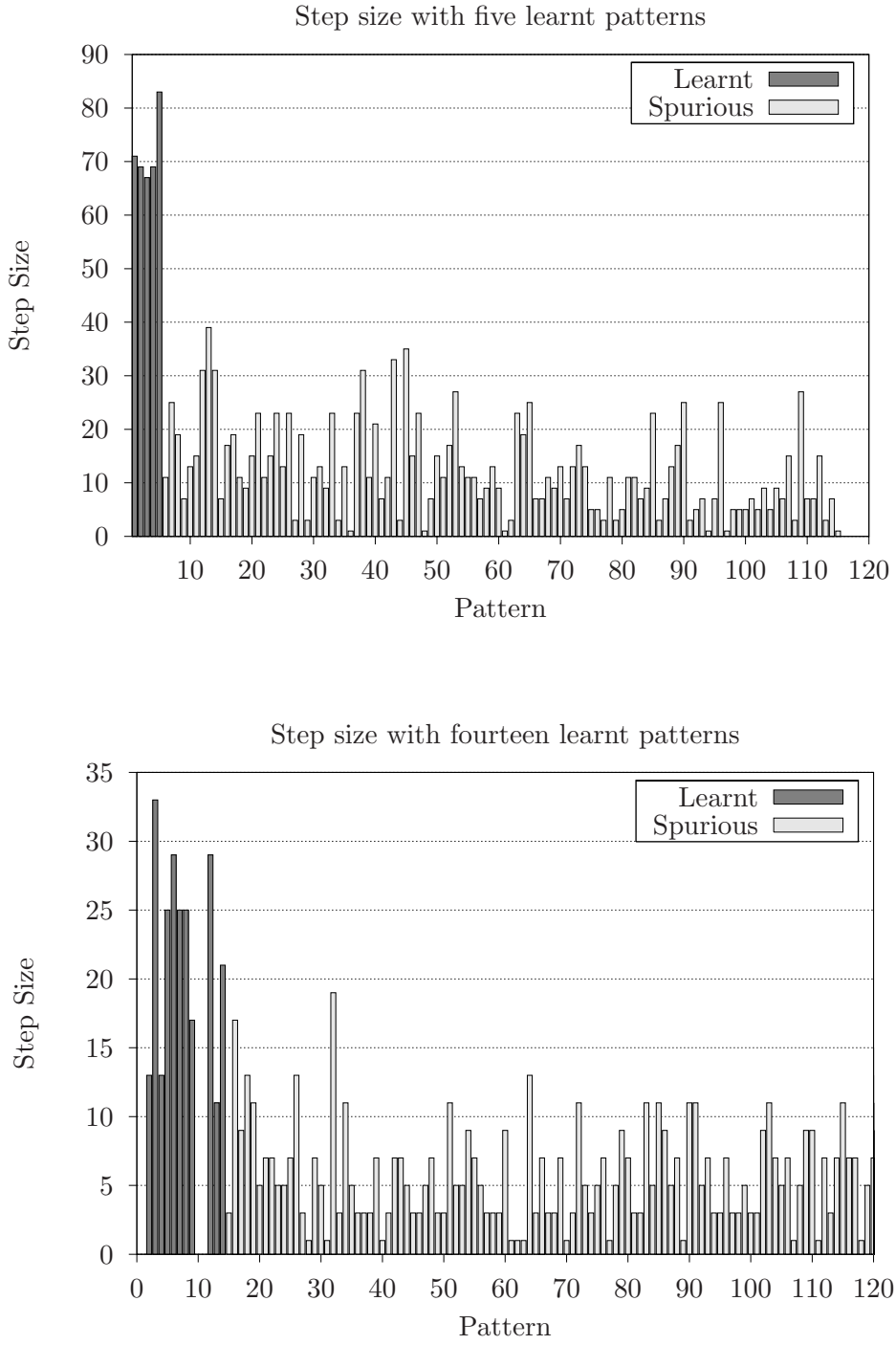


Figure 5.6: The size of the transition from negative input to positive input (step size) for learnt and spurious patterns, with 5 patterns learnt (top) and 14 patterns learnt (bottom).

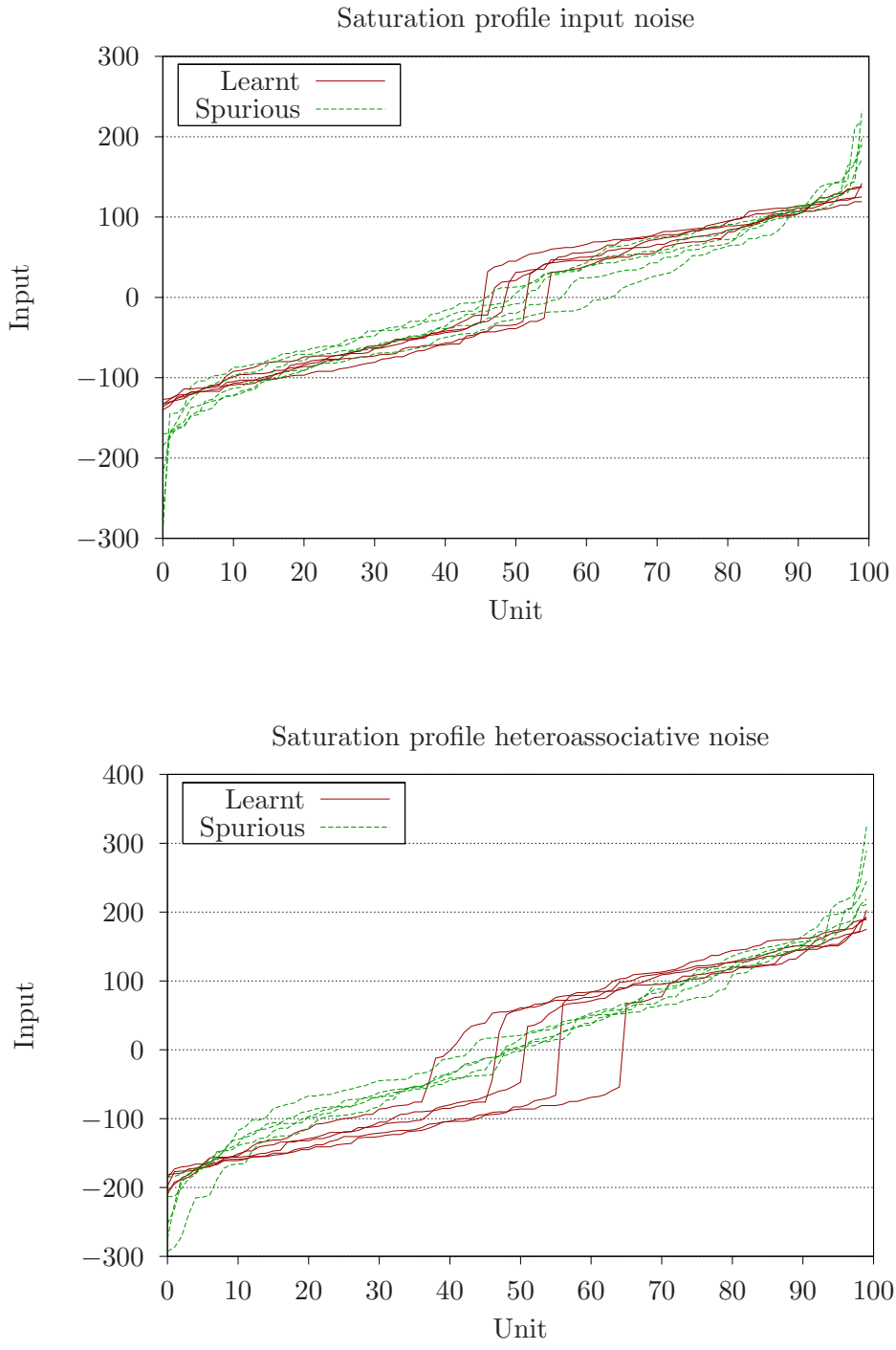


Figure 5.7: The saturation profile for the five most recently learnt and the first five spurious patterns in an $\mathcal{H}_{100,\pm}$ network with delta learning and different types of noise. Input noise ($\nu_i = 5$) (top); Heteroassociative noise ($\nu_h = 5\%$)(bottom).

energy, the very low energy units are grouped together. This changes the y–axis of subsequent graphs from "input" to "energy" of the units. When viewing the energy of units, the position on the y–axis is equivalent to the distance from the decision surface, with negative energy indicating units with more stability. We will call this type of graph an "energy profile".

5.3 Energy Profile

The energy profile of a pattern is the profile created from the energy of each unit sorted by the magnitude of the energy. Figure 5.8 shows the energy profile of four patterns after each of four Hebbian learning iterations. The total energy of the pattern is the sum of the individual units. For a pattern to be stable, all of the units have to have a negative energy[4]. The energy profile of an unstable pattern has units on the positive side of the y–axis.

Learning a pattern can be visualised as lowering each unit's energy until every unit in the pattern has a negative energy. For Hebbian learning with a learning constant η=0.5 the alteration changes the energy of each unit by $-(N-1)$ (remember that the connections between unit i and j are updated both by unit i and j, so the addition to each connection is 0.5×2). This can be seen in Figure 5.8 as the profiles move from being centred on zero to $-(N-1)$. After two patterns are learnt, "Pat 1" and "Pat 2" have identical energy profiles. This is always the case with Hebbian learning. As additional patterns are learnt, the crosstalk between patterns causes the slope of the energy profile to increase. This gives us an insight into the problem of catastrophic overloading and correlation when using Hebbian learning. If the highest energy unit has an energy greater than N before applying Hebbian learning to the pattern, the network will not learn the new pattern, as that unit's energy will not fall below 0 (the requirement for stability). It is at this point that catastrophic forgetting becomes a significant problem for Hebbian learning.

The act of creating a stable pattern results in an energy profile that is very different to the profiles of spurious patterns which arise from the crosstalk and correlations of the

[4]In our 2004 paper (Robins and McCallum, 2004) we used positive correlations between input and output for individual unit values and negative energy for patterns. For individual units the positive correlation between the input and the output is equivalent to the negative of the energy. For consistency we now only refer to energy with reference to the contribution it makes to the stability of the pattern. Thus, individual units that are stable are now referred to as having negative energy, rather than positive correlation between input and output.

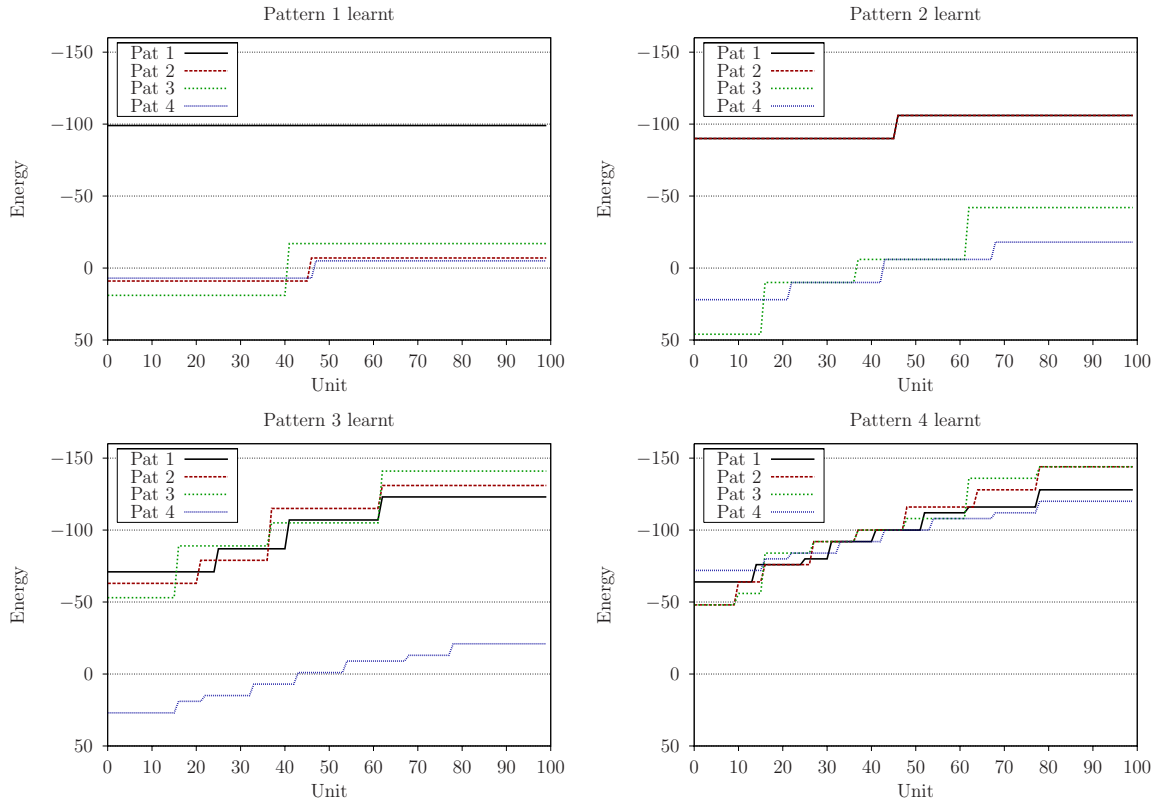


Figure 5.8: The energy profile of four patterns in an $\mathcal{H}_{100,\pm}$ Hopfield network as the patterns are learnt in succession.

learnt patterns. In Hebbian learning the learnt patterns are moved uniformly leaving the slope the same while decreasing the total energy.

While the network is lightly loaded the energy profile of the spurious patterns is very different to the profile of the learnt patterns. Figure 5.9 shows the profiles of learnt and spurious patterns after four patterns have been learnt with Hebbian learning. Note that both the position and slope of the learnt patterns is different to the position and slope of the spurious patterns. The total energy is lower for the learnt patterns[5] and they have a flatter profile than the spurious patterns. The most significant indicator of the difference in slope is at the beginning and end of the profile. The lowest energy units in a learnt pattern are higher than the spurious patterns and the highest energy (least stable) units are lower.

Although the total energy can no longer be used to differentiate patterns in a heavily loaded network, there is still a large difference in the profiles of the stable

[5]This is consistent with the ability to use total energy to differentiate patterns in a lightly loaded network.

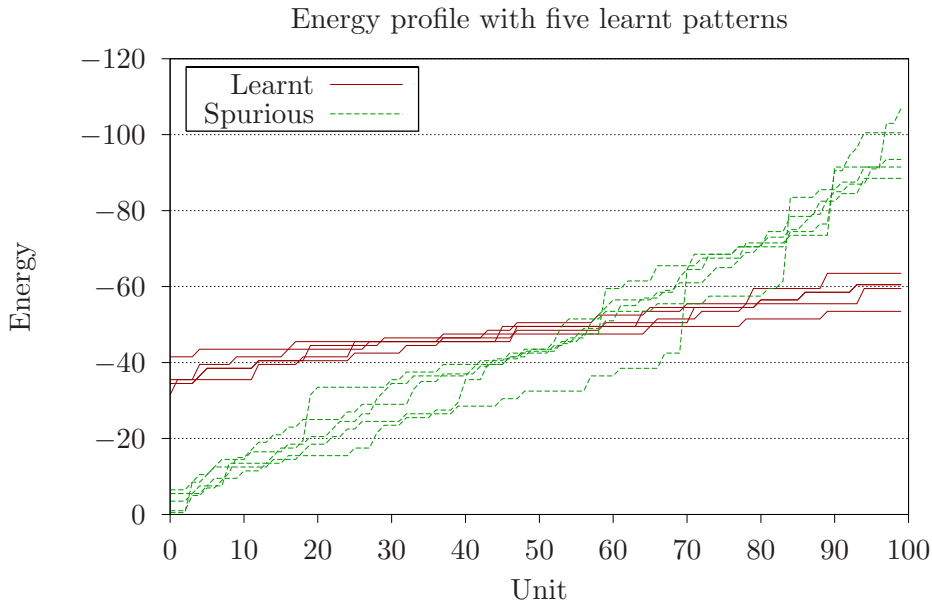


Figure 5.9: Energy profile of the first five learnt patterns and the first five spurious patterns found with probing. Hebbian learning applied to five patterns in an $\mathcal{H}_{100,\pm}$ network.

learnt patterns and spurious patterns. Figure 5.10 shows the profiles of the stable patterns with fourteen patterns learnt. There are now spurious patterns with total energies lower than the most stable of the learnt patterns, but the slopes of the profiles are still different.

Changing the learning procedure changes the energy profiles of the learnt patterns. For example, the inclusion of weight decay decreases the magnitude of the energy of every unit before each new pattern is learnt. This moves the units in the energy profile closer to 0. As long as the energy of an individual unit is not greater than N, Hebbian learning's adjustment of $-N-1$ will be large enough to learn the new pattern. Figure 5.11 shows the energy profile of the last five patterns learnt using Hebbian learning with weight decay of $d = 0.1$. This graph shows the profiles of patterns after the presentation of 50 patterns[6]. The most recently learnt pattern has the lowest energy as it has not experienced any weight decay. The older learnt patterns have units that are closer to 0, and patterns older than the immediate $0.06N$ are lost as the crosstalk from learning new patterns moves some of the units to the positive side of 0 on the y–axis (decision surface). Once any unit has positive energy in a pattern,

[6]50 patterns are used to ensure that the network has passed the initial learning phase where the number of stable patterns peaks and then drops back to a sustainable level.

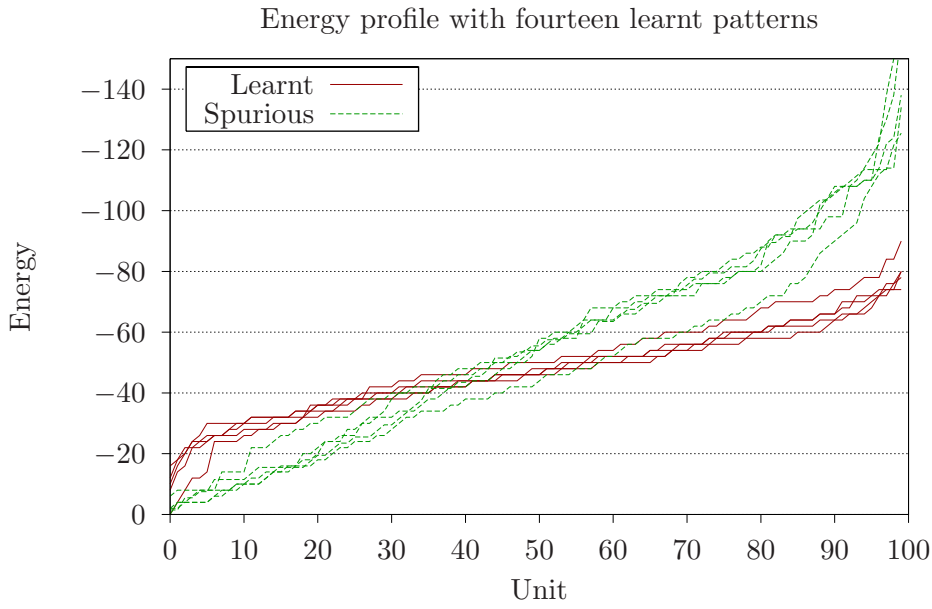


Figure 5.10: Energy profile of the first five learnt patterns and the first five spurious patterns found with probing. Hebbian learning applied to fourteen patterns in an $\mathcal{H}_{100,\pm}$ network.

that pattern becomes unstable.

The profiles of patterns that have been learnt with delta learning also have this distinctive shallow slope. Figure 5.12 shows the learning of 4 patterns after various numbers of epochs. After just a single epoch, about 50% of units have recieved a learning update of η= 0.1. As the weights are updated in both directions (as this is a 100 unit network), the energy of the updated units averages about -20. After a second epoch, a further 25% of the units have been updated. At the top end of the graph a few units have recieved a second learning update as the Gaussian noise will very occasionally force already learnt units to update. By the 50^{th} epoch the energy profile has flattened out significantly. All patterns become stable between approximately 5 and 10 epochs, but delta learning continues to alter the connections that are affected by noise. Units that already have a large input are far less likely to be affected by noise, and as a result the learnt patterns have a flatter energy profile.

Spurious patterns created during delta learning tend to have some units which are very close to the decision surface and some which are very far away. This observation can be explained by the spurious patterns having some units which are the correlations between learnt patterns, while other units are the uncorrelated activation across multiple learnt patterns.

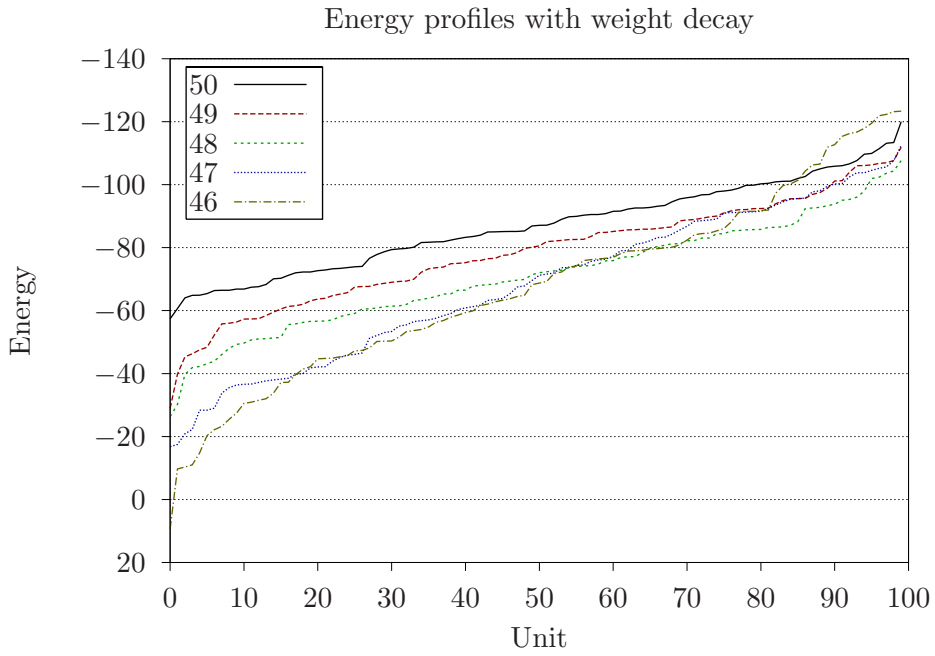


Figure 5.11: Energy profile of the last five patterns learnt in an $\mathcal{H}_{100,\pm}$ using Hebbian learning and weight decay of $d = 0.1$ after 50 patterns have been learnt.

