Stochastic Neural Networks

Olaf Dean Buhagiar
MSc Advanced Computer Science Computational Intelligence
University of Kent
September 2015
10360 Words

The research work disclosed in this publication is partially funded by the Master it! Scholarship Scheme (Malta). This Scholarship is part-financed by the European Union – European Social Fund (ESF) under Operational Programme II – Cohesion Policy 2007-2013, "Empowering People for More Jobs and a Better Quality Of Life.

Operational Programme II – Cohesion Policy 2007-2013
Empowering People for More Jobs and a Better Quality of Life
Scholarship part-financed by the European Union
European Social Fund (ESF)
Co-financing rate: 85% EU Funds;15% National Funds

Investing in your future

Abstract

Artificial neural networks are systems inspired by biological neural networks, which are used for various purposes. Neural networks are a black box, where small alterations in the design and values might drastically change its behaviour. The aim of this study is to provide some insight into the behaviour of a neural network when stochastic elements are introduced. This dissertation provides a background study of neural networks with particular emphasis on Hopfield networks as well as some known cases of stochastic element introduction in neural networks. The effects on the behaviour of a Hopfield Network resulting from the introduction of a constant recall malfunction is studied. The effect of introducing noise into the weights after training is also studied. This has shown that the network is capable of tolerating a lower levels of noise throughout which the network is still capable of producing good results. Once the tolerance is exceeded the performance of the network degrades drastically rendering it unusable. This behaviour was observed on networks using both the Hebb learning rule as well as the Storkey learning rule. The level of noise that the network is able to withstand depends on the learning rule, the number of memories stored as well as the size of the network.

Acknowledgements

I would like express my thanks to my supervisor Dr. Dominique Chu for his helpful guidance and support throughout the development of this project.

Contents

Chapter 1

Introduction

Inspired by biological neural networks, artificial neural networks are a problem solving approach following nature's design.

Artificial neural networks are able to perform tasks that are not easily achievable using traditional approaches, whether its handwriting recognition or content addressable memory you are unlikely to find a better approach. What makes artificial neural networks unique is that a neural network learns its function, unlike the classical approach where the algorithm contains specific steps that are able to return a result. This provides the neural networks with the power to solve many tasks, however this also makes the systems a black box, thus making it that much harder to understand how certain changes effect the Neural Network.

Biological neural networks are prone to malfunctions and noise. Limited knowledge is available about the effects of noise in an artificial neural network despite their importance and potential.

Stochasticity is an important component in artificial neural networks, introducing random noise into a network may result in making it unusable, or it may improve its power of generalization or even create alternate uses for a neural network such as solving constraint satisfaction problems.

The aim of this dissertation is to study how various forms of stochastic elements affect the neural network. Following an overview of the area, a study of the capabilities of a neural network built is performed. Stochastic elements will then be introduced at various components and of various sizes, where the effects of each case is analysed and studied.

This document is organised as follows: chapter 2, goes through the background of the area on neural networks. This will give a brief insight into biological neural networks, which will lead to a look into its artificial implementation, going into some detail on the some of the more common learning rules and recall methods used. The Hopfield neural network model

is discussed, followed by a look into some techniques and studies relating to stochastic elements and noise introduction in neural networks.

Chapter 3 will then introduce tests studying the behaviour of the network built, followed by a chapter studying the effects of stochasticity on the network's behaviour. The final chapter provides some comments and conclusions about the overall project.

Chapter 2

Background

This section will explore the background of neural networks, briefly touching upon the biological network, then going into how this inspired its artificial counterpart, delving further into the learning and recall methods then exploring how stochastic elements have been used within neural networks in the past.

2.1 Biological Neural Networks

2.1.1 Neurons

A biological neural network is comprised of nerve cells which have the ability to process information called neurons.

A neuron consists of 3 components:

soma: this is the cell body,

dendrites: the neuron has various dendrites, all used in order to receive input signals.

axom: a neuron has one axom, which then brunches out and connects to many other neurons. This sends out an output signal.

As mentioned above, one neuron's axom connects to another's dendrites in order to pass information from one neuron to another. Between the connections of the two neurons, there is a synapse[10].

A synapse generates neurotransmitters when a pulse reaches the synapse. Based on the synapse this may either enhance or inhibit the receptor neuron, affecting its likeliness to emit a pulse. The synapse's effectiveness is altered based on past experience, thus creating a way for the network to learn and store memories[10].

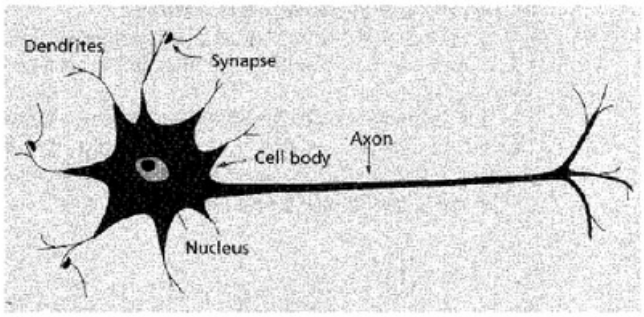


Figure 2.1: A sketch of a biological neuron, from [10]

Each neuron is connected to 10^3 to 10^4 other neurons. The cerebral cortex is known to have about 10^{11} neurons thus resulting in 10^{14} to 10^{15} neuron connections, forming a neural network[10].

2.2 Artificial Neural Networks

Artificial neural network is the result of building a system using the biological neural network as an inspiration.

When implementing the concepts learned from the biological model of neural network into a computational model, the basic building blocks must be implemented in a way to mimic the behaviour of its biological inspiration. The basic elements in question are:

Neuron: Much like the neurons are the building blocks of a biological neural network, so are the artificial neurons to an artificial neural network. Each individual neuron has a binary state which is based upon a number of input values. The following are the most common activation functions used:

- Binary threshold function: this computes a weighted sum, should the result be larger than the threshold, a state of *on* is produced, otherwise the state is set to *off*. This was proposed by McCulloch and Pitts [13].
- Piecewise linear: such a function provides a period which result in a value between the *on* and *off* values. The transition from one to the other is done in a linear function [10].
- Sigmoid function: this is perhaps the most used type of activation function. This uses a sigmoidal function to provide a smooth transition between the *on* and *off* values[5].

Weights: The connections between the neuron contain a weight. These mimic the functionality of the synapse. This weight provides the system with the capability of giving each input its importance, where an input with a large weight has a far greater effect then one with a small weight[10].

While in biology a synapse may either be one that excites or inhibits, here there is only one type of weight, this can mimic either behaviour based on the value of the weight. Should this be a positive one it excites the neuron, if this is negative then the behaviour of inhibition is created[10].

The process of learning consists of changing the value of weights.

There are also various ways of connecting these components in order to create the system. The two predominate types of Neural Networks are:

Feed Forward: In feed forward networks, activity always moves forward to the next layer (thus the name). These are networks that are presented with an input in one layer and generate an output based on it in the output layer[10]. The most common of such networks are the networks following the perceptron model.

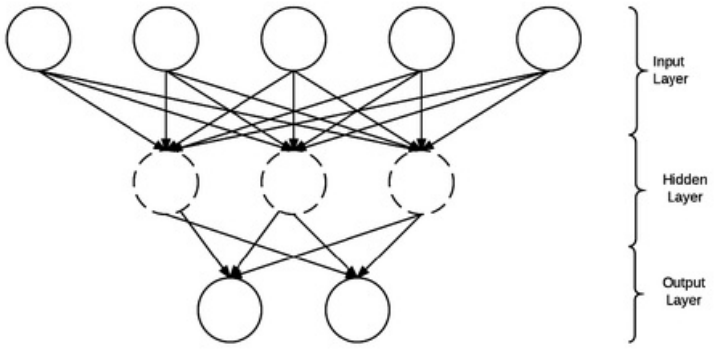


Figure 2.2: An abstract diagram of a feed forward neural network (multilayer perceptron)

Recurrent: Such network introduces loops where the result of a neuron may be passed back to neurons on the same or previous layers, resulting in loops. These are presented with an input pattern and then the network itself alters the states of the input layer resulting in a new state. This gives rise to a loop of altering states that then cause the states to get altered again. The output is presented on the same layer that was used for input once the network has stabilised (i.e. no longer changing the states)[10].

Some of the most common networks of this type are Hopfield networks. This neural network type will be discussed with in further detail below.

2.3 Perceptron Model

This is perhaps the most common type of neural network. This consists of at least two layers of neurons, every neuron in one layer is connected to every neuron in the next layer, where each connection has its own weight. This may have various intermediary layers that provide the system with the ability to solve more complex problems. The system is a feed forward type, in that no layer ever sends any data to the previous layer.

The state of the neurons in first layer, are set by the user as this is the input to the network (thus called the input layer). Based on these, the states of the other layers are set. The result is then produced as the states of the output layer. The states of any intermediary layers are not visible and are only used by the system to help produce the provided output[5].

The original aim of the perceptron model is to be able to classify a given input. In order for a basic single layer perceptron neural network to be able to classify between two classes, the inputs must be linearly separable. In order for such a network to be able to work with more complicated classes, hidden intermediary layer/s must be added[5].

2.4 Hopfield Network

In 1982, John Hopfield published a paper proposing a new type of neural network, this network is now (appropriately) known as Hopfield Network. The aim of a Hopfield neural network was that of memory storage, going so far as to propose that this might be a way to fix retrieval errors in hardware rather than software[8].

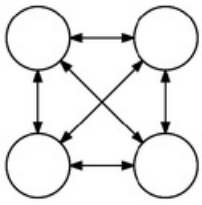


Figure 2.3: An abstract diagram of a Hopfield neural network with 4 neurons

Much like typical artificial neural network, the network proposed by Hopfield consists of neurons used as processing elements, having a Boolean state of 1 to represent firing state(*on*) and 0 as the alternative(*off*), and weighed connections between the neurons.

Hopfield also suggested that the neuron's states may be represented by positive and negative rather then 1 and 0, that is, using +1 to represent a state of *on* and -1 to indicate a state of *off*[8].

Unlike the perceptron model, a Hopfield network is a fully connected network, thus all neurons have an effect on each other, which the perceptron model cannot handle. Beyond this, the perceptron model is a synchronous model, where all the neurons are updated at once, however, this is a property that cannot be proven in biological neural networks, and thus Hopfield does not force this property on his proposed network[8].

2.4.1 Learning

Like any other neural network, the network proposed by Hopfield must first be trained before being operational. The training of a Hopfield network consists of updating the weights connecting the neurons. Since a neuron should not be connected to itself, the weight for connections between a neuron and itself must in all cases be set to 0. The other restriction with regards to the weights in a Hopfield Network is that weights are symmetric, thus the weight between neurons i and j is equivalent to that between neurons j and i[8].

The various properties that a learning rule might have are as follows:

Local: A local learning rule is one where the weights are trained based on information available to the neurons connected to the weight being updated, so when training weight *ij*, if a learning rule were to require the state of neuron *j+1*, then this cannot be considered local, however, if it were to only require the state of neurons i and/or j then it is local[17][9].

This offers two major benefits, making learning a process that could be done for multiple weights in parallel, while offering biological plausibility[4].

Incremental: as the name suggests, an incremental learning rule is one that builds upon previously learned patterns, and thus when learning a new pattern all the patterns already learned need not be known[17][9]. Thanks to this property the learning procedure need not be a batch process, thus making it an adaptive system, by making it possible for the system to learn as it goes along. This is also of extreme value when dealing with large sized data sets and streaming data[9].

Immediate: if a learning rule is such then it would perform an immediate update as opposed to a limited process. This makes learning procedure faster[17].

Capacity: a learning rule's capacity indicates the number of patterns(memories) it is able to teach the network as a ratio to the network's number of neurons. Capacity is only partially depended on the learning rule, as it also depends upon the patterns' correlation[17][9].

Furthermore, a capacity may be subdivided into two:

Absolute Capacity: this is the network's capacity if it is expected to recall the patterns perfectly[17].

Relative Capacity: this is the network's capacity if it is acceptable for it to recall the patterns with a degree of error[17]. The level of error considered as acceptable is not a fixed one, however it is usually a relatively small value.

The relative capacity is, if all properties are equal, higher than the absolute capacity, however it must be noted that in either case, even though a high level of capacity is desired this must not be achieved at the cost of generalization as may be the case[9].

Once trained, assuming the capacity has not been exceeded, the learned patterns (which will be referred to as memories), will become stable states, fixed points which will behave as attractors, having a basin of attraction, where all patterns in that area are attracted to the particular memory[14][1].

Hebb Rule

The simplest training rule is the Hebb rule, this is the rule used by Hopfield within his paper introducing the network being discussed[8].

This rule was first suggested by D.O. Hebb in his book *The Organization of Behaviour: A Neuropsychological Theory*[6]. Hebb's original conception of this rule may be summed up with the following quote:

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

[6, Chapter 4, p.62]

By converting the above idea into a mathematical formula, the result is:

$$\Delta W_{ij} = \xi_i \xi_j \tag{2.1}$$

Where W_{ij} is the weight connecting neuron i to neuron j and ξ_i is the state of the ith neuron of the current pattern.

While some sources[9] use this as the learning rule, many others make use of an updated version of this rule.

By simply analysing the formula above, an issue might become apparent, that is that the weights will continue to grow. To counter this many make use of two variations of the formula above.

The first being the following:

$$\Delta W_{ij} = \frac{1}{N}\xi_i\xi_j \tag{2.2}$$

where N is the number of neurons.

In equation (2.2) the product of the neuron's values is divided by the number of neurons in the system. This will keep the ratio of the weights consistent, thus still implementing Hebb's theory, while making sure the values do not grow[17][4].

The second variation, uses the same logic as the above formula, however, instead of dividing by the number of neurons, the product is divided by the number of patterns, such that:

$$\Delta W_{ij} = \frac{1}{P}\xi_i\xi_j \tag{2.3}$$

Where P is the number of memories in the network.

This is the variation used by Hopfield in his original paper[8].
Utilizing this rule or any of the other variation simply consists of setting up all the weights in the system to 0, then going through each one and updating it based on the formula being used.

The value of W_{ij} must be equal to 0 for all $i = j$, so as to preserve the Hopfield property where no neuron is connected to itself.

In its original form this learning rule is local, incremental as well as immediate. Should equation (2.3) be used, the rule loses its incremental property as the number of memories must be known before the first pattern is ever learned. As for the capacity of this rule, this has an absolute capacity of $\frac{N}{4ln\,N}$ where N is the size of the network (i.e. number of neurons)[14].

The Relative capacity of the Hebb rule is however more problematic since this depends on the level of error which is deemed acceptable. In fact multiple sources provide different values for this. As part of his research Hopfield identified the value to be $0.15N$[8], yet

Davey and Hunt use $0.14N$[4] while as part of the aforementioned research, McEliece states this is equal to $\frac{N}{2ln\ N}$[14].

Pseudo Inverse Rule

The Pseudo Inverse learning rule might very well be the learning rule with the highest storage capacity of N, this rule however is neither incremental, nor local, and thus a network using this must learn all patterns at once[17][9].

Pseudo Inverse learning follows the following rule:

$$W_{ij} = \frac{1}{n} \sum_{v=1}^{m} \sum_{\mu=1}^{m} \xi_i^v (Q^{-1})^{v\mu} \xi_j^\mu \tag{2.4}$$

where

$$Q = \frac{1}{n} \sum_{k=1}^{n} \xi_k^v \xi_k^\mu \tag{2.5}$$

Storkey Rule

The Storkey learning rule, suggested by Amos Storkey[17], is as follows:

$$W_{ij} = \frac{1}{N}\xi_i\xi_j - \frac{1}{N}\xi_i h_j - \frac{1}{N}\xi_j h_i \tag{2.6}$$

where

$$h_x = \sum_{k=1}^{n} W_{xk}\xi_k \tag{2.7}$$

which is a form of local field.

This rule may be seen as either an approximation to the pseudo inverse rule or a rule derived from the Hebb rule applied to residuals[9]. This rule, like the Hebb rule, is local, incremental as well as immediate. This provides a capacity of $\frac{N}{\sqrt{2ln\ N}}$ [17][4].

Hu also suggest the usage of the 2nd order of this rule, that is:

$$W_{ij} = \frac{1}{N}\xi_i xi_j - \frac{1}{N}\xi_i h_j - \frac{1}{N}\xi_j h_i + \frac{1}{N}h_i h_j \tag{2.8}$$

which based on his research provides for better results when considering the number of error bits per recall then its 1st order and the Hebb rule[9].

Local Learning Model

This approach is one which, unlike the previously mentioned learning algorithms, does not learn after one iteration, this must go through a learning phase, where the weights are

updated until all patterns are fixed points[4].

The weights are updated using the following:

$$W_{ij} = W_{ij} + \frac{\xi_i \xi_j}{n-1} \tag{2.9}$$

In this rule the weights are updated in cases where, after setting the states to equal the particular pattern, a recall would result in the neuron changing its state. This is repeated for all neurons of all patterns being learned, until they are all fixed points[4].

When using this type of training there is also the possibility that the system cannot handle the given patterns, in which case it will never converge and run into an infinite loop[4].

This is a local rule but not incremental (the weights build on top of the previous iteration's weights, however all patterns must be known at the start of training phase).

2.4.2 Recall

In his 1982 paper, Hopfield defined an energy function for the network[8]. The energy for a network is based on the states of the neurons.

$$E = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1, i \neq j}^{N} W_{ij} \xi_i \xi_j \tag{2.10}$$

Energy in a system should always move down, eventually reaching a minimum point, at which point the energy of the system remains constant. This implies that the result from an energy function should always be less then the previous energy level. This is true in the case of Hopfield networks as well.

A neuron's state is updated by checking its contribution to the overall energy for each of the states, the state producing the lower level of energy is selected, thus ensuring that it is not possible for a change of state to result in a larger energy level then the initial state.

However, when performing a recall, only one neuron is updated at a time, thus only the energy resulting from its connections may change. Taking this into account the difference from the previous energy state may be deduced as follows:

$$\Delta E = -\xi_i \sum_{j=0, j \neq i}^{N} W_{ij} \xi_j \tag{2.11}$$

If one considers that the possibilities for ξ_i are -1 or 1, it becomes clear that the resulting value for ΔE in either case is the same, with the exception of the sign.

Thus, the value of ξ_i is to be fully dependent upon whether $\sum_{j=1, j \neq i}^{N} W_{ij} \xi_j$ is a positive

or negative.

Therefore, a recall for ξ_i may be seen as:

$$\xi_i = F_h \left(\sum_{j=1, j \neq i}^{N} W_{ij} \xi_j \right) \tag{2.12}$$

where $F_h(x)$ return 1 when $x \geq 0$ and -1 when $x < 0$[3].

It is worth noting however, that had F_h been set to return -1 if the value of x is 0 (rather then 1), it would have also been correct as in either case the overall energy would not change (as a result of 0 indicates that there will be no change in energy) in fact another approach that could be taken is to leave ξ_i in its original state should $\sum_{j=1, j \neq i}^{N} W_{ij} \xi_j$ result to 0.

There are two known approaches to performing the overall recall of a Hopfield network:

Asynchronous Recall: When performing an asynchronous update, a number of random neurons are selected, and the new state for each is calculated using the current state, that is, if neuron i's state is updated, the state of neuron $i+1$ is updated based on i's new state. This is the approach used by Hopfield in this paper[8][14].

The probability of any neuron being selected is $\frac{1}{N}$, irrespective of the previously updated neurons and their initial or final state[14][8].

Synchronous Recall: In this case, the Neurons are all updated simultaneously, that is, if neuron i's state is updated, neuron $i+1$'s state is calculated using i's original state, rather than its new state[14]. Hopfield noted that this is not biologically plausible and thus he did not look into this approach, despite the fact that this opens the possibility of parallelising the recall procedure[8].

By running the above procedure several times, constantly driving the system's energy down in the process, the system will eventually settle at a stable state, indicating an energy minimum[8].

This process may also be seen as network's state being pulled to an attractor (a memory).
Settling in a pattern may indicate various things:

- A memory has been found. This is, of course, the ideal scenario. In such a case it is likely that the resulting memory is the one closest to the original pattern, however, this may not always be the case. Even in cases where the states may start off well within a memory's basin of attraction, the end result might end up being a different memory

based on the first neuron updated[14]. Using an example from [14] to illustrate this, if a neural network was trained and now stores the following memories:

- (1,1,1,1,1)
- (1,-1,-1,1,-1)
- (-1,1,-1,-1,-1)

the weight matrix would result as follows:

$$W = \begin{matrix} 0 & -1 & -1 & 1 & 3 \\ -1 & 0 & 1 & -1 & 1 \\ 1 & 1 & 0 & 1 & 3 \\ 3 & -1 & 1 & 0 & 1 \\ 1 & 1 & 3 & 1 & 0 \end{matrix}$$

If the test pattern was (1,-1,-1,1,1), the closest memory is (1,-1,-1,1,-1). If the first recall the fifth bit were to be updated, the weighed sum would result in -2, thus changing the fifth neuron to -1, and reaching the closest memory. However had the third bit been updated first, the weighed sum would be that of 4, thus updating the pattern to : (1,-1,1,1,1). If this is followed by an update to the second neuron, this would be updated to 1, resulting in a memory that was not the closest memory[14].

- A spurious minima has been reached. This may occur if two or more patterns are too similar to each other, as they might produce one minimum rather than two, thus resulting in a situation where the network is unable to recall the memories individually but is only able to settle on a pattern amalgamating them. This also starts occurring when the network is asked to store more memories then it could hold based on the current learning rule.

2.5 Stochastic features in Neural Networks

Introducing randomness into a neural network is not a novel idea, however detailed studies as to the effect of noise are scarce. Stochasticity has been seen as a possible solution to various problems found in neural networks, such as over fitting. What follows is a look into the various stochastic techniques used with neural networks and the problems they address.

2.5.1 Stochastic Activation Function

This is perhaps the most widely used technique of this type.

Simulated Annealing

The process of simulated annealing is a processed derived from the annealing process, which consists of melting a substance (such as metal), then slowly lowering its temperature until reaching freezing temperature (or close to), so that the substance will achieve equilibrium avoiding many defects in the resulting object[12].

Simulated annealing follows the process described above by calculating the system's energy, then a random neuron is given an amount of (randomly generated) displacement, if the energy has been reduced, this is used for the next step, otherwise, the decision whether to keep this displacement, or revert back for the next step is based upon the Boltzmann probability, that is:

$$\exp(-\frac{\Delta E}{k_B T}) \tag{2.13}$$

where T is the temperature and k_B is Boltzmann's constant[12].

By analysis of the probability above, it is easy to deduce that as the temperature decreases, so does the probability that a change leading to a higher energy state is used.

Thus while the system still has a high temperature it is capable of moving around, and escaping any local minimum it might get stuck on, however as this gets smaller, the likeliness of that occurring gets increasingly small. When the temperature reaches 0, this would be impossible, thus when used the temperature is initially set to a large value gradually reduced [12].

Boltzmann Machine

A Boltzmann Machine is a Hopfield network, where simulated annealing is used to allow the system to escape local minima by jumping to higher energy levels in some occasions. Another difference in a Boltzmann machine is that it also contains some hidden units[2].

Despite being very similar to a Hopfield machine, the resulting effect of the simulated annealing is that the Boltzmann machine finds the global minimum rather than a local one. This thus renders the system useless for memory storage as it will only remember the strongest memory, however Boltzmann machines have proved very useful in solving constraint satisfaction problems, where the aim is to find the global minimum[2][19].

2.5.2 Stochastic Weights

Although not very common, some research has also been done on the effects of having weights that are stochastic. Zhao and Shawe-Taylor[20] implemented such a network, *a Stochastically Connected Boltzmann Machine*.

It has been found that despite its differences from a Boltzmann machine the resulting network, if not too small, would still behave in a similar fashion to a Boltzmann machine[20].

2.5.3 Stochastic Patterns

A way in which randomness has been known to be used in the field of neural networks is during the training of the network. This technique is known as noise injection.

In noise injection, random values are added to components of the neural network throughout its training phases. In most cases noise is introduced in:

- the neuron: by randomly fixing a neuron's state, noise is being introduced into the system.
- the weights: random weight values are updated with a value based on a random integer. These may be added to the weights or multiplied to them, the resulting effect is the same.
- the input: here noise is introduced as part of the input patterns essentially jittering to the pattern.

These techniques have been used either alone or in conjunction with each other. Whatever the case, noise is usually introduced into the system before each training step, and thus this is a technique that is used with learning rules that require various iterations through the patterns[7][11].

The effect of noise during learning is that the network has a higher level of fault tolerance and increases generalisation[7][11].

The stochastic patterns techniques are used in feed forward neural networks (using supervised learning)[18], where generalisation is desired and fault tolerance is not a feature of the network itself.

Based on research in [18], applying noise injection on neural networks using unsupervised learning techniques results in no improvement and has an adverse effect on the system where the network's classification ability is degraded by the noise in the patterns[18].

2.5.4 Stochastic Learning

Sompolinsky[16], looked into various variations of the Hebb rule. The inspiration behind this study was, like artificial neural networks themselves, obtained from biology, where it is unrealistic to think that biological neural networks are fully connected, as well as to think that the value of the synapse is solely the result of the learning process. Beyond this, this study also aimed at trying to find a way for the Hebb learning rule, or rather, an upgraded version of it, to be used in order to learn correlated patterns. This would provide a higher level of biological plausibility, as well as making Hopfield networks more useful in some areas[16].

An approach taken was that during the learning phase, the weights were set using a Hebb rule with a level of static noise (obtained via a Gaussian distribution) added to it[16].

When comparing the performance of the network with the amount of noise, it is concluded that a Hopfield network is able to tolerate a "moderate level" of noise within the synapse[16]. Furthermore according to the conclusions drawn here noise only has a significant effect on the system when there is more then $\frac{1}{\sqrt{N}}$ level of relative noise in the system[16].

2.5.5 Diluted Hopfield network

A technique that has also been studied in Sompolinsky's study[16] is a technique mimicking an open circuit. This consists of disconnecting a number of connections in a fully connected Hopfield neural network, by making sure that the neurons linked via these connections have no effect on the state of each other[16].