In the next section we present the energy ratio measure, which compares the most stable and least stable units. This value can be used to differentiate patterns without having to perform extensive global calculation or massive probing of the network.

5.3.1 Energy Ratio

The energy ratio is a measure we presented in Robins and McCallum (2004), and is based on the energy profile of a stable pattern. It is similar to the saturation step metric above, but it takes into consideration both the low and the high end of the profile. It calculates a combination of the slope and magnitude of the energy profile. We define a pattern's energy ratio to be:

$$R = \frac{\sum_{i=1}^{k} h_i}{\sum_{j=N-k}^{N} h_j} \tag{5.2}$$

where the units have been sorted by their energy, k is the number of units contributing to the measure, and h_i is the energy of unit i. The flatter the energy profile of a pattern, the closer the energy ratio measure is to 1.0.

Figure 5.13 shows the energy profile and energy ratio of a learnt patterns and the first spurious pattern found when probing the network. The k units contributing to

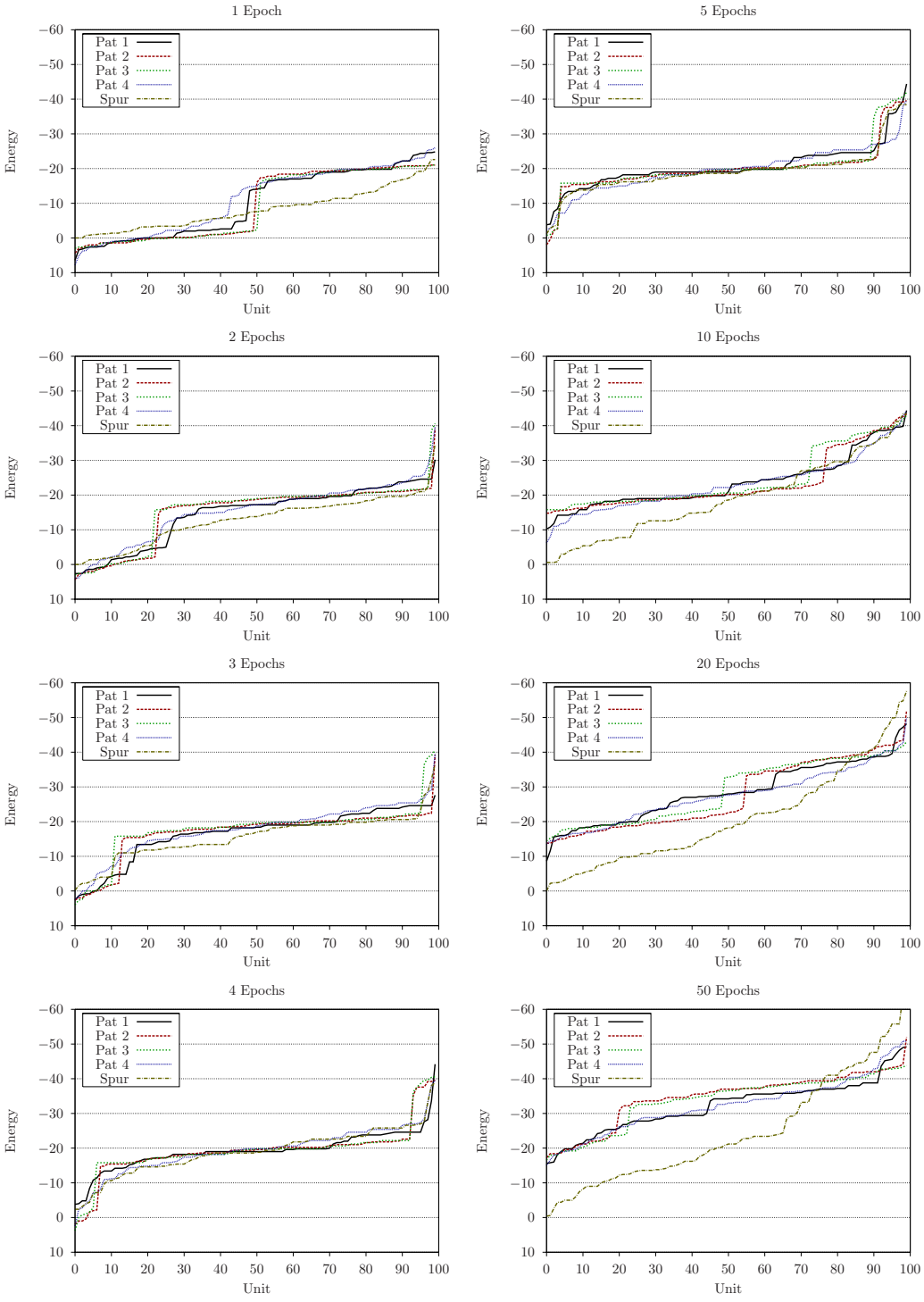


Figure 5.12: The energy profile of four learnt patterns in an $\mathcal{H}_{100,\pm}$ network as the patterns are learnt using delta learning η=0.1 with input noise ν_i=0.5. The "Spur" is the first spurious pattern found by random probing. The epochs are listed in columns with energy profiles shown for 1, 2, 3, 4, 5, 10, 20, and 50 epochs.

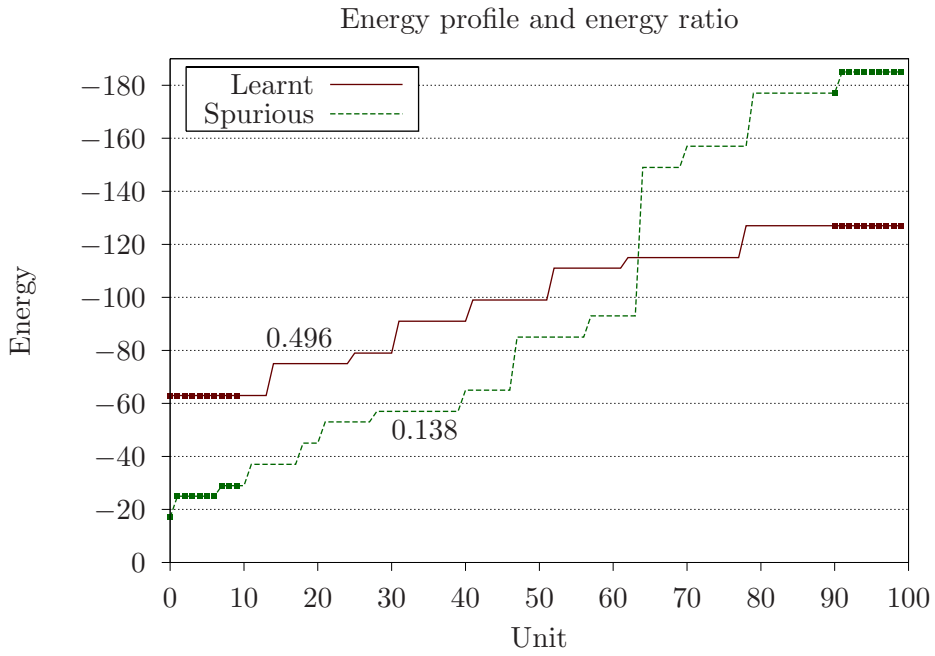


Figure 5.13: Energy profile and energy ratio for a learnt pattern and a spurious pattern in an $\mathcal{H}_{100\pm}$ with five patterns learnt using Hebbian learning. Energy ratio with $k = 10$.

the ratio are shown with points. The ratio for the spurious pattern is less than one third of the learnt pattern.

This metric can be seen as an estimate of the combination of standard deviation and mean, similar to the coefficient of variance, but done without calculating the sum of squares or averaging over the whole network. The energy ratio can be seen as:

$$R \approx \frac{\mu - (2\sigma)}{\mu + (2\sigma)} \tag{5.3}$$

where μ is the mean energy for the network in the current configuration, σ is the standard deviation, and R is the ratio measure defined above. The standard deviation measures how uniform the energies are, while the mean is equivalent to the total energy of a pattern. Once a pattern is stable, lowering the energy of the pattern lowers the mean and therefore increases the ratio, as $(a/b) < (a+c/b+c)$ where c is the increase, a is the sum of the least stable units and b is the sum of the most stable units. Delta learning also increases the ratio as it primarily makes changes to units which are part of a ($(a+c)/b < a/b$). When there are a large number of unstable units in a pattern the energy ratio of the pattern becomes negative. This is caused by the unstable units having a positive energy and the stable units having negative energy, resulting in a

negative value for the division a/b. Unstable patterns usually have a negative ratio, while most stable spurious patterns have ratios that are near zero. It is possible for an unstable pattern to have a positive ratio, if the sum of the energy of the unstable units in a is less than the stable units in a. The ratio will be low but not necessarily negative. This effect can be seen in Figure 5.14 where some of the learnt patterns are unstable, but still have a positive ratio. These patterns only have one or two units unstable out of the k=10 units used to calculate a.

5.3.2 Hebbian Learning and Energy Ratio

Hebbian learning uniformly lowers the energy of the units in the learnt patterns. This does not change the standard deviation of the energy profile, but does change the mean and total energy. Thus the learnt patterns have a standard deviation similar to a randomly selected pattern before relaxation, which is different to a stable spurious pattern. Learnt patterns also have a low mean energy and, when combined with the low standard deviation, this gives a high energy ratio.

As the number of learnt patterns presented increases, the network overloads and the learnt patterns become unstable. However, the ratio measure still performs reasonably well with only a few patterns being misclassified. Figure 5.14 shows the energy ratios for learnt and spurious patterns after learning either five patterns (top) or fourteen patterns (bottom). The energy ratio of the learnt patterns in a lightly loaded network are much higher than the energy ratio of the spurious patterns. In the heavily loaded network with $0.14N$ patterns learnt, five of the patterns have become unstable, while the rest retain a comparatively high energy ratio. In this example there is also a spurious pattern that has an energy ratio similar to the learnt patterns. Ideally the threshold for a lightly loaded network would be higher than for the heavily loaded network. A threshold of 0.4 would classify all the patterns correctly in the lightly loaded network while a threshold of 0.2 would classify all the learnt patterns correctly and misclassify only one spurious pattern when the network is heavily loaded.

The quality of a differentiator needs to be assessed using the analytical tools described in Table 5.1. The top graph in Figure 5.15 shows the PPV, NPV, TPR, and TNR for the simple energy ratio threshold of 0.2. The positive predictive value (how likely a pattern is to be a learnt pattern if the ratio measure defines it as learnt) drops to just over 50% with four patterns learnt. This is because a threshold of 0.2 is much lower than the ratios of the learnt patterns, and about half of the spurious patterns (TNR of 40%) make it above this value so are classified as learnt patterns. The TPR

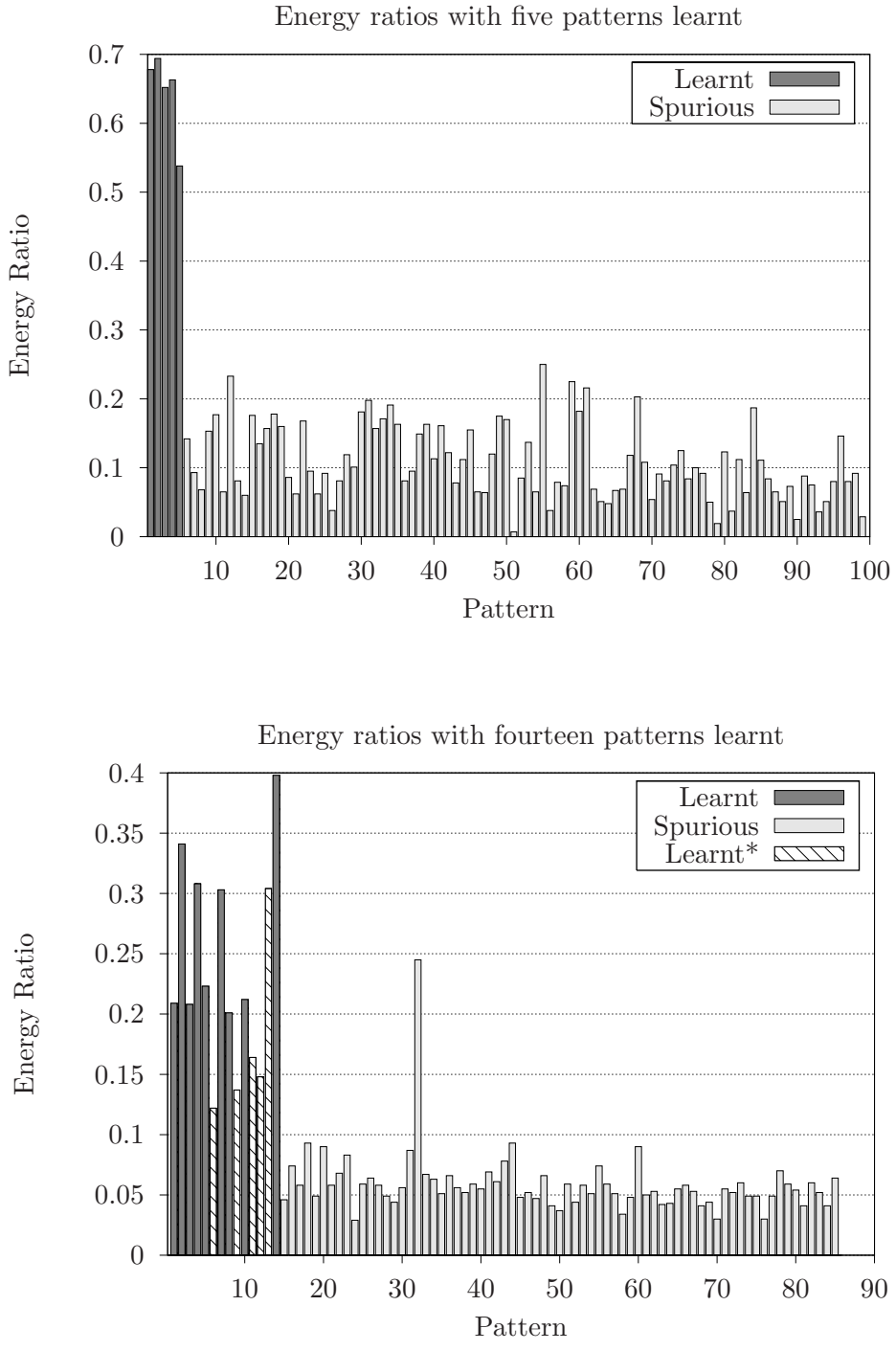


Figure 5.14: The energy ratio of all the learnt patterns and up to the first 100 spurious patterns in an $\mathcal{H}_{100,\pm}$ network after five patterns (top) and fourteen patterns (bottom). "Learnt*" are the unstable learnt patterns. This data is from the same simulation as the data in Figure 5.10.

represents the percentage of learnt patterns that are actually being classified as learnt patterns. This falls off quite rapidly as the network becomes increasingly overloaded.

Using other values for the energy ratio threshold changes the four analytical metrics above. The second graph in Figure 5.15 shows the results with a threshold of 0.25. Note that both the PPV and TNR are higher, as would be expected by making the threshold higher, while the TPR falls off faster given the more restrictive requirement of having a ratio above 0.25.

The energy ratio measure works well with Hebbian learning. If the number of patterns that have been learnt is known, the threshold can be set to maximally distinguish between learnt and spurious patterns. However, Hebbian learning is not the algorithm best suited to the energy ratio measure. Delta learning with its explicit alteration of units that are close to decision surface, creates patterns with much cleaner energy ratios.

5.3.3 Delta Learning and Energy Ratio

Delta learning only lowers the energy of units that generate an error (a delta). By only altering these units, delta learning generates very flat energy profiles which have low standard deviation. However, without noise the patterns are only just stable and so the energy ratio is still quite low. Figure 5.16 shows the energy profile of patterns that have been learnt without any form of noise applied. Note that a number of the units are very close to the decision surface (0 on the y–axis). Perhaps the most striking feature is the step in the middle of the energy profiles. This is a result of the units on the right of the step receiving an update from delta learning, while units on the left are stable merely from the cross talk of the other patterns.

There are two types of noise that we have applied to the delta learning process. These are described in Section 2.3.2. The two different noise types have different effects on the energy profiles and the energy ratio, but both still create learnt patterns with high ratios and spurious patterns with low ratios.

Figure 5.17 shows the energy profile of the stable learnt patterns after 50 patterns have been learnt in succession, using delta learning with a learning constant of $\eta = 0.1$ and input noise applied with an sigma of $\nu_i = N$. Only the most recently learnt patterns are still stable but they still have the characteristic shallow slope of learnt patterns. The energy ratio for the learnt patterns is very different to the spurious patterns as shown in Figure 5.18. With 30 patterns learnt, only the most recently learnt patterns are stable. Any threshold between 0.14 and 0.18 would correctly assess

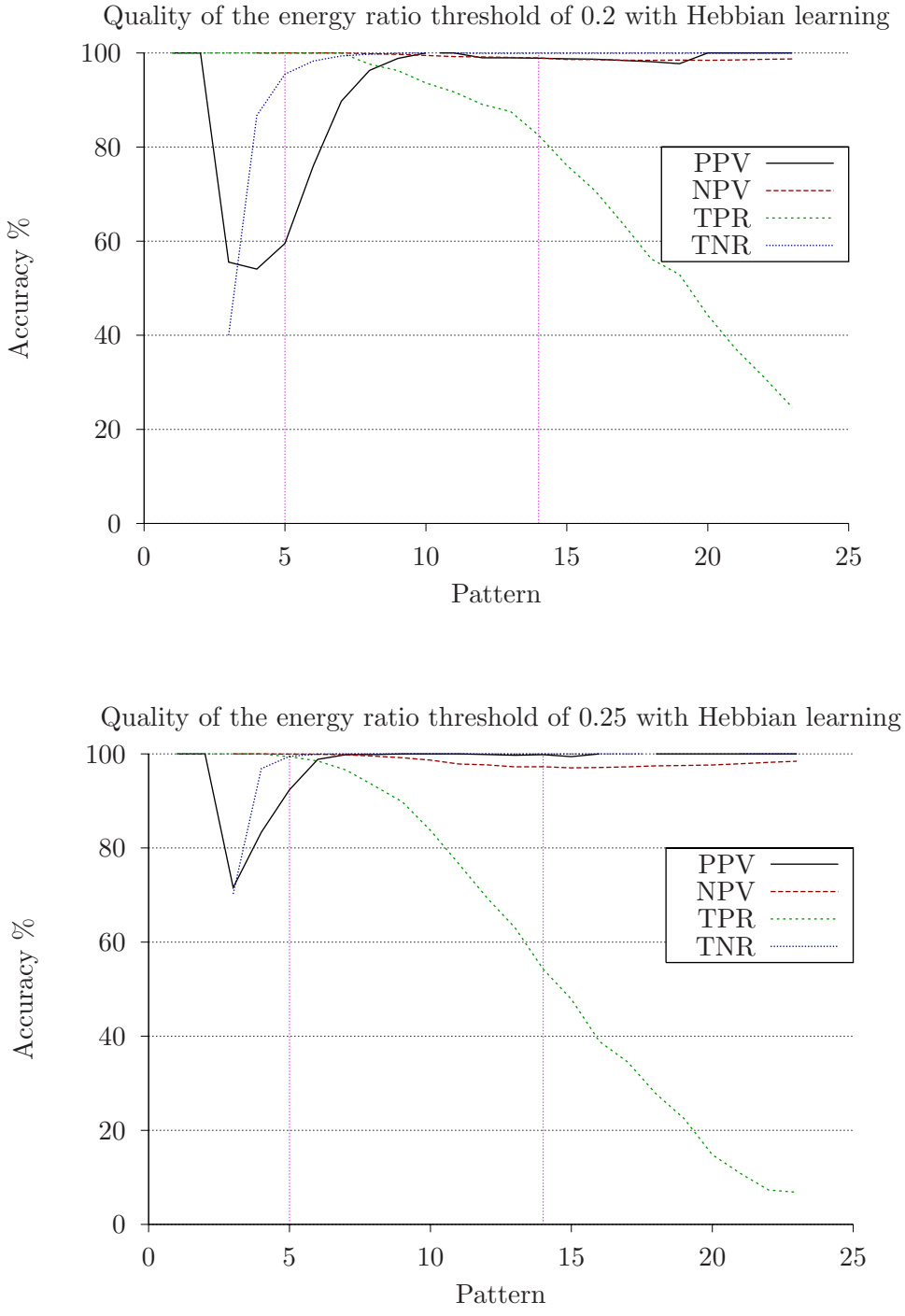


Figure 5.15: The positive predictive value (PPV), negative predictive value (NPV), true negative rate (TNR), and true positive rate (TPR) for an energy threshold of 0.2 (top) and 0.25 (bottom) in an $\mathcal{H}_{100,\pm}$ with Hebbian learning from one pattern learnt to 25 patterns (100 repetitions). Vertical lines represent the points used for graphs in Figure 5.14.

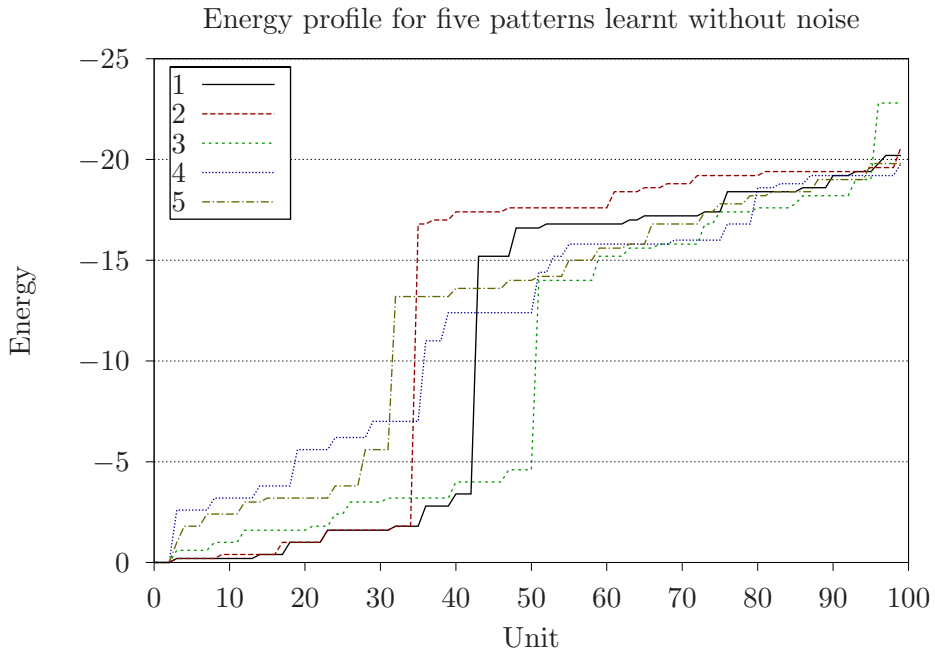


Figure 5.16: Energy profile of five patterns learnt with delta learning with a learning constant of 0.1 and no noise.

every stable pattern in this particular instance.

Changing the energy ratio threshold to the best possible value for each run would improve the results, but is unrealistic. Ideally we would be able to use a set value that works without knowing how many patterns have been learnt. Figure 5.19 shows the quality measures for energy ratio thresholds of 0.2 and 0.25. Both of these values perform reasonably well once a few patterns have been learnt. They also demonstrate the trade–off between misclassifying the learnt patterns with a high threshold and allowing too many spurious patterns into the learnt classification with a lower threshold. The PPV is about $98 - 99\%$ with a threshold of 0.2 and nearly 100% with the more restrictive 0.25. The TPR, the percentage of stable learnt patterns that are correctly identified, is around 84% with 0.2 and drops to about 70% for 0.25.

Heteroassociative noise is not directly focused on changing units which are close to the decision surface. Instead it alters the activation of some of the units in the network after the desired pattern has been set. Therefore it is actively building the basin of attraction of a pattern by forcing patterns close to the learnt pattern to relax to this desired pattern. Figure 5.20 shows the equivalent set of energy profiles as Figure 5.17, but with heteroassociative noise applied. In this example, 5% of the units in the learnt patterns had their output changed in each epoch. Only a few of the patterns are stable,

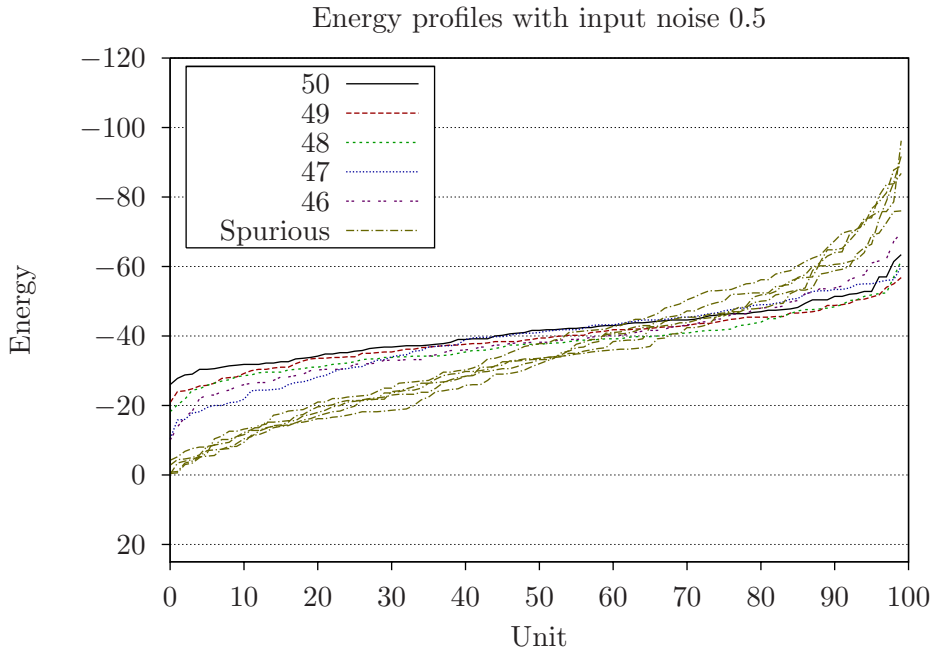


Figure 5.17: Energy profile of the last five patterns learnt and five spurious patterns with input noise and delta learning. Noise level with absolute range of sigma = N, learning constant $\eta = 0.1$ on the unit inputs, error criterion 0.001.

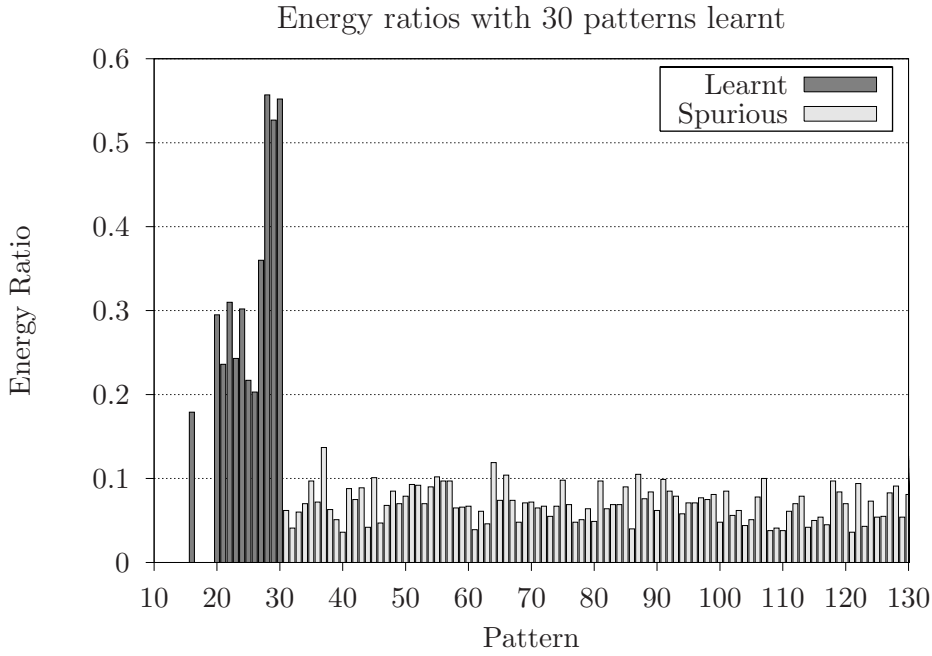


Figure 5.18: The energy ratio of the stable learnt patterns and up to the first 100 spurious patterns in an $\mathcal{H}_{100,\pm}$ Hopfield network after 30 patterns have been learnt with delta learning ($\eta = 0.1$, input noise $\nu_i = N$, 2000 probes).

but again they exhibit the shallow slope and a relatively high ratio.

Heteroassociative noise increases the size of the basin of attraction and in doing so also improves the energy ratio measure. Figure 5.21 shows the energy ratio for the stable learnt and spurious patterns after 30 patterns have been learnt. Figure 5.22 shows the quality measures for energy ratio thresholds of 0.2 and 0.25. The energy ratio measure is still performing well with this type of learning.

The energy ratio measure works well with both Hebbian learning and delta learning with two different types of noise. A threshold value of 0.2 performs well if the percentage of learnt patterns identified is important, while a threshold of 0.25 can be used when it is important to avoid misclassifying the spurious patterns.

5.4 Discussion

There have been many attempts to improve the quality of the results generated by Hopfield networks. Given that spurious patterns seem to be endemic to the Hopfield type networks, much of the effort has gone into either decreasing the number of spurious

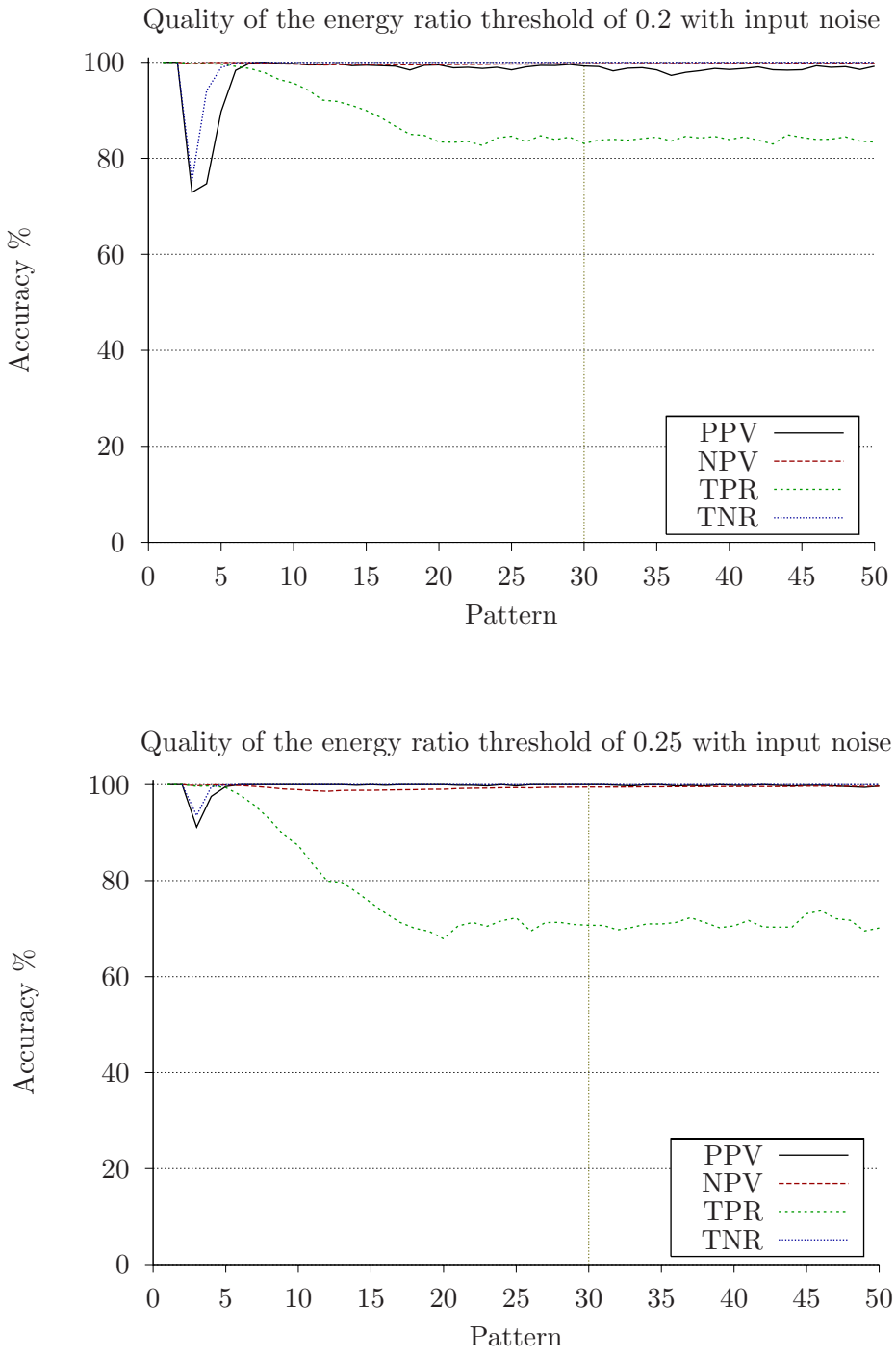


Figure 5.19: The PPV, NPV , TNR and TPR for an energy threshold of 0.2 (top) and 0.25 (bottom) in an $\mathcal{H}_{100,\pm}$ with delta learning (η=0.1) and input noise of 0.5 (100 repetitions). The vertical line represents the point used for Figure 5.18.

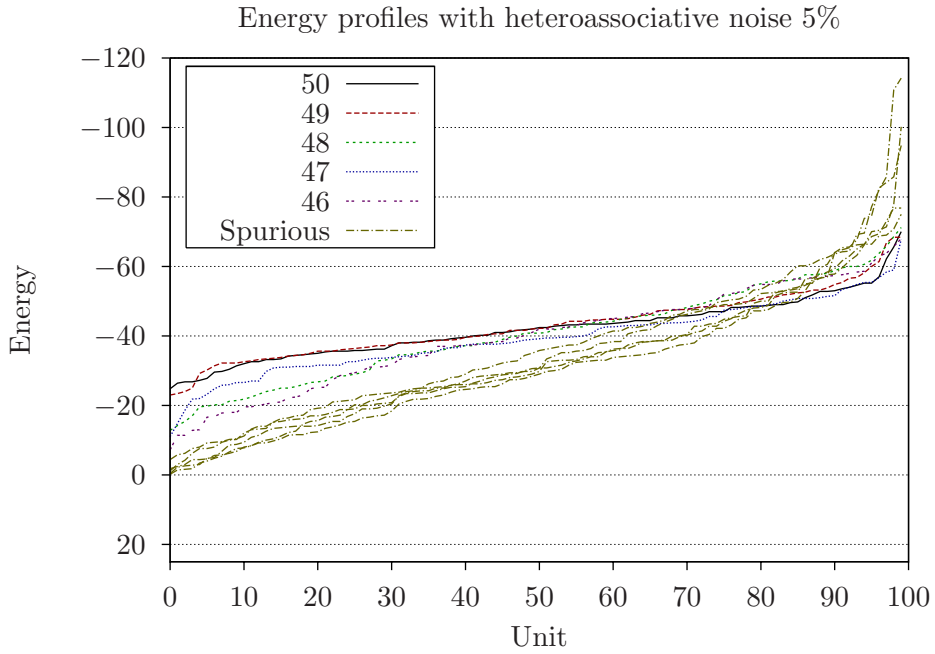


Figure 5.20: Energy profile of the last five patterns learnt and five spurious patterns with heteroassociative noise $\nu_h = 5\%$ delta learning, learning constant $\eta = 0.1$, error criterion 0.001, and epoch limit = 500.

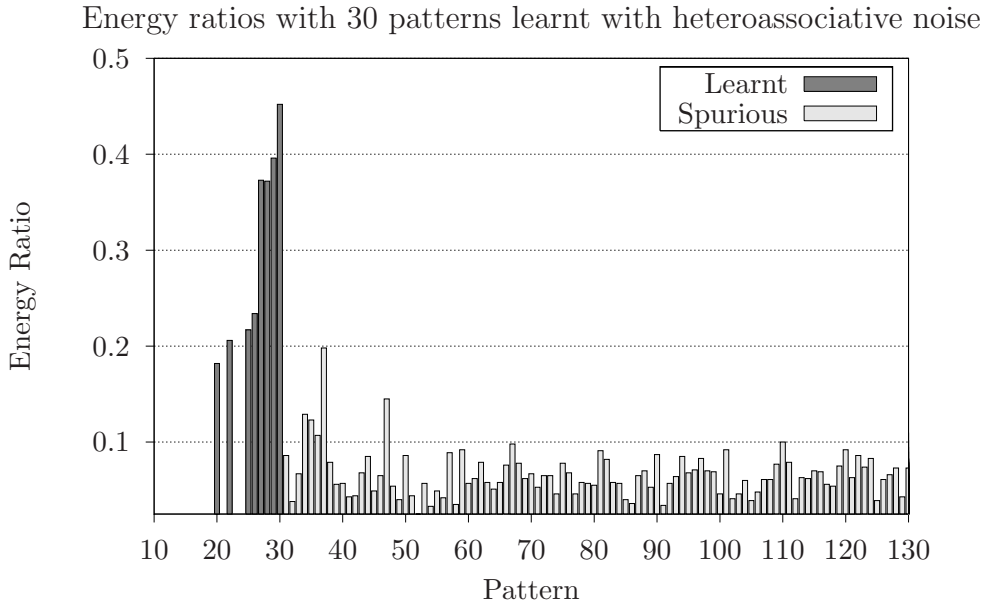


Figure 5.21: The energy ratio of the stable learnt patterns and up to the first 100 spurious patterns in an $\mathcal{H}_{100,\pm}$ Hopfield network after 30 patterns have been learnt with delta learning ($\eta = 0.1$, heteroassociative noise $\nu_h = 5\%$, 2000 probes).

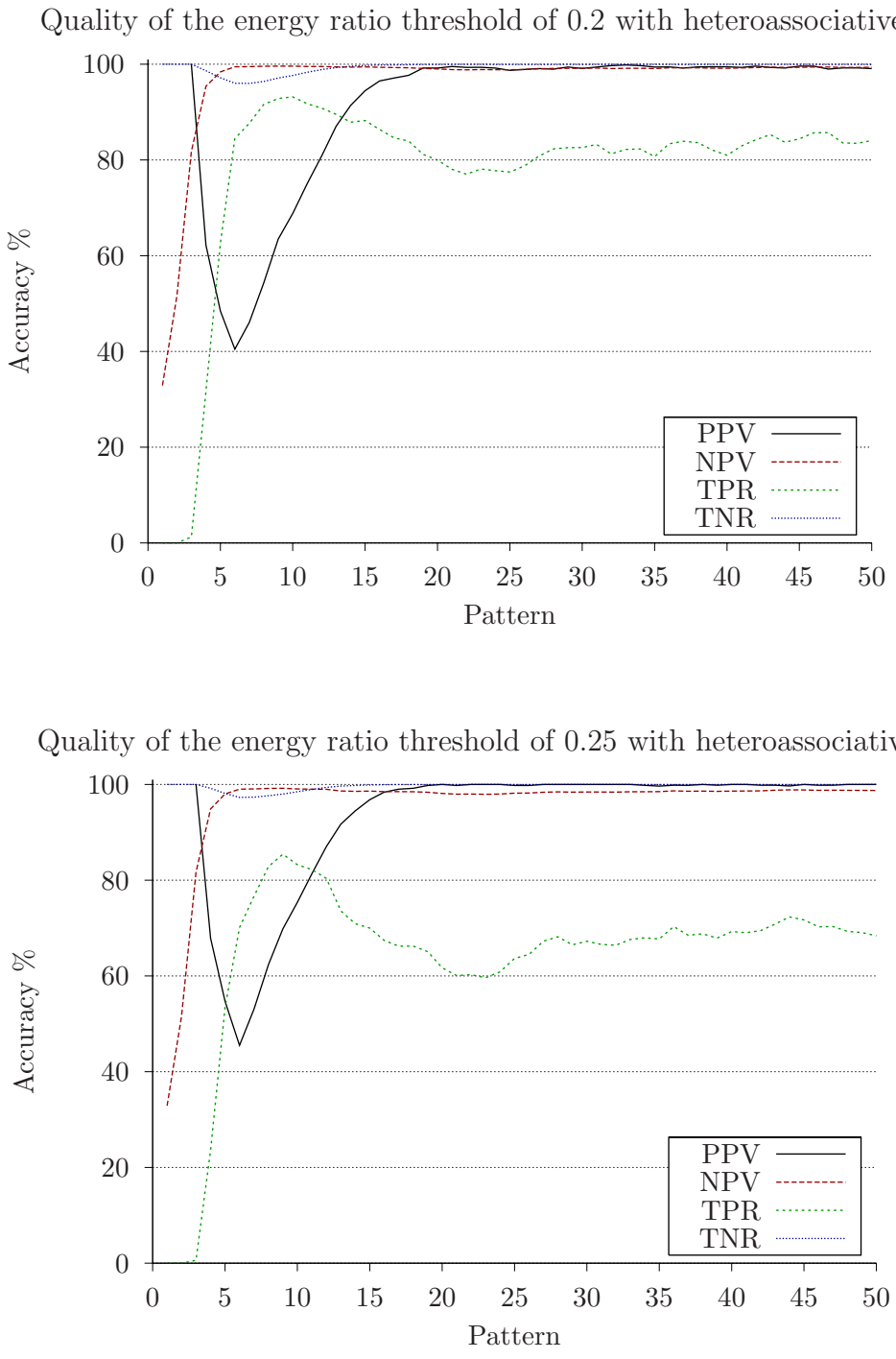


Figure 5.22: The PPV, NPV , TNR and TPR for an energy threshold of 0.2 (top) and 0.25 (bottom) in an $\mathcal{H}_{100,\pm}$ with delta learning (η=0.1) and heteroassociative noise $\nu_h = 5\%$ (100 repetitions).

patterns or differentiating the learnt patterns from the spurious patterns. Most of the measures perform well when the network is lightly loaded, but begin to fail as the network becomes heavily loaded.

When the task is simply to differentiate a learnt pattern from an unstable randomly generated pattern, there are a number of useful metrics, including total energy and number of units altering state. In this case the learnt pattern does not even need to be stable, it must merely have enough of an imprint of the changes made during learning to differentiate it from the majority of random patterns. Bogacz (2001) uses the total energy of unstable learnt patterns to generate a familiarity discrimination model of the perirhinal cortex.