To achieve this effect Sompolinsky proposed a variation on the Hebb learning rule, as follows:

$$W_{ij} = \frac{c_{ij}}{N} \sum_{k=1}^{P} \xi_i^k \xi_j^k \tag{2.14}$$

Where c_{ij}, may be 0 or 1. c_{ij}'s value is randomly generated based on a given probability.

Equation (2.14) is an upgraded version of the Hebb rule as seen in equation (2.2) for the weight ij. Thanks to the addition of c_{ij}, some weights end up with a value of 0, thus disconnecting the neurons they link. Symmetry is still being preserved here as $c_{ij} = c_{ji}$[16].

Chung and Krile built upon this and deduced that the disconnected weights reduce performance in the system especially as the network size is increased. Much like in a normal network the resulting performance of a network containing a number of disconnected weights also changes based on the amount of memories being stored[3].

In his work Sompolinsky found that when 40% of the connections were disconnected, the level of error was still below 2.5%, when taking 90% as the minimum acceptable level of recall (i.e. at least 90% of the result bits must be equal to the memory bits)[16].

Chang and Krile also look into another network technique, this time diluting the network by mimicking short circuit behaviour. In this technique a large signal is applied through the interconnections. In practice this is implemented by splitting up the recall function in two parts, the first part which deals with normal connections and the second deals with connections effected by the short circuit, as follows:

$$\xi_i = F_h\left(\sum_{j=1, j\notin A_i}^{N} W_{ij}\xi_j + \sum_{j=1, j\in A_i}^{N} sgn(W_{ij})\xi_j G\right) \tag{2.15}$$

where G is a positive noise value, $sgn(x)$ is a function returning 1 when $x > 0$, -1 when $x < 0$, and 0 when $x = 0$, and A_i is the set of neurons whose connection to i is effected by the short circuit[3].

Equation (2.15) effectively updates the recall equation by making it the result of two components, the first part dealing with the connections not effected, by simply performing a weighed sum (as is the case with a normal recall), while the second part deals with the effected connections. Here the value of the neuron on the other end is multiplied by a pre-set value G and -1 if the weight was negative. If the weight was 0, this is multiplied by 0, thus having the overall affect of not creating any connections from a neuron to itself (whose weight is 0).

Much like the previous technique the performance of the network decreases with the number of memories stored. Performance also decreases as the level of problematic connections added is increased. As for the noise creating variable G, the optimal value for this was found to be that of 5[3].

When comparing the two approaches, it was concluded that open circuited network perform better then short circuited once for the same amount of effected weights[3].

2.5.6 Anti-Symmetric Noise addition to weights

A study by M.P Singh[15], studied the effects of antisymmetric noise in a symmetric neural network. In this study, the weights of the network were split into two components, the first being the symmetric component, which is obtained using the Hebb rule, while the second is the antisymmetric part, which is randomly drawn from a Gaussian distribution. The asymmetric component was multiplied by a fixed constant k thus controlling its strength on the overall weight. The second component of the weight ij is not the same as that of ji,

these values are taken to be opposites (i.e. the same value, but one is positive, the other negative). Finally the connection between a neuron and itself is still set to 0.

From the tests performed on this system, it was deduced that for the initial recall the antisymmetric component is treated as noise resulting from overloading of memories. Beyond this point the noise starts effecting the result, where the larger the constant k, the larger the error[15].

Another effect that this has on the behaviour of the network is that the larger the k the smaller the basin of attraction of the memory became[15].

Chapter 3

Experiments

The main objective of this study is to understand how stochasticity affects the behaviour of a neural network. In order to achieve this goal, a Hopfield neural network was built which is used to analyse the effects of the noise.

The first step however is to test the network in its original form and ascertain its capabilities. These would later be used as the baseline and the benchmark when analysing results from a stochastic system.

3.1 Test Values

The first thing to go into with respect to the tests performed is the generation of the patterns.

Patterns used for tests, memories or test patterns, are always saved. These are saved in different files however they are linked. Whenever a pattern is needed, the patterns with the provided file name is loaded, if in existence, otherwise, a set of patterns with the given specifications are generated and saved with the given name. The same occurs in the case of the test set. This makes it possible to use the same set of memories and tests for multiple times whether it's for repeating the same tests or using the same patterns for new tests.

Generating the Test Patterns

The set of patterns used for memory are generated by simply generating a set of random lists of the desired size.

The process is not so straight forward when generating the patterns that are to be used as test patterns. Every test set consists of a subset of tests generated for each specific memory. When creating a test set the number of mutations is to be specified. A memory is

mutated, and the result is saved as a test pattern, this is then mutated again into another test pattern. This process is repeated for the number of test patterns required.

This process is then repeated for 10 times, and the whole process is repeated for every memory, thus creating a set of test patterns.

Therefore if generating a test set for 5 memories, and asking for 5 mutations, a test set of 250 test patterns is generated, which is comprised of 5 sets (one per memory), each of which is made up of 50 tests (10 groups of 5 mutations each).

In the tests below the test patterns used were all generated using 10 mutations.

3.2 Error Measurement

Error plays an integral part in most, if not all, tests and thus it is important to explain what error is used and how this is calculated. Two different types of errors are used throughout. These have already been mentioned in passing in the previous section, however due to their importance in this section it is best to discuss them in detail.

The errors here were designed to calculate error in a scenario where a pattern's intended result is known.

Recall Bit Error: this is a rather trivial way of calculating the amount of errors obtained, as this will add an error for every test pattern that fails to reach the desired memory. A particular pattern is considered to have reached the desired memory, if all its states are equivalent to that of the memory or all states are the inverse of the memory.

This error method does not take into account how far a pattern is from the memory, a pattern with one bit out of place is considered to be as errorous as one with half its bits incorrect.

Absolute Recall Error: This method does not look into how many patterns failed to reach the desired memory but by how much they failed to do so. For each pattern that fails to reach the intended memory this calculates the distance to the memory (and its inverse, choosing the smallest of the two), and calculates this as a percentage of the number of neurons in the pattern.

3.3 Preliminary Tests

3.3.1 Parameter Rationale

The following parameters are used to set-up various aspects of the network:

Neuron Selection: when a recall is performed only a percentage of the neurons are updated. The percentage is determined by this setting.

Number of Recalls: for every test performed, a number of recalls are performed, where the result of the previous recall is used as the input for the next recall (where the test pattern is used as the input for the first). The resulting pattern after all recalls is then compared to the memory and the error determined. The number of recalls performed is also set as a setting.

Recall Method: as implemented the recall method to be used must be set as well. Hopfield networks have two known types of recall methods, synchronous and asynchronous recall (details on both may be found in Section 2.4.2), both methods were implemented and thus the choice of which to use in each test must be taken.

Learning Rule: since multiple learning rules were implemented each network needed to know which rule to make use of. This is specified at the creation of the network instance.

Mutation percentage: this setting is used when creating test patterns. Thanks to this the tool know how different a mutated pattern is to be from its original form. Though this was adjustable, this was set to 10(%) throughout.

The affects that these parameters have on the network were tested.

Number of Updated Neurons per Recalls

This first test will investigate how the error changes based on the percentage of neurons updated every recall. This is done by first learning the memories provided, then performing a recall loop for multiple times each time changing the setting for the percentage of neurons updated per recall. All other variables remain constant throughout the test.

This test was repeated twice, one making use of the asynchronous recall method, and one making use of the synchronous recall method. For this test the number of neurons was set to 100, as was the number of recalls, while the three memories were thought in each case (the same memories and tests were used for both types of recall so as to investigate the performance in the same environment).

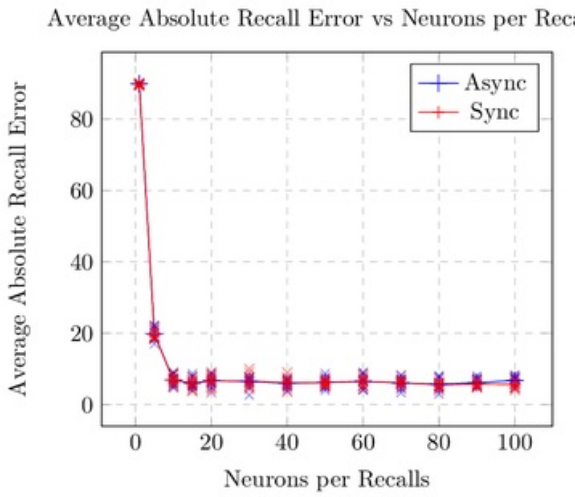


Figure 3.1: Average Absolute Recall Error vs Neurons per Recalls - This shows how the Average Absolute Recall error changes with the number of neurons updated for every recall. The settings used were 100 neurons per pattern, a 100 recalls, 3 memories

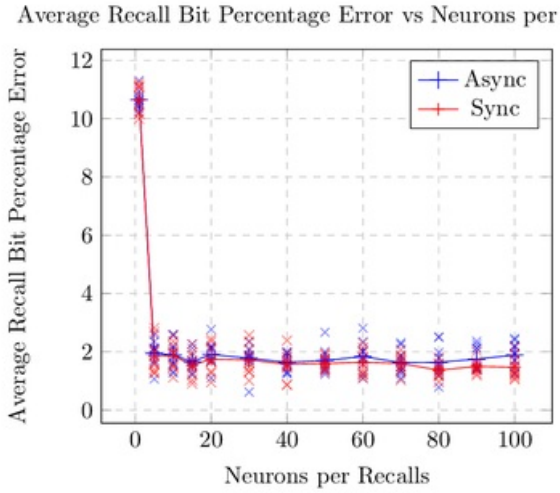


Figure 3.2: Average Recall Bit Percentage Error vs Neurons per Recalls - This shows how the Average recall bit percentage error changes with the number of neurons updated with every recall performed. The settings used were 100 neurons per pattern, a 100 recalls, 3 memories.

Perhaps the easiest observation when looking at figures 3.1 and 3.2 is that the difference in the performance between the async recall and the sync recall is very small. When looking at the absolute recall this is the same for almost all points.

When analysing the updates per recall, it is apparent that if this is set to a value less than 10% the number of resulting error is quite high, beyond this point however the error value remains rather consistent.

However, since the aim of this test was to determine the best value to be used as a default value for this property further inspection on the results was performed. From this it was deduced that the ideal value would be that of 80%. This performs slightly better than other values for both sync and async. This can best be seen in the sync result in figure 3.2.

Number of Recalls

Next we test the effect of the number of recalls. If the number of recalls is too small then the states would not have time to reach the attractor, however if ran for too long each test would consume computing resources on recalls where nothing is updated as the minimum has already been reached.

This test helps identify the behaviour of the network as the number of recalls is increased. This follows the same procedure as the previous test, however in this case the number of neurons per recall is fixed and the number of recalls is changed with every iteration instead.

Like the previous test, this was performed for both async and sync recall with 80% update per recall using 2 memories of a 100 neurons each. This test was also performed on a set of 3 memories of 1000 neurons so as to be able to identify if the behaviour remains the same with a network of a different size.

By analysing figures 3.3 and 3.4 it is clear that the error decreases to a point then remains constant. The difference between the 1000 neurons test and the 100 neurons is that the 1000 neurons takes slightly longer to settle, then the 100 neuron tests.

In either case the error seems to have settled by 20 Recalls. By close examination of the results obtained (namely the values on which the graphs are based), it would seem that 40 is the number of recalls that gives a slightly lower result then the other values. This was thus chosen as the default value.

When looking at the difference between the sync and async recall of both the previous test, it is clear that it is not worthwhile to proceed using both recalls as there is no significant difference in the results and thus the results of one type of recall are a good enough indicator for the results that would be obtained by the other.

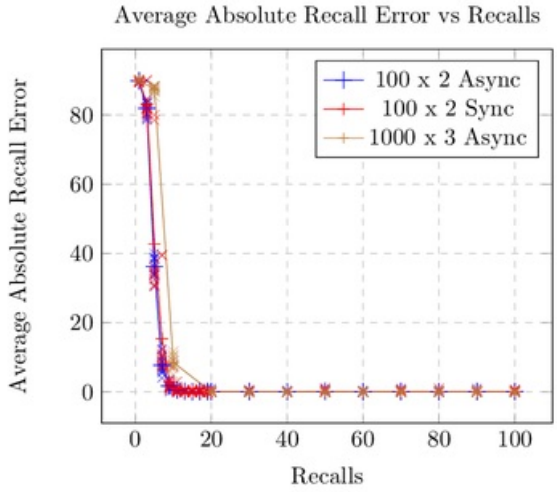


Figure 3.3: Average Absolute Recall Error vs Recalls -This shows how the Average Absolute Recall error changes with the number of recalls performed. This used 80% neurons per recall.

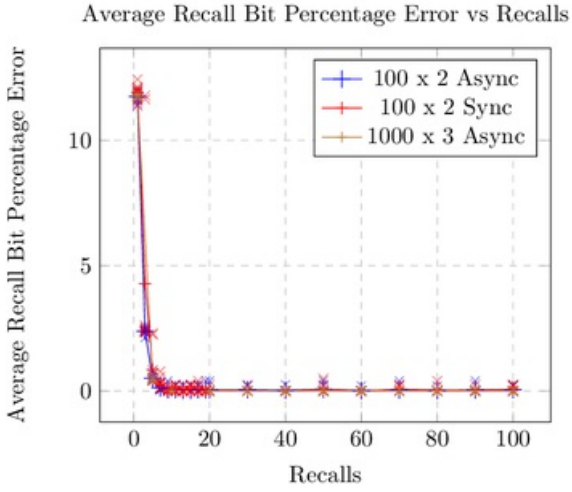


Figure 3.4: Average Recall Bit Percentage Error vs Recalls - This shows how the Average recall bit percentage error changes with the number of recalls performed. This used 80% neurons per recall.

From this point forth all tests will be run using the async recall. This was chosen over the other as this was the method used by Hopfield in his paper[8].

Thanks to the previous tests the behaviour of the parameters has been investigated, allowing informed decisions to be taken with regards to the aforementioned settings. A set of settings to be used as the default setting have been decided upon which will be used when moving forward.

From this point forward, if a particular setting not under test is not specified, then the default value is being used.

3.3.2 Storage Capabilities

What follows is an investigation into the network's capabilities. This predominantly consists of looking at the network's ability to store memories and how well it is able to recall them once they have been learnt.

Error vs Patterns Learned

This test is intended to evaluate the network's capacity. This test is provided with a set of patterns to be used as memories and a set derived from them to be used as test patterns. The tests consist of the following process:

```
learn the first p patterns
    errors = 0
    for each of the first p patterns
        for pattern i select the corresponding set of test patterns
            for each test pattern
                 set states to test pattern
                 recall (for a max of n times)
                 error += calculate error for pattern
        save p, average error
    p++
```

This process was followed to calculate both the absolute recall error as well as the recall bit error. This test was performed on a network using the Hebb learning rule, the Storkey rule and the 2nd order of the Storkey rule.

The error for the network when trained with the Hebb rule is larger for the same number of memories then that of the network when trained with the Storkey rule. This behaviour is expected as the ability to store more memories then the Hebb rule is one of the motivations behind Storkey's research.

Another unsurprising result is the close similarity in the results of the Storkey run and the Storkey 2nd derivation run. The latter is simply a derivation from the Storkey and thus similar results are expected. The derived rule provides approximately the same level of absolute error, and a (very) slightly lower level of recall error then the original.

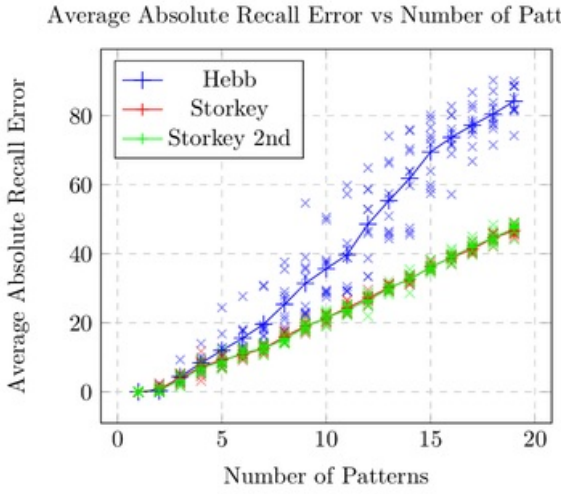


Figure 3.5: Average Absolute Recall Error vs Number of Patterns - This shows how the Average Absolute Recall error changes as the number of patterns increase. Memories of a 100 neurons each were used.

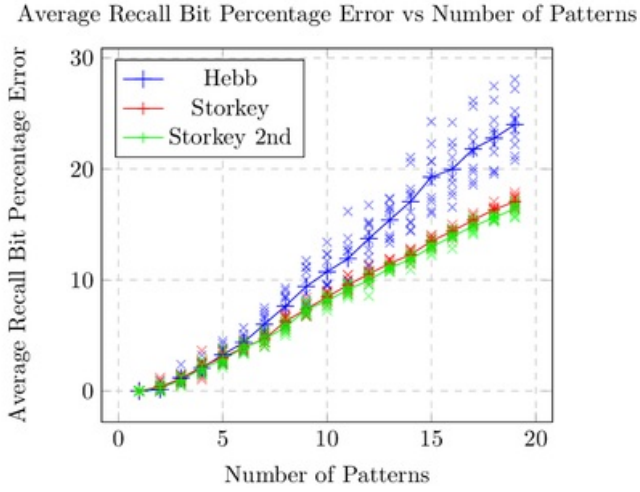


Figure 3.6: Average Recall Bit Percentage Error vs Number of Patterns - This shows how the Average recall bit percentage error changes as the number of patterns increase. This made use of 100 neuron patterns

Despite this, the rule does not provide a distinctive enough result to warrant further tests. From this point tests will not be run on the derived rule as the results of the original Storkey rule are a reliable indicator for its results.

Capacity Testing

Another test relating to storage capacity was to identify the maximum number of patterns a network is able to successfully store based on the number of neurons it contains. This test procedure is as follows:

```
for n in range
    p =0
    do
        p++
        learn the first p patterns cropped to n bits
        for each pattern
            check if it is a fixed point
        if one is not
            max capacity for n neurons is p-1
    while all patterns p are fixed points
```

A pattern is a fixed point if, after setting the states to the pattern, the weighed sum of all other neurons, for each neuron is larger than 0.

This test allows the system to be compared to the theoretical capacity of the learning rules discussed in section 2.4.1.

The results obtained for these tests, when executed for the Hebb rule and the Storkey rule, along with the theoretical capacity of both rules were plotted in figure 3.7.

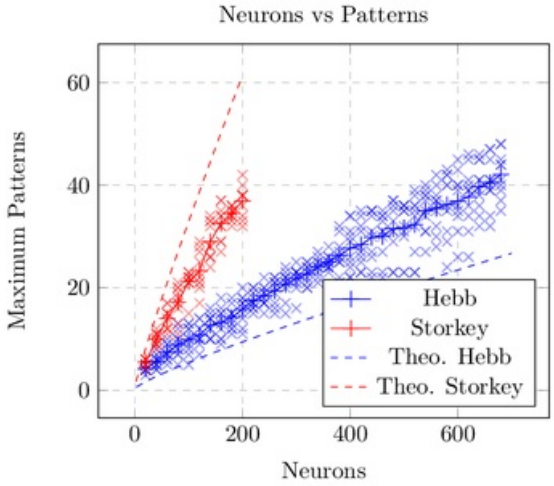


Figure 3.7: Maximum Pattern Capacity vs Number of Neurons - This graph illustrates how the Hopfield Network's capacity varies with the network size for the Hebb rule, the Storkey learning Rule.

By comparing the results obtained to the theoretical values, the results obtained for the Hebb rule's capacity performed somewhat better than the theoretical value, this is not the case with the Storkey rule as this fall short of the theoretical value.

Chapter 4

Stochastic Tests

Now that the capabilities of the network built are known, as are the settings' effect on the network, it is time to look into introducing stochasticity in the network and investigate the results.

4.1 Recall Malfunction

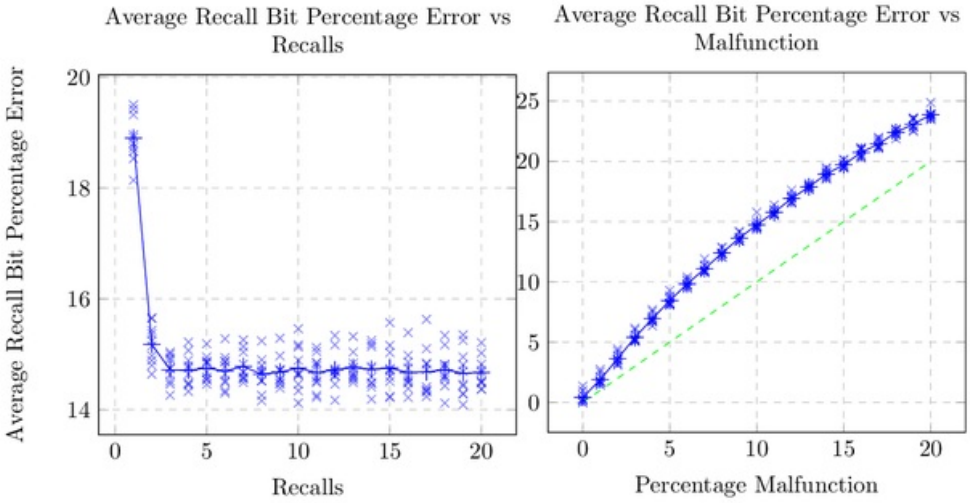
One of the original objectives behind the Hopfield model was that of fault tolerance. The basic function of such a network is built around this and thus in this section this will be the focus of these tests, namely we look into the question of whether a network can recover from a malfunctioning component that is constantly introducing new faults in the system, and how does this alter its behaviour.

These tests will test the principles stated above by introducing a percentage of error per recall (which will be referred to as the malfunction), that is, for n error, every recall the system will randomly select $n\%$ of the recall's result and flip their states. This does not guarantee that all will end up being incorrect, as if the bit from the result was incorrect, then by being flipped it is turned into the correct value.

Considering that the network is able to correct wrong states through the recall procedure, and in this case this very procedure is introducing the errors every time, it is not expected for the network to be able to produce results with less error then the malfunction, as even if a recall is able to correct all the errors of the previous recall, it will then introduce the malfunction error itself. However, whether the network is capable of recovering from the malfunction of previous recalls and only contain the errors from the last malfunction is to be seen.

The following figure, shows the result of two different tests. The first, investigates how

the error value changes based on the number of recalls. While the second shows the error value after stabilisation as the malfunction is increased. This graph also shows the base error per malfunction (i.e. the error introduced in the system due to the malfunction).



(a) Average Recall Bit Percentage Error vs Recalls

(b) Average Recall Bit Percentage Error vs Malfunction

Figure 4.1: This shows how the the recall bit percentage error as the percentage of incorrect bits per recall increases as well as how the error changes based on the number of recalls performed. A network of a 100 neurons storing 2 memories was used for both tests.

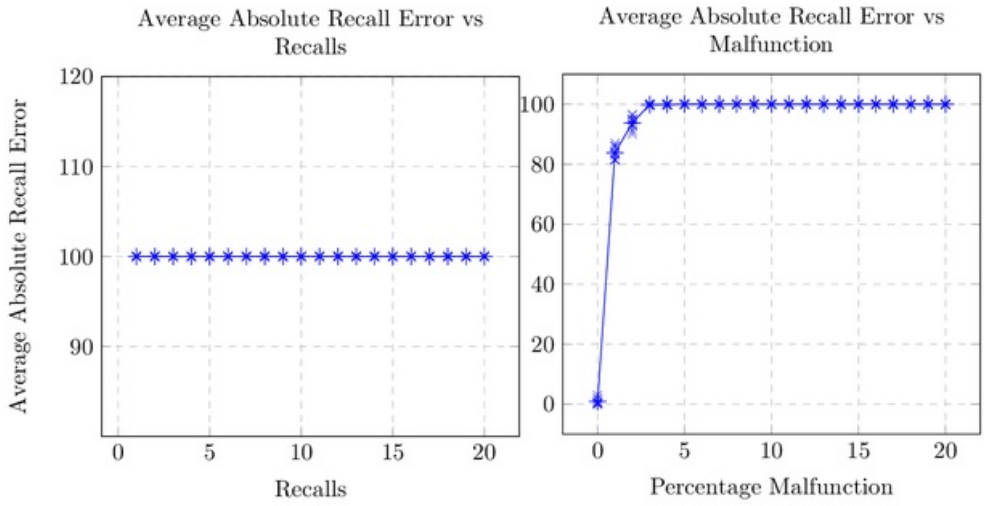
The results of the test shown in figure 4.2a can easily be compared to the results shown in figure 3.4, which consisted of the same investigation of the network under normal circumstances. Clearly the behaviour of the Hopfield network remains the same, in that, the number of incorrect bits starts at a maximum, quickly decreases then stabilizes. The major difference in the two however is the stabilization point.

While in the original test the stabilisation point was that of around 0% error, in this case this is now around 14.5% to 15%. It was expected that the results would contain at least 10% error, however, it would seem that the system is unable to correctly recall another 4.5%-5% due to the malfunction introduced.

Figure 4.2a does not provide much information, except to show that the recall error seen is distributed throughout all tests.

One can look at the previous result and conclude that with the current conditions a 10% recall malfunction results in around 14.5% error. The second test investigates whether the resulting error (after stabilization) is linear to the size of malfunction introduced.

The results seen suggest that this may not be the case as the results presented seem to grow for the first few malfunction, after which the results become rather linear.



(a) Average Absolute Recall Error vs Recalls

(b) Absolute Recall Error vs Malfunction

Figure 4.2: This shows how the the absolute error as the number of recalls increases as well as how the error changes based on the number of recalls performed. The networks were taught 2 memories of a 100 neurons.

4.2 Noise to Weights

Sompolinsky showed that a Hopfield network is able to withstand a small level of noise introduced into the weights via the Hebb rule during the learning phase[16]. At this point, the weights are still being set and thus it is possible that the weights are adapting to the noise as the learning continues. However, what happens if noise is introduced after the learning period has been completed. In this series of tests, noise is introduced to the weights after training, before the recall procedure is performed. During the recall procedure (consisting on a number of recalls) the weights remain constant. The amount of noise added is randomly drawn from a Gaussian distribution.