The task that we are interested in is the classification of the stable patterns. The two best metrics appear to be the saturation step size and the metric we propose, energy ratio. The saturation step size is similar to the energy ratio but it only uses the units that are close to the activation threshold. This is equivalent to the high energy area of the energy profile of a stable pattern. In randomly distributed patterns, the units involved in the step calculation are included in the energy ratio measure (they are the numerator (a) of the ratio a/b). The energy ratio also incorporates the difference between the extremely low energy units in learnt versus spurious patterns. Spurious patterns have some units which have very low energy. This helps to classify spurious patterns which the step misclassifies.

The energy ratio measure gives us the ability to adjust the responsiveness of the system between forgetfulness and fantasy. A high threshold will label more of the learnt patterns as spurious, and thus will be forgetful, but will very rarely misclassify spurious patterns as learnt patterns, and so avoids the fantasy of *déjà vu*. A low threshold will identify the learnt patterns but will also misclassify a larger number of spurious patterns. The results above for 0.2 and 0.25 give an indication that somewhere around these values is a reasonable threshold for creating a differentiator with high positive predictive value.

The energy ratio measure is a very useful metric for assessing whether a pattern is a learnt pattern. It works with both Hebbian learning and delta learning with various types of noise. This metric can be added to the response of a Hopfield network and it will improve the quality of the response by including an estimation of how likely the pattern is to be a spurious or learnt pattern.

In Chapter 6 we will use the energy ratio to augment the system level solutions of unlearning and pseudorehearsal presented in Chapter 4. The intention of the unlearning

procedure was to remove the spurious patterns leaving the learnt patterns, as the only stable patterns in the network. The energy ratio measure gives us a way of ensuring that this is the case. Pseudorehearsal is trying to reinforce the learnt patterns, but in doing so it also reinforces stable spurious patterns. The energy ratio measure can be used to selectively reinforce only those patterns that appear to be learnt patterns. The effectiveness of the energy ratio measure will be shown by an improvement in the performance of the Hopfield network, both in terms of the number of patterns and the size of their basins of attraction.

Chapter 6

Pattern Knowledge used with Unlearning and Pseudorehearsal

The results from Chapter 5 show that we can distinguish learnt patterns from the spurious patterns. In this chapter we discuss how this information might be used to improve the performance of the two system level solutions to catastrophic forgetting.

6.1 Perfect Unlearning

In the early papers on unlearning it was suggested that if unlearning could be restricted to just the spurious patterns then it might be possible to improve its performance (Hopfield *et al.*, 1983; van Hemmen, 1997).

Before testing the energy ratio measure as a way of restricting the unlearning to spurious patterns, we implemented a form of perfect unlearning with perfect information about the patterns. This is done by referring to the original learning set. Although this information would not be available to a real memory system, it forms an upper bound on performance for any pattern discrimination system, such as the energy ratio measure. By using perfect information about the types of patterns, we can analyse the assertion that identification of the nature of a pattern will assist unlearning.

6.1.1 Algorithm

The procedure for unlearning only the spurious patterns is similar to the Christos unlearning procedure presented in Section 4.4.5 with the addition of a selection phase. This selection phase labels spurious and learnt patterns and removes the learnt patterns from the unlearning set, thus protecting them from direct unlearning. The removed

learnt patterns are replaced by spurious patterns generated with additional probes. The new procedure with the filter as indicated (→) is:

- learn a small base population[1] of items
- perform unlearning of ($U \times$base population) items, each of which require
  - probing the network with a random pattern and relaxing to find a stable state
  - → • if the stable state is one of the learnt patterns continuing to probe until you find a pattern that is not a learnt pattern
  - performing unlearning on the pattern by applying Hebbian learning with a learning constant of $-\eta$
- for each new pattern to learn
  - learn the new item using Hebbian learning
  - perform unlearning of (U) items, each of which require
    - probing the network with a random pattern and relaxing to find a stable state
    - → • if the stable state is one of the learnt patterns continuing to probe until you find a pattern that is not a learnt pattern
    - performing unlearning on the pattern by applying Hebbian learning with a learning constant of $-\eta$

With the removal of the learnt patterns from the unlearning set, all the "power" can be focused on removing the unwanted spurious patterns. In theory, this should improve the performance of the learning system.

6.1.2 Results of Perfect Unlearning

Having implemented the above procedure we were initially surprised to discover that this "enhancement" did not improve the performance of unlearning at all. It certainly changes which patterns are remembered, but as shown in Figure 6.1 unlearning only the spurious patterns "Ul Spurious" causes the performance to collapse over time in a similar way to the original unlearning "Ul All". Unlearning only the spurious patterns delays the decline in performance for a while, but both types of unlearning are well below weight decay after 60 patterns. Although simple weight decay does not have as

[1]The base population is needed, as with only a few patterns learnt there are very few or no spurious patterns found by probing - and therefore nothing to unlearn.

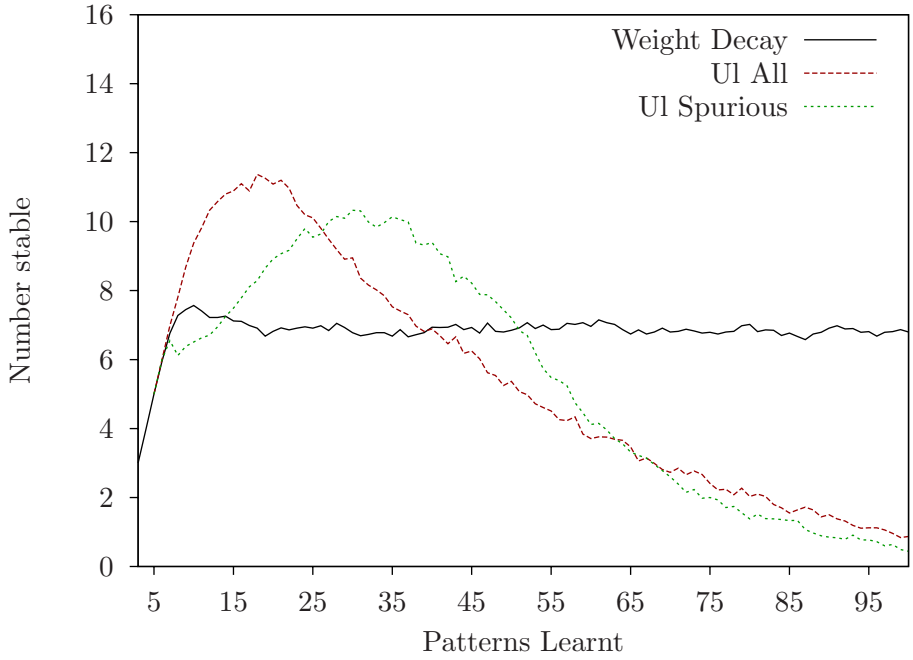


Figure 6.1: Unlearning of all patterns "Ul All", and unlearning of just the spurious patterns "Ul Spurious". Hebbian learning in $\mathcal{H}^5_{100,\pm}$ network with five patterns learnt followed by new patterns presented one at a time. Both unlearning conditions have an unlearning population of 50 patterns with unlearning constant of $-\eta = -0.01$, while the weight decay condition is set at $d = 0.1$ (100 repetitions, N.B. error bars removed for readability).

high a peak, its consistency over long periods indicates that it is still better than the unlearning systems for longer training runs.

The unlearning procedure suggested by van Hemmen (1997) is not altered at all by the inclusion of learnt pattern filtering. As the learnt patterns have virtually no basin of attraction, they are almost never included in the unlearning population. Out of 1,000,000 unlearning probes only 132 relaxed to learnt patterns. The exclusion of this small number of patterns had no perceivable influence on the performance of unlearning.

6.1.3 Assessment of Performance

The performance of unlearning was not greatly improved by the inclusion of perfect pattern identification. Part of the reason for this is that unlearning after each new

pattern, like weight decay, needs to decrease the influence of the learnt patterns as well as the spurious patterns. The most recently learnt pattern has the largest basin of attraction, and so receives the most unlearning. This pattern's large basin is slowly reduced by unlearning, until other patterns become stable. As a side effect the weights in the network are reduced. With perfect unlearning we remove the opportunity to decrease the size of the basin of attraction of the most recently learnt pattern, hence unlearning is unable to decrease many of the weights. Without this decrease, new patterns cannot be learnt. Thus, being able to distinguish spurious patterns from learnt patterns, even perfectly, does not improve unlearning.

6.2 Perfect Pseudorehearsal

When rehearsing the entire learnt population with every new pattern, the network must be able to retrieve a perfect copy of every pattern. This is obviously impractical for any real memory system. Instead of rehearsing every pattern from the entire learnt population, we could rehearse only the learnt patterns that are found by probing. This changes the task from being able to retrieve all the learnt patterns, to having perfect knowledge about a pattern.

During "Full" rehearsal the patterns are rehearsed with noise so that their basin of attraction is protected. Pseudorehearsal does not apply noise to the pseudoitems so that they do not generate basins and displace the learnt population. The new population containing only those patterns found by probing will be learnt with and without noise to find the best combination of pseudorehearsal and knowledge about patterns.

The results of altering the pseudorehearsal algorithm will be shown in two different networks. The first network, Net$A$, is a replication of the network and learning task that we presented in Robins and McCallum (1998). This is a 64 unit Hopfield network with 44 patterns in the base population ($\mathcal{H}^{44}_{64,\pm}$). There are 64 items in the learning population, all of which have been generated as simplified alpha-numeric characters. Examples of these are shown in Figure 4.20 with the full set of 64 shown in Appendix C. These patterns are correlated and have a coding ratio of below 30%. The patterns for the letter 'C' and the letter 'O' have a difference of only 2 units. These patterns are far too correlated for the unmodified Hebbian learning algorithm to learn more than four patterns. The 44 pattern base population is a large percentage of the theoretical capacity of the network, and this makes it difficult for delta learning to make all 44

learnt patterns stable and ensure that they have large basins of attraction.

The second network, Net$B$, is a 100 unit network with five patterns in the base population $\mathcal{H}^5_{100,\pm}$. The learning patterns for this network are randomly generated with a coding ratio of 50% (see Appendix B.4 for additional parameters). These patterns are ideal for Hebbian learning, and contain the maximum possible complexity per pattern of any distribution. The small base population allows pseudorehearsal to solve catastrophic plasticity, without the problem of catastrophic capacity. The results of pseudorehearsal will be shown for both of these networks for each of the following figures, unless stated otherwise.

6.2.1 Perfect Pseudorehearsal with Noise

Full rehearsal uses noise on both the new item and the rehearsal population. Figure 6.2 shows the performance of full rehearsal, "Full", compared to perfect pseudorehearsal with noise, "PrLν". The performance of full rehearsal is very close to 100%, confirming that the network is capable of learning all the patterns. The restriction to rehearsing only those patterns that were found by probing lowers the performance significantly, but it is still performing well above simple pseudorehearsal "Pr256".

The naming convention we will use in the following pseudorehearsal figures is: "Pr" for pseudorehearsal, followed by either a) the number of patterns in the pseudoitem population (e.g. "256") with an "*" if the learnt patterns have been removed; b) "L" indicating that only learnt patterns found by probing are included in the pseudoitem population; or c) "ER" indicating that only patterns that have an energy ratio above the number indicated (e.g. "0.25") are included in the pseudoitem population. If noise is applied to the population then "ν" is appended to the end of the name.

Perhaps the most interesting feature of Figure 6.2 is the difference between full rehearsal and perfect pseudorehearsal "PrLν". When only the found patterns are rehearsed, the number of patterns that are stable is approximately $0.6N$ in Net$A$ and below $0.3N$ in Net$B$. The difference between the performance is caused by the correlated patterns and large base population used with Net$A$. The performance of Net$B$ of $0.3N$ is relatively low compared to full rehearsal, and is only approximately double the number of patterns learnt by simple pseudorehearsal. This is using perfect knowledge of the patterns that are being rehearsed, and so will act as an upper bound for systems that use an estimate of pattern identity.

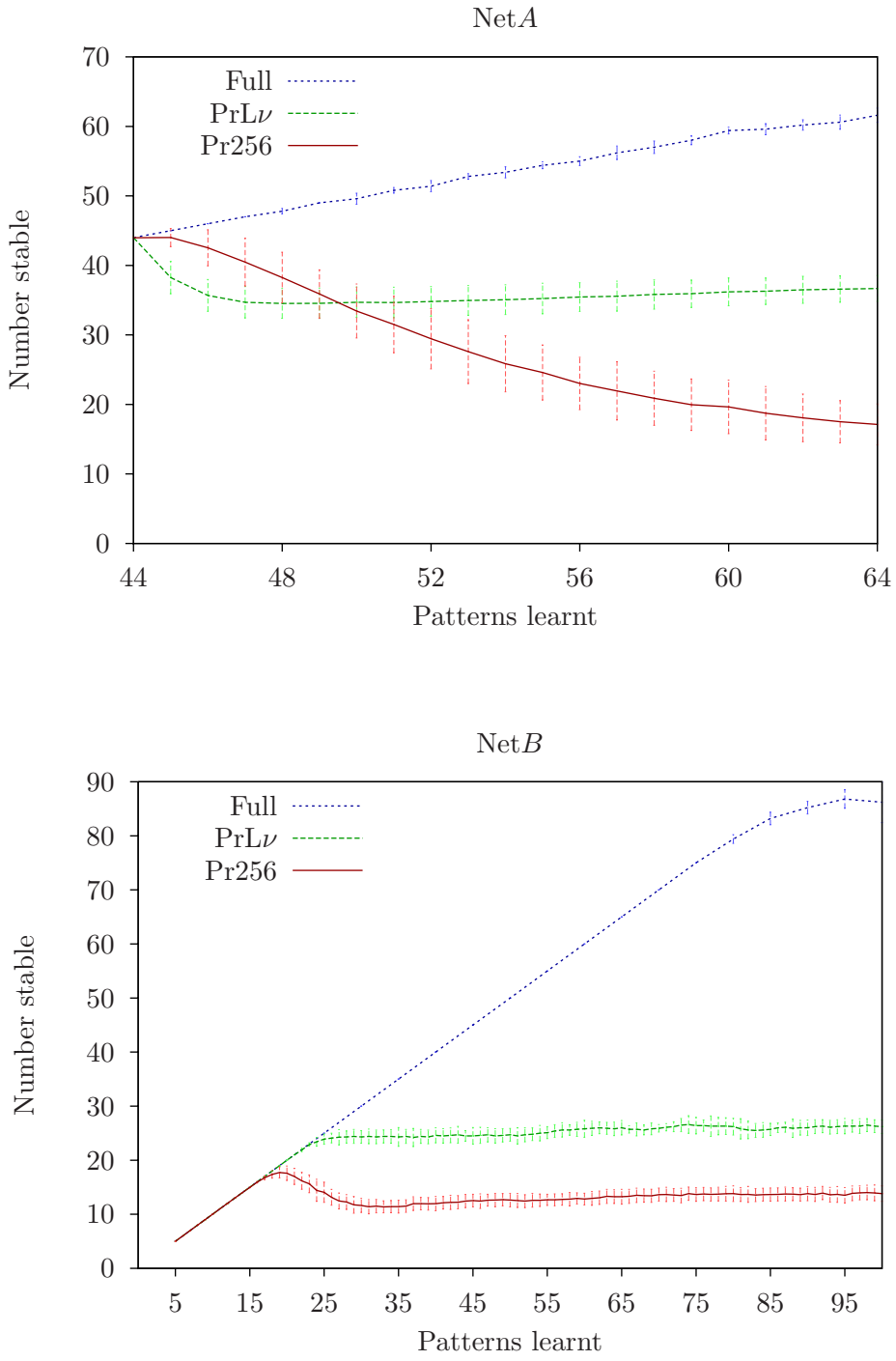


Figure 6.2: Stable learnt patterns in Net$A$ $\mathcal{H}^{44}_{64,\pm}$ and Net$B$ $\mathcal{H}^{5}_{100,\pm}$. "Full" is full rehearsal of every learnt pattern, "PrLν" is rehearsal of just the learnt patterns that were found with 2000 probes, and "Pr256" is simple pseudorehearsal of the first 256 patterns found with probing (100 repetitions).

6.2.2 Perfect Pseudorehearsal without Noise

The experiments presented in Section 4.5 do not apply noise to the pseudoitems when they are rehearsed. This allows the protection of the learnt patterns without creating new basins of attraction around the large number of spurious patterns that are usually included in the pseudoitem population.

In Robins and McCallum (1998) we compared simple pseudorehearsal with pseudorehearsal with the spurious patterns removed ,"Pr256*". In Section 4.5.3 we replicated this work, showing that pseudorehearsal is still relatively effective with all the learnt patterns removed from the pseudoitem population. Next we compared the use of perfect knowledge to remove either the learnt or the spurious patterns. Figure 6.3 shows the performance of pseudorehearsal without noise on a rehearsal population with the two types of patterns removed. "PrL" is pseudorehearsal of only the learnt patterns, and "Pr256*" of only spurious. Surprisingly, when we limit the rehearsal to only the learnt patterns, the number of stable learnt patterns decreases. This happens in both Net$A$ and Net$B$.

There are two reasons for this decrease in performance. The first is that the removal of the spurious patterns results in a very small pseudoitem population. The network is limited to 2000 probes to find patterns for the pseudorehearsal population. When limiting the patterns that are included in the population to just the learnt patterns, the size of the pseudoitem population cannot grow larger than the number of stable learnt patterns. In Net$B$ there are only about 10-15 learnt patterns found during probing. Thus the pseudoitem population is only about 5% of its normal size of 256 patterns. Without a large number of pseudoitems the network cannot preserve the original behaviour, as there are too few data points to correct the changes made by learning the new pattern.

The second reason relates to the application of noise. When learning patterns without noise, delta learning only makes adjustments when the pattern is unstable. As most of the learnt patterns have relatively large basins of attraction, it takes a large number of changes before they generate errors. Thus the majority of the damage to the pattern's stability is done well before the pattern becomes unstable. Preserving the learnt patterns without noise does not protect the basin of attraction of the pattern, and although the learnt patterns remain stable for a while, they quickly lose their basin of attraction and so are not found with random probing, and will not appear in pseudoitem population. When a learnt pattern is no longer part of the pseudoitem population it quickly disappears as there is no mechanism to preserve it.

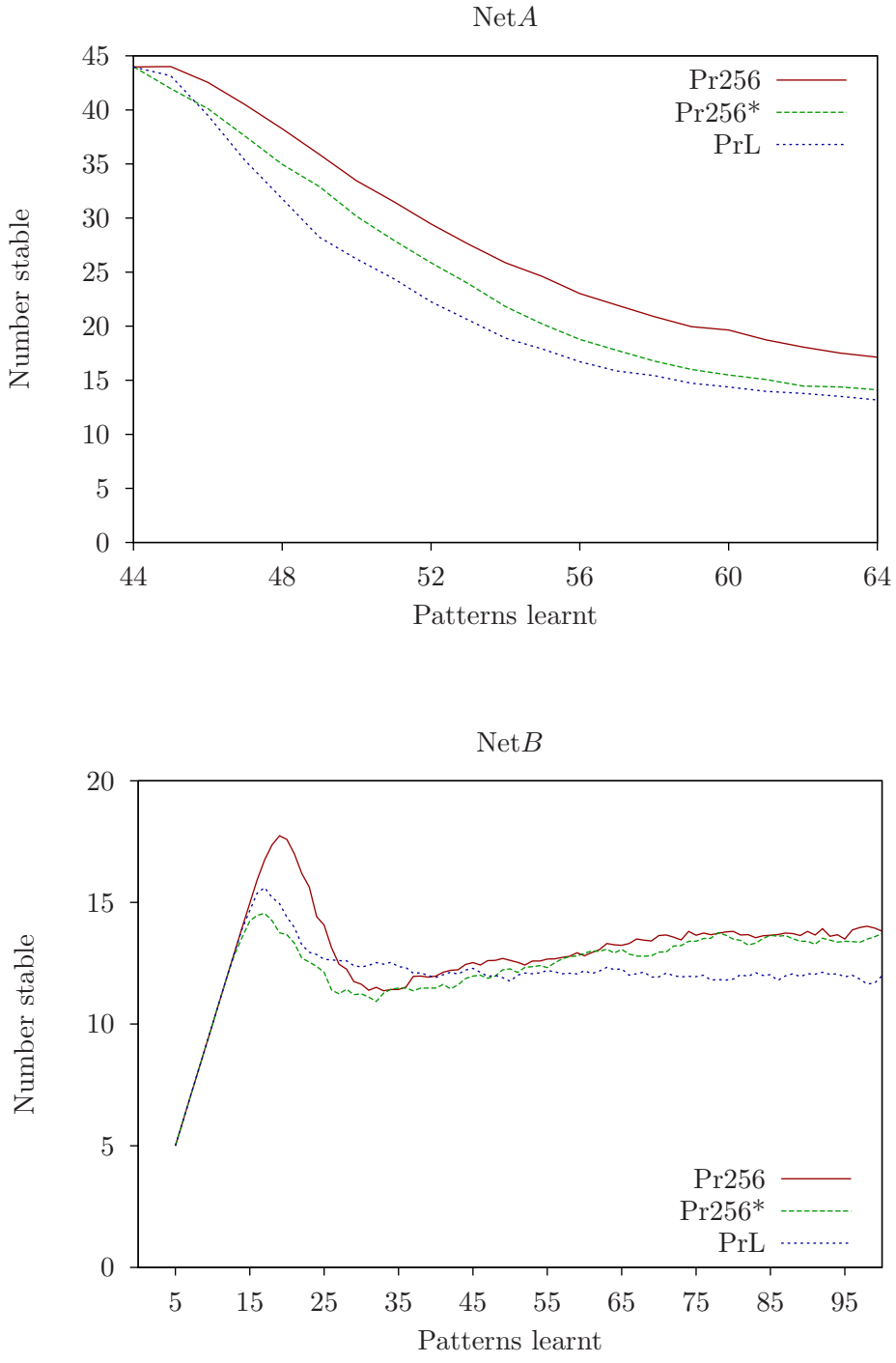


Figure 6.3: Stable learnt patterns in Net$A$ $\mathcal{H}^{44}_{64,\pm}$ and Net$B$ $\mathcal{H}^{5}_{100,\pm}$. "Pr256" is pseudorehearsal of 256 patterns, "PrL" is pseudorehearsal with only the learnt patterns found with 2000 probes, and "Pr256*" is the first 256 spurious patterns (with learnt patterns removed and replaced) found with 2000 probes (100 repetitions, N.B. error bars removed for readability).

Simply adding noise to all types of pseudorehearsal does not improve the performance of pseudorehearsal. Adding noise to the rehearsal of every pattern found during sampling quickly degrades the performance of the network. Figure 6.4 shows the performance with noise added to all pseudoitems, "Pr256ν", and with the learnt patterns removed, "Pr256*ν". The performance of both is very poor, with fewer than five patterns stable. Not only is the number of stable patterns low, but they are usually only the first five patterns learnt. These learnt patterns have no basin of attraction, and only remain stable as a result of the large spurious states that were formed as linear combinations of these patterns when the base population was learnt. The "Pr256ν" and "Pr256*ν" are almost identical as the learnt patterns are only found when there are less than five or six patterns learnt. After that point they have no basin of attraction and so are never found, making their filtering irrelevant.

The reason for the original five base population items remaining stable is linked to the discussion of spurious patterns in Section 4.5.4. After only five patterns have been learnt, there are only a few large spurious attractors in the network. These are the linear combinations of the learnt patterns. These combination patterns are found in the probing stage, and become part of the pseudoitem population for every new pattern. As these combination patterns are not part of the learnt population they are not removed by the learnt pattern filter. Thus they are rehearsed with every pattern and become very stable with large basins of attraction. The five base population patterns are now the linear combinations of these very stable spurious states. The rehearsal of the spurious states ensures that the learnt patterns that created them remain stable, without ever being directly rehearsed. After a moderate number of new patterns are presented, the stable base population patterns have small basins of attraction and are no longer found with random probing.

With access to perfect knowledge about the patterns found by probing, pseudorehearsal is able to learn approximately $0.3N$ patterns. Unfortunately, it still requires access to perfect external knowledge about which patterns were learnt. As shown in Chapter 5, the energy ratio measure can distinguish learnt patterns from spurious patterns with a high degree of accuracy without accessing the original patterns. We can use this metric with pseudorehearsal to assess the type of pattern found by probing, instead of the infeasible approach of accessing the original base population.

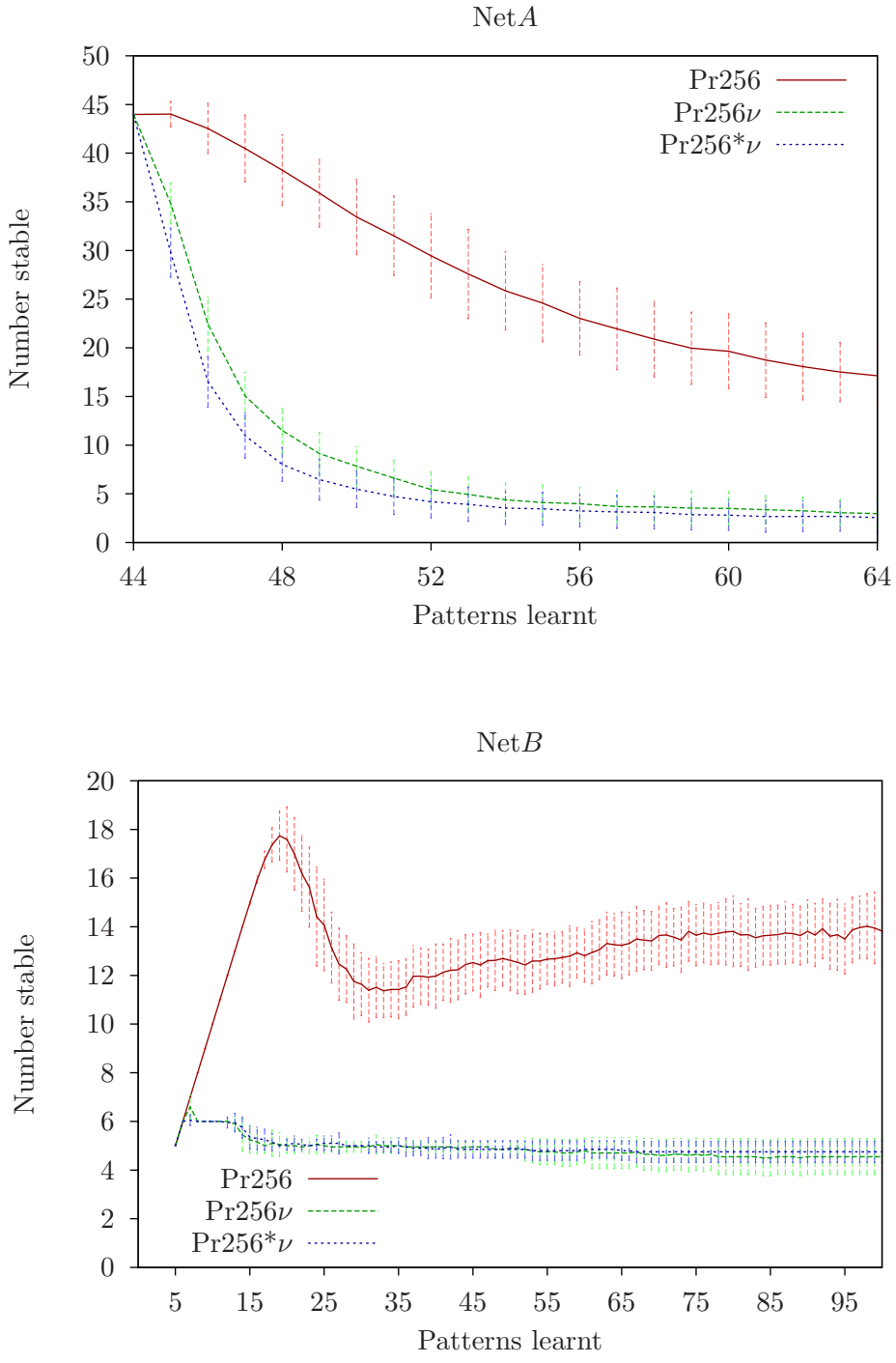


Figure 6.4: Stable learnt patterns in Net$A$ $\mathcal{H}^{44}_{64,\pm}$ and Net$B$ $\mathcal{H}^{5}_{100,\pm}$. "Pr256" is simple pseudorehearsal of 256 patterns, "Pr256ν" is simple pseudorehearsal with noise added to the all the pseudoitems, and "Pr256*ν" is pseudorehearsal with perfect removal of learnt patterns and noise added to the remaining pseudoitems (Pr256 100 repetitions, with ν 20 repetitions).

6.3 Energy Ratio Based Pseudorehearsal

Pseudorehearsal of learnt patterns with noise is able to store $0.3N$ patterns. In this section we will use the energy ratio measure to replace perfect knowledge about the rehearsal population. This requires the learnt patterns to have relatively high energy ratios. To achieve this, two types of noise are applied to the network – heteroassociative noise for the basin of attraction, and absolute input noise to move units away from the decision surface. For the following examples we used heteroassociative noise of $\nu_h = 5\%$ and absolute input noise of $\nu_i = 0.5$. Figure 6.5 shows the performance of pseudorehearsal with an energy ratio threshold of 0.15 ($PrER^{0.15}$) in Net$A$ and 0.25 ($PrER^{0.25}$) in Net$B$. In Net$B$ the performance of pseudorehearsal with only the high energy ratio patterns (above the threshold for recognition as a learnt pattern) in the rehearsal population is much better than simple pseudorehearsal, and is similar to perfect pseudorehearsal which has access to perfect knowledge about patterns.

Pseudorehearsal using only high ratio patterns performs well in Net$B$, but poorly in Net$A$. There are two reasons for this difference in performance – 1) the large base population in A lowers the energy ratios as the patterns compete for weights, and 2) the correlated patterns in the alphanumeric task create a number of units that are only just stable. The two units that are the difference between the 'C' and the 'O' are continually pulled back and forth between being active and inactive. This results in the units being close to the decision surface and thus the patterns have a low energy ratio. Patterns with low energy ratios are not included in the pseudoitem population and so become unstable.

In the 100 unit Net$B$ learning task the energy ratio is able to distinguish learnt patterns with high accuracy. In the 100 repetitions of learning 100 patterns with 2000 probes per new item (20,000,000 probes), not a single spurious pattern was misclassified as a learnt pattern. The number of pseudoitems in each population averaged at 18.41. This indicates that of the approximately 25 stable patterns, about 18 of these were being found and rehearsed.

6.3.1 Additional Measures of Performance

The number of stable learnt patterns is not the only important measure of performance. As discussed earlier, it is possible to have a large number of patterns stable without the network being a good content addressable memory. The size of the basins of attraction and the number of random probes that relax to learnt patterns is also important.

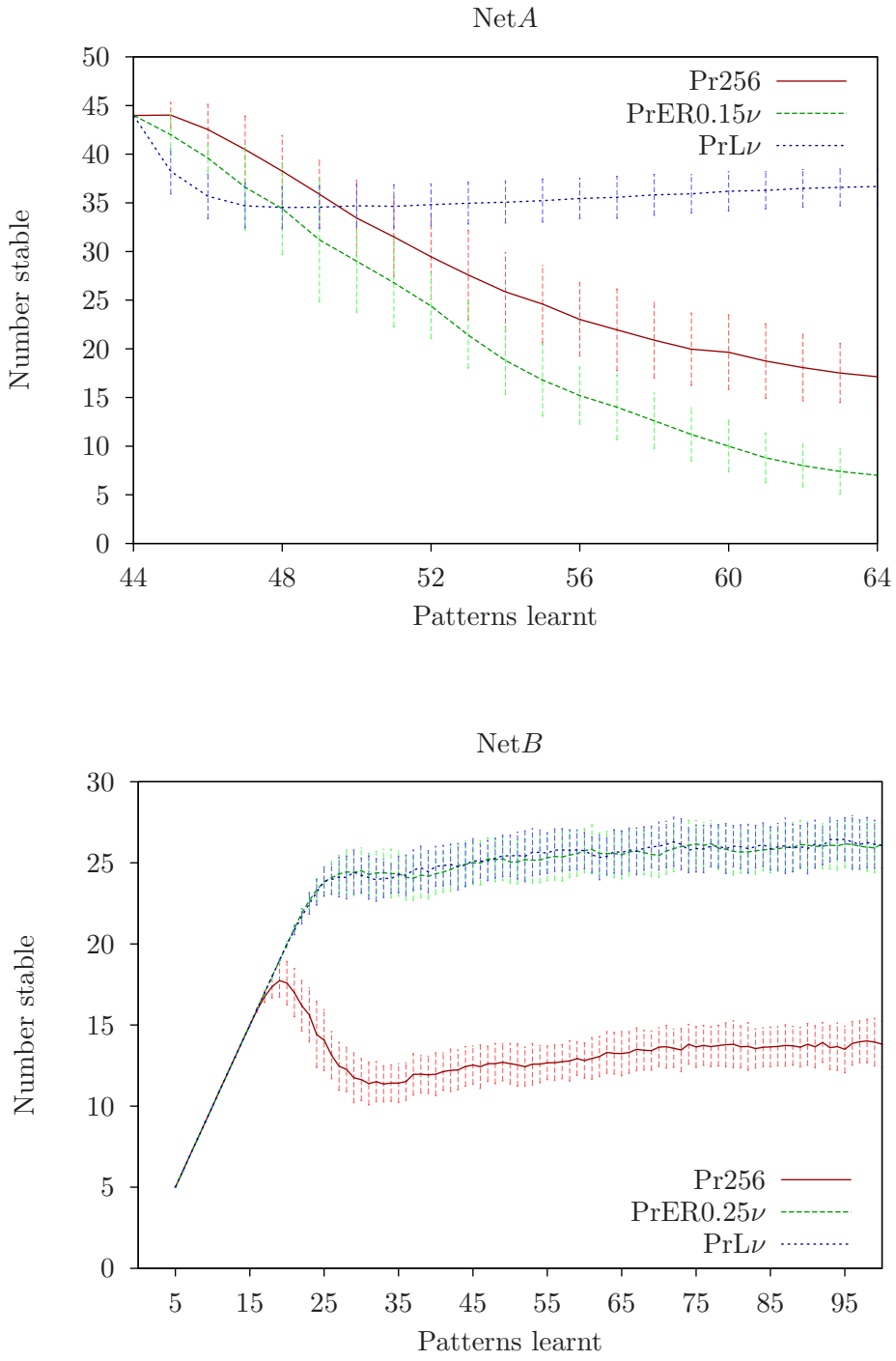


Figure 6.5: Stable learnt patterns in Net$A$ $\mathcal{H}^{44}_{64,\pm}$ and Net$B$ $\mathcal{H}^{5}_{100,\pm}$. "Pr256" is simple pseudorehearsal, "PrLν" is perfect pseudorehearsal, and "PrERν" is pseudorehearsal of the patterns that are above the energy ratio threshold, 0.15 for Net$A$ and 0.25 for Net$B$ (η=0.1 , ν_i= 0.5, ν_h=5%) (40 Repetitions).

For the network to be considered to be learning new patterns, a balance must be found between remembering old patterns and learning new patterns. A system which learns the first few patterns presented, and then ignores new patterns, is "solving" the plasticity/stability dilemma by ignoring the need for plasticity.

With learning applied equally to the new items and the pseudoitems there is no guarantee that any given new item will be learnt. If a pattern is found by probing, and passes the energy ratio threshold, it will be learnt in exactly the same way as a new pattern. Using this approach the network tends to learn only the first $0.3N$ patterns presented and then stops learning new patterns. This can be seen in the probability of patterns in different positions being stable in Net$B$ shown in Figure 6.6. Pseudoitems in condition "1.0" have the same learning constant and noise ratios as the new items. After 100 patterns have been learnt, most of the stable patterns are in the first $0.3N$ patterns presented, with the small addition of the most recently learnt patterns.

To ensure that new patterns are stable, we need to apply a different amount of learning to the new pattern as compared to the rehearsed patterns. To differentiate the new pattern from the previously learnt patterns we decrease the learning constant η and noise constant ν for the rehearsed pseudoitems. Figure 6.6 shows the marked difference in the probability of patterns being stable and the basin sizes with different learning on the pseudoitems. Pseudoitems in the "0.5" condition have their learning parameters halved – learning constant $\eta = 0.05$, heteroassociative noise $\nu_h = 2.5\%$, and absolute input noise $\nu_i = 0.25$. With this differentiation, the stable patterns cluster closer to the most recently learnt pattern. A new pattern receives more learning than any other pattern, and so has a larger basin of attraction and survives to the next iteration. The lower learning constant and noise allows older patterns to be forgotten, so that new items can be learnt.

The average basin of attraction size, indicated by the overlap m_f, is much larger and more uniform for the "1.0" condition than the "0.5" condition. The pseudoitems in the "1.0" condition receive as much rehearsal as any other pattern, and so all the patterns converge on a similar size of basin. The overlaps m_f presented are the average for the patterns that were stable. In the 100 runs, after 100 patterns have been learnt, the 87^{th} pattern was only stable in one of the 100 repetitions. In the one run that it was stable it had a basin overlap of m_f= 0.772 (the furthest right point of the relatively uniform basin overlaps in the "100 Overlap" graph). The number of data points used to find the average can be seen by comparing the overlap graph with the probability graph directly above it. If a pattern is stable in 5% of the repetitions, then the overlap

is the average of those 5 stable patterns. The patterns in the range 60 to 87 are rarely stable, but once in the pseudoitem population they become similar in basin size and overlap to any other rehearsed pattern. The last 10 patterns (90–100) are much more likely to be stable, not because of rehearsal, but because of their recent learning. The probability of these patterns being stable is very similar to standard delta learning without any form of rehearsal. These most recently learnt patterns generally do not have a large enough basin of attraction to be found by random probing, and so are not receiving rehearsal after they were first learnt. Without rehearsal the basin size and the probability of being stable both decrease sharply. This explains the discontinuity in the "1.0" condition in the "100 Overlap" graph at position 87. The development of this discontinuity can be seen after 50 patterns at the 45^{th} position.

Pseudorehearsal with the energy ratio measure is also effective in low coding environments. Figure 6.7 shows that pseudorehearsal with an energy ratio threshold of 0.25 is also effective in networks with low coding ratios. The performance with patterns that are generated with 10% or 20% of their units active shows that this approach is robust with respect to the coding ratio of the patterns.

6.4 Summary of Performance

Pseudorehearsal with the energy ratio threshold consistently stores a large number of patterns. We can now compare this solution with the other feasible solutions presented in this thesis. For a solution to be feasible it must have access to a learnt pattern only while the pattern is being learnt. Figure 6.8 compares the best of each of the feasible solutions in the 100 unit $\mathcal{H}^5_{100,\pm}$ network. These are broken into two groups – the Hebbian learning group of weight decay, neuronal regulation, and Christos style unlearning; and the delta learning group of simple pseudorehearsal and pseudorehearsal with the energy ratio threshold. Simple pseudorehearsal is already better than any of the Hebbian learning solutions, and with the enhancement of the energy ratio measure "PrER0.25ν" it stores almost four times as many learnt patterns as simple weight decay. The significant gap between the solutions is the clearest indication of the success of this combination of delta learning, pseudorehearsal and energy ratio.

The various experiments that have been conducted throughout this thesis have been drawn together in a matrix for easier review, Table 6.1. The best of the feasible solutions, PrER0.25ν, is in the bottom right corner of the table.

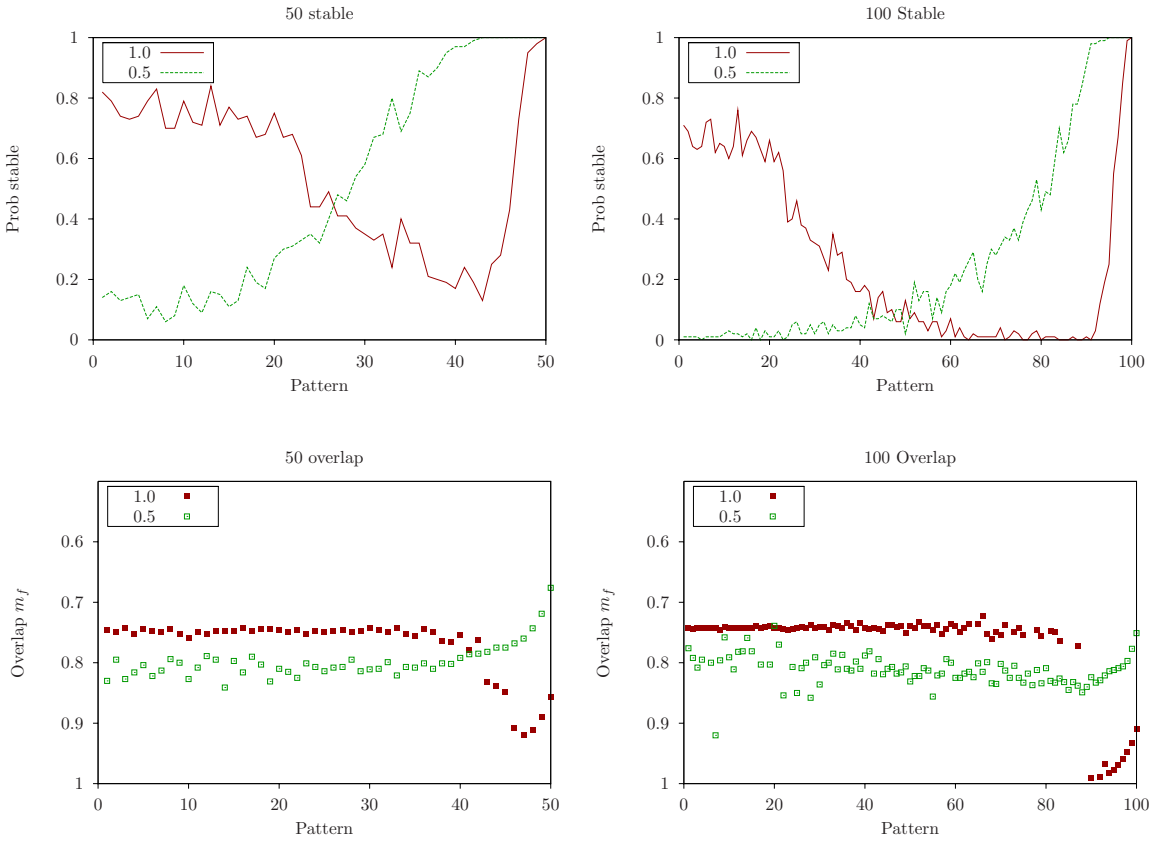


Figure 6.6: Probability of being stable and the average size of the basins of attraction for patterns after either 50 patterns have been learnt or 100 patterns have been learnt for Net$B$. "1.0" is the same noise on pseudoitems as new items ($\eta = 0.1,\ \nu_h = 5\%$ and $\nu_i = 0.5$) and "0.5" has pseudoitems with half the learning and noise constants ($\eta = 0.05,\ \nu_h = 2.5\%$ and $\nu_i = 0.25$).