When applying noise to the weights, there are two possible approaches, one consists of introducing symmetric noise, where the weight symmetry of the network is observed, thus applying the same amount of noise for weight ij as is applied to weight ji(or simply applying noise to only half the weights and updating the rest accordingly). The second approach consists of introducing asymmetric noise to the system, where the symmetric property is ignored and different noise is applied to all weights in the system thereby losing the network's symmetry. Both approaches were implemented.

In whatever case however, the value of the weights for i,i remain 0, thus preserving the property of not having neurons connected to themselves.

In each test the value of the sigma value used to generate the random noise is iterated

over, thus providing a higher value of noise as this increases.

4.2.1 Hebb Network

The following tests were executed on a network using the Hebb rule as its learning rule.

The results for the average number of absolute recall error per sigma and for the average number of error bits per sigma can be seen below:

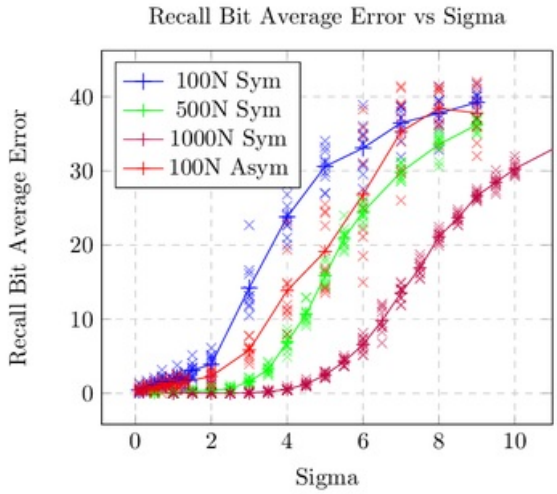


Figure 4.3: Recall Bit Average Error vs Sigma - The results of how the network's Recall bit Average Error varies as the value of sigma increases. This shows the results of symmetrical noise to a network of 100, 500 and 1000 neurons as well as that of asymmetric noise to a 100 neuron network.

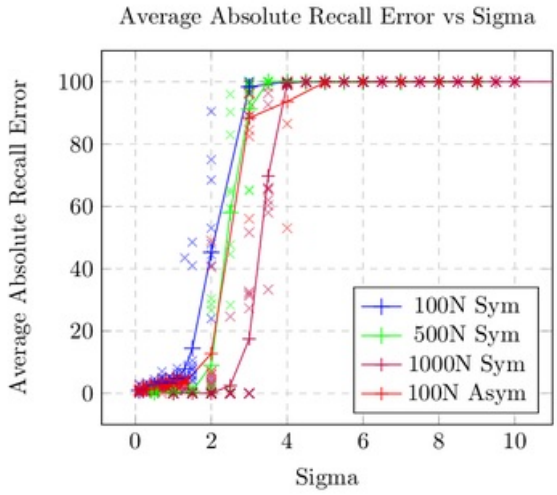


Figure 4.4: Average Absolute Recall Error vs Sigma - The results of how the network Average Absolute Recall Error varies as the value of sigma increases. This shows the results for symmetrical noise on a 100,500 and 1000 neuron network and asymmetric noise on a 100 neuron network.

As expected, the larger the amount of noise introduced, the larger the error. What was unexpected however is the way the error increases. The system is able to handle lower level of noise, however the network's recall ability degrades drastically. This can be seen more easily in the absolute error (seen in figure 4.3), where the network goes from less than 15% incorrect results to a 100% incorrect in the space of 1 sigma.

Also worth nothing is that when comparing the symmetrical results to the asymmetrical results. In both cases the behaviour is similar, however in the case of asymmetrical noise, the behaviour provides a smoother transition once the tolerance level is exceeded. Moving forward only symmetric noise will be taken into account so as to preserve one of Hopfield's fundamental rules.

Whatever the size of the network, the behaviour is consistent, however the amount of noise the network can withstand is not. Based on this it seems clear that the larger the network the more noise it is able to tolerance before escalating.

It is worth noting that a similar effect is seen when making use of different variations of the Hebb rule. In this case the variation of the Hebb rule in use is the one dividing by the number of patterns. Further detail as to the difference between the Hebb rule may be found in the Appendix.

4.2.2 Storkey Network

What follows is the same test used in the previous section, however this time it is performed on a neural network using the Storkey learning rule. The same settings, memories and tests were used, however the amount of noise introduced in this case was far less.

The rationale behind this is that the weights resulting from the Storkey learning rule are considerably smaller than those resulting from the Hebb rule, and thus the relative noise added would be far higher had the same values been used. Since the behaviour of the network is being analysed and not the noise levels in themselves this was seen as an acceptable solution.

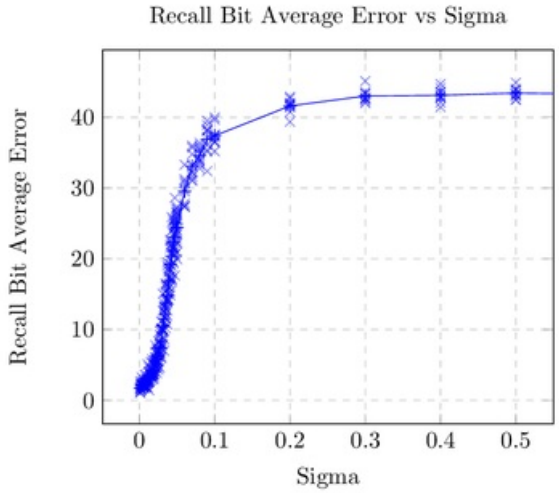


Figure 4.5: Average Recall Bit Percentage Error vs Sigma - The results of how the network Average Recall Bit Percentage Error varies as the value of sigma increases. This shows the result when using the Storkey learning rule while preserving symmetry on a network of a 100 neurons and 3 memories.

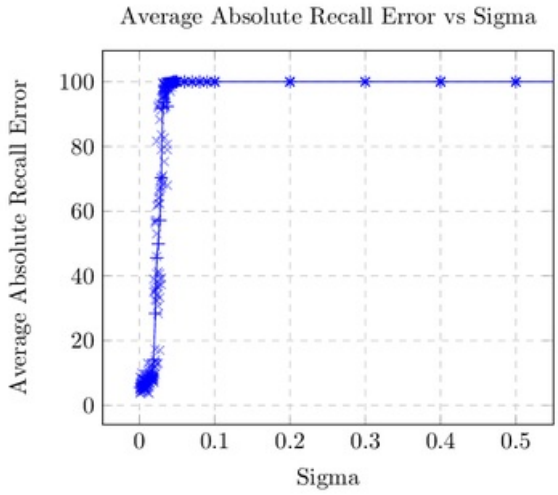


Figure 4.6: Average Absolute Recall Error vs Sigma - The results of how the network Average Absolute Recall Error varies as the amount of symmetrical noise increases. The tests here used Storkey's learning rule for a network storing 3 memories of a 100 Neurons.

The noise tolerance period seen in the Hebb rule can also be seen in the Storkey rule, thus it would seem that the properties discussed for the Hebb rule apply in the case of the Storkey learning rule as well.

4.2.3 Noise Tolerance

The previous tests have shown that the overall behaviour of the network remains the same whatever the network size, however the point at which the drastic error increase presented itself was not the same for all sizes.

Recalls

In the previous tests, the number of recalls was kept constant. The following test investigates how varying the number of recalls effects the tolerance period. The test process followed the idea of teaching the neural network a number of patterns, noise is introduced based on an initial value of sigma, the average recall bit error resulting is then compared to a threshold value, if this is below this value the sigma value is increased, and the process repeated, if the error is above the threshold, then the sigma value is decreased to the previous value (as the current sigma resulted in an error higher then the threshold) and saved as the max for the current number of neurons. The number of neurons is then increased and the process is repeated.

This test was run using patterns of 100 neurons each. The value of 4 was used as the

threshold value as previous tests have shown that this is approximately the point at which the error value starts escalating.

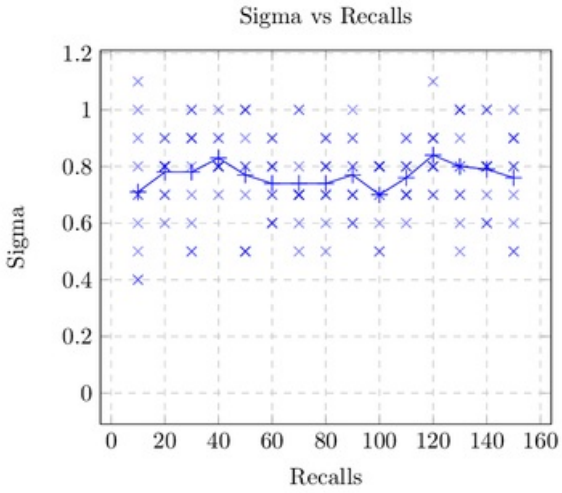


Figure 4.7: Sigma vs Recalls - This shows the maximum value of sigma the network can tolerate before the error value escalates based on the number of recalls performed. This made use of a network with a 100 Neurons and 3 patterns

Based on the results in graph 4.7, one can conclude that the number of recalls do not have an effect on the amount of noise that the Hopfield network is able to tolerate as the maximum sigma tolerated is mostly consistent throughout.

4.2.4 Neurons

The next test takes a similar approach to the previous test, however in this case the variable is the number of neurons in the network.

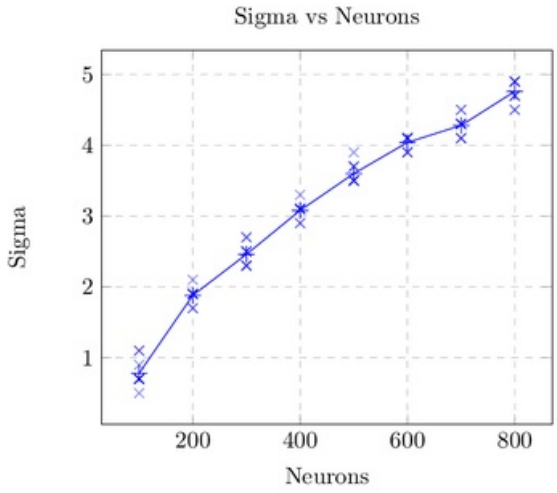


Figure 4.8: Sigma vs Neurons - This shows the maximum value of sigma the network can tolerate as the size of the network changes between 100 and 800. 3 memories were used for this test, with an error threshold of 4.

The graph of Sigma vs Neurons show that the network's noise tolerance grows along with the size of the network.

4.2.5 Memories

It is known that the recall accuracy of a Hopfield network depends upon the number of memories stored, among other factors. This would lead to the belief that the number of memories stored would also play a role in the amount of noise that a network can tolerate.

To prove this, a test following the methods used for figures 4.7 and 4.8 was performed. In this case the number of recalls, and the size of the network were kept constant at 40(the default) and 500 neurons respectively, however the number of memories stored was varied.

Another difference from the previous tests in this section was the variation of the Hebb rule used. So far the rule dividing by the number of patterns was used ((2.3) in section 2.4.1), in this case using this rule was problematic since the number of patterns was varied (thus the relative noise would change with the number of memories making the resultant behaviour unreliable). The Hebb rule dividing by the number of neurons (2.2) was made use of instead. Since the behaviour of all the variations of the Hebb rule is seen to be the same (the results for tests proving this may be seen in the appendix) the behaviour can be reliably studied regardless.

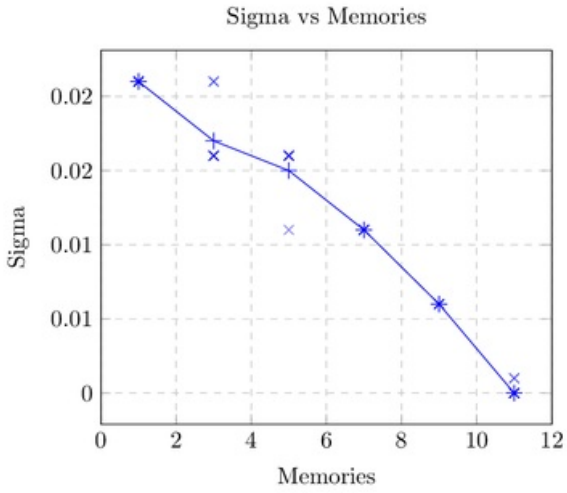


Figure 4.9: Sigma vs Memories - this shows the maximum amount of sigma the network can withstand before exceeding the threshold error based on the number of memories stored. The threshold error was set to 4, and memories of 500 neurons were used.

As expected, figure 4.9 shows that the more memories that are stored the less tolerant the network is to noise. This makes sense as the more memories in the network, the larger the number of minimums, and thus the easier it is for the noise to throw the system into an incorrect minimum.

Thanks to these tests, it can now be said that a Hopfield neural network has a level of tolerance to noise when injected in the weights after the memories have been learned. The amount of noise the network is capable of withstanding before the level of error escalates, is dependent on the learning rule used, the size of the patterns and the number of memories stored. This behaviour is consistent when using networks that use the Hebb learning rule and networks making use of the Storkey learning rule.

Chapter 5

Conclusions

In this dissertation, the area of neural networks was studied, going into some detail on various studies involving noise introduction in various types of neural networks. A neural network was built following the principles discussed including various learning rules and recall methods, which was used to execute tests that provided an insight into Hopfield neural network's behaviour and the effect that various settings have. This was further used to investigate how the behaviour documented is altered with the introduction of malfunction and noise. Various aspects of the resulting behaviour were analysed and discussed.

Following an introductory chapter, a comprehensive look at neural networks was presented. This provided an introduction to the area through a brief overview of the components of a biological neural network, the inspiration behind this whole area. This was followed by an overview of the artificial neural network and how the components in this relate to their biological counterpart. This lead to a discussion of the predominate network models, with a focus on Hopfield network including the most widely used learning rules and recall methods. The final section of this chapter focuses on stochastisity and studies of noise introduction in neural networks, mostly focusing of networks based on a Hopfield network.

Utilising a network built upon the various principles and components discussed in the background chapter a series of tests were performed. First to understand how the behaviour is altered based upon the changes of the various variables in the system, then to study of the capabilities of the network and the capabilities resulting from the different learning rules and recall methods used.

The fourth chapter introduced noise into the network. This looked into two predominate types of test, the first centred around the concept of a malfunctioning recall, where every recall a percentage of the output pattern is flipped from its original output. This has shown

that the overall behaviour of the neural network still remains consistent, despite the increase in error throughout. Tests have also shown that the resulting error after stabilisation is higher then the error introduced via the malfunction, showing that the network is incapable of fully recovering from the previous recall's malfunction.

The second focused on the effects of noise introduction into the weights of a fully trained network. These tests have shown that the Hopfield neural network is able to tolerate lower levels of noise in the weights. At these lower levels, although not resulting in a perfect recall for all patterns, the resulting error is of low enough level, and occurs to only a small to still consider the system useful. The system ceases to be useful beyond the tolerance period as the number of errorous results along with the severity of the errors have a drastic increase to a point where the system is no longer capable of recalling a single correct memory.

This focus of this study was on the behaviour of the network, and thus deducing the way to identify the exact amount of noise a particular network is able to handle was outside the scope of the study, however the amount of noise the Hopfield network can tolerate has been seen to depend on the size of the network as well as the number of memories stored. The learning rule also plays a part in error tolerance, where even the rule variation is seen to make a difference in the level of noise tolerated. In all cases however, the behaviour is observed to remain consistent.

We can now deduce that a Hopfield network may be found useful in situations where the weights might be subjected to low levels of noise after being trained. Coupled with Sompolinsky's discovery that low levels of noise may also be acceptable during training of the weights[16], and its own basic property of error correction seem to suggest that a Hopfield network is capable of functioning in a noisy environment as long as the noise is at a relevantly low volume.

Bibliography

[1] Y.S. Abu-Mostafa and J. St.Jacques. "Information capacity of the Hopfield model". In: *Information Theory, IEEE Transactions on* 31.4 (1985), pp. 461–464. ISSN: 0018-9448. DOI: 10.1109/TIT.1985.1057069.

[2] H. Ackley, E. Hinton, and J. Sejnowski. "A learning algorithm for Boltzmann machines". In: *Cognitive Science* (1985), pp. 147–169.

[3] Pau-Choo Chung and T.F. Krile. "Characteristics of Hebbian-type associative memories having faulty interconnections". In: *Neural Networks, IEEE Transactions on* 3.6 (1992), pp. 969–980. ISSN: 1045-9227. DOI: 10.1109/72.165598.

[4] N. Davey and S.P. Hunt. "The capacity and attractor basins of associative memory models". English. In: *Foundations and Tools for Neural Modeling*. Ed. by Jos Mira and JuanV. Snchez-Andrs. Vol. 1606. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1999, pp. 330–339. ISBN: 978-3-540-66069-9. DOI: 10.1007/BFb0098189. URL: http://dx.doi.org/10.1007/BFb0098189.

[5] S.S. Haykin. *Neural Networks: A Comprehensive Foundation*. International edition. Prentice Hall, 1999. ISBN: 9780139083853. URL: https://books.google.co.uk/books?id=M5abQgAACAAJ.

[6] D.O. Hebb. *The Organization of Behavior: A Neuropsychological Theory*. Taylor & Francis, 2005. ISBN: 9781135631901. URL: https://books.google.co.uk/books?id=ddB4AgAAQBAJ.

[7] Kevin Ho, Chi-sing Leung, and John Sum. "On Weight-Noise-Injection Training". English. In: *Advances in Neuro-Information Processing*. Ed. by Mario Kppen, Nikola Kasabov, and George Coghill. Vol. 5507. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, pp. 919–926. ISBN: 978-3-642-03039-0. DOI: 10.1007/978-3-642-03040-6_112. URL: http://dx.doi.org/10.1007/978-3-642-03040-6_112.

[8] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[9] Xiao Hu. "Storkey Learning Rules for Hopfield Networks". In: (2013).

[10] Anil K Jain, Jianchang Mao, and KM Mohiuddin. "Artificial neural networks: A tutorial". In: *Computer* 29.3 (1996), pp. 31–44.

[11] Kam-Chuen Jim, C.L. Giles, and B.G. Horne. "An analysis of noise in recurrent neural networks: convergence and generalization". In: *Neural Networks, IEEE Transactions on* 7.6 (1996), pp. 1424–1438. ISSN: 1045-9227. DOI: 10.1109/72.548170.

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by simulated annealing". In: *SCIENCE* 220.4598 (1983), pp. 671–680.

[13] WarrenS. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". English. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133. ISSN: 0007-4985. DOI: 10.1007/BF02478259. URL: http://dx.doi.org/10.1007/BF02478259.

[14] R. J. McEliece et al. "The Capacity of the Hopfield Associative Memory". In: *IEEE Trans. Inf. Theor.* 33.4 (July 1987), pp. 461–482. ISSN: 0018-9448. DOI: 10.1109/TIT.1987.1057328. URL: http://dx.doi.org/10.1109/TIT.1987.1057328.

[15] Manoranjan P Singh. "Synchronous Dynamics of a Hopfield Model with Random Asymmetric Interactions". In: *arXiv preprint cond-mat/0307132* (2003).

[16] H. Sompolinsky. "The theory of neural networks: The Hebb rule and beyond". English. In: *Heidelberg Colloquium on Glassy Dynamics*. Ed. by J.L. van Hemmen and I. Morgenstern. Vol. 275. Lecture Notes in Physics. Springer Berlin Heidelberg, 1987, pp. 485–527. ISBN: 978-3-540-17777-7. DOI: 10.1007/BFb0057531. URL: http://dx.doi.org/10.1007/BFb0057531.

[17] Amos Storkey. "Increasing the capacity of a hopfield network without sacrificing functionality". English. In: *Artificial Neural Networks ICANN'97*. Ed. by Wulfram Gerstner et al. Vol. 1327. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1997, pp. 451–456. ISBN: 978-3-540-63631-1. DOI: 10.1007/BFb0020196. URL: http://dx.doi.org/10.1007/BFb0020196.

[18] Lipo Wang. "Noise injection into inputs in sparsely connected Hopfield and winner-take-all neural networks." In: *IEEE transactions on systems, man, and cybernetics Part B: cybernetics* (1997). ISSN: 868-70.

[19] Lipo Wang and Kate Smith. "On chaotic simulated annealing". In: *IEEE Trans. Neural Net* 9 (1998), pp. 716–718.

[20] Jieyu Zhao and J. Shawe-Taylor. "A recurrent network with stochastic weights". In: *Neural Networks, 1996., IEEE International Conference on*. Vol. 2. 1996, 1302–1307 vol.2. DOI: 10.1109/ICNN.1996.549086.

Appendix A

Additional Tests

What follows are some additional tests that were not deemed of enough importance to be included in the main chapters.

A.1 Hebb rule comparison

In section 2.4.1, three variations of the Hebb rules were explored.

- equation (2.1) which does not the divide the weights,
- equation (2.2) which divides by the number of neurons,
- equation (2.3), which divides by the number of patterns.

A.1.1 Error vs Patterns

All three of the aforementioned variations of the Hebb rule are stated to provide the same results, however tests were run on all three in order to verify this fact. What follows are the results of the error vs patterns.

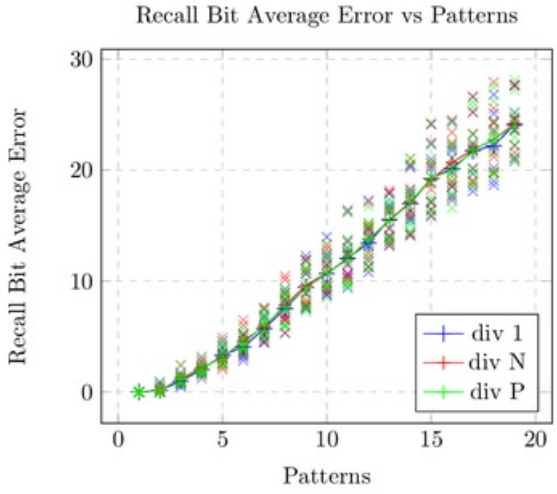


Figure A.1: Recall Bit Average Error vs Patterns - The results of how the network's recall bit average error varies as the number of saved patterns. This shows the results tests of all Hebb rule variations. Patterns of a 100 neurons were used for these tests

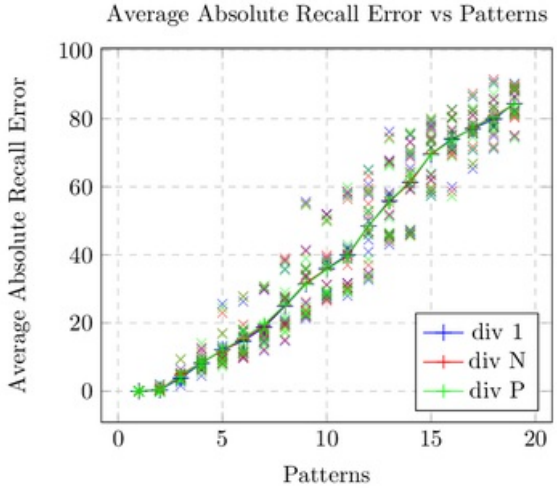


Figure A.2: Average Absolute Recall Error vs Patterns - The results of how the network Average Absolute Recall Error varies as the number of stored patterns increases for the Hebb rule variations. These tests made use of patterns containing a 100 neurons

A.1.2 Error Vs Sigma

Having confirmed that all three variations of the Hebb rule behave the same in normal circumstances, additional tests were performed to study their behaviour when noise is introduced to the weights. Since the variation in the rules result in the weights having the same ratio between them (thus explaining the same behaviour), the actual values vary, thus the amount of sigma that each variation can take is expected to vary, however the overall behaviour should remain consistent.

The following shows the results of all three on the same graph followed by a plot of each by itself, where the behaviour can be easily observed.

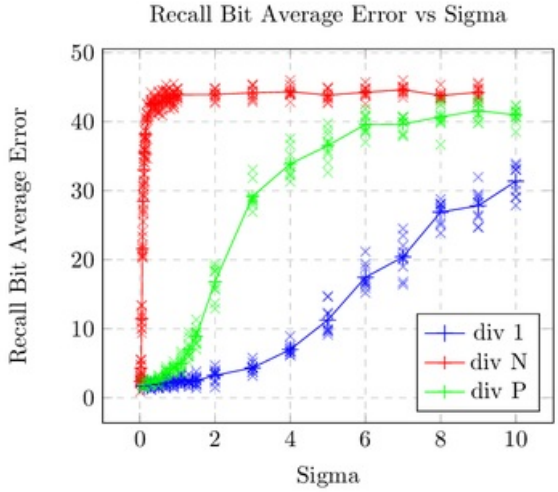


Figure A.3: Recall Bit Average Error vs Sigma - The results of how the network's Recall Bit average Error varies as the value of sigma increases. This shows the results for symmetric tests of all Hebb rule variations. Memories of a 100 neurons were used here.

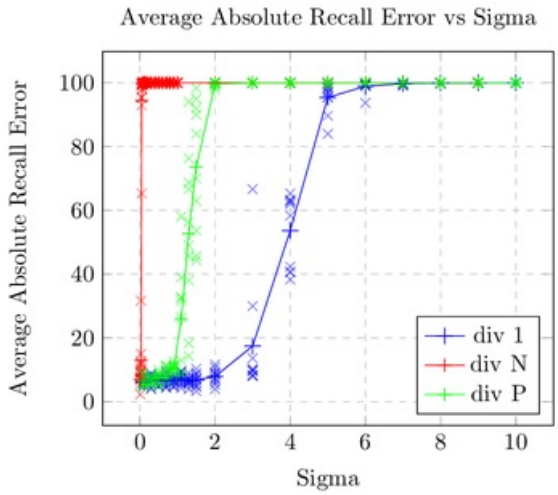


Figure A.4: Average Absolute Recall Error vs Sigma - The results of how the network Average Absolute Recall Error varies as the value of sigma increases for the Hebb rule variations. Patterns of 100 neurons were used for these tests.