	Basic algorithm	Simple solutions		Enhanced Solutions (Improved by distinguishing learnt/spurious)	
		Assume perfect access to prev. learnt population	Assume no access to prev. learnt population	Assume perfect knowledge via probe	Use estimates of pattern knowledge via probe
Hebbian Learning	Suffers complete CF because of catastrophic capacity & correlation. Sect. 4.1	**Rehearsal**: Perfect access does not alter performance (merely doubles weights). Sect. 4.5.2.	**Unlearning**: Unlearning of probed stable states is somewhat effective. Christos, van Hemmen. Sect. 4.4.4-6. **Pseudorehearsal**: Doesn't apply to Hebbian (rehearsal ineffective). Sect. 4.5.2. **Weight decay, capping & regulation**: Limited protection of most recent items. Sect. 4.4	**Unlearning**: Perfect recognition of stable patterns from probing is detrimental to performance. Sect. 6.1. This is contrary to the original predictions of Hopfield *et al.* (1983).	Ratio based recognition of patterns works well, but if perfect knowledge is ineffective estimates will also fail.
Delta Learning	Suffers CF because of catastrophic plasticity & capacity. Some retention of most recent items. Sect. 4.4.7	**Rehearsal**: Perfect access to all patterns solves CF and allows performance to approach theoretical max of $1N$. Sect. 4.5.1.	**Pseudorehearsal**: Relearning of probed stable patterns is somewhat effective. Sect. 4.5. **Unlearning**: Does not apply to delta learning. **Weight decay, capping & regulation**: just degrades performance. 4.4.7	**Pseudorehearsal**: Perfect recognition of patterns allows protection from CF. Twice performance of simple Pr, $1/3^{rd}$ performance of full rehearsal. Establishes upper bound. Sect. 6.2.	**Pseudorehearsal**: Ratio based recognition twice the performance of simple Pr. Performance similar to perfect recognition in some conditions. Best practical solution! Sect. 6.3.

Table 6.1: Matrix of experimental conditions

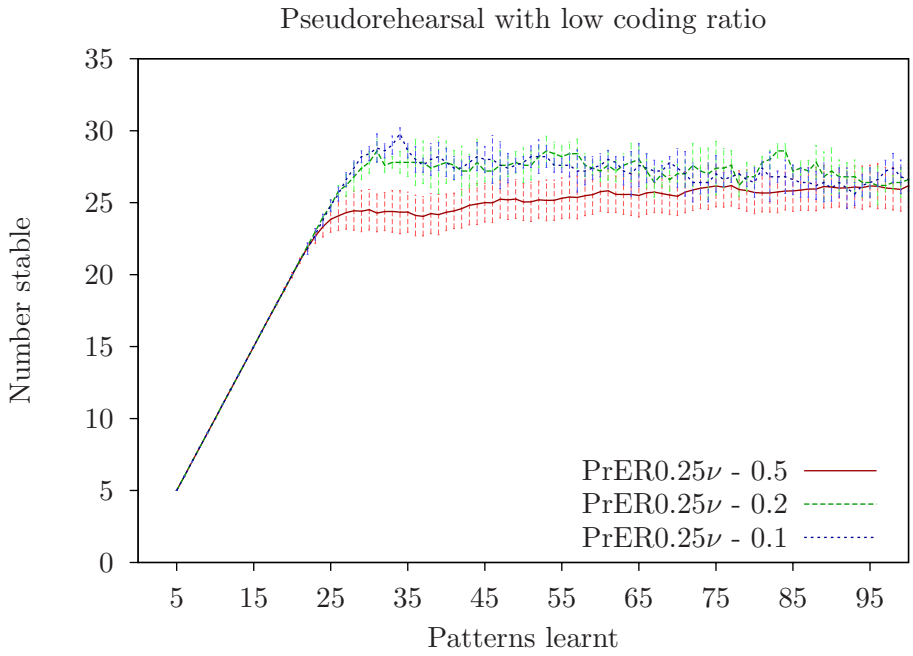


Figure 6.7: Pseudorehearsal applied to learning patterns with low coding ratios. "PrER0.25ν - 0.2" has 20% of the units active and "PrER0.25ν - 0.1" has 10%. $\mathcal{H}^{5}_{100,\pm}$ with energy ratio threshold of 0.25, learning ratio of 0.5, 2000 probes to find high energy ratio patterns (0.5 has 100 repetitions, while 0.1 and 0.2 have 20 repetitions).

6.5 Conclusion

One of the early claims made about unlearning is that it removes the spurious "fantasy" states from the network. Crick and Mitchison (1986) even suggested the catch phase "we dream to reduce fantasy and obsession". If the unlearning process is aimed at removing the spurious "fantasy" states, then by explicitly filtering all the learnt patterns from the unlearning population there should be an improvement in the performance of the system, which there is not. The unlearning of learnt items is covered by the "obsession" part of the phrase. Unlearning needs to decrease the strength of the most recently learnt pattern to allow new patterns to be learnt. The experimental results support this need to remove "obsession". Unlearning must remove the large basins of attraction regardless of their origin, and therefore is not able to benefit from additional knowledge about the types of patterns being unlearnt.

Pseudorehearsal, however, can be improved by knowledge about the types of patterns that are being rehearsed. Full rehearsal prevents catastrophic forgetting caused

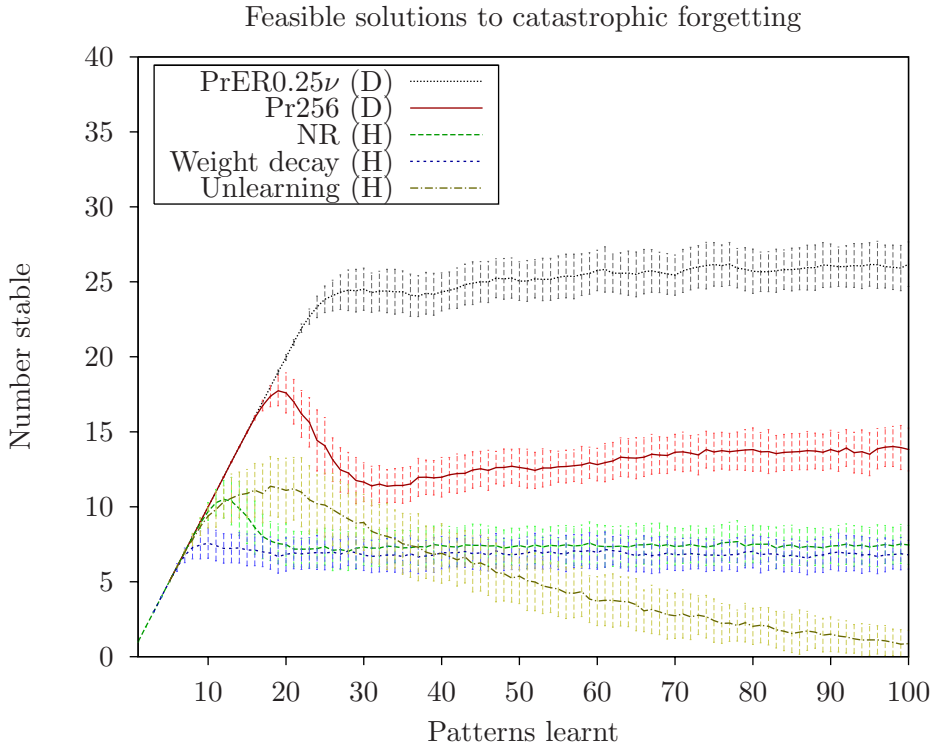


Figure 6.8: Comparison of the best practical solutions to catastrophic forgetting. Conditions with (D) are based on delta learning and (H) on Hebbian learning. The conditions are "PrER0.25ν (D)" pseudorehearsal with the energy ratio threshold of 0.25 with noise ratio of 0.5, "Pr256 (D)" simple pseudorehearsal of 256 patterns, "NR (H)" neuronal regulation at $2N$, "Weight decay (H)" weight decay of $d = 0.1$, and "Unlearning (H)" unlearning of 50 patterns at $-\eta$=0.01 (100 repetitions).

by excessive plasticity, but requires access to all the learnt patterns. Limiting the rehearsal to only the patterns that can be found by probing, results in the capacity of the network being much lower, approximately $0.3N$. This is still much higher than simple delta learning, and is about double the number stored with standard pseudorehearsal. Remarkably, we can replace perfect knowledge with the energy ratio measure and for certain conditions retain almost the same level of performance. The main conditions are that the base population be less than $0.3N$ and that the patterns be relatively independent.

The improvement of storage to $0.3N$ is significantly better than the maximum capacity of any of the Hebbian learning based solutions. It is also higher than the maximum theoretical capacity of Hebbian learning in the static case of approximately $0.14N$. The inclusion of the energy ratio measure allows the system to provide a level of confidence about the origin of a stable pattern, and a threshold for familiarity. Increasing the memory capacity and the quality of the information returned from the network makes the combination of pseudorehearsal and energy ratio a very attractive solution to the problem of catastrophic forgetting. This solution does not require a second copy of the learnt population, nor does it need to process all the patterns before learning them. It is robust with respect to different numbers of patterns, network size, coding ratios, and noise levels.

The significant improvement in performance provided by energy ratio pseudorehearsal, and its robustness, make this process a strong candidate for a computational explanation of how catastrophic forgetting may be solved by biological systems.

Chapter 7

Discussion

One of the motivations for this research is to propose a computational metaphor for what might be happening to memory in the sleeping brain. In this chapter we investigate if a link can be made between memory processing during sleep and pseudorehearsal. We then go on to discuss the possible future of this research.

7.1 Memory Consolidation and Sleep

The classic model for memory consolidation is that the hippocampus serves as a short term memory that binds various sensory components of an event into an episode. These components are processed and identified by the neocortex but are bound together by the hippocampus. Over time this binding is transferred to the neocortex, and it is able to re-activate the whole memory from a subset of the original sensory input, thus consolidating short term episodic memory into long term memory (Alvarez and Squire, 1994; McClelland *et al.*, 1995).

Meeter and Murre (2004) summarise several different interpretations of the psychological data, including the work of Nadel, Samsonovich, Ryan, and Moscovitch (2000). Nadel *et al.* propose that the hippocampus is always required to recover episodic memory regardless of the age of the memory. Their proposal is that the retrograde amnesia associated with hippocampal damage (as seen in the Ribot curve), is actually conflating two different effects on different memory types, episodic and semantic. Episodic memory is lost, while semantic memory has been successfully retained in the neocortex. This relies on differences in the testing stimulus for very distant memory.

While the disagreement is ongoing, we will use the classic model of consolidation as our preferred model for this thesis, as both pseudorehearsal (Robins and McCallum,

1998) and unlearning (Crick and Mitchison, 1983) have been linked to the classic model of consolidation of memory, and in particular to the role of sleep.

7.1.1 The Consolidation of Learning During Sleep

There are several stages of sleep, each of which is physiologically and neurologically distinct. The most obvious distinction is REM (rapid eye movement) and NREM (non REM) sleep. While engaged in REM sleep the brain is being heavily stimulated by input from the brain stem, with waves of activation passing up through the visual areas and into the cortex. These seemingly random pulses cause the eyes to shudder or move rapidly, hence the name rapid eye movement. This stage of sleep has also been called "paradoxical sleep" as the activity of the brain on an EEG appears to be similar to fully awake behaviour. NREM sleep is very different to both REM and wakeful activity, and has been broken down into four stages, with Stages 3 and 4 being the deepest. Stage 3 and 4 sleep are also called slow wave sleep (SWS) as there are slow waves of activation visible on an EEG.

The link between sleep and memory consolidation is still a contentious issue. There are those who deny any link exists. Vertes argues that:

> "In sum, there is no compelling evidence to support a relationship between sleep and memory consolidation."(Vertes, 2004)

The basis for his argument is that REM deprived individuals do not show significant memory deficits, and that sleep is an amnesiac state and so seems unlikely to be related to memory. On the other side of the debate Stickgold states:

> "The past 10 years have shown an explosive growth in our knowledge of the relationship between sleep and memory, providing consistent and strong support for the existence of sleep-dependent memory consolidation."(Stickgold, 2005)

This conflict over the interpretation of the biological and psychological data can only be resolved by additional animal and human studies. Computational models, such as the one presented in this thesis, can only provide a limited amount of support for psychological theories. The support they lend is to demonstrate that a particular process is possible and provide direction for future investigation. This support is limited because there are usually a large number of simplifications and assumptions required to implement the model in a computer.

Studies of the link between memory consolidation and sleep initially focused on REM sleep (Greenberg, 1981), but recent research suggests that the stage of sleep necessary for memory consolidation, may depend on the type of memory being processed. Procedural memory, both visual and motor, appear to be linked to NREM sleep. Positive correlations between the quantity of NREM sleep and post sleep improvements in performance have been shown by Smith and MacNeill (1994) and Walker, Liston, Hobson, and Stickgold (2002) for motor skills, and Maquet, Schwartz, Passingham, and Frith (2003) and Huber, Ghilardi, Massimini, and Tononi (2004) for visual tasks.

The activity of the brain during sleep has also been used to support the theory of consolidation of previously learnt material. Brain imaging shows that temporal patterns of activation that occurred during training, reoccurred in subsequent REM sleep periods (Pennartz, Uylings, Barnes, and McNaughton, 2002; Maquet, Laureys, Peigneux, Fuchs, Petiau, Phillips, Aerts, Del Fiore, Degueldre, Meulemans, Luxen, Franck, Van Der Linden, Smith, and Cleeremans, 2000). This matching of temporal sequence did not occur in animals that had not received training. Reactivation of task specific areas of the brain have also been shown in SWS, and the amount of reactivation is positively correlated with next-day improvement (Peigneux, Laureys, Fuchs, Collette, Perrin, Reggers, Phillips, Degueldre, Del Fiore, Aerts, Luxen, and Maquet, 2004).

Walker and Stickgold (2006) summarises recent research at three levels: molecular level, where there is an up-regulation of memory specific genes during sleep; electrophysiological level, with robust results for correlations between neural activity during sleep and improved performance; and the most complex level of behavioural research. The behavioural evidence shows that some types of learning are dependent on sleep, and some are independent of this state. However they conclude that:

> "It is now clear that sleep mediates learning and memory processing, but the way in which it does so remains largely unknown."(Walker and Stickgold, 2006)

7.1.2 Sleep and Pseudorehearsal

If we accept the hypothesis that sleep is important for memory processing, there is still the question of why there is a separate stage of memory processing, and why this occurs during sleep. Most of the psychological work in this area focuses on defining the effect of sleep, and the lack thereof, on memory, rather than the mechanism by which sleep consolidates memory. Robins (1996) proposed that catastrophic forgetting may have

been the reason for an offline memory processing system, and that pseudorehearsal may be functionally similar to cognitive activity during sleep. This proposal was based on an MLP network using back prorogation learning. In this thesis, and work generated as part of the thesis (Robins and McCallum, 1998, 1999, 2004) we have shown that the pseudorehearsal approach works with a biologically plausible Hopfield network.

There are several features of the neurological research that link well with rehearsal, and indeed pseudorehearsal, as an explanation of sleep processing. The replaying of patterns of activation that were active during the learning of a task (Pennartz *et al.*, 2002; Peigneux *et al.*, 2004) fits very well with the requirement for the replaying of a new pattern to integrate into long term memory. The waves of random activation flowing from the brain stem could be used to sample the current behaviour of the network, generating the pseudoitem population.

7.2 Related Work

The concept of pseudorehearsal, as presented by Robins (1996), and the Hopfield variant which was developed as part of this PhD (Robins and McCallum, 1998), have already been extended by various authors.

French (1997) and Ans and Rousset (1997) were inspired by the concept of pseudorehearsal to develop a "reverberating" dual network model of the hippocampus and neocortex. In this model the pseudoitems are used to transfer patterns from one part of the network to a secondary storage area. This model used pseudoitems generated by the same feedforward approach describe in Robins (1996). Ans *et al.* (2002) extend the memory task to temporal sequences rather than static patterns that are used in this thesis. The pseudoitems used to protect these temporal sequences are similar to those in Robins and McCallum (1998). One of the surprising results from this approach is that static pseudoitems preserve the temporal relationships required to recall the original temporal sequences. The forgetting curves demonstrated for this dual reverberating simple recurrent network are very similar to those in humans (Ans *et al.*, 2004).

One of the major issues when using pseudorehearsal with Hebbian learning is "runaway" rehearsal (Meeter, 2003), where one pattern receives all the reinforcement and becomes the only stable pattern in the network. In some of our earlier work (Robins and McCallum, 1998) the pseudopopulation remained the same across all of the new patterns. This led to the dual problems of favouring items that had been learnt as part

of the base population, and required a store for the pseudoitems that remained static over many learning cycles. The experiments in this thesis, like those in Meeter (2003), generate a pseudopopulation after each new pattern is learnt. This allows new items to become part of the rehearsal population, and so behaves more like real memory. Meeter (2003) demonstrates that the problem of runaway consolidation also exists in the "Trace-Link" model proposed by Murre (1996), and suggests three possible solutions: the weights can be suppressed during consolidation; the weights can be capped (as discussed in Section 4.4.1); or unlearning can be used in conjunction with rehearsal. These methods improved performance marginally. As a process for selecting which patterns to rehearse, the energy ratio measure proposed in Chapter 5 may also work in the TraceLink model. Given that the problem to overcome is the runaway effect, selectively reinforcing patterns which had an energy ratio within a band of perhaps 0.25–0.4 may provide consolidation for unstable learnt patterns without the associated problem of strengthening already strong patterns.

Walker and Russo (2004) extend the original pseudorehearsal with a process of generating noise during rehearsal that creates a form of unlearning. This dual rehearsal process is equated with the two main distinct phases of sleep, REM and SWS. The conclusion of this work is that pseudorehearsal is indeed a good candidate for consolidation, and when using Hebbian learning, the addition of noise to the rehearsal process, allows rehearsal of patterns other than the dominant pattern. These results are for a network using Hebbian learning, and so a direct comparison of capacity is difficult.

The brain is organised in to a remarkably hierarchical structure, which can be analysed at many different levels. The content addressable memory in this thesis is homogeneous, but could serve as part of a larger heterogeneous system. Káli and Dayan (2004) have modelled the interaction between the hippocampus and the neocortex using a replay of memory system similar to pseudorehearsal. In this implementation the replay of memories from the hippocampus helps to form semantic memories in the neocortex, while individual episodes are retained in the hippocampus. Káli and Dayan are interested in the slow learning of semantic information rather than the core content addressable memory. As a simplification they use an idealised attractor which returns a perfect copy of a learnt pattern when presented with a slightly corrupted version (overlap $m_f = 0.95$). The content addressable memory presented in this thesis could be used as a substitute for the idealised sub–system.

7.3 Future Work

As part of the research for this thesis we have contributed the Hopfield autoassociative variant for pseudorehearsal, a model of familiarity discrimination for stable patterns, and shown that by filtering the rehearsal population using the energy ratio measure we can achieve much higher memory capacity that any of the other proposed solutions for the serial learning task. Pseudorehearsal is already being extended by various authors as shown above. The energy ratio measure could also be extremely useful in Hopfield network research.

7.3.1 Energy Ratio

The energy profile and the energy ratio measure that we have used to improve pseudorehearsal could be incorporated into many applications of Hopfield networks. The ability to determine the type of pattern that has been found by random probing allows the system to differentiate between fantasy and reality.

The energy ratio can be used to improve the core feature of a content addressable memory system, by improving the information returned by the system. The measure allows us to use the standard Hopfield network model, with its guarantee of finding a stable pattern, and augment it with the ability to give a "yes, I remember X", or a "no, that does not seem familiar" response. Without the ability to differentiate the spurious memories, a negative result for a probe of X result in the response "yes, I remember Y", which may or may not have actually been part of the learning population. Additional research is required to assess how useful this measure will be for the large variety of Hopfield like models being used in many research environments.

7.3.2 Biological Analogue of the Energy Ratio Measure

The energy ratio measure is a very effective computational solution to the problem of familiarity. There is still the open question of how a biological equivalent could be calculated. We propose two possible mechanisms: differential firing rates, and extracellular neurotransmitter. Using the first mechanism, the energy ratio measure could be equated to calculating the difference in the firing rates of cell assemblies, as a high input results in a high firing rate. If an assembly of active neurons have relatively uniform firing rates then this would be equivalent to a high energy ratio (suggesting that this state is familiar / learnt), whereas a wide range of firing rates would be equivalent to a low ratio (suggesting that the state is novel / spurious). The extracellular

neurotransmitter technique uses the neurotransmitter that is released when a neuron fires. Excess neurotransmitter would indicate saturation. It is possible that synapses close by could respond to these fluctuations in the amount of free neurotransmitter, and use this to calculate a ratio between neurons which are saturated and those that are only just over their decision threshold.

Given the implication of the perirhinal cortex in familiarity detection, perhaps a calculation similar to the energy ratio measure is occurring in these neurons. If the neurons are predicting familiarity using the energy ratio measure, then we would expect that heavy external stimulation of a small subset of hippocampal cells would be linked with feelings of unfamiliarity and uncertainty. Laboratory experiments using this technique may be able to support the current computational model, or they may require the development of other mechanisms to perform the task of determining familiarity.

7.4 Conclusion

This thesis has explored a wide range of issues relating to the capacity and stability of learning in Hopfield networks. Pseudorehearsal is clearly the best of the proposed solutions to the pervasive problem of catastrophic forgetting. We have shown that the pseudorehearsal mechanism can be enhanced by using an estimate of the familiarity of randomly retrieved memories, the energy ratio measure. Using this enhanced pseudorehearsal it is possible to store a relatively large amount of information in a Hopfield network using purely local information.