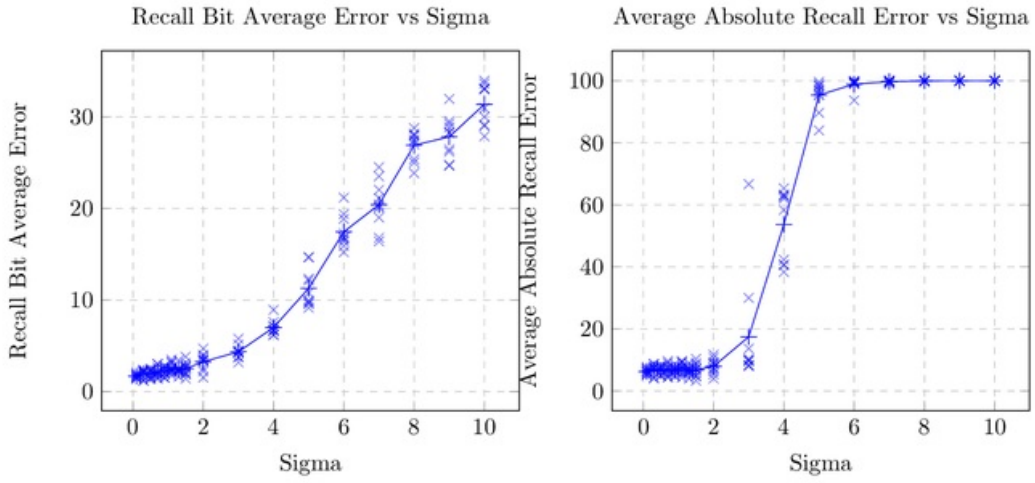


Figure A.5: This shows the behaviour of the original Hebb rule. The network used for this test made use of patterns of a 100 neurons.

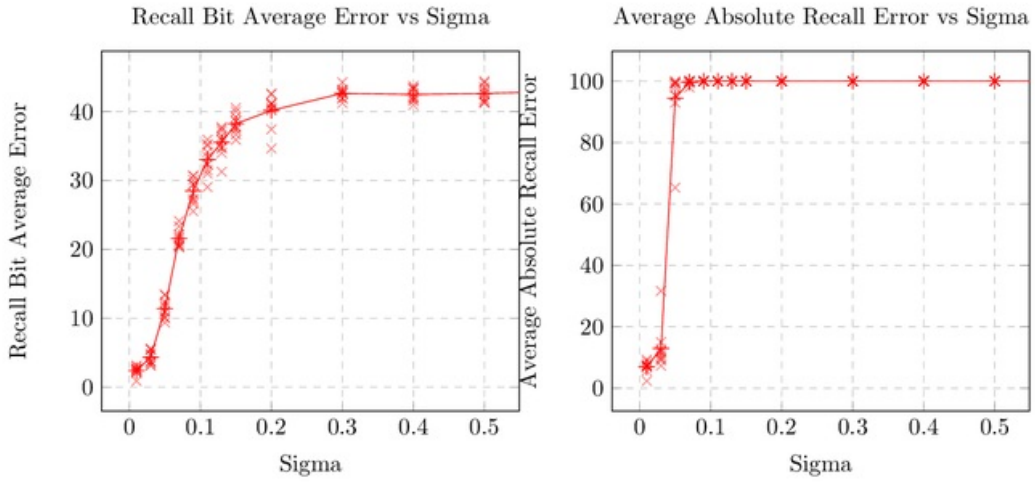


Figure A.6: This shows the behaviour of the Hebb rule dividing by the number of neurons. This made use of a network of a 100 neurons.

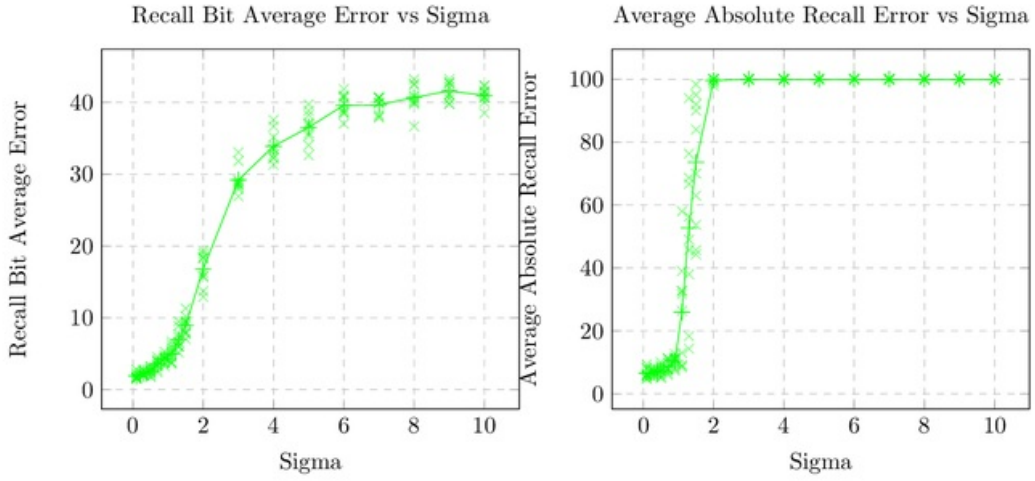


Figure A.7: This shows the behaviour of the Hebb rule dividing by the number of patterns. The size of the network used was that of a 100 neurons

Clearly it seems that the behaviour is the same, however the amount of noise each rule can take is different.

The test that follows investigates the error compared to the relative noise, where the relative noise is obtained by:

$$\frac{\delta_{ij}}{W_{ij}} \tag{A.1}$$

where δ represents the noise introduced to weight ij.

Since relative noise is a derived value, the original test was performed by increasing sigma, however the average relative noise for each case was recorded. This was then plotted vs the error for the three Hebb rules as seen below.

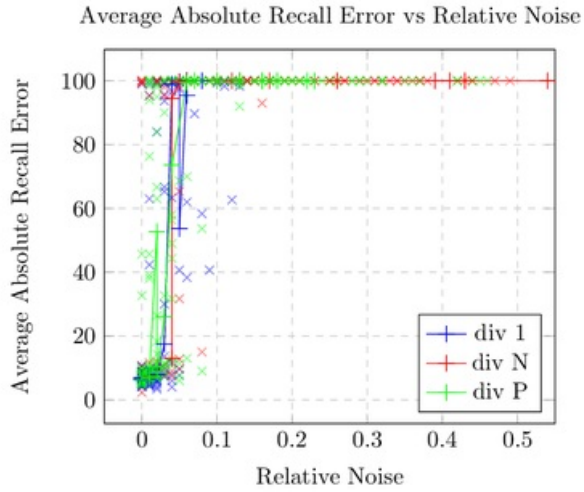


Figure A.8: Average Absolute Recall Error vs Relative Noise - This shows the behaviour of the absolute error as the relative noise is increased for networks making use of all 3 Hebb rule variations.

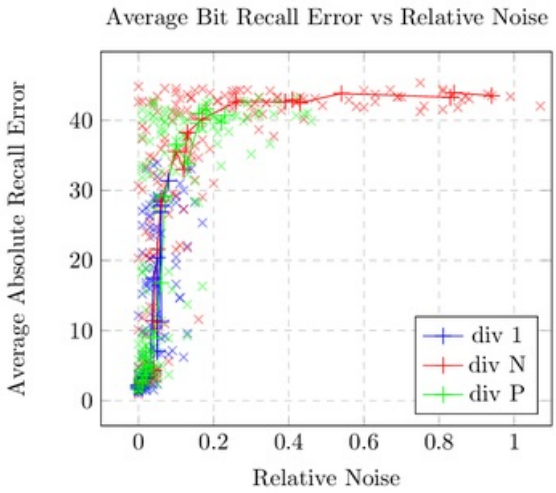


Figure A.9: Average Recall Bit Error vs Relative Noise - This shows the behaviour of the bit recall error as the relative noise increases for a networks making use of the three Hebb rule variations

Appendix B

The Neural Network Model

Chapters 3 and 4, explored the various tests performed in order to investigate the behaviour of a neural network. These were performed through the use of a Hopfield network built as a tool to provide a way to perform such tests.

This section will provide some details with regards to the Hopfield network built.

B.1 Design of the Hopfield Neural Network

The Hopfield Network needed to be built with flexibility in mind, due to the nature of the project the network's functionality needed to be altered based on the test in question, and thus it was required to have a structure that allows for different components (methods/modules/classes) of the system to be altered without requiring many other components to be updated as well, thus allowing for swapping out components with a similar component that provides altered behaviour, while still interacting and making use of the other components whose function requires no change for the particular test. This would provide a way of going back and forth between the original and the updated behaviour, possibly even in the same run. This also makes it possible to save the tests performed so as to be able to go back to them at a later stage. This bring up the importance of an intuitive structure so as to make it easy and fast to change between components that offer different functionality. Due to this it was important to have a high level of cohesion within the classes/modules where functions/methods need to perform specific tasks.

In order to achieve the aforementioned objectives, it was decided to divide the components into separate modules, centred around a Hopfield network core class that offers the functionality of the network.

The Hopfield network class itself needs to keep track of the weights, no additional data needs to be stored as this is passed in as part of the request. This class needs to provide methods to train the network with the given patterns and perform a recall given the initial state, thus having all required methods to perform the basic functions of a Hopfield network.

The various tests are contained within their own module, where each test is provided with a pre-initialised network and any settings required as parameters, so as not to be tied to any one type of network.

Due to the possibility of various learning algorithms, methods for training are contained within their own module. The particular method to be used is then specified when initialising the particular network. This makes it possible to have two different networks using different learning techniques at the same time.

B.1.1 Building Blocks

This section will go through the components of the artefact giving an insight as to how it is comprised.

HopfieldNetwork class: This is the core of the system. Whenever a neural network is to be used an instance of this class is initiated then used as required. By having the Hopfield Network as its own class, it is possible to have multiple instances of Hopfield networks set up and used within the same test. Each instance could be provided with the learning algorithm to use, which is saved within the instance. The instance also keeps track of its weights, and thus it is impossible to change the size of the network after it has been trained without retraining it.

- *init* is the constructor for the HopfieldNetwork class where the name of the learning algorithm to be used (corresponding to the method in the training module) may be provided as a string parameter. If not provided the "*trainhebb*" method is used which uses the Hebb rule.
- the *fromOther* method initializes the network by copying the properties of a provided network.
- *train*, is given a list of patterns which it uses to train the network using the learning algorithm defined on class initialisation. This methods initialises the weights before starting training, and thus any previous memories are lost. The initialising of the weights also specifies the size of the network to be that of the given pattern. From this point forth only patterns of this size may be used unless the network is retrained.
- the *recall* method is provided with the initial state of the network and the percentage of neurons to recall (as well as a dummy object that is not used in the standard method), it would then perform a recall operation on a randomly selected percentage of the neurons and return the resulting states. The *recallSync* method follows the same procedure however performs a synchronous recall, where the original states are used as the starting point for the recall of every neuron.

  The *recallMalfucntion* and *recallMalfucntionSync* methods take the same parameters as the standard recall methods, with the exception of the dummy parameter, where in this case this is replaced by a malfunction parameter. These methods perform a recall like the previous methods, however when done the specified percentage of neurons are randomly selected and have their states flipped.
- The methods *MutateWeights* and *MutateWeightsSymmetrically* are provided with an array whose values are appended to the current weights. In the case of the *MutateWeightsSymmetrically* method, it is made sure that symmetry is observed.
- *isStable* is given a pattern and determines if it is a fixed point or not based on the network's current weights.
- *getWeights* and *getTrainingAlg*, return the current weights and the name of the training algorithm in use respectively.

additionalFunctions: this contains a collection of methods that are used by test methods, but cannot be considered as test methods themselves. These are methods that would also be out of place should they be included in the Hopfield network class as they are not part of the core functionality of the network.

- this module contains three global variables, one storing the mutation percentage, which is used when creating test patterns so as to know how much to mutate

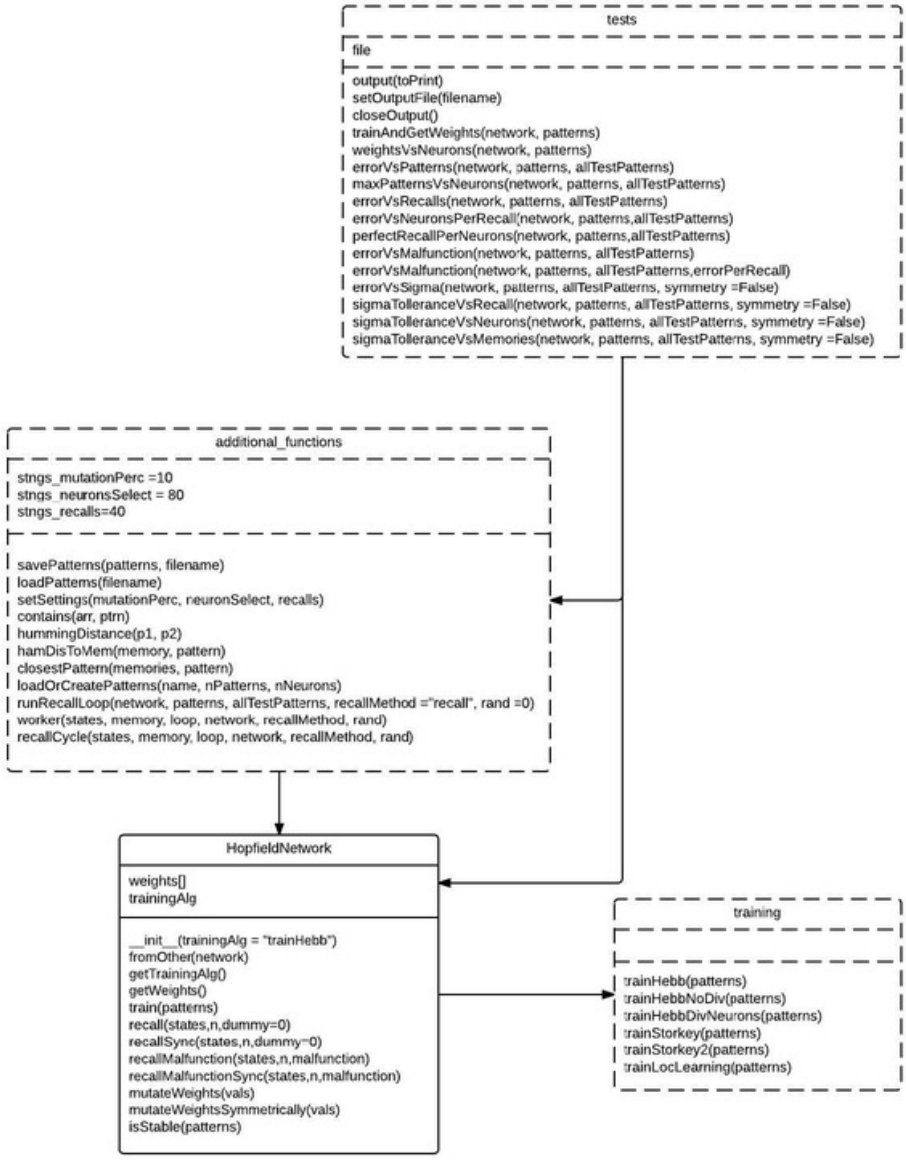


Figure B.1: A uml like diagram showing the major class and modules in the network built. Dashed boxes are used to represent modules, while normal lines represent classes.

from one pattern to the next, another is used to store the number of neurons to select per recall, while the third is used to store the number of recalls to be performed before calculating the error. After the first set of tests described in chapter 3, the default values of 10, 80 and 40 respectively were set to these variables. The method *setSettings* could be used so as to change these to the required value. The variables themselves could also be updated directly should only one require updating.

- the *savePatterns* method is provided with a list of patterns and saves them to a file with the provided filename. The *loadPatterns* method is used to load patterns previously saved with the given filename. This is used by the *loadOrCreatePatterns* method, which given a filename checks of the file's existence, if found this is loaded, otherwise a set of patterns are created and saved to the provided filename. The size of the patterns and the number of patterns are provided as parameters along with the filename.
- the *hammingDistance* method calculates the distance between the two patterns provided. This is used by *hamDistanceToMemory* which calculates the distance of the pattern from both the given memory as well as its inverse and returns the shortest value. Since the network learns both a given pattern as well as its inverse, both have to be considered since matching the inverse is considered as matching the memory itself. This is most prominently used by the aforementioned method when calculating the error. The *closestPattern* method is provided with a set of memories and returns the one that is closest tho the provided pattern.
- the *recallCycle* method is responsible for calling the specified recall method for the number of times specified in the module's settings, then comparing the result and returning the error values. The initial states are provided as a parameter as is the memory to compare its results to. The *worker* method is a method used so as to be able to run the *recallcycle* method in multiple threads.
- the *runRecallLoop* method is a very important method as it is used for most of the tests. This is provided with a HopfieldNetwork object, the recall method(name) to use, the list of memories in the network, a list of memories and a corresponding list of test patterns (ie all patterns for the first memory should result in the first memory for perfect recall) along with a randomness percentage which is by default set to 0. This method will then call the *recallCycle* method for every test provided and telly up the results.

training: this contains various methods that are used to train the network. These methods all expect the patterns that are to be learned as inputs. The methods create a new set of weights based upon the size of the given patterns, these are returned after training.

- *trainhebb*, this method trains the network using the Hebb rule. The implementation used follows the Hebb rule used by Hopfield, that is the one where the product of the neurons is divided by the number of patterns. This particular variation was chosen as a default over the version which is not divided in order to keep the value of the weights from growing too much. This version was preferred to the other alternative as had the other been chosen the weights of a network with a 100 neurons would not be comparable to those of a 1000, which is a requirement for some of the tests performed. These variations of the rule were also implemented as *trainHebbNoDiv* and *trainHebbDivNeurons* respectively.
- the *trainStorkey* is used to train the network based on Storkey's learning algorithm, while *trainStorkey2* is used to train the network using the second derivative of Storkey's rule.

tests: this module contains the test performed.

- the application is able to save the results to a file apart from printing it to console. This is achieved by calling the *setOutputFile* with the file's name, then using the *output* method to print to console as well as to the file. Finally *closeOutput* is used to close the file.
- The remainder of the methods in this module are all different types of tests. For the most part the name of the method indicates the results it provide, and the graph that can eventually plotted with the results. The basic test procedure that is followed is to first set the settings of the *additionalfunctions* when the default are not to be used. The network is trained, then in a loop which changes the control variable the *runRecallLoop* method is called and the result are saved along with the current setting. The training may also be done in the loop in cases where the memories learned depend on the variable under test (like memories learned or network size).

  The logic of the individual tests has been discussed in the experiments chapters and thus they will not be explored here.

B.2 Programming Language and External Packages

Various programming languages were considered, however it was decided that Python is the best fit for what is required.

Python provides a object oriented paradigm while still providing a procedural approach, which allows the network itself to be built as an object while the tests and various other functions required may be built as procedures. Python also has a dynamic type system, which would make it easier to make changes in types with little code modification, this would also open the door to having the prototype use the same variable in different way simply based on which test is run(and how it makes use of the particular variable).

Beyond this, python also offers a wide variety of external packages that could help in the implementation, and efficiency of the artefact.

B.2.1 NumPy

Numpy is a package intended for scientific uses that provides various mathematical functions and operations related to array and list manipulation, which prove very useful a project such as this.

Appendix C

Data

This section contains the data resulting from the tests performed. This data was used to generate the graphs in the main document and the tests in the additional tests section in the appendix.

C.1 PreTests

The following data corresponds the tests run tun explore the network's capabilities, whose results are explored in Chapter 3.

Table C.1: Recall Bit Error vs Neurons per Recall - 100 Neurons Async (Graph 3.2)

NeuronsPerRecall	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1	10.95	11.31	10.54	10.63	10.35	10.73	10.46	10.24	10.62	10.65	10.65
5	2.04	2.35	1.07	1.88	2.13	2.12	2.01	1.61	1.70	2.59	1.95
10	2.17	2.58	1.32	1.63	1.58	2.09	2.60	1.27	1.81	1.85	1.89
15	2.27	1.60	1.54	2.13	1.28	1.68	1.50	1.21	1.65	1.70	1.66
20	2.14	2.77	1.67	1.77	1.69	1.85	2.21	1.10	1.74	2.15	1.91
30	2.36	2.11	1.74	1.83	1.70	1.89	1.81	0.62	1.89	1.80	1.78
40	1.94	2.01	1.62	1.83	1.24	1.95	1.56	1.31	1.29	1.69	1.64
50	1.78	2.67	1.51	2.01	1.33	1.39	1.68	1.23	1.67	1.71	1.70
60	2.00	2.81	1.23	2.35	1.79	2.03	1.72	1.10	1.59	1.81	1.84
70	1.36	2.31	1.09	1.77	1.33	2.25	1.24	1.63	1.48	1.88	1.63
80	1.20	2.51	1.98	1.29	1.49	1.36	2.50	0.79	1.44	1.80	1.64
90	2.13	2.36	1.36	2.12	1.44	1.41	1.42	1.45	1.51	2.25	1.75
100	2.22	2.20	1.50	1.94	1.27	2.41	2.46	1.25	1.69	1.94	1.89

Table C.2: Absolute Recall Error Vs Neurons per Recall - 100 Neurons Async (Graph 3.1)

NeuronsPerRecall	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1	90.00	90.00	90.00	90.00	90.00	90.00	90.00	89.67	90.00	89.67	89.93
5	19.00	21.00	22.00	17.00	18.33	20.00	22.33	19.00	18.67	20.00	19.73
10	7.33	8.67	5.67	6.00	6.00	7.33	9.00	4.67	7.00	7.33	6.90
15	8.67	5.67	5.00	8.00	5.00	5.33	5.67	5.00	6.00	5.67	6.00
20	7.33	9.00	6.00	6.33	5.67	6.33	8.33	4.00	6.67	7.67	6.73
30	8.00	7.33	6.00	7.00	5.67	6.67	7.33	2.67	7.33	6.67	6.47
40	7.00	7.00	5.33	6.67	4.33	6.67	6.00	5.00	5.00	6.67	5.97
50	7.00	8.67	5.33	7.67	4.00	5.00	6.67	4.67	7.00	6.33	6.23
60	7.33	9.00	4.33	9.00	5.33	6.67	6.33	4.00	6.67	6.67	6.53
70	5.67	8.33	3.33	7.00	4.67	7.33	5.00	6.33	5.67	7.00	6.03
80	4.67	8.00	6.00	5.00	5.33	5.00	8.00	3.00	5.00	7.33	5.73
90	7.00	7.33	5.33	7.33	5.00	5.00	5.67	5.00	5.67	8.00	6.13
100	8.00	7.33	5.67	7.33	5.00	7.33	8.33	4.67	7.00	7.00	6.77

Table C.3: Recall Bit Error vs Neurons per Recall - 100 Neurons Sync (Graph 3.2)

NeuronsPerRecall	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1	11.05	11.23	11.14	10.46	9.99	11.09	10.79	10.15	10.19	10.55	10.66
5	2.68	2.82	1.29	2.09	1.26	1.39	1.84	1.55	1.62	2.05	1.86
10	2.22	2.55	1.99	1.45	1.52	1.97	2.36	1.12	1.54	1.93	1.87
15	1.87	2.26	1.65	1.54	0.91	1.22	1.99	1.03	1.53	1.42	1.54
20	2.35	2.44	0.95	1.68	1.33	1.67	2.11	1.27	1.66	2.07	1.75
30	2.57	1.85	1.33	1.60	1.30	1.46	2.34	1.01	1.67	2.06	1.72
40	1.83	2.39	0.87	1.96	1.59	1.50	1.44	0.87	1.53	1.73	1.57
50	1.94	1.90	1.41	1.76	1.59	1.41	1.68	1.45	1.31	1.58	1.60
60	1.57	2.30	1.27	1.35	1.09	1.79	2.01	1.19	1.69	2.14	1.64
70	1.87	1.82	1.50	1.42	1.51	1.69	1.69	1.03	1.36	2.07	1.60
80	1.32	1.60	0.96	1.40	1.51	1.15	1.78	1.19	1.21	1.58	1.37
90	1.64	2.02	1.41	1.48	1.26	1.42	1.37	1.19	1.61	1.56	1.50
100	1.53	2.15	1.05	1.44	1.19	1.25	1.90	1.11	1.35	1.73	1.47

Table C.4: Absolute Recall Error Vs Neurons per Recall - 100 Neurons Async (Graph 3.1)

NeuronsPerRecall	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1	89.67	89.33	90.00	90.00	89.60	90.00	90.00	90.00	89.33	89.67	89.76
5	21.33	20.00	19.00	21.67	19.00	20.67	18.67	18.67	18.67	18.33	19.60
10	9.00	8.33	6.67	5.33	5.33	7.00	8.33	5.00	6.33	7.33	6.87
15	7.00	7.67	6.00	6.33	3.67	4.33	7.67	4.00	6.67	5.67	5.90
20	9.00	8.33	3.33	6.67	5.00	6.33	8.33	4.67	7.00	7.67	6.63
30	10.00	6.33	5.00	6.33	5.00	5.33	9.00	4.33	7.33	7.67	6.63
40	7.33	9.00	3.67	7.67	5.33	6.00	5.67	3.67	6.00	6.67	6.10
50	7.67	6.67	5.00	7.00	6.33	5.33	6.00	5.67	5.67	6.33	6.17
60	6.33	8.33	5.33	5.33	4.33	6.67	7.67	5.00	7.00	8.33	6.43
70	6.67	6.33	5.67	5.67	6.00	6.33	6.67	4.33	6.00	8.00	6.17
80	5.33	6.00	4.00	5.33	6.00	4.33	7.33	5.00	5.33	6.33	5.50
90	6.33	7.00	4.67	5.67	5.00	5.33	5.67	5.00	6.67	6.33	5.77
100	5.67	7.33	4.00	5.67	4.67	4.67	7.67	4.67	6.00	6.33	5.67

Table C.5: Recall Bit Error vs Recalls - 100 Neurons x 2 Memories Async (Graph 3.4)

Recalls	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1	11.71	11.43	11.74	11.79	11.86	11.80	11.92	11.63	11.61	11.96	11.75
3	2.54	2.13	2.42	2.40	2.34	2.35	2.43	2.48	2.33	2.40	2.38
5	0.53	0.68	0.43	0.53	0.46	0.50	0.58	0.50	0.45	0.47	0.51
7	0.07	0.35	0.11	0.08	0.09	0.06	0.32	0.06	0.07	0.04	0.13
9	0.01	0.01	0.03	0.00	0.01	0.01	0.40	0.02	0.01	0.03	0.05
11	0.00	0.25	0.00	0.01	0.01	0.00	0.21	0.01	0.01	0.01	0.05
13	0.00	0.25	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.01	0.03
15	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03
17	0.00	0.25	0.00	0.00	0.00	0.00	0.20	0.00	0.00	0.00	0.05
19	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.04
20	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.04
30	0.00	0.25	0.00	0.00	0.00	0.00	0.20	0.00	0.00	0.00	0.05
40	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03
50	0.00	0.25	0.00	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.07
60	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.00	0.00	0.00	0.02
70	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.04
80	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.00	0.00	0.00	0.02
90	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.04
100	0.00	0.25	0.00	0.00	0.00	0.00	0.20	0.00	0.00	0.00	0.05

Table C.6: Absolute Recall Error Vs Recalls - 100 Neurons x 2 Memories Async (Graph 3.3)

Recalls	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1	90.00	90.00	90.00	89.50	89.50	90.00	90.00	90.00	90.00	89.50	89.85
3	82.00	79.00	82.50	83.50	78.50	83.50	82.00	80.00	84.50	83.50	81.90
5	40.00	34.00	37.00	38.50	35.00	39.00	30.50	33.50	36.00	38.50	36.20
7	7.00	9.00	10.50	7.50	8.50	5.50	12.50	6.00	6.50	4.00	7.70
9	1.50	1.00	3.50	0.00	1.50	1.50	1.50	2.00	1.50	3.50	1.75
11	0.00	0.50	0.00	1.50	0.50	0.00	1.50	0.50	0.50	0.50	0.55
13	0.00	0.50	0.00	0.50	0.50	0.00	0.00	0.00	0.00	0.50	0.20
15	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05
17	0.00	0.50	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.10
19	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.10
20	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.10
30	0.00	0.50	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.10
40	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05
50	0.00	0.50	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.15
60	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.05
70	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.10
80	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.05
90	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.10
100	0.00	0.50	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.10

Table C.7: Recall Bit Error vs Recalls - 100 Neurons x 2 Memories Sync (Graph 3.4)

Recalls	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1	12.12	11.37	11.87	12.07	12.43	11.92	11.54	12.00	11.63	12.01	11.90
3	2.37	2.33	2.40	2.40	2.50	2.34	2.40	2.61	11.64	11.78	4.28
5	0.40	0.43	0.42	0.50	0.47	0.48	0.83	0.44	2.25	2.30	0.85
7	0.09	0.35	0.10	0.10	0.12	0.12	0.27	0.30	0.78	0.56	0.28
9	0.01	0.01	0.01	0.01	0.02	0.03	0.21	0.03	0.08	0.08	0.05
11	0.01	0.25	0.01	0.00	0.00	0.01	0.01	0.22	0.01	0.23	0.08
13	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.00	0.00	0.01	0.02
15	0.00	0.25	0.00	0.00	0.00	0.00	0.20	0.22	0.00	0.00	0.07
17	0.00	0.25	0.00	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.07
19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03
30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.22	0.00	0.00	0.02
40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	0.00	0.00	0.00	0.00	0.49	0.00	0.00	0.00	0.00	0.21	0.07
60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
70	0.24	0.25	0.00	0.00	0.00	0.00	0.20	0.00	0.00	0.00	0.07
80	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.00	0.00	0.00	0.04
90	0.00	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03
100	0.00	0.00	0.00	0.00	0.24	0.00	0.20	0.00	0.24	0.00	0.07

Table C.8: Absolute Recall Error Vs Recalls - 100 Neurons x 2 memories Sync (Graph 3.3)

Recalls	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1	90.00	90.00	89.50	90.00	90.00	90.00	90.00	90.00	89.50	89.50	89.85
3	81.50	82.00	80.50	84.00	81.50	82.00	80.50	79.50	90.00	90.00	83.15
5	31.00	34.00	30.50	36.00	34.50	36.00	34.00	32.50	78.50	80.00	42.70
7	8.00	10.00	10.00	9.00	12.00	11.00	7.00	7.50	39.50	39.50	15.35
9	1.50	1.50	1.00	0.50	2.00	3.00	1.00	2.50	8.00	7.50	2.85
11	0.50	0.50	0.50	0.00	0.00	0.50	1.00	0.50	0.50	3.00	0.70
13	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.50	0.10
15	0.00	0.50	0.00	0.00	0.00	0.00	0.50	0.50	0.00	0.00	0.15
17	0.00	0.50	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.15
19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
20	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05
30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.05
40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.50	0.15
60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
70	0.50	0.50	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.15
80	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.10
90	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05
100	0.00	0.00	0.00	0.00	0.50	0.00	0.50	0.00	0.50	0.00	0.15

Table C.9: Recall Bit Error vs Recalls - 1000 Neurons x 3 Memories Async (Graph 3.4)

Recalls	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1	11.83	11.76	11.83	11.74	11.79	11.77	11.78	11.72	11.77	11.74	11.77
5	0.48	0.46	0.48	0.46	0.49	0.47	0.48	0.49	0.48	0.51	0.48
10	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table C.10: Absolute Recall Error Vs Recalls - 1000 Neurons x3 memories Async (Graph 3.3)

Recalls	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00
5	88.33	88.00	87.33	85.33	88.00	88.33	87.67	87.00	88.33	86.67	87.50
10	7.00	8.00	10.67	7.67	5.00	7.33	9.00	11.67	9.00	10.00	8.53
20	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
70	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
80	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
90	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table C.11: Recall Bit Error vs Patterns - using Hebb Rule - 100Neurons(Graph 3.6)

Patterns	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.48	0.00	0.00	0.00	0.92	0.14
3	0.94	1.05	0.72	0.90	1.21	2.38	1.07	1.28	0.79	1.49	1.18
4	1.82	1.51	1.59	2.12	2.02	3.21	2.25	2.30	1.79	1.98	2.06
5	2.60	2.33	4.25	2.63	2.90	4.51	3.74	2.85	3.63	3.15	3.26
6	4.17	3.73	5.69	3.76	3.55	4.85	4.30	4.50	4.01	5.13	4.37
7	4.83	5.22	7.49	5.34	4.96	6.83	6.56	7.16	6.07	5.63	6.01
8	7.44	6.05	9.05	7.51	5.98	8.61	8.39	9.28	8.41	5.80	7.65
9	7.27	8.10	11.17	8.77	7.50	11.15	9.97	11.73	10.75	7.65	9.41
10	9.46	9.65	13.40	9.57	8.49	11.71	10.86	12.31	12.33	9.50	10.73
11	9.68	9.98	16.17	11.91	9.89	12.60	13.03	13.74	12.71	9.54	11.93
12	11.20	11.37	16.74	15.09	12.05	14.97	13.62	15.24	14.25	12.72	13.73
13	14.01	13.32	17.35	17.27	13.73	16.57	13.74	16.20	16.46	15.39	15.40
14	14.40	16.10	20.60	18.18	14.48	17.81	15.24	17.58	21.05	15.14	17.06
15	16.21	22.26	20.67	20.71	16.00	19.38	17.42	18.83	24.26	17.10	19.28
16	17.96	22.40	21.05	21.94	17.60	19.21	18.77	20.04	24.20	16.56	19.97
17	18.56	23.27	22.50	26.14	19.76	21.66	20.07	21.52	25.66	18.87	21.80
18	19.54	24.40	23.58	27.47	19.80	23.21	21.52	22.20	26.20	19.82	22.77
19	22.20	24.54	25.21	27.22	21.15	24.37	22.93	23.56	28.07	20.79	24.00

Table C.12: Absolute Recall Error Vs Patterns - using Hebb Rule - 100 neurons(Graph 3.6)

Patterns	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	2.00	0.30
3	3.33	3.33	3.00	3.67	4.67	9.33	3.33	5.00	3.67	4.67	4.40
4	7.50	5.50	7.00	6.50	8.50	14.00	10.00	9.25	7.25	8.25	8.38
5	9.00	7.20	24.40	8.20	8.60	16.80	12.60	10.20	12.80	10.40	12.02
6	13.17	11.00	27.67	11.67	11.17	17.67	15.17	17.67	12.67	18.17	15.60
7	14.29	16.00	30.57	16.43	13.57	19.43	18.86	30.86	18.14	18.29	19.64
8	21.00	18.75	35.50	21.88	17.25	26.75	29.00	37.62	27.88	17.62	25.33
9	22.44	23.00	39.00	24.78	23.00	35.56	36.11	54.67	33.33	22.33	31.42
10	28.40	28.90	49.80	29.50	29.10	37.00	38.20	50.60	37.60	27.50	35.66
11	29.45	29.18	57.27	38.91	30.45	41.36	43.00	59.73	38.91	29.09	39.74
12	33.42	33.42	58.58	58.75	45.92	52.83	52.92	64.83	46.58	38.67	48.59
13	46.54	45.38	66.15	74.08	46.08	56.00	50.92	66.77	57.38	44.38	55.37
14	46.29	64.00	70.43	75.93	47.43	59.79	63.71	69.71	75.64	45.50	61.84
15	58.80	74.13	73.13	79.07	59.87	63.93	71.67	76.60	80.13	57.47	69.48
16	71.31	76.12	75.00	82.69	68.94	71.81	73.38	80.19	80.12	57.19	73.68
17	71.00	78.41	77.47	86.76	75.88	74.59	76.12	82.18	83.18	66.76	77.24
18	76.22	78.89	77.28	90.22	77.72	82.39	79.61	83.78	87.11	71.50	80.47
19	81.74	81.95	81.84	90.00	81.26	88.42	85.05	88.58	88.68	74.21	84.17

Table C.13: Recall Bit Error vs Patterns - using Storkey Rule - 100 Neurons (Graph 3.6)

Patterns	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.48	0.23	0.47	0.00	0.24	0.96	0.24	0.24	1.18	0.00	0.40
3	0.75	0.59	0.82	0.97	1.15	1.49	0.90	1.38	1.36	1.36	1.08
4	2.02	1.82	1.04	2.45	1.27	3.62	2.44	2.56	2.37	2.60	2.22
5	3.00	2.49	2.42	3.16	2.52	3.61	3.85	2.71	2.94	3.43	3.01
6	2.99	3.90	3.71	3.57	4.59	4.22	4.02	3.80	4.08	4.27	3.92
7	4.53	4.69	4.36	4.98	4.38	5.14	5.38	4.05	4.94	4.43	4.69
8	5.46	5.85	6.74	5.58	6.17	6.99	6.61	6.72	6.52	5.92	6.26
9	7.44	6.71	7.82	6.79	7.83	7.14	6.86	7.80	8.39	6.88	7.37
10	8.33	7.76	9.02	8.23	8.23	8.94	8.76	8.77	8.50	8.02	8.46
11	10.10	9.14	9.06	8.71	8.93	10.46	10.03	10.43	9.24	9.40	9.55
12	10.66	9.85	10.72	10.81	10.33	10.72	11.36	10.54	10.28	10.11	10.54
13	11.77	11.17	11.89	11.86	10.98	11.76	11.16	11.29	11.55	11.32	11.48
14	12.50	11.46	12.27	12.73	11.98	12.82	12.46	11.94	12.05	12.63	12.28
15	14.13	13.08	13.64	13.01	13.54	13.92	14.05	12.78	13.67	13.18	13.50
16	14.98	14.39	14.63	14.48	14.07	14.97	14.41	13.93	14.38	13.75	14.40
17	15.91	14.99	15.46	16.01	14.86	15.79	15.39	15.27	15.02	15.17	15.39
18	17.25	15.83	17.06	16.32	16.28	16.57	16.44	15.89	16.33	15.39	16.34
19	17.16	16.47	17.88	17.34	17.44	17.07	17.61	16.34	16.76	16.57	17.06

Table C.14: Absolute Recall Error Vs Patterns - using Storkey Rule - 100 Neurons (Graph 3.6)

Patterns	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	1.00	0.50	1.00	0.00	0.50	2.00	0.50	0.50	2.50	0.00	0.85
3	2.67	2.33	3.00	4.00	4.33	5.33	3.00	5.00	5.33	5.00	4.00
4	6.50	5.25	3.25	7.50	4.50	11.75	8.50	9.00	8.50	8.25	7.30
5	8.80	7.20	7.00	10.60	8.20	11.40	11.40	8.40	8.80	10.60	9.24
6	9.17	10.67	9.83	9.83	12.67	12.17	10.83	10.17	11.17	12.00	10.85
7	12.57	13.00	11.57	13.14	11.71	13.43	13.86	10.86	13.43	12.00	12.56
8	14.25	14.37	17.25	14.50	16.25	18.00	16.25	17.12	16.25	15.62	15.99
9	19.44	16.89	19.89	18.56	19.89	18.44	17.67	19.89	21.56	17.78	19.00
10	20.40	19.70	22.60	22.00	20.60	22.60	22.30	22.30	21.60	20.30	21.44
11	25.00	22.64	23.36	23.18	22.55	26.45	25.45	26.64	24.27	24.00	24.35
12	27.25	24.92	27.83	27.42	27.33	27.75	29.58	26.92	26.67	26.17	27.18
13	30.08	29.69	31.08	31.31	29.15	31.31	29.92	29.46	30.92	29.08	30.20
14	32.64	31.07	32.93	33.14	31.36	33.29	33.86	31.93	31.86	33.71	32.58
15	37.47	35.33	36.20	35.00	36.20	35.93	38.20	35.07	36.73	35.47	36.16
16	39.44	39.81	39.06	39.56	38.44	38.69	39.00	39.25	38.94	37.00	38.92
17	41.76	40.35	40.88	43.53	40.06	41.71	42.24	41.18	41.00	40.53	41.32
18	45.72	43.44	45.50	46.11	44.33	46.00	44.11	44.67	44.17	42.11	44.62
19	46.00	46.42	47.47	49.11	47.63	47.05	47.16	45.47	45.89	44.63	46.68

Table C.15: Recall Bit Error vs Patterns - using Storkey 2nd Rule - 100 Neurons (Graph 3.6)

Patterns	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.23	0.00	0.00	0.00	0.72	0.24	0.24	0.47	0.69	0.26
3	0.44	1.04	0.53	0.74	1.37	1.33	0.73	1.27	0.99	1.29	0.97
4	1.89	1.71	1.49	1.96	1.52	2.76	2.33	2.69	1.66	2.30	2.03
5	2.55	2.42	2.08	2.30	2.71	3.82	3.64	2.92	2.81	2.80	2.81
6	3.50	4.11	4.15	4.26	3.37	4.30	3.79	3.49	3.36	4.52	3.89
7	4.42	5.10	5.53	4.47	4.03	4.75	5.22	3.97	4.77	4.33	4.66
8	5.69	6.22	5.93	6.01	5.24	6.49	6.16	5.34	5.00	5.44	5.75
9	7.85	6.81	8.08	7.37	6.94	7.93	7.39	6.76	7.07	7.12	7.33
10	8.03	7.24	8.11	8.45	7.51	8.72	8.50	7.72	8.46	8.42	8.12
11	9.55	8.67	9.63	8.94	8.80	9.64	9.71	9.33	8.39	8.63	9.13
12	10.26	8.56	10.82	10.38	9.60	10.49	10.21	9.52	10.16	9.59	9.96
13	11.19	11.04	11.35	11.32	10.69	11.32	11.23	10.80	10.81	11.08	11.08
14	12.06	11.43	12.18	11.72	10.93	12.55	11.85	11.63	12.12	12.22	11.87
15	13.77	12.99	12.95	12.92	12.42	13.23	12.71	12.21	13.47	13.15	12.98
16	14.46	13.33	13.87	14.06	13.94	14.63	14.25	12.82	13.83	13.97	13.92
17	15.72	14.57	15.18	14.69	14.28	15.39	14.34	14.09	15.19	14.99	14.84
18	15.53	15.45	15.96	15.83	15.35	16.02	15.47	15.21	15.73	15.52	15.61
19	16.94	15.86	16.85	16.11	16.52	16.61	17.12	15.65	16.90	15.63	16.42

Table C.16: Absolute Recall Error Vs Patterns - using Storkey 2nd Rule - 100 Neurons (Graph 3.6)

Patterns	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
2	0.	0.50	0.	0.	0.	1.5	0.50	0.50	1.0	1.5	0.55
3	1.7	3.3	1.7	3.0	4.7	5.3	2.0	4.0	3.3	5.0	3.4
4	7.0	5.8	5.3	6.0	6.3	9.3	7.8	9.0	6.0	7.3	7.0
5	8.2	7.4	6.6	7.4	8.8	12	11	8.8	8.8	9.0	8.9
6	10	12	12	12	10	13	11	9.7	10	13	11
7	12	14	14	13	11	14	13	11	13	13	13
8	14	16	15	16	14	17	15	15	14	14	15
9	21	17	20	19	18	20	19	17	19	18	19
10	20	19	21	23	19	24	22	21	23	22	21
11	24	22	25	24	23	25	25	25	23	23	24
12	27	22	29	28	25	28	27	25	29	25	26
13	30	29	32	31	28	31	30	29	31	30	30
14	32	32	34	33	29	36	34	32	34	32	33
15	36	36	36	38	34	36	35	35	38	35	36
16	39	38	39	41	39	41	40	37	40	38	39
17	42	41	42	45	40	44	41	41	43	42	42
18	42	44	46	48	44	47	43	44	45	43	45
19	47	47	48	49	47	49	48	47	48	44	47

Table C.17: Patterns vs Neurons - Hebb rule part 1 (Graph 3.7)

Neurons	Patterns1	Patterns2	Patterns3	Patterns4	Patterns5	Patterns6	Patterns7	Patterns8	Patterns9	Patterns10	PatternsAvg
20	5	4	4	4	5	3	4	4	4	5	4.20
40	6	5	5	5	7	6	8	4	6	4	5.60
60	7	6	7	9	5	9	10	6	8	7	7.40
80	7	8	7	13	5	9	12	10	8	10	8.90
100	9	8	7	13	5	12	13	10	10	12	9.90
120	10	10	9	12	7	12	14	10	10	14	10.80
140	10	12	14	15	10	12	14	10	16	14	12.70
160	13	15	9	15	10	12	14	10	18	16	13.20
180	11	15	14	17	12	12	15	13	18	15	14.20
200	11	17	18	17	14	12	15	16	18	20	15.80
220	17	17	20	17	16	15	18	16	20	19	17.50
240	17	17	20	17	15	19	22	18	20	19	18.40
260	21	21	20	17	15	22	21	16	19	22	19.40
280	21	23	21	17	15	24	22	22	20	22	20.70

Table C.18: Patterns vs Neurons - Hebb rule part 2 (Graph 3.7)

Neurons	Patterns1	Patterns2	Patterns3	Patterns4	Patterns5	Patterns6	Patterns7	Patterns8	Patterns9	Patterns10	PatternsAvg
300	23	23	23	23	16	25	22	22	20	21	21.80
320	25	23	24	25	20	23	22	24	20	21	22.70
340	25	21	24	23	24	26	27	25	22	24	24.10
360	25	23	24	25	24	26	28	24	25	26	25.00
380	31	21	22	25	27	26	28	25	32	26	26.30
400	31	22	24	25	29	34	30	26	34	22	27.70
420	34	23	25	26	28	35	31	27	32	23	28.40
440	35	23	30	26	34	35	35	27	31	22	29.80
460	31	23	35	30	34	34	33	27	31	22	30.00
480	33	23	37	31	35	36	37	29	31	23	31.50
500	36	23	32	30	35	36	36	35	31	23	31.70
520	38	23	39	27	35	36	36	34	32	23	32.30
540	38	38	39	31	34	37	40	35	31	27	35.00
560	36	38	40	30	35	37	42	37	32	28	35.50
580	36	40	42	31	35	40	43	42	24	29	36.20
600	36	40	43	34	35	39	45	40	26	31	36.90
620	36	39	46	34	41	39	45	40	26	31	37.70
640	45	38	45	34	41	39	46	41	36	32	39.70
660	45	39	47	34	40	39	47	41	41	32	40.50
680	45	44	48	35	44	40	48	41	39	37	42.10

Table C.19: Patterns vs Neurons - Storkey rule (Graph 3.7)

Neurons	Patterns1	Patterns2	Patterns3	Patterns4	Patterns5	Patterns6	Patterns7	Patterns8	Patterns9	Patterns10	PatternsAvg
20	4	6	6	6	6	5	5	6	5	6	5.50
40	9	9	11	10	12	11	9	11	8	10	10.00
60	15	15	12	15	17	11	12	14	17	12	14.00
80	20	16	11	17	19	13	20	18	21	17	17.20
100	22	15	22	23	21	22	21	22	23	21	21.20
120	28	17	27	21	25	20	29	21	22	23	23.30
140	33	26	26	26	28	33	28	24	34	32	29.00
160	37	29	37	34	32	34	29	32	35	27	32.60
180	36	34	36	33	34	36	34	32	35	35	34.50
200	38	34	40	33	38	38	36	32	42	38	36.90

C.2 Stochastic Tests

What follows are the results obtained from the tests used to explore the recall malfunction and noisy weights. These results were explored in Chapter 4.

Table C.20: Recall Bit Error vs Recalls - 100 Neurons x 2 Memories, 10% Malfunction (Graph 4.1b)

Recalls	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1	19.32	18.14	18.91	18.80	19.50	18.53	18.66	18.96	18.73	19.42	18.90
2	15.65	14.64	15.15	14.80	15.65	14.97	14.89	15.43	15.37	15.24	15.18
3	15.00	14.26	14.90	14.74	15.04	14.64	14.46	14.94	14.75	14.51	14.72
4	14.79	14.75	14.89	14.64	15.07	14.48	14.47	15.22	14.44	14.33	14.71
5	14.84	14.46	14.85	14.54	15.19	14.95	14.67	14.88	14.59	14.54	14.75
6	14.78	14.30	14.86	14.52	15.28	14.88	14.34	14.94	14.64	14.49	14.70
7	15.04	14.67	14.68	14.82	15.26	14.57	14.49	15.14	14.55	14.59	14.78
8	14.90	14.52	14.52	14.66	15.24	14.67	14.16	14.80	14.71	14.26	14.64
9	14.54	14.46	14.55	14.49	15.29	14.70	14.39	14.72	14.72	14.97	14.68
10	15.46	14.34	14.66	14.62	15.17	14.64	14.12	15.13	14.89	14.46	14.75
11	14.79	14.26	14.84	14.49	15.05	14.52	14.58	15.12	14.85	14.22	14.67
12	14.62	14.68	14.74	14.71	15.34	14.46	14.17	15.17	14.61	14.70	14.72
13	14.74	14.60	14.77	14.49	15.32	14.57	14.54	14.64	15.04	14.84	14.76
14	14.75	14.41	14.78	14.49	15.25	14.47	14.19	15.19	15.14	14.59	14.73
15	14.74	14.24	14.59	15.16	15.57	14.73	14.51	15.03	14.70	14.22	14.75
16	14.98	14.54	14.49	14.49	15.39	14.59	14.38	14.86	14.75	14.22	14.67
17	14.82	14.24	14.41	14.32	15.63	14.72	14.34	15.10	14.84	14.36	14.68
18	14.82	14.54	14.80	14.43	15.34	14.59	14.12	15.17	14.61	14.75	14.72
19	14.43	14.09	14.29	14.69	15.35	14.46	14.28	15.25	14.95	14.81	14.66
20	14.71	14.50	14.82	14.67	15.02	14.51	14.37	15.21	14.48	14.37	14.67

Table C.21: Absolute Recall Error Vs Recalls - 100 Neurons x 2 Memories, 10% Malfunction (Graph 4.2b)

Recalls	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
2	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
3	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
4	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
5	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
6	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
7	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
8	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
9	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
10	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
11	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
12	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
13	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
14	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
15	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
16	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
17	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
18	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
19	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
20	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table C.22: Recall Bit Error vs Recalls - 100 Neurons x 2 Memories (Graph 4.1b)

Malfunction	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
0	0.24	0.25	0.24	0.00	0.98	0.21	0.00	1.35	0.24	0.61	0.41
1	1.80	1.69	1.69	1.49	2.54	1.73	1.44	2.75	1.68	2.10	1.89
2	3.26	3.57	3.58	3.08	4.23	3.44	3.20	4.44	3.54	3.89	3.62
3	5.06	5.30	5.17	5.12	6.02	5.20	5.10	6.17	5.27	5.59	5.40
4	6.84	6.91	6.72	6.35	7.54	6.68	6.75	7.71	6.93	7.10	6.95
5	8.23	8.54	8.22	8.09	8.84	8.24	8.06	9.29	8.19	8.69	8.44
6	9.64	9.78	9.79	9.77	10.18	9.50	9.47	10.39	9.73	10.04	9.83
7	10.79	11.04	10.79	10.79	11.31	10.89	10.93	11.91	10.93	11.41	11.08
8	12.09	12.31	12.40	12.35	12.90	12.37	12.02	12.80	12.28	12.38	12.39
9	13.43	13.59	13.66	13.40	14.13	13.39	13.38	14.19	13.39	13.69	13.63
10	14.67	14.45	14.52	14.55	14.99	14.33	14.60	15.78	14.62	14.86	14.74
11	15.62	15.62	15.80	15.65	15.97	15.53	15.63	16.38	15.53	16.00	15.77
12	16.93	16.63	16.70	16.59	17.19	17.07	16.52	17.62	16.98	16.97	16.92
13	17.83	17.88	17.73	17.93	18.15	17.66	17.66	17.95	17.91	18.07	17.88
14	18.57	18.68	18.83	18.53	19.36	19.14	19.00	19.48	18.80	19.06	18.95
15	19.54	19.79	19.49	19.50	20.11	19.45	19.57	20.15	19.82	19.66	19.71
16	20.36	21.03	20.63	20.26	21.05	20.91	20.46	21.12	20.46	21.11	20.74
17	21.32	21.26	21.10	21.22	22.00	21.19	21.07	21.77	21.59	21.92	21.44
18	22.64	22.25	22.29	22.17	22.45	22.62	22.34	22.69	21.88	22.62	22.40
19	22.86	22.89	22.96	22.93	23.46	23.59	22.61	23.57	22.49	23.57	23.09
20	23.45	23.61	23.54	23.93	23.96	23.83	23.71	24.88	23.88	23.82	23.86