As highly interconnected, dynamical, content addressable memory systems, Hopfield networks are a plausible computational approximation of the properties of memory mechanisms in the mammalian brain. If it is indeed the case that during the course of its evolution the brain has met and solved the problem of catastrophic forgetting (perhaps, as argued here, via consolidation of newly learned information during sleep), then pseudorehearsal is a plausible candidate for a computational model of this mechanism.

Hence, after a thorough investigation of the problem of catastrophic forgetting, the pseudorehearsal solution stands out as robust and efficient in practical terms, and as a biologically plausible model of an important aspect of human learning and memory.

References

Abel, T., Nguyen, P. V., Barad, M., Deuel, T. A. S., Kandel, E. R., and Bourtchouladze, R. (1997). Genetic demonstration of a role for PKA in the late phase of LTP and in hippocampus-based long-term memory. *Cell*, *88*(5), 615–626.

Abraham, W. C. (2000). *Neuronal mechanisms of memory formation: Concepts of long-term potentiation and beyond*, Chapter Persisting with LTP as a memory mechanism: Clues from variations in LTP maintenance., 37–57. Cambridge University Press.

Abraham, W. C. (2003). How long will long-term potentiation last? *Phil. Trans. R. Soc. B*, *358*(1432), 735–744.

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning Algorithm for Boltzmann Machines. *Cognitive Science*, *9*(1).

Alvarez, P. and Squire, L. R. (1994). Memory Consolidation and the Medial Temporal Lobe: a Simple Network Model. *Proceedings of the National Academy of Sciences of the United States of America*, *91*(15), 7041–7045.

Amit, D. and Brunel, N. (1995). Learning internal representations in an attractor neural network with analogue neurons. *Network*, *6*(3), 359–388.

Amit, D., Gutfreund, H., and Sompolinsky, H. (1987a). Statistical mechanics of neural networks near saturation. *Annals of Physics*, *173*(1), 30–67.

Amit, D. J., Gutfreund, H., and Sompolinsky, H. (1987b). Information storage in neural networks with low levels of activity. *Physical Review A (General Physics)*, *35*(5), 2293–2303.

Ans, B. and Rousset, S. (1997). Avoiding catastrophic forgetting by coupling two reverberating neural networks. *Comptes Rendus de l'Academie des Sciences – Series III – Sciences de la Vie*, *320*(12), 989–997.

Ans, B., Rousset, S., French, R. M., and Musca, S. (2002). Preventing catastrophic interference in multiple-sequence learning using coupled reverberating Elman networks. In W. D. Gray and C. D. Schunn (Eds.), *Proceedings of the 24th Annual Meeting of the Cognitive Science Society*, Mahwah, NJ, 71–76. Erlbaum.

Ans, B., Rousset, S., French, R. M., and Musca, S. (2004). Self-refreshing memory in artificial neural networks: learning temporal sequences without catastrophic forgetting. *Connection Science*, *16*(2), 71–99.

Athithan, G. (2002). A local and neurobiologically plausible method of learning correlated patterns. *Neural Networks*, *15*(3), 327–335.

Atkinson, R. C. and Shiffrin, R. M. (1968). *The Psychology of learning and motivation: Advances in research and theory*, Volume 2, Chapter Human memory: A proposed system and its control processes, 89–105. New York: Academic Press.

Bliss, T. V. P., Collingridge, G. L., and Morris, R. G. M. (2003). Introduction. *Phil. Trans. R. Soc. B*, *358*(1432), 607–611.

Bliss, T. V. P. and Lømo, T. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the unanaesthetized rabbit following stimulation of the perforant path. *Journal of Physiology*, *232*(2), 331–356.

Blomfield, S. (1974). Arithmetical Operations Performed by Nerve-cells. *Brain Research*, *69*(1), 115–124.

Bogacz, R. (2001). *Computational models of familiarity discrimination in the perirhinal cortex*. Ph. D. thesis, Department of Computer Science, University of Bristol.

Bogacz, R. and Brown, M. W. (2003). Comparison of computational models of familiarity discrimination in the perirhinal cortex. *Hippocampus*, *13*(4), 494–524.

Bogacz, R., Brown, M. W., and Giraud-Carrier, C. (2001). Model of Familiarity Discrimination in the Perirhinal Cortex. *Journal of Computational Neuroscience*, *10*(1), 5–23.

Bovier, A. (1999). Sharp upper bounds on perfect retrieval in the Hopfield model. *Journal of Applied Probability*, *36*(3), 941–950.

Brown, M. W. and Xiang, J. Z. (1998). Recognition memory: Neuronal substrates of the judgement of prior occurrence. *Progress In Neurobiology*, *55*(2), 149–189.

Burgess, N., Reece, M., and O'Kefee, J. (1994). A Model of Hippocampal Function. *Neural Networks*, *7*(6-7), 1065–1081.

Burgess, N., Shapiro, J. L., and Moore, M. A. (1991). Neural Network Models of List Learning. *Network – Computation in Neural Systems*, *2*, 399–422.

Chechik, G., Meilijson, I., and Ruppin, E. (2001). Effective Neuronal Learning with Ineffective Hebbian Learning Rules. *Neural Computation*, *13*(4), 817–840.

Chen, T. P. and Amari, S. I. (2001). Stability of asymmetric Hopfield networks. *IEEE Transactions on Neural Networks*, *12*(1), 159–163.

Chengxiang, Z., Dasgupta, C., and Singh, M. P. (2000). Retrieval Properties of a Hopfield Model with Random Asymmetric Interactions. *Neural Computation*, *12*(4), 865–880.

Christos, G. A. (1996). Investigation of the Crick-Mitchison reverse-learning dream sleep hypothesis in a dynamical setting. *Neural Networks*, *9*(3), 427–434.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley-Interscience.

Crick, F. and Mitchison, G. (1983). The Function of Dream Sleep. *Nature*, *304*, 111–114.

Crick, F. and Mitchison, G. (1986). REM sleep and neural nets. *Journal of Mind and Behaviour*, *7*, 229–250.

Crook, P., Marsland, S., Hayes, G., and Nehmzow, U. (2002). A Tale of Two Filters – On–line Novelty Detection. In *Proceedings of International Conference on Robotics and Automations(ICRA'02)*, Volume 4, Washington D.C., 3894–3899.

Davachi, L., Mitchell, J. P., and Wagner, A. D. (2003). Multiple routes to memory: Distinct medial temporal lobe processes build item and source memories. *Proceedings of the National Academy of Sciences*, *100*, 2157–2162.

Davey, N., Adams, R. G., and Hunt, S. P. (2000). High Performance Associative Memory Models and Symmetric Connections. In *Proceedings of the International ICSC Congress on Intelligent Systems and Applications (ISA 2000): Symposium on Computational Intelligence (CI 2000)*, Volume 2, Wollongong, Australia, 326–331.

Dayan, P. and Willshaw, D. J. (1991). Optimizing synaptic learning rules in linear associative memories. *Biological Cybernetics*, *65*(4), 253–265.

Diederich, S. and Opper, M. (1987). Learning Of Correlated Patterns In Spin-Glass Networks By Local Learning Rules. *Physical Review Letters*, *58*(9), 949–952.

Durrett, R. (1996). *Probability: Theory and Examples* (Second Edition ed.). Duxbury Press.

Fishbein, W. (1997). Memory Consolidation in REM sleep: Making Dreams Out of Chaos. A Response to R. P. Vertes. *Sleep Research Society Bulletin*, *2*(4), 55–56.

Fishbein, W. and Gutwein, B. M. (1981). *Sleep, Dreams and Memory*, Chapter Paradoxical Sleep and a Theory of Long-Term Memory, 147–182. New York: SP Medical and Scientific Books.

Frean, M. (1990). The Upstart Algorithm: A Method for Constructing and Training Feed-Forward. Technical report, Edinburgh Physics Department.

Frean, M. and Robins, A. (1999). Catastrophic forgetting in simple networks: an analysis of the pseudorehearsal solution. *Network-Computation in Neural Systems*, *10*(3), 227–236.

French, R. M. (1992). Semi-Distributed Representations and Catastrophic Forgetting in Connectionist Networks. *Connection Science*, *4*(3-4), 365–377.

French, R. M. (1994). Dynamically Constraining Connectionist Networks to Produce Distributed, Orthogonal Representations to Reduce Catastrophic Interference. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society*, 335–340. Erlbaum.

French, R. M. (1997). Pseudo-recurrent Connectionist Networks: An Approach to the Sensitivity – Stability Dilemma. *Connection Science*, *9*, 353–380.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Scienes*, *3*(4), 128–135.

Gandolfo, D., Laanait, L., Messager, A., and Ruiz, J. (1999). Memory capacity in neural networks with spatial correlations between attractors. *Physics A–Statistical Mechanics and its Applications*, *264*(1), 305–317.

Gardner, E. (1987). Maximum storage capacity in neural networks. *Europhysics Letters*, *4*, 481–485.

Gardner, E. (1988). The Space of Interactions in Neural Network Models. *Journal of Physics A: Mathematical and General*, *21*, 257–270.

Gardner, E., Stroud, N., and Wallace, D. (1989). Training with noise and the storage of correlated patterns in a neural network model. *Journal of Physics A: Mathematics and General*, *22*, 2019–2030.

Gasparini, S. and Magee, J. C. (2006). State-dependent dendritic computation in hippocampal CA1 pyramidal neurons. *Journal of Neuroscience*, *26*(7), 2088–2100.

Greenberg, R. (1981). *Sleep, Dreams and Memory*, Chapter Dreams and REM Sleep – An Intergrative Approach, 125–133. New York: SP Medical and Scientific Books.

Grossberg, S. (1987). Competitive Learning – From Interactive Activation to Adaptive Resonance. *Cognitive Science*, *11*(1), 23–63.

Hanlon, A. G. (1966). Content-Addressable and Associative Memory Systems. *IEEE Transactions on Electronic Computers*, *EC-15*(4), 509–521.

Hebb, D. O. (1949). *The Organization of Behaviour*. New York: John Wiley & Sons Inc.

Hennevin, E., Hars, B., and Maho, C. (1995). Memory Processing in Paradoxical Sleep. *Sleep Research Society Bulletin*, *1*(3), 44–50.

Hennevin, E., Hars, B., Maho, C., and Bloch, V. (1995). Processing of Learned Information in Paradoxical Sleep: Relevance for Memory. *Behavioural Brain Research*, *69*(1-2), 125–135.

Hertz, J., Krough, A., and Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Redwood City, CA: Westview Press.

Hetherington, P. A. and Seidenberg, M. S. (1989). Is there "catastrophic interference" in connectionist networks? In *Proceedings of the 11th Annual conference of the Cognitive Society*, 26–33. Erlbaum.

Hodgkin, A. L. and Huxley, A. F. (1952). A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve. *Journal of Physiology*, *117*(4), 500–544.

Hopfield, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. In *Proceedings of the National Acadamy of Science, USA*, Volume 79, 2554–2558.

Hopfield, J. J., Feinstein, D. I., and Palmer, R. G. (1983). 'Unlearning' has a stabilizing effect in collective memories. *Nature*, *304*, 158–159.

Horas, J. A. and Bea, E. A. (2002). Distinguishing Spurious and Nominal Attractors Applying Unlearning to an Asymmetric Neural Network. *International Journal of Neural Systems*, *12*(2), 109–116.

Horn, D., Levy, N., and Ruppin, E. (1998a). Memory Maintenance via Neuronal Regulation. *Neural Computation*, *10*(1), 1–18.

Horn, D., Levy, N., and Ruppin, E. (1998b). Neuronal regulation versus synaptic unlearning in memory maintenance mechanisms. *Network – Computation in Neural Systems*, *9*(4), 577–586.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Huber, R., Ghilardi, M. F., Massimini, M., and Tononi, G. (2004). Local sleep and learning. *Nature*, *430*(6995), 78–81.

Izui, Y. and Pentland, A. (1990). Analysis of neural networks with redundancy. *Neural Computation*, *2*(2), 226–238.

Jonas, J. B., Schmidt, A. M., Muller-Bergh, J. A., Schldrzer-Schrehardr, U. M., and Naumann, G. O. H. (1992). Human Optic Nerve Fiber Count and Optic Disc Size. *Investigative Ophthalmology & Visual Science*, *33*(6), 2012–2018.

Káli, S. and Dayan, P. (2004). Off-line replay maintains declarative memories in a model of hippocampal-neocortical interactions. *Nat Neurosci*, *7*(3), 286–294.

Kim, Y. I. and Dudek, F. E. (1993). Intracellular electrophysiological study of suprachiasmatic nucleus neurons in rodents : inhibitory synaptic mechanisms. *The Journal of Physiology*, *464*, 229–243.

Kimoto, T. and Okada, M. (2002). Sparsely encoded associative memory model with forgetting process. *IEICE Transactions on Information and Systems*, *E85–D*(12), 1938–1945.

Kohonen, T. (1987). *Content-Addressable Memories*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Kortge, C. A. (1990). Episodic Memory in Connectionist Networks. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, 764–771. Erlbaum.

Koyama, H., Fujie, N., and Fujiwara, T. (1996). Analysis of the recalling processes of associative memory by an integral representation of the sign function. *Neural Networks*, *9*(5), 737–746.

Kühn, R., Bös, S., and van Hemmen, J. L. (1991). Statistical mechanics for networks of graded-response neurons. *Physical Review A (General Physics)*, *43*(4), 2084–2087.

Lewandowsky, S. (1991). Gradual unlearning and catastrophic interference: A comparison of distributed architectures. In W. E. Hockley and S. Lewandowsky (Eds.), *Relating Theory and Data: Essays on Human Memory in Honour of Bennet B. Murdock*, 445–476. Erlbaum.

Lewandowsky, S. and Li, S.-C. (1994). *Interference and inhibition in cognition*, Chapter Catastrophic Interference in Neural Networks: Causes, solutions, and data, 329–361. San Diago: Academic Press.

Lømo, T. (2003). The discovery of long-term potentiation. *Phil. Trans. R. Soc. B*, *358*(1432), 617–620.

London, M. and Hausser, M. (2005). Dendritic computation. *Annual Review of Neuroscience*, *28*, 503–532.

Löwe, M. (1998). On the storage capacity of Hopfield models with correlated patterns. *Annals of Applied Probability*, *8*(4), 1216–1250.

Löwe, M. (1999). On the storage capacity of the Hopfield model with biased patterns. *IEEE Transactions on Information Theory*, *45*(1), 314–318.

Löwe, M. and Vermet, F. (2005). The storage capacity of the Hopfield model and moderate deviations. *Statistics & Probability Letters*, *75*(4), 237–248.

Malenka, R. C. and Nicoll, R. A. (1999). Long-Term Potentiation–A Decade of Progress? *Science*, *285*, 1870–1874.

Maquet, P., Laureys, S., Peigneux, P., Fuchs, S., Petiau, C., Phillips, C., Aerts, J., Del Fiore, G., Degueldre, C., Meulemans, T., Luxen, A., Franck, G., Van Der Linden, M., Smith, C., and Cleeremans, A. (2000). Experience-dependent changes in cerebral activation during human REM. *Nature Neuroscience*, *3*(8), 831–836.

Maquet, P., Schwartz, S., Passingham, R., and Frith, C. (2003). Sleep-Related Consolidation of a Visuomotor Skill: Brain Mechanisms as Assessed by Functional Magnetic Resonance Imaging. *Journal of Neuroscience*, *23*(4), 1432–1440.

Markou, M. and Singh, S. (2003). Novelty detection: A Review – Part 2: Neural network based approaches. *Signal Processing*, *83*(12), 2499–2521.

Marr, D. and Poggio, T. (1979). Computational Theory of Human Stereo Vision. *Proceedings of the Royal Society of London Series B-Biological Science*, *204*(1156), 301–328.

Marsaglia, G., Zaman, A., and Tsang, W. W. (1990). Toward a Universal Random Number Generator. *Statistics and Probability Letters*, *8*, 35–39.

Mayford, M., Abel, T., and Kandel, E. R. (1995). Transgenic approaches to cognition. *Current Opinion in Neurobiology*, *5*(2), 141–148.

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). Why There Are Complementary Learning Systems in the Hippocampus and Neocortex: Insights From the Successes and Failures of Connectionist Models of Learning and Memory. *Psychological Review*, *102*(3), 419–457.

McCloskey, M. and Cohen, N. J. (1989). *The Psychology of Learning and Motivation*, Volume 23, Chapter Catastrophic interference in connectionist networks: The sequential learning problem., 109–164. New York: Academic Press.

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, *5*, 115–133.

McEliece, R. J., Posner, E. C., Rodemich, E. R., and Venkatesh, S. S. (1987). The Capacity of the Hopfield Associative Memory. *IEEE Transactions on Information Theory*, *33*(4), 461–482.

McRae, K. and Hetherington, P. A. (1993). Catastrophic interference is eliminated in pretrained networks. In *Proceedings of the15th Annual Meeting of the Cognitive Science Society*, Hillsdale, NJ, 723–728. Erlbaum.

Medin, D. L. and Smith, E. E. (1984). Concepts and Concept Formation. *Annual Review of Psychology*, *35*, 113–138.

Meeter, M. (2003). Control of consolidation in neural networks: avoiding runaway effects. *Connection Science*, *15*(1), 45–61.

Meeter, M. and Murre, J. M. J. (2004). Consolidation of Long-Term Memory: Evidence and Alternatives. *Psychological Bulletin*, *130*(6), 843–857.

Mercer, A. R. (2001). *Toward a Theory of Neuroplasticity*, Chapter The predictable plasticity of honey bees., 64–81. Psychology Press.

Minsky, M. and Papert, S. (1969). *Perceptrons, An Introduction to Computational Geometry*. The MIT Press, 1969.

Murre, J. M. J. (1992). *Psychological Plausibility*. Erlbaum.

Murre, J. M. J. (1996). TraceLink: A model of amnesia and consolidation of memory. *Hippocampus*, *6*(6), 675–684.

Muzur, A. (2005). Toward an integrative theory of sleep and dreaming. *Journal of Theoretical Biology*, *233*(1), 103–118.

Nadal, J. P., Toulouse, G., Changeux, J. P., and Dehaene, S. (1986). Networks of Formal Neurons and Memory Palimpsests. *Europhysics Letters*, *1*, 535–542.

Nadel, L., Samsonovich, A., Ryan, L., and Moscovitch, M. (2000). Multiple trace theory of human memory: Computational, neuroimaging, and neuropsychological results. *Hippocampus*, *10*(4), 352–368.

Naundorf, B., Wolf, F., and Volgushev, M. (2006). Unique features of action potential initiation in cortical neurons. *Nature*, *440*, 1060–1063.

Norman, K. A. and O'Reilly, R. C. (2003). Modeling Hippocampal and Neocortical Contributions to Recognition Memory: A Complementary-Learning-Systems Approach. *Psychological Review*, *110*(4), 611–646.

Novikoff, A. B. J. (1962). On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, Volume 12, 615–622. Polytechnic Institute of Brooklyn.

Okada, M. (1996). Notions of associative memory and sparse coding. *Neural Networks*, *9*(8), 1429–1458.

O'Reilly, R. C., Norman, K. A., and McClelland, J. L. (1998). A Hippocampal Model of Recognition Memory. *Advances in Neural Information Processing Systems*, *10*, 73–79.

Pagiamtzis, K. and Sheikholeslami, A. (2006). Content-addressable memory (CAM) circuits and architectures: A tutorial and survey. *IEEE Journal of Solid-state Circuits*, *41*(3), 712–727.

Palm, G. and Sommer, F. (1996). *Models of neural networks III: Association, generalization and representation*, Chapter Associative data storage and retrieval in neural networks., 79118. Berlin: Springer.

Palmer, M. J., Isaac, J. T. R., and Collingridge, G. L. (2004). Multiple, Developmentally Regulated Expression Mechanisms of Long-Term Potentiation at CA1 Synapses. *The Journal of Neuroscience*, *24*(21), 4903–4911.

Pastur, L. A. and Figotin, A. L. (1977). Exactly solvable model of a spin glass. *Soviet Journal of Low Temperature Physics*, *3*, 378–383.

Pavlov, I. P. (1927). *Conditioned Reflexes : An Investigation of the Physiological Activity of the Cerebral Cortex*. Dover Publications.

Pearlman, C. (1971). Latent learning impaired by REM sleep deprivation. *Psychonomic Science*, *25*, 135–136.

Pearlman, C. (1979). REM sleep and information processing: Evidence from animal studies. *Neuroscience & Biobehavioural Reviews*, *3*, 57–68.

Pearlman, C. and Becker, M. (1973). Brief posttrial REM-sleep deprivation impairs discrimination-learning in rats. *Physiological Psychology*, *1*, 373–376.

Pearlman, C. and Becker, M. (1974). REM sleep deprivation impairs bar-press acquisition in rats. *Physiology & Behavior*, *13*, 813–817.

Peigneux, P., Laureys, S., Fuchs, S., Collette, F., Perrin, F., Reggers, J., Phillips, C., Degueldre, C., Del Fiore, G., Aerts, J., Luxen, A., and Maquet, P. (2004). Are spatial memories strengthened in the human hippocampus during slow wave sleep? *Neuron*, *44*(3), 535–545.

Pennartz, C. M. A., Uylings, H. B. M., Barnes, C. A., and McNaughton, B. L. (2002). Memory reactivation and consolidation during sleep: from cellular mechanisms to human performance. *Plasticity in the Adult Brain: From Genes to Neurotherapy*, *138*, 143–166.

Pfeiffer, J. (1965). What's on Your Mind? (Just 10 Million, Million Things). *Popular Mechanic*, *123*(1), 23–27.

Ratcliff, R. (1990). Connectionist Models of Recognition Memory: Constraints Imposed by Learning and Forgetting Functions. *Psychological Review*, *97*(2), 285–308.