Table C.23: Absolute Recall Error Vs Recalls - 100 Neurons x 2 Memories (Graph 4.2b)

Malfunction	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
0	0.50	0.50	0.50	0.00	2.00	0.50	0.00	3.00	0.50	1.50	0.90
1	86.00	83.50	84.00	83.00	87.00	86.00	81.50	81.50	81.50	84.00	83.80
2	90.00	95.00	96.50	91.00	94.50	93.50	93.00	95.00	96.50	93.00	93.80
3	100.00	100.00	99.50	100.00	99.50	100.00	100.00	100.00	100.00	100.00	99.90
4	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.50	100.00	99.95
5	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
6	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
7	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
8	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
9	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
10	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
11	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
12	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
13	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
14	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
15	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
16	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
17	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
18	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
19	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
20	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table C.24: Recall Bit Error vs Sigma - 100 Neurons x 2 Memories Symmetric (Graph 4.4)

Sigma	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
0.10	0.47	0.00	0.24	0.24	0.73	0.43	0.40	1.35	0.47	0.00	0.43
0.30	0.70	0.25	1.18	0.24	1.71	0.21	0.60	0.90	0.47	0.82	0.71
0.50	2.11	0.25	0.47	1.23	1.71	1.07	1.20	1.57	0.47	1.03	1.11
0.70	2.35	0.75	1.65	0.73	1.96	2.15	1.00	3.15	1.41	0.82	1.60
0.90	1.41	1.00	0.94	1.47	1.71	1.50	2.20	2.25	1.18	1.64	1.53
1.10	3.76	1.00	1.88	1.47	2.45	2.37	2.00	2.02	1.88	2.67	2.15
1.30	2.11	0.75	1.88	0.49	2.94	1.91	2.76	2.25	1.65	1.64	1.84
1.50	3.52	3.00	2.44	2.21	5.15	3.40	2.15	4.05	1.88	3.28	3.11
2.00	5.17	1.82	4.62	1.33	4.74	4.32	3.54	6.12	2.21	5.06	3.89
3.00	13.16	13.05	10.50	11.95	13.81	12.61	22.69	11.70	16.10	16.58	14.22
4.00	26.80	20.97	24.18	28.18	22.60	22.69	23.89	20.01	23.40	25.25	23.80
5.00	31.34	32.65	34.03	31.11	33.18	30.64	29.79	29.23	27.01	26.92	30.59
6.00	33.67	30.94	34.98	36.03	33.66	29.53	38.86	34.78	27.55	30.69	33.07
7.00	35.80	31.86	38.55	38.62	38.73	38.00	34.04	36.36	37.11	35.52	36.46
8.00	32.81	39.48	39.38	35.69	36.30	37.09	37.65	38.30	39.53	41.35	37.76
9.00	40.11	41.61	39.30	39.06	40.47	40.06	37.06	37.06	39.45	38.05	39.22

Table C.25: Absolute Recall Error Vs Sigma - 100 Neurons x 2 Memories Symmetric (Graph 4.3)

Sigma	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
0.10	1.00	0.00	0.50	0.50	1.50	1.00	1.00	3.00	1.00	0.00	0.95
0.30	1.50	0.50	2.50	0.50	3.50	0.50	1.50	2.00	1.00	2.00	1.55
0.50	4.50	0.50	1.00	2.50	3.50	2.50	3.00	3.50	1.00	2.50	2.45
0.70	5.00	1.50	3.50	1.50	4.00	5.00	2.50	7.00	3.00	2.00	3.50
0.90	3.00	2.00	2.00	3.00	3.50	3.50	5.50	5.00	2.50	4.00	3.40
1.10	8.00	2.00	4.00	3.00	5.00	5.50	5.00	4.50	4.00	6.50	4.75
1.30	4.50	1.50	4.00	1.00	6.00	5.00	43.50	5.00	3.50	4.00	7.80
1.50	7.50	6.00	5.50	4.50	10.50	48.50	41.00	9.00	4.00	8.00	14.45
2.00	90.50	41.00	75.00	3.00	44.50	53.00	68.50	24.00	5.00	48.00	45.25
3.00	100.00	99.00	99.50	97.50	100.00	100.00	100.00	89.00	100.00	99.00	98.40
4.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
5.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
6.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
7.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
8.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
9.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table C.26: Recall Bit Error vs Sigma - 500 Neurons x 3 Memories Symmetric (Graph 4.4)

Sigma	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
0.50	0.08	0.34	0.00	0.17	0.00	0.00	0.31	0.00	0.00	0.16	0.11
1.00	0.17	0.08	0.00	0.17	0.00	0.00	0.16	0.00	0.39	0.15	0.11
1.50	0.65	0.24	0.48	0.32	0.00	0.00	0.39	0.48	0.24	0.15	0.30
2.00	0.00	0.30	0.46	0.77	0.00	0.16	0.89	0.16	0.31	0.65	0.37
2.50	0.73	0.78	0.36	0.58	0.74	0.64	0.45	0.58	1.09	0.45	0.64
3.00	1.19	1.45	1.83	1.44	1.45	1.20	2.13	1.14	2.07	1.58	1.55
3.50	2.86	4.27	4.04	3.25	2.87	3.39	2.39	2.63	3.20	3.44	3.23
4.00	4.90	8.63	5.63	6.41	7.38	7.23	7.97	6.54	6.29	8.03	6.90
4.50	9.08	10.82	9.89	11.05	10.78	9.46	12.93	9.86	10.26	12.85	10.70
5.00	16.38	16.34	14.56	17.74	18.19	15.72	14.85	14.36	16.13	14.49	15.88
5.50	21.49	21.55	20.02	19.06	20.43	23.97	19.39	21.26	20.00	22.15	20.93
6.00	25.92	24.90	24.91	23.00	25.49	24.25	25.80	24.01	23.80	22.39	24.45
7.00	28.57	31.23	29.02	29.03	30.26	28.64	29.80	31.03	29.81	31.05	29.84
8.00	34.01	35.29	34.00	34.94	31.20	33.63	33.06	34.36	34.47	30.58	33.55
9.00	36.88	35.03	35.02	36.97	36.00	36.06	37.18	36.89	35.47	36.63	36.21

Table C.27: Absolute Recall Error Vs Sigma - 500 Neurons x 3 Memories Symmetric (Graph 4.3)

Sigma	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
0.50	0.33	1.33	0.00	0.67	0.00	0.00	1.00	0.00	0.00	0.33	0.37
1.00	0.67	0.33	0.00	0.67	0.00	0.00	0.67	0.00	1.00	0.33	0.37
1.50	1.67	0.67	1.00	1.00	0.00	0.00	1.00	1.00	0.67	0.67	0.77
2.00	0.00	30.33	28.67	26.67	0.00	0.33	2.33	0.33	0.67	1.33	9.07
2.50	47.67	90.33	65.33	64.33	44.67	1.33	59.33	28.33	83.00	96.00	58.03
3.00	65.33	93.33	99.67	98.00	95.67	97.00	99.67	65.00	98.67	100.00	91.23
3.50	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
4.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
4.50	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
5.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
5.50	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
6.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
7.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
8.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
9.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table C.28: Recall Bit Error vs Sigma - 1000 Neurons x 3 Memories Symmetric (Graph 4.4)

Sigma	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.50	0.00	0.00	0.00	0.00	0.00	0.16	0.00	0.09	0.00	0.00	0.03
2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.00	0.02
2.50	0.00	0.00	0.00	0.00	0.08	0.00	0.00	0.00	0.00	0.02	0.01
3.00	0.00	0.00	0.00	0.08	0.00	0.03	0.03	0.20	0.35	0.00	0.07
3.50	0.09	0.45	0.23	0.07	0.16	0.36	0.12	0.23	0.12	0.23	0.21
4.00	0.40	0.53	0.40	0.51	0.49	0.56	0.40	0.49	0.56	0.94	0.53
4.50	1.43	1.51	1.16	0.93	1.13	1.11	0.92	1.92	1.07	1.16	1.23
5.00	2.21	2.54	2.07	2.12	2.80	2.81	2.06	2.82	2.38	3.01	2.48
5.50	4.63	3.66	4.57	3.80	3.65	5.07	4.61	3.70	4.30	4.71	4.27
6.00	5.24	7.46	5.80	7.42	6.73	8.13	6.09	5.12	7.15	6.51	6.57
6.50	12.08	6.78	9.03	11.82	11.52	8.88	10.98	9.46	9.05	8.28	9.79
7.00	14.98	14.04	12.84	12.88	14.84	13.96	13.96	11.85	14.09	11.95	13.54
7.50	16.94	16.16	18.71	17.32	18.21	15.40	15.55	16.99	17.54	16.19	16.90
8.00	20.31	21.45	21.51	20.34	21.31	22.42	21.02	20.90	19.91	21.95	21.11
8.50	23.48	24.38	23.77	22.72	23.77	23.24	24.76	22.66	23.38	24.20	23.64
9.00	26.80	26.04	27.57	26.36	26.25	26.58	26.42	27.23	25.88	27.09	26.62
9.50	27.31	28.66	28.14	29.39	28.46	26.90	28.86	28.47	27.99	30.08	28.43
10.00	29.27	29.65	29.93	30.85	31.34	29.71	30.06	31.96	30.51	29.42	30.27
11.00	32.99	33.91	32.60	32.15	32.58	31.94	32.78	34.99	32.97	32.27	32.92
12.00	37.66	33.82	34.08	34.47	34.81	35.35	36.02	34.89	35.78	34.97	35.19
13.00	35.33	37.58	36.05	37.42	35.56	36.49	36.82	35.98	37.42	37.08	36.57
14.00	38.01	37.99	37.41	37.91	37.56	37.06	37.32	37.75	37.93	37.49	37.64
15.00	39.82	38.48	37.60	39.26	39.88	40.55	38.99	39.10	38.34	38.73	39.08
16.00	39.10	38.05	39.12	39.71	40.45	40.03	39.54	39.16	39.21	39.90	39.43
17.00	40.93	40.75	39.66	39.78	39.88	39.98	40.67	40.65	39.25	40.92	40.25
18.00	42.20	40.19	41.86	40.03	41.16	41.30	41.00	41.21	40.23	41.60	41.08
19.00	42.84	40.72	40.99	41.89	41.13	41.39	41.45	40.38	40.92	41.61	41.33

Table C.29: Absolute Recall Error Vs Sigma - 1000 Neurons x 3 Memories Symmetric (Graph 4.3)

Sigma	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.50	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.33	0.00	0.00	0.07
2.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.03
2.50	0.00	0.00	0.00	0.00	0.33	0.00	0.00	0.00	0.00	24.67	2.50
3.00	0.00	0.00	0.00	51.67	0.00	31.33	27.33	32.67	32.00	0.00	17.50
3.50	60.33	62.00	98.33	33.33	65.67	65.33	92.67	96.33	58.00	66.00	69.80
4.00	99.33	99.33	99.33	99.67	100.00	99.33	100.00	99.00	99.67	99.00	99.47
4.50	100.00	100.00	100.00	100.00	100.00	99.67	100.00	100.00	100.00	100.00	99.97
5.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
5.50	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
6.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
6.50	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
7.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
7.50	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
8.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
8.50	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
9.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
9.50	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
10.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
11.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
12.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
13.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
14.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
15.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
16.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
17.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
18.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
19.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table C.30: Recall Bit Error vs Sigma - 1000 Neurons x 3 Memories Asymmetric (Graph 4.4)

Sigma	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
0.10	0.24	0.25	0.47	0.24	0.73	0.21	0.20	1.12	0.24	0.41	0.41
0.30	1.41	0.25	0.94	0.73	1.23	0.00	0.60	0.68	0.47	0.21	0.65
0.50	0.70	0.25	0.24	0.49	1.71	1.07	0.80	0.90	0.47	0.41	0.70
0.70	1.65	0.50	1.18	0.98	0.98	1.29	0.00	2.02	1.18	0.82	1.06
0.90	1.65	1.00	1.18	1.23	1.23	0.65	0.60	2.25	1.65	1.23	1.27
1.10	2.35	0.50	2.35	0.49	2.45	1.07	1.00	1.80	0.94	1.23	1.42
1.30	2.11	1.00	1.88	2.21	1.47	1.50	1.40	1.80	1.65	0.41	1.54
2.00	2.82	1.25	2.92	0.98	3.43	2.56	1.40	3.15	1.88	2.46	2.29
3.00	4.44	3.96	5.22	7.67	6.07	4.88	7.78	6.07	6.30	5.91	5.83
4.00	7.94	14.90	11.06	23.07	13.73	11.28	14.70	11.91	10.49	19.39	13.85
5.00	22.60	16.54	14.22	19.26	24.55	16.84	25.50	13.98	13.55	24.32	19.14
6.00	22.05	18.50	14.96	35.77	34.62	25.05	21.09	38.29	31.01	26.52	26.79
7.00	33.90	37.20	38.83	41.41	39.02	29.75	26.00	41.16	35.23	30.16	35.27
8.00	37.08	39.00	36.23	41.12	40.76	38.42	33.36	36.07	40.83	41.37	38.42
9.00	38.75	33.72	42.02	35.89	41.07	41.75	32.00	37.12	39.41	36.05	37.78

Table C.31: Absolute Recall Error Vs Sigma - 1000 Neurons x 3 Memories Asymmetric (Graph 4.3)

Sigma	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
0.10	0.50	0.50	1.00	0.50	1.50	0.50	0.50	2.50	0.50	1.00	0.90
0.30	3.00	0.50	2.00	1.50	2.50	0.00	1.50	1.50	1.00	0.50	1.40
0.50	1.50	0.50	0.50	1.00	3.50	2.50	2.00	2.00	1.00	1.00	1.55
0.70	3.50	1.00	2.50	2.00	2.00	3.00	0.00	4.50	2.50	2.00	2.30
0.90	3.50	2.00	2.50	2.50	2.50	1.50	1.50	5.00	3.50	3.00	2.75
1.10	5.00	1.00	5.00	1.00	5.00	2.50	2.50	4.00	2.00	3.00	3.10
1.30	4.50	2.00	4.00	4.50	3.00	3.50	3.50	4.00	3.50	1.00	3.35
2.00	6.00	2.50	40.50	2.00	7.00	49.00	3.50	7.00	4.00	6.00	12.75
3.00	96.00	82.50	84.50	99.00	56.00	96.50	96.50	93.50	89.00	90.50	88.40
4.00	86.50	99.50	99.00	100.00	100.00	53.00	100.00	99.50	100.00	99.00	93.65
5.00	100.00	99.50	100.00	99.50	100.00	100.00	100.00	100.00	100.00	100.00	99.90
6.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
7.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
8.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
9.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table C.32: Recall Bit Error vs Sigma - 100 Neurons x 2 Memories -Storkey Rule Symmetric (Graph 4.5)

Sigma	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
0.001	1.760	1.100	1.330	2.020	2.300	1.750	1.070	1.530	2.640	1.680	1.720
0.003	2.280	2.130	2.230	1.470	2.090	2.320	1.690	3.250	2.660	2.010	2.210
0.005	2.590	1.940	2.930	1.880	2.030	1.940	1.730	2.200	1.910	2.320	2.150
0.007	2.220	1.690	2.020	2.050	2.030	2.380	2.020	2.580	3.590	2.470	2.310
0.009	3.580	2.610	3.070	1.940	2.220	1.890	3.010	2.790	2.440	2.430	2.600
0.011	3.000	2.650	2.800	2.660	5.080	2.440	1.790	2.410	2.790	3.040	2.870
0.013	3.980	3.050	2.640	1.390	4.000	2.610	2.690	3.350	4.650	3.820	3.220
0.015	3.320	4.050	3.410	2.810	3.550	3.660	3.570	3.590	5.350	3.440	3.680
0.017	3.730	4.360	4.030	3.410	5.380	3.010	4.050	3.520	5.790	3.670	4.100
0.019	4.220	5.130	3.670	3.260	3.780	3.950	3.610	3.660	5.510	2.970	3.980
0.021	5.550	4.890	4.720	5.650	5.920	6.810	4.190	5.050	7.810	6.120	5.670
0.023	5.270	5.730	5.190	5.810	3.830	5.440	6.840	6.700	7.290	4.670	5.680
0.025	6.860	5.510	7.130	4.870	6.380	5.570	6.840	4.320	5.390	8.370	6.120
0.027	9.790	7.360	4.580	7.860	8.330	7.000	6.570	9.250	6.720	7.000	7.450
0.029	8.080	8.540	7.830	5.970	8.490	6.190	11.140	10.110	10.210	7.400	8.400
0.031	10.010	7.870	10.040	9.960	9.850	10.080	9.920	12.660	15.080	9.650	10.510
0.033	12.540	11.710	8.580	6.990	13.770	13.260	10.920	11.710	12.750	13.300	11.550
0.035	12.960	15.270	11.580	10.180	16.450	16.470	11.610	16.070	13.400	12.180	13.620

Table C.33: Recall Bit Error vs Sigma - 100 Neurons x 2 Memories -Storkey Rule Symmetric (Graph 4.5)...continued

Sigma	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
0.037	19.490	15.530	12.690	14.440	14.300	19.050	10.750	14.030	13.780	14.120	14.820
0.039	18.750	14.200	15.940	16.010	18.260	17.400	17.130	16.600	18.670	15.110	16.810
0.041	15.360	22.500	17.500	18.530	21.900	22.700	21.500	14.840	18.160	19.230	19.220
0.043	25.900	22.080	17.140	16.730	20.530	17.030	18.950	20.770	23.910	20.920	20.400
0.045	25.300	23.520	19.440	21.710	21.460	25.610	21.380	21.690	24.170	19.470	22.380
0.047	20.480	28.620	24.780	21.140	24.530	20.700	25.000	24.460	17.640	23.600	23.100
0.049	22.480	25.540	26.040	25.060	27.070	24.760	22.870	26.570	25.730	23.250	24.940
0.050	27.030	20.930	23.960	23.510	26.400	27.360	19.940	26.210	26.650	22.210	24.420
0.060	31.270	27.470	29.520	33.290	27.400	30.700	27.790	27.320	31.010	29.860	29.560
0.070	33.070	31.020	34.790	36.000	31.630	31.210	35.700	32.080	32.320	33.310	33.110
0.080	35.350	33.710	34.860	33.820	33.300	35.590	32.780	34.100	35.980	36.410	34.590
0.090	39.500	35.140	37.180	38.080	38.280	32.390	35.360	38.140	38.920	36.070	36.910
0.100	36.650	37.140	36.510	37.370	35.200	37.530	39.660	40.000	36.440	37.500	37.400
0.200	42.070	41.580	41.910	41.660	41.800	41.390	39.370	42.920	42.670	40.750	41.610
0.300	43.350	41.980	42.660	45.120	42.490	42.200	42.640	43.270	43.590	42.820	43.010
0.400	42.880	44.150	42.700	43.480	41.450	43.100	43.630	42.890	42.090	44.700	43.110
0.500	43.810	44.890	44.040	43.280	42.950	43.180	43.800	42.520	43.380	42.440	43.430
0.600	42.540	43.310	44.690	43.090	43.400	42.980	43.670	42.730	43.730	42.700	43.280
0.700	44.190	44.360	42.830	43.700	42.710	43.720	44.950	41.690	43.830	43.510	43.550
0.800	44.910	44.040	43.470	43.190	42.670	42.250	43.620	42.170	44.130	45.310	43.580
0.900	43.730	45.200	43.960	43.990	43.790	44.320	41.880	44.480	43.750	44.560	43.970
1.000	43.350	45.040	44.800	43.300	43.600	43.590	44.680	43.160	41.990	44.730	43.820

Table C.34: Absolute Recall Error Vs Sigma - 100 Neurons x 2 Memories Storkey Rule Symmetric (Graph 4.6)

Sigma	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
0.001	5.000	4.000	4.670	5.330	7.000	6.330	3.670	5.330	8.670	5.330	5.530
0.003	7.000	6.330	6.670	4.330	6.000	7.330	5.670	10.000	8.330	5.670	6.730
0.005	7.670	6.000	8.330	4.330	5.000	5.670	5.670	6.330	5.670	7.670	6.230
0.007	6.330	4.670	5.330	5.670	5.330	5.670	5.330	8.000	9.670	7.670	6.370
0.009	9.670	6.670	8.670	5.330	6.000	5.000	8.000	7.670	7.000	6.330	7.030
0.011	8.330	6.330	7.670	6.670	13.000	5.330	4.000	6.330	7.330	7.330	7.230
0.013	9.670	7.000	7.000	3.670	9.330	5.670	7.000	9.330	11.330	10.330	8.030
0.015	8.330	9.330	8.330	6.670	7.670	8.000	8.330	8.330	12.330	8.330	8.570
0.017	9.000	10.670	9.000	7.670	11.330	8.000	9.670	8.000	12.670	8.670	9.470
0.019	10.000	39.000	8.000	7.330	8.670	9.000	8.000	8.670	35.000	7.000	14.070
0.021	57.000	36.000	32.670	35.330	14.000	56.330	9.670	12.000	17.330	13.330	28.370
0.023	46.000	61.670	57.330	12.670	32.330	36.670	81.670	40.670	53.000	34.000	45.600
0.025	92.000	37.670	36.670	62.330	41.330	13.000	64.000	30.670	28.330	92.670	49.870
0.027	93.330	62.330	33.330	88.330	67.000	17.000	66.000	64.330	39.000	40.670	57.130
0.029	70.330	79.000	37.330	39.330	70.330	67.330	95.000	69.330	91.330	83.670	70.300
0.031	71.000	97.000	98.000	97.670	95.330	95.330	99.670	99.670	68.330	96.000	91.800
0.033	97.000	99.330	75.330	82.670	98.670	92.670	99.330	99.000	95.670	97.670	93.730
0.035	96.330	99.000	95.670	92.330	98.330	100.000	97.330	99.670	95.670	97.670	97.200

Table C.35: Absolute Recall Error Vs Sigma - 100 Neurons x 2 Memories Storkey Rule Symmetric (Graph 4.6)...continued

Sigma	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
0.037	99.330	100.000	79.000	99.330	98.330	99.670	68.000	100.000	80.670	99.330	92.370
0.039	100.000	99.670	98.000	100.000	99.670	99.000	99.330	98.670	100.000	99.670	99.400
0.041	97.000	100.000	100.000	100.000	100.000	100.000	100.000	99.330	99.670	99.670	99.570
0.043	100.000	100.000	100.000	98.330	100.000	99.670	99.670	100.000	99.330	100.000	99.700
0.045	100.000	100.000	100.000	99.670	100.000	100.000	100.000	100.000	100.000	100.000	99.970
0.047	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.049	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.050	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.060	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.070	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.080	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.090	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.100	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.200	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.300	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.400	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.500	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.600	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.700	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.800	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
0.900	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000
1.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000	100.000

Table C.36: Sigma Vs Recalls (Graph 4.7)

Recalls	Sigma1	Sigma2	Sigma3	Sigma4	Sigma5	Sigma6	Sigma7	Sigma8	Sigma9	Sigma10	SigmaAvg
10	0.50	0.60	0.80	0.40	1.00	0.70	0.40	0.90	0.70	1.10	0.71
20	0.70	0.80	0.70	0.80	0.90	0.80	0.60	0.80	0.80	0.90	0.78
30	0.70	0.60	0.90	0.50	1.00	0.90	0.80	1.00	0.50	0.90	0.78
40	0.80	0.80	0.80	0.90	1.00	0.80	0.70	0.80	0.90	0.80	0.83
50	0.50	0.50	0.70	1.00	1.00	0.80	0.50	1.00	0.80	0.90	0.77
60	0.70	0.60	0.80	0.60	0.90	0.70	0.80	0.80	0.60	0.90	0.74
70	0.70	0.50	0.70	0.70	1.00	0.70	0.70	1.00	0.80	0.60	0.74
80	0.70	0.70	0.80	0.70	0.90	0.60	0.50	0.90	0.80	0.80	0.74
90	0.60	0.70	0.90	0.80	1.00	0.90	0.70	0.80	0.70	0.60	0.77
100	0.50	0.80	0.80	0.80	0.80	0.50	0.70	0.70	0.60	0.80	0.70
110	0.70	0.80	0.70	0.60	0.70	0.90	0.80	0.70	0.80	0.90	0.76
120	0.80	0.70	0.90	0.70	0.80	0.90	0.80	0.90	0.80	1.10	0.84
130	0.70	0.60	0.90	0.80	1.00	0.50	0.80	1.00	0.70	1.00	0.80
140	0.80	0.60	0.70	0.60	1.00	0.80	0.80	1.00	0.80	0.80	0.79
150	0.80	0.60	0.80	0.50	1.00	0.90	0.50	0.90	0.70	0.90	0.76

Table C.37: Sigma Vs Neurons (Graph 4.8)

Neurons	Sigma1	Sigma2	Sigma3	Sigma4	Sigma5	Sigma6	Sigma7	Sigma8	Sigma9	Sigma10	SigmaAvg
100	0.70	0.70	0.70	0.70	1.10	0.70	1.10	0.50	0.70	0.90	0.78
200	1.90	1.90	1.70	1.70	2.10	1.90	1.90	1.90	1.90	1.90	1.88
300	2.30	2.30	2.50	2.50	2.70	2.50	2.30	2.50	2.30	2.70	2.46
400	3.10	2.90	3.10	3.10	3.10	3.10	3.10	3.30	2.90	3.10	3.08
500	3.50	3.90	3.70	3.50	3.50	3.50	3.70	3.70	3.50	3.50	3.60
600	3.90	4.10	3.90	4.10	4.10	3.90	4.10	4.10	4.10	4.10	4.04
700	4.30	4.30	4.30	4.10	4.30	4.10	4.30	4.50	4.50	4.10	4.28
800	4.70	4.90	4.50	4.50	4.90	4.90	4.70	4.70	4.90	4.90	4.76

Table C.38: Sigma Vs Memories (Graph 4.9)

Memories	Sigma1	Sigma2	Sigma3	Sigma4	Sigma5	Sigma6	Sigma7	Sigma8	Sigma9	Sigma10	SigmaAvg
1	0.021	0.021	0.021	0.021	0.021	0.021	0.021	0.021	0.021	0.021	0.021
3	0.016	0.016	0.016	0.021	0.016	0.016	0.016	0.016	0.021	0.016	0.017
5	0.016	0.016	0.016	0.011	0.016	0.016	0.016	0.016	0.016	0.016	0.015
7	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011
9	0.006	0.006	0.006	0.006	0.006	0.006	0.006	0.006	0.006	0.006	0.006
11	0.000	0.000	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.000

C.3 Additional Tests

The results that follow were discussed in the first section of the appendix.

Table C.39: Recall Bit Error vs Patterns - using Hebb Rule (No div) - 100Neurons(Graph A.1)

Patterns	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.48	0.00	0.00	0.00	0.92	0.14
3	0.66	0.79	1.15	0.99	0.86	1.72	0.49	0.86	0.83	1.46	0.98
4	2.03	1.22	1.75	1.71	1.90	2.85	2.03	2.23	2.28	2.08	2.01
5	3.10	2.60	4.58	3.16	2.55	4.03	3.54	3.37	2.94	3.37	3.32
6	3.59	3.32	5.25	2.84	3.79	4.28	5.12	4.57	3.14	4.60	4.05
7	4.47	5.10	7.48	4.70	4.48	6.72	5.74	6.90	5.63	5.92	5.71
8	7.51	6.01	8.93	7.05	5.30	8.36	7.91	9.27	8.27	6.53	7.51
9	7.62	7.47	12.26	8.98	7.56	10.67	9.50	11.84	10.66	7.74	9.43
10	8.80	8.73	13.97	9.47	9.13	11.51	10.63	12.15	12.46	9.80	10.67
11	10.35	10.64	16.40	11.59	10.57	12.39	12.95	12.54	13.66	9.39	12.05
12	11.26	10.80	17.28	14.51	11.78	13.94	13.21	14.82	13.43	13.13	13.42
13	14.01	13.20	17.94	17.90	13.57	16.90	13.92	16.69	16.46	14.79	15.54
14	14.32	15.90	20.41	18.14	14.11	17.89	15.60	17.59	20.21	15.50	16.97
15	15.83	22.35	20.98	20.21	16.46	19.35	17.06	18.87	24.12	17.17	19.24
16	17.72	21.76	21.10	22.05	18.48	19.84	18.55	20.19	24.38	17.27	20.13
17	18.52	23.23	22.21	26.58	19.65	21.77	20.10	21.83	25.19	18.09	21.72
18	18.64	24.38	23.63	26.79	19.02	22.34	20.19	22.16	25.11	19.77	22.20
19	22.24	24.58	25.15	27.70	21.14	24.53	23.54	23.66	27.58	20.83	24.10

Table C.40: Absolute Recall Error Vs Patterns - using Hebb Rule (No div) - 100 neurons(Graph A.2)

Patterns	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	2.00	0.30
3	2.67	3.33	4.67	4.00	3.33	6.33	1.67	3.67	3.67	5.00	3.83
4	7.50	4.50	8.50	6.25	7.75	12.25	9.00	10.00	8.25	8.00	8.20
5	11.20	8.40	25.60	10.80	8.00	13.20	12.20	12.00	10.00	10.20	12.16
6	11.17	10.17	26.33	9.67	11.17	15.33	17.33	17.83	11.67	16.00	14.67
7	14.29	15.71	29.71	14.71	11.86	18.29	16.71	30.00	17.57	18.43	18.73
8	20.75	19.50	35.88	20.25	14.75	26.12	27.50	38.25	27.00	19.62	24.96
9	22.56	21.33	41.44	25.78	22.11	33.89	35.67	55.56	33.44	22.00	31.38
10	28.00	27.00	51.90	28.50	31.20	36.60	38.70	51.90	37.70	28.00	35.95
11	30.45	31.55	57.64	38.82	31.73	39.91	42.18	58.36	41.55	28.00	40.02
12	34.67	32.67	60.67	57.75	46.00	49.92	52.75	65.00	44.42	40.92	48.48
13	46.23	45.38	67.38	76.15	45.69	56.92	51.62	67.77	58.23	43.08	55.85
14	46.14	60.86	69.79	75.21	47.07	59.43	65.43	69.07	73.21	46.71	61.29
15	58.40	74.20	73.53	78.20	62.13	63.67	71.40	77.27	79.20	57.40	69.54
16	70.94	76.44	75.06	82.69	70.69	71.69	72.69	80.12	80.50	59.94	74.08
17	70.82	78.18	77.35	86.47	75.59	76.88	76.12	81.12	82.35	65.29	77.02
18	75.00	78.72	77.94	90.67	77.61	81.06	77.39	83.44	85.61	71.22	79.87
19	81.37	81.79	82.42	90.26	82.11	88.32	86.11	88.21	88.84	74.84	84.43

Table C.41: Recall Bit Error vs Patterns - using Hebb Rule /Neurons - 100 Neurons(Graph A.1)

Patterns	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.24	0.00	0.24	0.24	0.69	0.14
3	1.01	1.03	0.89	1.35	1.31	2.44	1.09	1.05	0.81	1.44	1.24
4	2.32	1.88	1.52	1.97	1.76	3.11	2.29	2.61	2.22	1.96	2.16
5	3.13	2.55	4.22	3.11	2.05	4.89	3.44	2.27	2.57	2.97	3.12
6	3.44	3.61	6.04	3.74	3.74	6.48	4.65	4.92	4.03	5.21	4.59
7	4.95	5.34	7.64	5.31	4.48	6.40	5.75	7.44	5.83	5.76	5.89
8	6.71	6.85	10.44	6.50	5.36	8.94	8.12	10.14	8.72	6.40	7.82
9	7.77	8.28	11.94	8.94	7.94	11.00	9.34	11.93	10.95	7.96	9.61
10	8.96	9.46	13.22	9.87	9.26	11.36	11.08	11.95	12.87	9.50	10.75
11	10.20	9.88	16.23	11.39	10.70	12.36	12.24	12.58	13.48	10.31	11.94
12	11.32	11.23	17.11	15.17	12.46	14.89	14.04	14.66	14.78	12.10	13.78
13	13.43	13.19	17.89	18.17	13.97	16.16	14.13	16.31	16.56	14.68	15.45
14	14.13	16.33	20.47	18.32	14.14	17.88	15.81	17.22	21.03	16.18	17.15
15	16.39	22.48	20.62	20.41	15.91	18.80	17.12	18.27	24.13	17.14	19.13
16	18.55	23.39	22.43	22.27	18.97	19.83	18.85	20.33	24.51	17.53	20.67
17	18.87	23.26	22.40	26.64	19.09	21.99	20.18	21.82	25.25	18.87	21.84
18	19.86	24.30	24.18	27.91	19.83	23.40	21.20	21.65	25.55	19.74	22.76
19	22.21	24.77	24.52	27.66	21.59	24.45	24.07	23.70	27.62	21.06	24.17

Table C.42: Absolute Recall Error Vs Patterns - using Hebb Rule /Neurons - 100 neurons(Graph A.2)

Patterns	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.50	0.50	1.50	0.30
3	3.67	3.33	4.00	5.00	4.67	9.33	4.00	5.00	3.67	5.33	4.80
4	9.00	6.25	7.75	7.00	6.75	12.75	9.00	11.25	7.75	7.00	8.45
5	10.80	9.40	23.00	10.40	8.00	17.00	12.20	8.20	9.40	9.20	11.76
6	10.33	9.83	27.83	11.50	11.00	19.50	15.67	18.00	12.33	16.00	15.20
7	15.00	16.29	30.00	16.29	12.29	18.57	17.00	30.71	18.43	17.71	19.23
8	18.75	20.12	39.00	19.50	15.12	26.88	27.75	38.38	27.12	19.38	25.20
9	23.44	22.44	41.11	26.22	23.78	34.22	36.11	54.89	33.44	22.00	31.77
10	26.50	27.20	50.10	28.90	31.20	36.60	39.20	51.80	39.80	27.30	35.86
11	30.27	29.82	56.45	37.00	31.64	39.73	40.27	58.45	41.45	30.64	39.57
12	34.25	33.33	59.42	58.33	47.00	51.67	55.00	63.17	46.75	37.00	48.59
13	45.00	45.92	66.92	74.54	46.15	55.31	51.85	67.54	57.69	44.08	55.50
14	45.86	62.07	70.50	75.57	46.07	59.07	64.43	68.64	73.00	47.36	61.26
15	59.53	74.40	73.73	77.73	61.80	63.33	71.47	76.60	79.80	58.80	69.72
16	69.81	76.88	75.38	82.50	69.31	70.69	73.25	80.19	80.50	58.63	73.71
17	71.76	78.59	77.47	87.41	75.65	76.65	76.06	81.00	82.53	66.71	77.38
18	75.67	78.83	77.72	91.50	80.22	82.28	79.67	83.33	85.83	71.89	80.69
19	81.42	80.79	80.37	89.79	82.42	89.26	86.68	89.05	88.16	75.11	84.31

Table C.43: Recall Bit Error vs Patterns - using Hebb Rule /Patterns - 100Neurons(Graph A.1)

Patterns	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.48	0.00	0.00	0.00	0.92	0.14
3	0.94	1.05	0.72	0.90	1.21	2.38	1.07	1.28	0.79	1.49	1.18
4	1.82	1.51	1.59	2.12	2.02	3.21	2.25	2.30	1.79	1.98	2.06
5	2.60	2.33	4.25	2.63	2.90	4.51	3.74	2.85	3.63	3.15	3.26
6	4.17	3.73	5.69	3.76	3.55	4.85	4.30	4.50	4.01	5.13	4.37
7	4.83	5.22	7.49	5.34	4.96	6.83	6.56	7.16	6.07	5.63	6.01
8	7.44	6.05	9.05	7.51	5.98	8.61	8.39	9.28	8.41	5.80	7.65
9	7.27	8.10	11.17	8.77	7.50	11.15	9.97	11.73	10.75	7.65	9.41
10	9.46	9.65	13.40	9.57	8.49	11.71	10.86	12.31	12.33	9.50	10.73
11	9.68	9.98	16.17	11.91	9.89	12.60	13.03	13.74	12.71	9.54	11.93
12	11.20	11.37	16.74	15.09	12.05	14.97	13.62	15.24	14.25	12.72	13.73
13	14.01	13.32	17.35	17.27	13.73	16.57	13.74	16.20	16.46	15.39	15.40
14	14.40	16.10	20.60	18.18	14.48	17.81	15.24	17.58	21.05	15.14	17.06
15	16.21	22.26	20.67	20.71	16.00	19.38	17.42	18.83	24.26	17.10	19.28
16	17.96	22.40	21.05	21.94	17.60	19.21	18.77	20.04	24.20	16.56	19.97
17	18.56	23.27	22.50	26.14	19.76	21.66	20.07	21.52	25.66	18.87	21.80
18	19.54	24.40	23.58	27.47	19.80	23.21	21.52	22.20	26.20	19.82	22.77
19	22.20	24.54	25.21	27.22	21.15	24.37	22.93	23.56	28.07	20.79	24.00

Table C.44: Absolute Recall Error Vs Patterns - using Hebb Rule /Patterns - 100 neurons(Graph A.2)

Patterns	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	2.00	0.30
3	3.33	3.33	3.00	3.67	4.67	9.33	3.33	5.00	3.67	4.67	4.40
4	7.50	5.50	7.00	6.50	8.50	14.00	10.00	9.25	7.25	8.25	8.38
5	9.00	7.20	24.40	8.20	8.60	16.80	12.60	10.20	12.80	10.40	12.02
6	13.17	11.00	27.67	11.67	11.17	17.67	15.17	17.67	12.67	18.17	15.60
7	14.29	16.00	30.57	16.43	13.57	19.43	18.86	30.86	18.14	18.29	19.64
8	21.00	18.75	35.50	21.88	17.25	26.75	29.00	37.62	27.88	17.62	25.33
9	22.44	23.00	39.00	24.78	23.00	35.56	36.11	54.67	33.33	22.33	31.42
10	28.40	28.90	49.80	29.50	29.10	37.00	38.20	50.60	37.60	27.50	35.66
11	29.45	29.18	57.27	38.91	30.45	41.36	43.00	59.73	38.91	29.09	39.74
12	33.42	33.42	58.58	58.75	45.92	52.83	52.92	64.83	46.58	38.67	48.59
13	46.54	45.38	66.15	74.08	46.08	56.00	50.92	66.77	57.38	44.38	55.37
14	46.29	64.00	70.43	75.93	47.43	59.79	63.71	69.71	75.64	45.50	61.84
15	58.80	74.13	73.13	79.07	59.87	63.93	71.67	76.60	80.13	57.47	69.48
16	71.31	76.12	75.00	82.69	68.94	71.81	73.38	80.19	80.12	57.19	73.68
17	71.00	78.41	77.47	86.76	75.88	74.59	76.12	82.18	83.18	66.76	77.24
18	76.22	78.89	77.28	90.22	77.72	82.39	79.61	83.78	87.11	71.50	80.47
19	81.74	81.95	81.84	90.00	81.26	88.42	85.05	88.58	88.68	74.21	84.17

Table C.45: Recall Bit Error vs Sigma - Hebb Rule (No Div) - 100 Neurons x 2 Memories Symmetric (Graph A.3)

Sigma	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
0.10	1.81	1.92	1.38	2.08	1.49	1.52	1.66	1.49	1.83	1.72	1.69
0.30	2.38	2.16	1.64	2.09	1.25	2.29	2.13	1.20	2.33	2.05	1.95
0.50	2.34	2.20	1.38	2.36	1.83	1.45	1.50	1.69	2.51	2.11	1.94
0.70	3.02	3.00	1.84	2.08	1.57	2.62	1.95	1.41	2.25	2.05	2.18
0.90	2.73	2.00	1.53	2.29	2.06	2.51	2.51	2.81	1.53	1.96	2.19
1.10	3.48	3.13	1.73	2.48	2.07	3.24	2.60	2.27	2.39	2.43	2.58
1.30	3.41	2.47	2.40	2.40	2.11	2.55	2.24	1.62	2.27	2.34	2.38
1.50	2.85	3.11	1.43	1.96	1.87	2.75	3.81	2.12	2.51	2.65	2.51
2.00	2.53	3.45	3.31	3.57	3.42	4.70	3.83	2.22	4.02	1.54	3.26
3.00	5.78	4.37	4.36	4.41	4.23	3.77	4.43	3.77	3.19	5.14	4.35
4.00	6.69	6.59	7.13	8.93	6.20	7.57	6.73	6.78	6.19	7.15	7.00
5.00	14.66	14.69	9.16	10.80	9.68	9.95	9.58	9.85	12.33	12.13	11.28
6.00	21.18	16.84	17.29	16.84	15.23	16.37	16.60	18.86	19.37	15.91	17.45
7.00	23.51	21.11	19.01	24.51	16.41	20.23	20.43	16.79	22.03	20.08	20.41
8.00	27.01	25.37	28.77	28.17	27.93	27.51	28.01	23.86	25.02	27.30	26.90
9.00	29.52	29.24	28.27	26.19	26.45	31.96	28.30	24.71	24.71	28.86	27.82
10.00	29.11	33.08	27.88	33.62	33.09	33.94	33.00	30.96	29.99	29.02	31.37

Table C.46: Absolute Recall Error Vs Sigma - Hebb Rule (No Div) - 100 Neurons x 2 Memories Symmetric (Graph A.4)

Sigma	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
0.10	7.00	6.67	5.00	7.33	5.33	5.33	6.00	6.00	7.67	6.33	6.27
0.30	8.67	7.67	6.00	7.00	4.33	7.33	7.67	4.00	8.67	7.67	6.90
0.50	7.67	7.00	5.00	7.67	6.00	5.00	5.00	6.67	8.33	7.67	6.60
0.70	9.00	9.67	5.33	7.33	4.67	7.67	6.67	4.33	7.67	6.33	6.87
0.90	7.67	7.00	5.00	7.33	4.67	6.33	7.00	8.67	6.00	6.33	6.60
1.10	9.33	9.67	4.33	7.00	5.33	9.00	8.00	6.00	7.67	7.33	7.37
1.30	9.00	7.00	6.00	7.33	5.67	6.33	6.00	4.00	7.33	6.67	6.53
1.50	9.00	8.33	3.33	5.00	4.67	7.00	10.33	6.00	7.33	7.00	6.80
2.00	6.33	9.33	7.33	8.67	7.33	10.67	8.67	5.33	11.67	4.00	7.93
3.00	13.67	66.67	9.67	10.33	9.67	8.33	10.00	8.33	8.00	30.00	17.47
4.00	42.33	58.33	65.33	38.33	63.00	63.33	62.00	40.67	62.67	40.67	53.67
5.00	98.33	99.67	89.67	95.67	84.00	96.67	98.33	95.33	97.33	99.00	95.40
6.00	99.67	93.67	99.67	99.67	99.00	99.67	99.33	99.00	100.00	99.33	98.90
7.00	100.00	100.00	99.67	100.00	99.67	99.00	100.00	99.33	99.67	100.00	99.73
8.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.67	99.97
9.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
10.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table C.47: Recall Bit Error vs Sigma - Hebb Rule /Neurons - 100 Neurons x 2 Memories Symmetric (Graph A.3)

Sigma	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
0.01	2.97	2.60	0.88	2.44	2.13	2.69	3.14	2.59	2.12	2.36	2.39
0.03	3.71	4.67	3.77	3.53	3.24	5.44	4.82	5.44	5.60	3.11	4.33
0.05	12.35	11.92	11.59	10.11	9.37	13.43	13.36	10.97	9.98	10.96	11.40
0.07	21.11	21.79	21.54	20.69	24.10	23.35	22.16	20.59	20.31	20.35	21.60
0.09	27.02	27.33	25.54	28.12	30.09	26.61	29.34	30.66	29.52	30.67	28.49
0.11	32.38	35.05	35.92	29.05	33.47	31.50	31.00	34.10	32.36	35.29	33.01
0.13	35.88	37.75	37.55	34.85	36.35	37.22	33.99	31.27	35.92	34.65	35.54
0.15	38.60	37.70	40.52	35.91	36.69	40.03	37.80	37.33	39.20	38.66	38.24
0.20	41.25	40.35	42.61	41.08	37.42	40.59	42.54	34.65	40.16	41.07	40.17
0.30	42.87	43.12	42.66	42.72	41.97	42.55	41.90	44.24	41.23	43.20	42.65
0.40	42.03	41.82	43.37	42.20	42.61	43.22	40.95	41.55	43.73	43.42	42.49
0.50	42.49	43.08	41.85	41.47	43.52	41.40	42.70	44.42	41.23	44.18	42.63
0.60	43.33	43.15	42.03	44.51	42.52	43.32	40.88	44.56	43.31	41.65	42.93
0.70	45.34	44.23	44.48	43.08	43.30	44.33	44.61	42.50	43.68	42.65	43.82
0.80	44.22	44.00	43.10	42.90	43.40	42.89	44.83	42.58	41.80	43.10	43.28
0.90	42.10	44.15	43.54	41.97	44.11	42.53	45.51	44.35	42.63	44.30	43.52
1.00	42.87	43.70	44.75	44.81	43.90	44.10	43.43	44.69	43.49	43.32	43.91
2.00	44.67	42.69	43.47	43.70	44.08	44.84	44.17	43.93	42.93	44.85	43.93
3.00	45.40	45.30	42.83	43.88	44.48	43.36	45.24	44.35	43.12	43.80	44.18
4.00	46.01	44.68	43.88	43.90	44.02	44.24	45.36	44.37	44.09	42.84	44.34
5.00	43.66	44.83	44.09	43.73	44.96	43.36	42.66	43.38	44.83	42.86	43.84
6.00	44.88	44.70	43.78	42.56	44.31	43.75	44.01	44.10	44.70	45.71	44.25
7.00	44.37	44.69	45.06	44.01	44.94	44.96	43.51	45.99	44.35	44.35	44.62
8.00	43.46	43.06	43.47	43.57	45.34	43.92	43.57	44.06	43.96	43.46	43.79
9.00	42.77	43.65	45.40	43.59	45.65	44.71	44.49	43.26	43.99	44.87	44.24

Table C.48: Absolute Recall Error Vs Sigma - Hebb Rule /Neurons - 100 Neurons x 2 Memories Symmetric (Graph A.4)

Sigma	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
0.01	9.33	7.67	2.33	7.00	5.67	7.00	8.67	7.33	7.33	7.00	6.93
0.03	9.67	12.67	9.67	9.00	7.33	11.67	10.67	12.00	15.00	31.67	12.94
0.05	100.00	95.33	93.00	99.33	99.00	95.33	99.33	98.00	65.33	99.67	94.43
0.07	99.67	100.00	100.00	100.00	100.00	98.00	100.00	100.00	99.33	100.00	99.70
0.09	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.11	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.13	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.15	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.20	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.30	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.40	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.50	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.60	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.70	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.80	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.90	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
1.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
2.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
3.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
4.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
5.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
6.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
7.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
8.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
9.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table C.49: Recall Bit Error vs Sigma - Hebb Rule /Patterns - 100 Neurons x 2 Memories Symmetric (Graph A.3)

Sigma	BitErr1	BitErr2	BitErr3	BitErr4	BitErr5	BitErr6	BitErr7	BitErr8	BitErr9	BitErr10	BitErrAvg
0.10	2.30	2.76	1.90	1.70	1.55	1.47	1.77	1.62	2.21	1.72	1.90
0.30	2.12	2.60	2.73	1.92	2.14	1.98	2.73	2.07	2.35	1.92	2.26
0.50	2.74	3.17	1.89	2.88	1.81	3.27	2.41	2.69	2.21	2.48	2.56
0.70	4.52	4.14	3.35	3.89	3.79	3.43	3.25	2.72	4.02	2.40	3.55
0.90	4.44	4.54	5.27	4.36	3.65	4.28	4.81	3.25	3.79	4.35	4.27
1.10	5.38	6.72	4.82	3.62	3.85	6.29	5.11	4.52	3.71	5.84	4.99
1.30	6.66	8.88	9.50	6.51	6.01	5.16	7.70	7.29	6.08	5.97	6.98
1.50	10.21	9.26	11.32	7.05	9.54	7.91	8.90	9.47	7.74	8.02	8.94
2.00	18.93	16.32	19.37	18.24	12.91	15.66	18.44	18.44	15.85	13.71	16.79
3.00	28.67	28.21	28.50	31.82	29.01	33.06	26.91	28.72	27.89	29.15	29.19
4.00	32.20	35.16	32.68	32.31	37.57	36.64	33.48	33.54	31.30	33.99	33.89
5.00	34.04	38.83	39.75	38.24	37.09	35.71	37.00	32.64	36.68	35.33	36.53
6.00	40.60	38.41	38.62	37.03	40.98	38.73	40.81	39.84	39.11	41.87	39.60
7.00	40.69	39.86	39.90	40.72	37.88	38.30	40.65	38.05	40.58	40.00	39.66
8.00	43.20	40.76	40.43	42.43	40.93	39.76	36.66	39.98	40.12	42.61	40.69
9.00	42.78	41.15	39.83	40.46	39.80	42.80	42.67	41.06	43.23	42.08	41.59
10.00	40.85	40.69	42.39	40.65	40.03	41.49	41.27	42.28	41.56	38.50	40.97

Table C.50: Absolute Recall Error Vs Sigma - Hebb Rule /Patterns - 100 Neurons x 2 Memories Symmetric (Graph A.4)

Sigma	AbsErr1	AbsErr2	AbsErr3	AbsErr4	AbsErr5	AbsErr6	AbsErr7	AbsErr8	AbsErr9	AbsErr10	AbsErrAvg
0.10	8.00	9.33	6.00	6.00	4.67	5.00	5.67	5.67	8.33	6.33	6.50
0.30	7.00	7.33	7.67	6.67	6.00	5.33	8.33	6.33	7.00	6.00	6.77
0.50	7.00	10.00	5.00	8.00	5.33	9.00	6.00	8.33	7.33	7.33	7.33
0.70	11.00	11.33	7.67	10.00	8.00	8.00	7.67	6.67	11.33	6.00	8.77
0.90	10.67	12.67	11.67	10.00	8.00	9.67	11.33	7.33	11.00	11.00	10.33
1.10	31.67	26.33	58.00	9.00	8.33	39.00	13.00	32.67	9.00	32.67	25.97
1.30	45.67	94.00	49.00	56.33	14.33	18.33	68.67	76.33	66.67	38.00	52.73
1.50	84.00	92.00	70.00	90.00	98.33	63.00	53.67	44.33	95.00	45.67	73.60
2.00	100.00	100.00	100.00	100.00	98.00	99.00	100.00	100.00	100.00	99.00	99.60
3.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
4.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
5.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
6.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
7.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
8.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
9.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
10.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table C.51: Recall Bit Error vs Sigma - Hebb Rule (No Div) - 100 Neurons x 2 Memories Symmetric (Graph A.9)

Noise1	BitErr1	Noise2	BitErr2	Noise3	BitErr3	Noise4	BitErr4	Noise5	BitErr5
0.00	1.81	0.00	1.92	0.00	1.38	0.00	2.08	0.00	1.49
0.00	2.38	0.00	2.16	0.00	1.64	0.00	2.09	0.00	1.25
0.00	2.34	0.01	2.20	0.00	1.38	0.00	2.36	0.00	1.83
0.00	2.73	0.00	2.00	0.00	1.53	0.01	2.29	0.01	2.06
0.01	3.02	0.01	3.00	0.00	1.84	0.01	2.08	0.00	1.57
0.01	3.48	0.02	3.13	0.01	1.73	0.00	2.48	0.01	2.07
0.01	2.85	0.01	3.11	0.02	1.43	0.02	1.96	0.01	1.87
0.02	3.41	0.02	2.47	0.00	2.40	0.01	2.40	0.02	2.11
0.02	2.53	0.01	3.45	0.03	3.31	0.00	3.57	0.01	3.42
0.03	5.78	0.03	4.37	0.05	4.36	0.02	4.41	0.02	4.23
0.05	21.18	0.03	16.84	0.03	17.29	0.02	16.84	0.01	15.23
0.04	23.51	0.01	21.11	0.04	19.01	0.04	24.51	0.02	16.41
0.01	6.69	0.08	6.59	0.03	7.13	0.06	8.93	0.01	6.20
0.01	27.01	0.17	25.37	0.05	28.77	0.05	28.17	0.03	27.93
0.11	14.66	0.11	14.69	0.07	9.16	0.03	10.80	0.02	9.68
0.07	29.52	0.05	29.24	0.10	28.27	0.03	26.19	0.03	26.45
0.10	29.11	0.14	33.08	0.08	27.88	0.04	33.62	0.02	33.09