Riesenhuber, M. and Poggio, T. (2002). Neural mechanisms of object recognition. *Current Opinion In Neurobiology*, *12*(2), 162–168.

Robins, A. (1995). Catastrophic forgetting, rehearsal, and pseudorehearsal. *Connection Science*, *7*(2), 123–146.

Robins, A. (1996). Consolidation in Neural Networks and in the Sleeping Brain. *Connection Science*, *8*(2), 259–75.

Robins, A. and McCallum, S. (1998). Catastrophic forgetting and the pseudorehearsal solution in Hopfield-type networks. *Connection Science*, *10*(2), 121–135.

Robins, A. and McCallum, S. (1999). The consolidation of learning during sleep: Comparing the pseudorehearsal and unlearning accounts. *Neural Networks*, *12*(7-8), 1191–1206.

Robins, A. and McCallum, S. (2004). A robust method for distinguishing between learned and spurious attractors. *Neural Networks*, *17*(3), 313–326.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Representations by Back-Propogation Errors. *Nature*, *323*(9), 533–536.

Schultz, K. J. and Gulak, P. G. (1996). Fully parallel integrated CAM/RAM using preclassification to enable large capacities. *IEEE Journal of Solid-state Circuits*, *31*(5), 689–699.

Sharkey, N. E. and Sharkey, A. J. C. (1995). An Analysis of Catastrophic Interference. *Connection Science*, *7*(3 & 4), 301–329.

Siegel, J. M. (2001). The REM Sleep-Memory Consolidation Hypothesis. *Science*, *294*(5544), 1058–1063.

Siegel, J. M. (2005). Clues to the functions of mammalian sleep. *Nature*, *437*(7063), 1264–1271.

Silver, D. and Mercer, R. (1996). The parallel transfer of task knowledge using dynamic learning rates based on a measure of relatedness. *Connection Science Special Issue: Transfer in Inductive Systems*, *8*(2), 277–294.

Smith, C. (1993). REM Sleep and Learning: Some Recent Findings. In A. Moffit and M. Kramer (Eds.), *The Functions of Dreaming*. Albany NY: Sate University of New York Press.

Smith, C. (1995). Sleep States and Memory Processes. *Behavioural Brain Research*, *69*(1-2), 137–45.

Smith, C. (1996). Sleep States, Memory Processes and Synaptic Plasticity. *Behavioural Brain Research*, *78*(1), 49–56.

Smith, C. and Lapp, L. (1991). Increases in Number of REMs and REM Density in Humans Following an Intensive Learning Period. *Sleep*, *14*(4), 325–330.

Smith, C. and MacNeill, C. (1994). Impaired Motor Memory For A Pursuit Rotor Task Following Stage-2 Sleep Loss In College Students. *Journal Of Sleep Research*, *3*(4), 206–213.

Squire, L. R., Stark, C. E. L., and Clark, R. E. (2004). The medial temporal lobe. *Annual Review of Neuroscience*, *27*, 279–306.

Stickgold, R. (2005). Sleep-dependent memory consolidation. *Nature*, *437*(7063), 1272–1278.

Stickgold, R., Hobson, J. A., Fosse, R., and Fosse, M. (2001). Sleep, learning, and dreams: Off-line memory reprocessing. *Science*, *294*(5544), 1052–1057.

Stinchcombe, M. B. (1999). Neural network approximation of continuous functionals and continuous functions on compactifications. *Neural Networks*, *12*(3), 467–477.

Storkey, A. (1997). Increasing the Capacity of a Hopfield Network without Sacrificing Functionality. *Lecture Notes in Computer Science*, *1327*, 451–456.

Storkey, A. J. and Valabregue, R. (1999). The basins of attraction of a new Hopfield learning rule. *Neural Networks*, *12*(6), 869–876.

Tulving, E. (1987). Multiple memory systems and consciousness. *Human Neurobiology*, *6*(2), 67–80.

van Hemmen, J. (1997). Hebbian learning, its correlation catastrophe, and unlearning. *Network: Computation in Neural Systems*, *8*(3), V1–V17.

Vertes, R. P. (2004). Memory Consolidation in Sleep: Dream or Reality. *Neuron*, *44*(1), 135–148.

Walker, M. P., Liston, C., Hobson, J. A., and Stickgold, R. (2002). Cognitive flexibility across the sleep-wake cycle: REM-sleep enhancement of anagram problem solving. *Cognitive Brain Research*, *14*(3), 317–324.

Walker, M. P. and Stickgold, R. (2006). Sleep, memory, and plasticity. *Annual Review of Psychology*, *57*, 139–166.

Walker, R. and Russo, V. (2004). Memory consolidation and forgetting during sleep: A neural network model. *Neural Processing Letters*, *19*(2), 147–156.

Widrow, B. and Hoff, M. E. (1960). Adaptative switching circuits. In *IRE Western Electric Show and Convention Record*, 96–104. Part 4.

Willshaw, D. J., Buneman, O. P., and Longuet-Higgins, H. C. (1969). Non-holographic associative memory. *Nature*, *222*, 960–962.

Wimbauer, S., Klemmer, N., and van Hemmen, J. L. (1994). Universality of Learning. *Neural Networks*, *7*(2), 261–270.

Appendix A

Parameters

This appendix contains the actual parameter names used for each simulation using the Hopfield Network program (as per Appendix D, the major piece of code generated during this thesis).

Parameters	Default / Values	Description
N	100	The number of units in the network
activationType	Threshold, Sigmoid	The activation function to use
ON	1	The value of the active response from the threshold function
OFF	-1	The inactive value, -1 in $\mathcal{H}_{N\pm}$, 0 in $\mathcal{H}_{N,0}$
maxCycles	4*N	The number of cycles of relaxation before quitting
relaxation	Asynchronous, Synchronous	Type of relaxation to use. Synchronous updates every unit every step
bias	True, False	Inclusion of the bias unit
randomSeed	1182452015	The seed used for the random number generator
learnTiming	Online, Batch	When to perform weight updates, online after each error or batch at the end of an epoch
symmetricWeights	True, False	Force weights to be symmetric after each learning iteration
weightCappingType	NoCap, HardCap, SoftCap	Weight capping can be hard – fixed total that cannot be exceeded, or soft – were changes are logarithmic above the cap
weightCappingValue	5	The cap value for weight capping
weightDecay	off, on	Applying weight decay after each new learning
weightDecayValue	0.1	Amount of weight decay – 0.1 is 10% where weight = weight*(1-val)

weightNormType	Nonormalisation, NormaliseNet, NormaliseUnit, NormUnit_ZM, NormUnit_Both	Normalisation on a per unit or whole network basis, with either normalisation to a zero sum or resource limiting so that the sum of the absolute value of the input weights is limited to a set value
weightNormValue	$2 * N$	The resource limit value for above
calcHammingDist	on	Calculate the closest learnt pattern to the pattern found by relaxation
noiseRelaxation	False	Apply noise to units as they relax - this moves the Hopfield network toward a Boltzman machine
GausNoiseType	Absolute, Relative	The type of noise to apply during relaxation
stableTimeRelax	100	With noise applied, the length of time a pattern has to remain stable before we consider the network to have relaxed to a particular pattern
GausRelRange	0.1	The relative noise for relaxation
GausAbsRange	$0.1 * N$	The relative range for noise
sampleProbOn	0.5	The coding ratio for probing the network
gnuplotGraphs	off	Change outputs for gnuplot graph package
debugLevel	0	Debugging level for printing information
calcInitChanges	no	Calculate the number of units that change in the first step of relaxation

Learning parameters		
learningType	Hebbian, StrictDelta, PseudoDelta	The learning type to use
numTrials	100	Number of repetitions of the current trial
learningConst	0.1	Learning constant for delta learning
momentum	0	Momentum for delta learning
errorCriteria	0.001	The criterion used to stop learning when there is low error
errorTail	0.9	A decay term used to allow the error to be zero for a few epochs waiting for noise to generate a error. Current error = (previous error * errorTail) + epoch error
errorCalcType	DeltaError, SigmoidError	The type of error signal to use. Delta error generates a -1, 0 or +1, whereas sigmoid error generates an error based on the unit input. Delta error was used in all of the tests in this thesis

allEpochLearning	False	Ignore the error and learn for all the specified epochs
maxLearntPatts	$1 * N$	Number of patterns to learn
pattsFromFile	no	Load the patterns to learn from an external file. If "no" then randomly generate patterns
pattFilename	letters.txt	Filename to load external patterns
generateProbOn	0.5	If we are not reading from a file use this probability for a unit being on in the generated patterns
genExactAct	False	This forces the patterns to have exactly 50% coding ratio
hebbianCycles	1	The number of times that a pattern is presented to the network
ProbBaseLearning	False	This is using Hebbian learning of (Si - a)(Sj - a) rather than (Si)(Sj)
trainingEpochs	500	The number of training epochs for new patterns
initTrainingEpochs	2000	Number of epochs used to learn the base population
numCheck	2000	Number of probes used to calculate the percentage of all the patterns that relax to each attractor
maxSpuriousPatts	500	The maximum number of spurious patterns to store for display and comparison

Repetition Parameters		
initialNumberPatts	5	The size of the base population
step	1	The size of the block of new items to learn - usually 1
numLearntPatts	1*maxLearntPatts	The number of learnt patterns for this particular run
noiseOnInput	yes	Apply ν_i to the input of each unit
GausNoiseTypeIn	Absolute, Relative	The type of noise to apply to the input to units
GausRelativeRange	0.1	The range for relative noise ν_i
GausAbsoluteRange	0.5	The range for absolute noise ν_h
noiseHetroAssociative	yes	Apply noise to the output of the units between setting the desired pattern and setting the initial pattern. Generates a heteroassociative learning task
hetroNoiseLevel	0.05	Percent of patterns to alter the activation state of 5%

Pseudorehearsal parameters		
maxPseudoPatts	256	The number of pseudoitems in the rehearsal population
maxSampPseudoPatts	2000	The total number of attempts to find enough pseudoitems for the rehearsal population
unitsInRatio	0.1	The percentage of the units in the network to include in the energy ratio measure
EnergyRatioCutoff	0.25	The threshold for the energy ratio measure used in "PrER0.25"
useMinLearntvsCutoff	False	A method to lower the energy ratio threshold as patterns are learnt. The threshold is set to either the lowest learnt pattern or the specified cutoff whichever is higher. This is effective, but knowing the lowest learnt patterns energy ratio defeats the purpose of determining which are the learnt patterns
uniquePseudoPatts	True	When generating a pseudoitem population ensures that the patterns only occur once each
delLearntPseudoPatts	False	Actively remove all the learnt patterns from the pseudoitem population
includePseudoInError	True	Add the error on the pseudoitem population to the normal error
pseudoRehearsalReal	Flase	This uses **perfect** knowledge to remove spurious patterns
noiseOnPseudoPatts	True	Use noise on the psuedoitems with the ratios below. Simple pseudorehearsal has this set to False
noiseRatioPseudoPatts	0.5	Noise ratio for pseudoitem patterns
LCRatioPseudoPatts	0.5	Learning constant ratio for pseudoitem patterns

Unlearning Parameters		
maxUnlearningPatts	50	Maximum number of patterns that can be unlearnt
maxProbesUnLPatts	1000	Maximum number of probes used to find unlearning patterns. Mainly useful when removing either learnt patterns or spurious patterns
unlearningCutoffHigh	1	This is used to only unlearnt certain units in the network, unlearn if unit input is >= to this value

unlearningCutoffLow	1	As above but <=. With both set to 1 this is normal unlearning
unlearningCycleSize	50	The number of probes to use for unlearning – Christos uses 50
unlearningConst	0.01	$-\eta$ is the amount of unlearning to apply to each stable pattern
unlearningType	unlearningStd	Various unlearning approaches were tried, none of which proved to be better than standard unlearning
unlearningSuperHeated	False	Unlearning applied only to units with high activation values
uniqueUnlearningPatts	False	Check that a pattern has not already been unlearnt in this cycle of unlearning

Display parameters		
reorderUnitsinPatts	no	Sort the units so they display with the units that are active in the most patterns first
calcSpuriousBasins	no	Calculate the basin of attraction size m_f for the spurious patterns
calcLearntBasins	yes	Calculate the basin of attraction size m_f for the learnt patterns
saveStabProfPattern	0,1	Save the stability profile for this list of patterns, pattern 0 and pattern 1
dispTestedPatts	off	The learnt patterns that were tested
dispLrntPattUnits	off	Activation of every unit in every learnt pattern
dispLrntPattInput	off	Total input to every unit in every learnt pattern
dispLrntPattNetInput	off	Net input to every unit in every learnt pattern
dispLrntPattSummary	on	The summary of the performance of learnt patterns in every learnt pattern
dispSpurPattUnits	off	As above but for spurious
dispSpurPattInput	off	As above but for spurious
dispSpurPattNetInput	off	As above but for spurious
dispSpurPattSummary	off	As above but for spurious
dispSpurPatts	off	
dispSpurPattDetails	off	
dispPrPattUnits	off	As above but for pseudoitems
dispPrPattInput	off	As above but for pseudoitems
dispPrPattNetInput	off	As above but for pseudoitems
dispPrPattSummary	on	As above but for pseudoitems

Appendix B

Simulation Runs

This appendix contains parameters for individual runs as referenced within the body of the thesis.

B.1 Hebbian Learning with Weight Decay

The same parameter set was run in two conditions, the first with the "weightDecay" parameter in the first column set to "off", and second with the parameter set to "on":

Parameter	Value	Parameter	Value
N	100	numTrials	100
activationType	Threshold	learningConst	1
ON	1	maxLearntPatts	$0.5 * N$
OFF	-1	patternsFromFile	no
maxCycles	$4 * N$	trainingEpochs	1
relaxation	Asynchronous	learningType	Hebbian
bias	FALSE	numCheck	0
learnTiming	Online	maxSpuriousPatts	10
weightCappingType	NoCap	initialNumberPatts	3
weightDecay	on	step	1
weightDecayValue	0.1	numLearntPatts	1*maxLearntPatts
weightNormType	NoNormalisation	hebbianCycles	1
calcHammingDist	on	reorderUnitsinPatts	False
noiseRelaxation	no	calcSpuriousBasins	False
sampleProbOn	0.5	calcLearntBasins	False
randomSeed	11		

B.2 Simple Pseudorehearsal

Pseudorehearsal with a small base population of five patterns and 256 randomly generated pseudoitems:

Parameter	Value	Parameter	Value
N	100	numTrials	100
activationType	Threshold	learningConst	0.1
ON	1	maxLearntPatts	2*N
OFF	-1	patternsFromFile	no
maxCycles	$4 * N$	trainingEpochs	500
relaxation	Asynchronous	learningType	PseudoDelta
bias	FALSE	numCheck	1000
learnTiming	Online	maxSpuriousPatts	100
weightCappingType	NoCap	initialNumberPatts	5
weightDecay	off	step	1
weightNormType	NoNormalisation	numLearntPatts	1*maxLearntPatts
calcHammingDist	on	reorderUnitsinPatts	False
noiseRelaxation	no	calcSpuriousBasins	False
sampleProbOn	0.5	calcLearntBasins	True
randomSeed	1182665743		
noiseOnInput	yes	maxPseudoPatts	256
GausNoiseTypeIn	Absolute	maxSampPseudoPatts	2000
GausAbsoluteRange	0.5	EnergyRatioCutoff	0.0
noiseHetroAssociative	no	uniquePseudoPatts	True
noiseOnPseudoPatts	False	delLearntPseudoPatts	False
		includePseudoInError	False
		pseudoRehearsalReal	False
		noiseOnPseudoPatts	False

B.3 Robins and McCallum (1998) Pseudorehearsal

The results from Robins and McCallum (1998) were replicated with variants of the following parameter file:

Parameter	Value	Parameter	Value
N	64	numTrials	100
activationType	Threshold	learningConst	0.1
ON	1	maxLearntPatts	64
OFF	-1	patternsFromFile	yes
maxCycles	$4 * N$	pattFilename	letters.txt
relaxation	Asynchronous	trainingEpochs	500
bias	FALSE	learningType	PseudoDelta
learnTiming	Online	numCheck	1000
weightCappingType	NoCap	maxSpuriousPatts	100
weightDecay	off	initialNumberPatts	44
weightNormType	NoNormalisation	step	1
calcHammingDist	on	numLearntPatts	1*maxLearntPatts
noiseRelaxation	no	reorderUnitsinPatts	False
sampleProbOn	0.5	calcSpuriousBasins	False
randomSeed	42	calcLearntBasins	True
noiseOnInput	yes	maxPseudoPatts	256
GausNoiseTypeIn	Absolute	maxSampPseudoPatts	2000
GausAbsoluteRange	0.5	EnergyRatioCutoff	0.0
noiseHetroAssociative	yes	uniquePseudoPatts	True
hetroNoiseLevel	0.05	delLearntPseudoPatts	False
noiseOnPseudoPatts	False	includePseudoInError	True
		pseudoRehearsalReal	False

B.4 Enhanced Pseudorehearsal

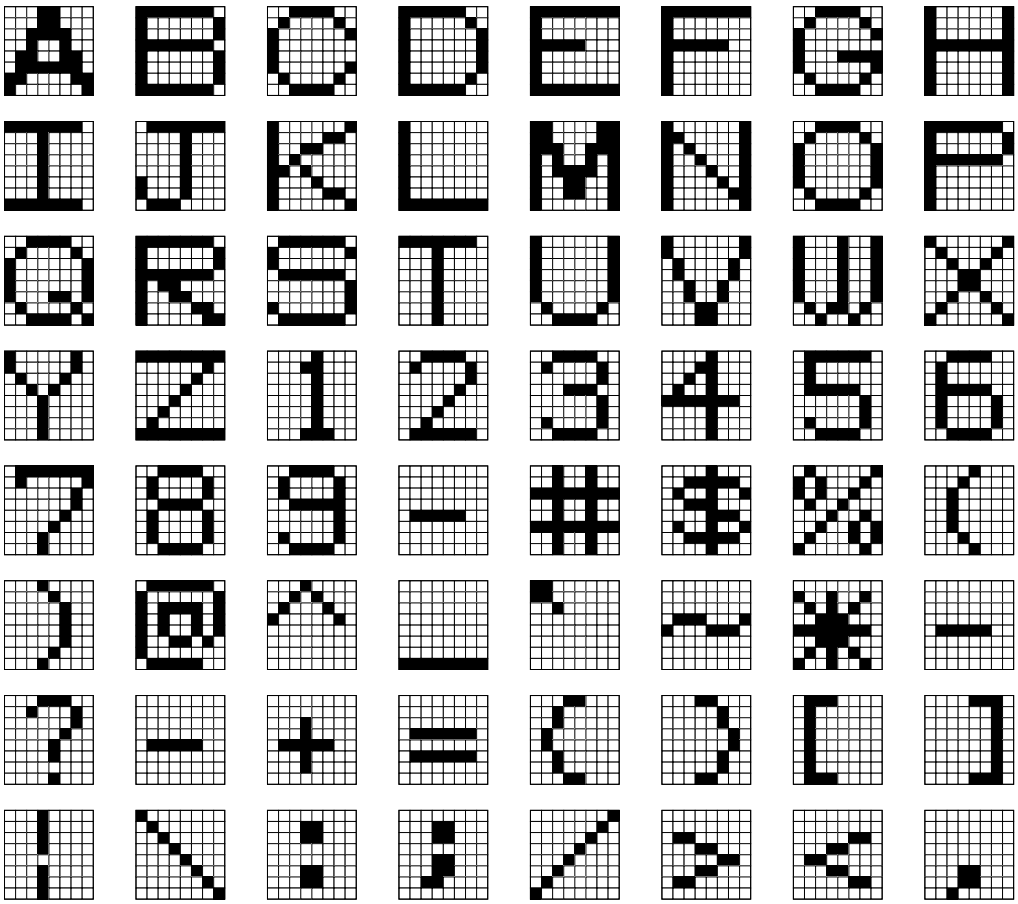
The basic parameter file for pseudorehearsal with the addition of the energy ratio threshold:

Parameter	Value	Parameter	Value
N	100	numTrials	100
activationType	Threshold	learningConst	0.1
ON	1	maxLearntPatts	1*N
OFF	-1	patternsFromFile	yes
maxCycles	$4 * N$	pattFilename	letters.txt
relaxation	Asynchronous	trainingEpochs	500
bias	FALSE	learningType	PseudoDelta
learnTiming	Online	numCheck	1000
weightCappingType	NoCap	maxSpuriousPatts	100
weightDecay	off	initialNumberPatts	44
weightNormType	NoNormalisation	step	1
calcHammingDist	on	numLearntPatts	1*maxLearntPatts
noiseRelaxation	no	reorderUnitsinPatts	False
sampleProbOn	0.5	calcSpuriousBasins	False
randomSeed	1182686337	calcLearntBasins	True
noiseOnInput	yes	maxPseudoPatts	256
GausNoiseTypeIn	Absolute	maxSampPseudoPatts	2000
GausAbsoluteRange	0.5	EnergyRatioCutoff	**0.25**
noiseHetroAssociative	yes	uniquePseudoPatts	True
hetroNoiseLevel	0.05	delLearntPseudoPatts	False
noiseOnPseudoPatts	True	includePseudoInError	True
noiseRatioPseudoPatts	0.5	pseudoRehearsalReal	False
LCRatioPseudoPatts	0.5		

Appendix C

Alphanumeric Data Set

The following are 64 artificially constructed patterns that resemble alphanumeric characters. There are a number of patterns in this set which are highly correlated - the average activation level is approximately 30%.



Appendix D

Source Code and Data

The DVD attached contains:

Thesis Code	All the source code for this thesis
Data	The data gathered during the thesis
Latex Thesis	The full latex source for this document, including all gnuplot scripts, spreadsheets and summary data