Table C.52: Recall Bit Error vs Sigma - Hebb Rule (No Div) - 100 Neurons x 2 Memories Symmetric (Graph A.9) ...continued

Noise6	BitErr6	Noise7	BitErr7	Noise8	BitErr8	Noise9	BitErr9	Noise10	BitErr10	NoiseAvg	BitErrAvg
0.00	1.52	0.00	1.66	0.00	1.49	0.00	1.83	0.00	1.72	0.00	1.69
0.00	2.29	0.01	2.13	0.00	1.20	0.00	2.33	0.00	2.05	0.00	1.95
0.01	1.45	0.00	1.50	0.00	1.69	0.00	2.51	0.01	2.11	0.00	1.94
0.01	2.51	0.00	2.51	0.00	2.81	0.00	1.53	0.00	1.96	0.00	2.19
0.00	2.62	0.00	1.95	0.00	1.41	0.01	2.25	0.00	2.05	0.00	2.18
0.00	3.24	0.01	2.60	0.00	2.27	0.05	2.39	0.01	2.43	0.01	2.58
0.00	2.75	0.00	3.81	0.02	2.12	0.03	2.51	0.00	2.65	0.01	2.51
0.00	2.55	0.04	2.24	0.02	1.62	0.01	2.27	0.01	2.34	0.01	2.38
0.00	4.70	0.02	3.83	0.00	2.22	0.04	4.02	0.04	1.54	0.02	3.26
0.03	3.77	0.03	4.43	0.01	3.77	0.04	3.19	0.03	5.14	0.03	4.35
0.05	16.37	0.10	16.60	0.02	18.86	0.10	19.37	0.03	15.91	0.04	17.45
0.03	20.23	0.10	20.43	0.04	16.79	0.13	22.03	0.05	20.08	0.05	20.41
0.04	7.57	0.06	6.73	0.05	6.78	0.12	6.19	0.09	7.15	0.05	7.00
0.02	27.51	0.05	28.01	0.06	23.86	0.01	25.02	0.12	27.30	0.06	26.90
0.04	9.95	0.13	9.58	0.01	9.85	0.05	12.33	0.00	12.13	0.06	11.28
0.05	31.96	0.15	28.30	0.03	24.71	0.05	24.71	0.04	28.86	0.06	27.82
0.05	33.94	0.13	33.00	0.11	30.96	0.12	29.99	0.05	29.02	0.08	31.37

Table C.53: Absolute Recall Error Vs Sigma - Hebb Rule (No Div) - 100 Neurons x 2 Memories Symmetric (Graph A.8)

Noise1	AbsErr1	Noise2	AbsErr2	Noise3	AbsErr3	Noise4	AbsErr4	Noise5	AbsErr5
0.00	7.00	0.00	6.67	0.00	5.00	0.00	7.33	0.00	5.33
0.00	8.67	0.00	7.67	0.00	6.00	0.00	7.00	0.00	4.33
0.00	7.67	0.01	7.00	0.00	5.00	0.00	7.67	0.00	6.00
0.01	9.00	0.01	9.67	0.00	5.33	0.01	7.33	0.00	4.67
0.00	7.67	0.00	7.00	0.00	5.00	0.01	7.33	0.01	4.67
0.01	9.33	0.02	9.67	0.01	4.33	0.00	7.00	0.01	5.33
0.02	9.00	0.02	7.00	0.00	6.00	0.01	7.33	0.02	5.67
0.01	9.00	0.01	8.33	0.02	3.33	0.02	5.00	0.01	4.67
0.02	6.33	0.01	9.33	0.03	7.33	0.00	8.67	0.01	7.33
0.03	13.67	0.03	66.67	0.05	9.67	0.02	10.33	0.02	9.67
0.01	42.33	0.08	58.33	0.03	65.33	0.06	38.33	0.01	63.00
0.11	98.33	0.11	99.67	0.07	89.67	0.03	95.67	0.02	84.00
0.05	99.67	0.03	93.67	0.03	99.67	0.02	99.67	0.01	99.00
0.04	100.00	0.01	100.00	0.04	99.67	0.04	100.00	0.02	99.67
0.01	100.00	0.17	100.00	0.05	100.00	0.05	100.00	0.03	100.00
0.07	100.00	0.05	100.00	0.10	100.00	0.03	100.00	0.03	100.00
0.10	100.00	0.14	100.00	0.08	100.00	0.04	100.00	0.02	100.00

Table C.54: Absolute Recall Error Vs Sigma - Hebb Rule (No Div) - 100 Neurons x 2 Memories Symmetric (Graph A.8)...continued

Noise6	AbsErr6	Noise7	AbsErr7	Noise8	AbsErr8	Noise9	AbsErr9	Noise10	AbsErr10	NoiseAvg	AbsErrAvg
0.00	5.33	0.00	6.00	0.00	6.00	0.00	7.67	0.00	6.33	0.00	6.27
0.00	7.33	0.01	7.67	0.00	4.00	0.00	8.67	0.00	7.67	0.00	6.90
0.01	5.00	0.00	5.00	0.00	6.67	0.00	8.33	0.01	7.67	0.00	6.60
0.00	7.67	0.00	6.67	0.00	4.33	0.01	7.67	0.00	6.33	0.00	6.87
0.01	6.33	0.00	7.00	0.00	8.67	0.00	6.00	0.00	6.33	0.00	6.60
0.00	9.00	0.01	8.00	0.00	6.00	0.05	7.67	0.01	7.33	0.01	7.37
0.00	6.33	0.04	6.00	0.02	4.00	0.01	7.33	0.01	6.67	0.01	6.53
0.00	7.00	0.00	10.33	0.02	6.00	0.03	7.33	0.00	7.00	0.01	6.80
0.00	10.67	0.02	8.67	0.00	5.33	0.04	11.67	0.04	4.00	0.02	7.93
0.03	8.33	0.03	10.00	0.01	8.33	0.04	8.00	0.03	30.00	0.03	17.47
0.04	63.33	0.06	62.00	0.05	40.67	0.12	62.67	0.09	40.67	0.05	53.67
0.04	96.67	0.13	98.33	0.01	95.33	0.05	97.33	0.00	99.00	0.06	95.40
0.05	99.67	0.10	99.33	0.02	99.00	0.10	100.00	0.03	99.33	0.04	98.90
0.03	99.00	0.10	100.00	0.04	99.33	0.13	99.67	0.05	100.00	0.05	99.73
0.02	100.00	0.05	100.00	0.06	100.00	0.01	100.00	0.12	99.67	0.06	99.97
0.05	100.00	0.15	100.00	0.03	100.00	0.05	100.00	0.04	100.00	0.06	100.00
0.05	100.00	0.13	100.00	0.11	100.00	0.12	100.00	0.05	100.00	0.08	100.00

Table C.55: Recall Bit Error vs Sigma - Hebb Rule /neurons - 100 Neurons x 2 Memories Symmetric (Graph A.9)

Noise1	BitErr1	Noise2	BitErr2	Noise3	BitErr3	Noise4	BitErr4	Noise5	BitErr5
0.03	2.97	0.03	2.60	0.00	0.88	0.01	2.44	0.00	2.13
0.02	3.71	0.05	4.67	0.05	3.77	0.04	3.53	0.04	3.24
0.03	12.35	0.01	11.92	0.16	11.59	0.00	10.11	0.02	9.37
0.03	21.11	0.13	21.79	0.10	21.54	0.02	20.69	0.02	24.10
0.05	27.02	0.12	27.33	0.03	25.54	0.06	28.12	0.01	30.09
0.06	35.88	0.13	37.75	0.13	37.55	0.03	34.85	0.15	36.35
0.04	32.38	0.14	35.05	0.13	35.92	0.17	29.05	0.02	33.47
0.02	38.60	0.36	37.70	0.14	40.52	0.19	35.91	0.07	36.69
0.05	41.25	0.29	40.35	0.26	42.61	0.10	41.08	0.07	37.42
0.21	42.87	0.11	43.12	0.21	42.66	0.27	42.72	0.30	41.97
0.42	42.49	0.88	43.08	0.60	41.85	0.13	41.47	0.23	43.52
0.69	43.33	0.14	43.15	1.07	42.03	0.10	44.51	0.34	42.52
0.76	42.03	0.85	41.82	0.37	43.37	0.10	42.20	0.00	42.61
0.75	45.34	0.60	44.23	0.17	44.48	0.44	43.08	0.32	43.30
0.26	44.22	0.10	44.00	1.34	43.10	0.56	42.90	0.99	43.40
1.26	42.87	0.63	43.70	1.51	44.75	0.00	44.81	0.19	43.90
0.49	42.10	2.11	44.15	0.25	43.54	0.58	41.97	0.25	44.11

Table C.56: Recall Bit Error vs Sigma - Hebb Rule /neurons - 100 Neurons x 2 Memories Symmetric (Graph A.9) ...continued

Noise6	BitErr6	Noise7	BitErr7	Noise8	BitErr8	Noise9	BitErr9	Noise10	BitErr10	NoiseAvg	BitErrAvg
0.01	2.69	0.00	3.14	0.00	2.59	0.00	2.12	0.00	2.36	0.01	2.39
0.01	5.44	0.00	4.82	0.02	5.44	0.08	5.60	0.05	3.11	0.04	4.33
0.03	13.43	0.04	13.36	0.05	10.97	0.05	9.98	0.07	10.96	0.04	11.40
0.03	23.35	0.11	22.16	0.04	20.59	0.00	20.31	0.03	20.35	0.05	21.60
0.08	26.61	0.10	29.34	0.00	30.66	0.09	29.52	0.06	30.67	0.06	28.49
0.17	37.22	0.09	33.99	0.05	31.27	0.12	35.92	0.04	34.65	0.10	35.54
0.15	31.50	0.20	31.00	0.14	34.10	0.07	32.36	0.10	35.29	0.12	33.01
0.09	40.03	0.13	37.80	0.01	37.33	0.01	39.20	0.32	38.66	0.13	38.24
0.15	40.59	0.03	42.54	0.16	34.65	0.44	40.16	0.11	41.07	0.17	40.17
0.04	42.55	0.23	41.90	0.15	44.24	0.75	41.23	0.37	43.20	0.26	42.65
0.38	41.40	0.01	42.70	0.81	44.42	0.09	41.23	0.35	44.18	0.39	42.63
0.69	43.32	0.33	40.88	0.07	44.56	0.43	43.31	0.25	41.65	0.41	42.93
0.14	43.22	0.31	40.95	0.81	41.55	0.93	43.73	0.05	43.42	0.43	42.49
0.29	44.33	1.77	44.61	0.31	42.50	0.13	43.68	0.63	42.65	0.54	43.82
0.86	42.89	2.40	44.83	1.16	42.58	0.18	41.80	0.47	43.10	0.83	43.28
0.81	44.10	0.87	43.43	1.44	44.69	0.94	43.49	0.70	43.32	0.84	43.91
0.34	42.53	2.40	45.51	0.67	44.35	2.07	42.63	0.27	44.30	0.94	43.52

Table C.57: Absolute Recall Error Vs Sigma - Hebb Rule /neurons - 100 Neurons x 2 Memories Symmetric (Graph A.8)

Noise1	AbsErr1	Noise2	AbsErr2	Noise3	AbsErr3	Noise4	AbsErr4	Noise5	AbsErr5
0.03	9.33	0.03	7.67	0.00	2.33	0.01	7.00	0.00	5.67
0.02	9.67	0.05	12.67	0.05	9.67	0.04	9.00	0.04	7.33
0.03	100.00	0.01	95.33	0.16	93.00	0.00	99.33	0.02	99.00
0.03	99.67	0.13	100.00	0.10	100.00	0.02	100.00	0.02	100.00
0.05	100.00	0.12	100.00	0.03	100.00	0.06	100.00	0.01	100.00
0.04	100.00	0.14	100.00	0.13	100.00	0.17	100.00	0.02	100.00
0.06	100.00	0.13	100.00	0.13	100.00	0.03	100.00	0.15	100.00
0.02	100.00	0.36	100.00	0.14	100.00	0.19	100.00	0.07	100.00
0.05	100.00	0.29	100.00	0.26	100.00	0.10	100.00	0.07	100.00
0.21	100.00	0.11	100.00	0.21	100.00	0.27	100.00	0.30	100.00
0.76	100.00	0.85	100.00	0.37	100.00	0.10	100.00	0.00	100.00
0.42	100.00	0.88	100.00	0.60	100.00	0.13	100.00	0.23	100.00
0.69	100.00	0.14	100.00	1.07	100.00	0.10	100.00	0.34	100.00
0.75	100.00	0.60	100.00	0.17	100.00	0.44	100.00	0.32	100.00
0.26	100.00	0.10	100.00	1.34	100.00	0.56	100.00	0.99	100.00
0.49	100.00	2.11	100.00	0.25	100.00	0.58	100.00	0.25	100.00
1.26	100.00	0.63	100.00	1.51	100.00	0.00	100.00	0.19	100.00
2.18	100.00	8.04	100.00	2.65	100.00	3.76	100.00	12.73	100.00
2.89	100.00	14.57	100.00	5.75	100.00	3.39	100.00	11.80	100.00
1.42	100.00	12.40	100.00	1.96	100.00	31.55	100.00	32.60	100.00
14.12	100.00	64.06	100.00	24.40	100.00	15.68	100.00	31.79	100.00
6.88	100.00	31.68	100.00	15.42	100.00	16.13	100.00	19.03	100.00
39.06	100.00	0.42	100.00	25.91	100.00	1.93	100.00	72.72	100.00
29.78	100.00	4.82	100.00	37.30	100.00	38.38	100.00	95.79	100.00
40.67	100.00	60.37	100.00	9.47	100.00	80.34	100.00	34.85	100.00

Table C.58: Absolute Recall Error Vs Sigma - Hebb Rule /neurons - 100 Neurons x 2 Memories Symmetric (Graph A.8)...continued

Noise6	AbsErr6	Noise7	AbsErr7	Noise8	AbsErr8	Noise9	AbsErr9	Noise10	AbsErr10	NoiseAvg	AbsErrAvg
0.01	7.00	0.00	8.67	0.00	7.33	0.00	7.33	0.00	7.00	0.01	6.93
0.01	11.67	0.00	10.67	0.02	12.00	0.08	15.00	0.05	31.67	0.04	12.94
0.03	95.33	0.04	99.33	0.05	98.00	0.05	65.33	0.07	99.67	0.04	94.43
0.03	98.00	0.11	100.00	0.04	100.00	0.00	99.33	0.03	100.00	0.05	99.70
0.08	100.00	0.10	100.00	0.00	100.00	0.09	100.00	0.06	100.00	0.06	100.00
0.15	100.00	0.20	100.00	0.14	100.00	0.07	100.00	0.10	100.00	0.12	100.00
0.17	100.00	0.09	100.00	0.05	100.00	0.12	100.00	0.04	100.00	0.10	100.00
0.09	100.00	0.13	100.00	0.01	100.00	0.01	100.00	0.32	100.00	0.13	100.00
0.15	100.00	0.03	100.00	0.16	100.00	0.44	100.00	0.11	100.00	0.17	100.00
0.04	100.00	0.23	100.00	0.15	100.00	0.75	100.00	0.37	100.00	0.26	100.00
0.14	100.00	0.31	100.00	0.81	100.00	0.93	100.00	0.05	100.00	0.43	100.00
0.38	100.00	0.01	100.00	0.81	100.00	0.09	100.00	0.35	100.00	0.39	100.00
0.69	100.00	0.33	100.00	0.07	100.00	0.43	100.00	0.25	100.00	0.41	100.00
0.29	100.00	1.77	100.00	0.31	100.00	0.13	100.00	0.63	100.00	0.54	100.00
0.86	100.00	2.40	100.00	1.16	100.00	0.18	100.00	0.47	100.00	0.83	100.00
0.34	100.00	2.40	100.00	0.67	100.00	2.07	100.00	0.27	100.00	0.94	100.00
0.81	100.00	0.87	100.00	1.44	100.00	0.94	100.00	0.70	100.00	0.84	100.00
5.95	100.00	6.27	100.00	7.29	100.00	7.37	100.00	7.11	100.00	6.34	100.00
20.45	100.00	3.00	100.00	5.38	100.00	0.11	100.00	11.40	100.00	7.87	100.00
15.12	100.00	29.60	100.00	16.15	100.00	7.75	100.00	13.04	100.00	16.16	100.00
15.19	100.00	16.24	100.00	14.43	100.00	25.84	100.00	27.63	100.00	24.94	100.00
16.00	100.00	33.75	100.00	22.13	100.00	28.24	100.00	6.59	100.00	19.58	100.00
43.32	100.00	43.76	100.00	31.87	100.00	14.93	100.00	2.35	100.00	27.63	100.00
21.87	100.00	92.07	100.00	35.36	100.00	20.81	100.00	91.44	100.00	46.76	100.00
22.61	100.00	7.40	100.00	8.98	100.00	37.38	100.00	78.26	100.00	38.03	100.00

Table C.59: Recall Bit Error vs Sigma - Hebb Rule /Patterns - 100 Neurons x 2 Memories Symmetric (Graph A.9)

Noise1	BitErr1	Noise2	BitErr2	Noise3	BitErr3	Noise4	BitErr4	Noise5	BitErr5
0.01	2.30	0.00	2.76	0.00	1.90	0.00	1.70	0.00	1.55
0.01	2.74	0.01	3.17	0.00	1.89	0.01	2.88	0.00	1.81
0.01	2.12	0.01	2.60	0.00	2.73	0.01	1.92	0.00	2.14
0.01	6.66	0.01	8.88	0.04	9.50	0.04	6.51	0.02	6.01
0.03	4.44	0.04	4.54	0.03	5.27	0.01	4.36	0.02	3.65
0.02	4.52	0.05	4.14	0.05	3.35	0.00	3.89	0.00	3.79
0.04	5.38	0.02	6.72	0.04	4.82	0.03	3.62	0.02	3.85
0.02	10.21	0.13	9.26	0.06	11.32	0.03	7.05	0.01	9.54
0.03	18.93	0.17	16.32	0.01	19.37	0.01	18.24	0.02	12.91
0.08	28.67	0.16	28.21	0.15	28.50	0.01	31.82	0.05	29.01
0.03	34.04	0.17	38.83	0.07	39.75	0.21	38.24	0.13	37.09
0.03	32.20	0.30	35.16	0.10	32.68	0.13	32.31	0.01	37.57
0.04	40.60	0.24	38.41	0.06	38.62	0.10	37.03	0.03	40.98
0.36	40.85	0.25	40.69	0.18	42.39	0.26	40.65	0.01	40.03
0.04	42.78	0.09	41.15	0.07	39.83	0.20	40.46	0.04	39.80
0.42	40.69	0.28	39.86	0.46	39.90	0.11	40.72	0.12	37.88
0.32	43.20	0.32	40.76	0.18	40.43	0.29	42.43	0.07	40.93

Table C.60: Recall Bit Error vs Sigma - Hebb Rule /Patterns - 100 Neurons x 2 Memories Symmetric (Graph A.9) ...continued

Noise6	BitErr6	Noise7	BitErr7	Noise8	BitErr8	Noise9	BitErr9	Noise10	BitErr10	NoiseAvg	BitErrAvg
0.00	1.47	0.00	1.77	0.00	1.62	0.00	2.21	0.00	1.72	0.00	1.90
0.01	3.27	0.00	2.41	0.00	2.69	0.00	2.21	0.00	2.48	0.01	2.56
0.00	1.98	0.00	2.73	0.01	2.07	0.02	2.35	0.00	1.92	0.01	2.26
0.02	5.16	0.05	7.70	0.01	7.29	0.02	6.08	0.01	5.97	0.02	6.98
0.00	4.28	0.03	4.81	0.03	3.25	0.04	3.79	0.01	4.35	0.02	4.27
0.01	3.43	0.02	3.25	0.02	2.72	0.02	4.02	0.05	2.40	0.02	3.55
0.01	6.29	0.06	5.11	0.03	4.52	0.08	3.71	0.00	5.84	0.03	4.99
0.02	7.91	0.08	8.90	0.04	9.47	0.02	7.74	0.00	8.02	0.04	8.94
0.02	15.66	0.01	18.44	0.08	18.44	0.09	15.85	0.13	13.71	0.06	16.79
0.01	33.06	0.05	26.91	0.08	28.72	0.04	27.89	0.05	29.15	0.07	29.19
0.10	35.71	0.03	37.00	0.06	32.64	0.12	36.68	0.07	35.33	0.10	36.53
0.21	36.64	0.22	33.48	0.01	33.54	0.15	31.30	0.13	33.99	0.13	33.89
0.17	38.73	0.44	40.81	0.12	39.84	0.20	39.11	0.17	41.87	0.16	39.60
0.01	41.49	0.32	41.27	0.04	42.28	0.07	41.56	0.07	38.50	0.16	40.97
0.19	42.80	0.28	42.67	0.45	41.06	0.42	43.23	0.04	42.08	0.18	41.59
0.11	38.30	0.19	40.65	0.06	38.05	0.29	40.58	0.13	40.00	0.22	39.66
0.09	39.76	0.35	36.66	0.22	39.98	0.06	40.12	0.37	42.61	0.23	40.69

Table C.61: Absolute Recall Error Vs Sigma - Hebb Rule /Patterns - 100 Neurons x 2 Memories Symmetric (Graph A.8)

Noise1	AbsErr1	Noise2	AbsErr2	Noise3	AbsErr3	Noise4	AbsErr4	Noise5	AbsErr5
0.01	8.00	0.00	9.33	0.00	6.00	0.00	6.00	0.00	4.67
0.01	7.00	0.01	7.33	0.00	7.67	0.01	6.67	0.00	6.00
0.01	7.00	0.01	10.00	0.00	5.00	0.01	8.00	0.00	5.33
0.02	11.00	0.05	11.33	0.05	7.67	0.00	10.00	0.00	8.00
0.03	10.67	0.04	12.67	0.03	11.67	0.01	10.00	0.02	8.00
0.04	31.67	0.02	26.33	0.04	58.00	0.03	9.00	0.02	8.33
0.01	45.67	0.01	94.00	0.04	49.00	0.04	56.33	0.02	14.33
0.02	84.00	0.13	92.00	0.06	70.00	0.03	90.00	0.01	98.33
0.03	100.00	0.17	100.00	0.01	100.00	0.01	100.00	0.02	98.00
0.08	100.00	0.16	100.00	0.15	100.00	0.01	100.00	0.05	100.00
0.03	100.00	0.30	100.00	0.10	100.00	0.13	100.00	0.01	100.00
0.03	100.00	0.17	100.00	0.07	100.00	0.21	100.00	0.13	100.00
0.04	100.00	0.24	100.00	0.06	100.00	0.10	100.00	0.03	100.00
0.42	100.00	0.28	100.00	0.46	100.00	0.11	100.00	0.12	100.00
0.32	100.00	0.32	100.00	0.18	100.00	0.29	100.00	0.07	100.00
0.04	100.00	0.09	100.00	0.07	100.00	0.20	100.00	0.04	100.00
0.36	100.00	0.25	100.00	0.18	100.00	0.26	100.00	0.01	100.00

Table C.62: Absolute Recall Error Vs Sigma - Hebb Rule /Patterns - 100 Neurons x 2 Memories Symmetric (Graph A.8)...continued

Noise6	AbsErr6	Noise7	AbsErr7	Noise8	AbsErr8	Noise9	AbsErr9	Noise10	AbsErr10	NoiseAvg	AbsErrAvg
0.00	5.00	0.00	5.67	0.00	5.67	0.00	8.33	0.00	6.33	0.00	6.50
0.00	5.33	0.00	8.33	0.01	6.33	0.02	7.00	0.00	6.00	0.01	6.77
0.01	9.00	0.00	6.00	0.00	8.33	0.00	7.33	0.00	7.33	0.01	7.33
0.01	8.00	0.02	7.67	0.02	6.67	0.02	11.33	0.05	6.00	0.02	8.77
0.00	9.67	0.03	11.33	0.03	7.33	0.04	11.00	0.01	11.00	0.02	10.33
0.01	39.00	0.06	13.00	0.03	32.67	0.08	9.00	0.00	32.67	0.03	25.97
0.02	18.33	0.05	68.67	0.01	76.33	0.02	66.67	0.01	38.00	0.02	52.73
0.02	63.00	0.08	53.67	0.04	44.33	0.02	95.00	0.00	45.67	0.04	73.60
0.02	99.00	0.01	100.00	0.08	100.00	0.09	100.00	0.13	99.00	0.06	99.60
0.01	100.00	0.05	100.00	0.08	100.00	0.04	100.00	0.05	100.00	0.07	100.00
0.21	100.00	0.22	100.00	0.01	100.00	0.15	100.00	0.13	100.00	0.13	100.00
0.10	100.00	0.03	100.00	0.06	100.00	0.12	100.00	0.07	100.00	0.10	100.00
0.17	100.00	0.44	100.00	0.12	100.00	0.20	100.00	0.17	100.00	0.16	100.00
0.11	100.00	0.19	100.00	0.06	100.00	0.29	100.00	0.13	100.00	0.22	100.00
0.09	100.00	0.35	100.00	0.22	100.00	0.06	100.00	0.37	100.00	0.23	100.00
0.19	100.00	0.28	100.00	0.45	100.00	0.42	100.00	0.04	100.00	0.18	100.00
0.01	100.00	0.32	100.00	0.04	100.00	0.07	100.00	0.07	100.00	0.16	100.00