Imperial College London

Department of Computing

BEng Computing Individual Project

Stochastic Hopfield Networks To Model Secure And Insecure Attachment Types With Noise

Author:
Rebecca Tucker

Supervisor:
Professor Abbas Edalat
Second Marker:
Dr Aldo Faisal

June 2013

Path to code archive:
`/homes/rjt09/project-archive`

Abstract

With some sources estimating that at least a quarter of us will suffer from mental illness at some point within our lifetime, maintaining good mental health is becoming more of a prevalent issue for modern day society. In the past couple of years the development of a new therapeutic technique called self-attachment therapy has started, its focus being to reduce mental health problems that can be attributed to an insecure attachment in childhood. Its development is being based on our understanding of associative memory, and as such we seek to confirm theoretical results and improve our knowledge of these models.

Current studies on this topic are focused on the Hopfield network, a mathematical model of associative memory, and the newly developing theory of strong attractors within these networks. The Hopfield model is known to have its limitations, especially regarding the presence of spurious states. However it is possible to resolve some of these issues by adding the concept of temperature to the model.

We have conducted an investigation into the effects of temperature in a stochastic Hopfield network containing one or two strong attractors, plus a number of other random patterns. We show that at low temperature it is still possible to converge fairly close to a nearby strong attractor and that in a network containing two strong attractors the stronger of the two will maintain its dominance in the network over the other. We also present results from our investigation that confirms the characteristics of the energy landscape in classical Hopfield network theory, how this landscape changes with respect to spin glass states when implemented with strong attractors, and the implications this has in the emerging theories and self-attachment therapy.

Acknowledgements

I would like to acknowledge the following people for their assistance during the course of this project:

- My supervisor Abbas Edalat for his time, assistance and feedback throughout the project. Also thank you to Aldo Faisal for his feedback at the interim report stage.
- David Mooney for SpLD tutoring and assistance during the write up of this report.
- Francesca Toni, Margaret Cunningham and Duncan Gillies for their pastoral support over the past 4 years.
- Andreja, Benedict and David for listening whenever I'd start to talk about this project, and for providing useful discussion and feedback related to it.
- My family for their ongoing support.

Thank you, one and all.

Contents

Chapter 1

Introduction

1.1 Motivation

The prevalence of mental health disorders is becoming a growing issue for modern day society. Various estimates propose that over a quarter of us will experience issues with our mental health at some point in our lifetime [18], reducing our quality of life and having an impact on our communities.

A new therapeutic technique call self-attachment therapy is currently under development, lead by Prof. Abbas Edalat [9]. It is based on attachment theory and is aimed at those who are affected psychologically because they did not form a secure attachment during early life with their primary caregiver.

This project aims to support the mathematical protocol/proof for self-attachment therapy. We explore using simulations of the Hopfield network, particularly the stochastic Hopfield network, to model attachment theory by studying the emerging theory of strong attractors [10] within these networks. We build on previous studies such as [10], [16] and [13] by investigating the effects of the concept of temperature in the stochastic Hopfield network. We also study the energy landscape of the Hopfield network for classical theory and strong attractor theory.

1.2 Contributions

We were able to make the following contributions during the course of this project:

- Support results already obtained in the research carried out by [16] and [13], particularly in the investigation of modelling strong attractors.
- An investigation into how the temperature of a stochastic Hopfield network affects convergence towards strong attractors in networks containing one or two strong attractors.
  - The experiment containing one strong attractor demonstrates that at low temperature, the strong attractor is still a relatively stable attractor and is able to mostly converge towards the strong attractor or stay relatively within its basin of attraction. However, as temperature increases, it becomes less stable, eventually reaching a point where each neuron has a 50% chance of being in the firing state. Past a certain point, increasing the temperature

does not make the strong attractor any more unstable, at least up to temperature = degree of strong attractor.

- The experiment containing two strong attractors demonstrates that at low temperatures, a second strong attractor with higher degree can be introduced to lessen the influence of an undesired first strong attractor. As temperature is increased, the strong attractor with the higher degree is less unstable than the other strong attractor, however they both reach a point where each neuron has 50% chance of being in the firing state (i.e. half the neurons match the strong attractor pattern). Again past a certain point increasing the temperature does not make the strong attractor any more unstable, at least up to temperature = degree of strongest attractor.

- An implementation of the energy landscape of a Hopfield network to check theoretical results in networks that do and do not contain strong attractors.
  - We demonstrate that the behaviour of mixture states for classical Hopfield network theory holds, then introduce a strong attractor to the network. We observe that for a strong attractor of low degree, mixture states have higher average energy than that of the strong attractor, with some margin of error. However as an attractor becomes stronger, we see it becomes more likely that the mixture states actually have lower energy than that of the strong attractor, and that this behaviour mostly happens within phase B of the phase diagram. However the methodology used in this experiment is not believed to be entirely suitable for the problem.
  - We use a heuristic method to investigate the behaviour of spin glass states for classical and strong attractor Hopfield network theory. In a network containing no strong attractors we show that within region B of the phase diagram spin glass states become more likely to be the lowest energy minima in a network, which is in line with the theorems. When we introduce a strong attractor to the network we stop observing spin glass states as the lowest energy minima and as the degree of the strong attractor increases it becomes likely at low alpha that no spin glass states are detected using the heuristic method.

Chapter 2

Background

2.1 Attachment Theory

Attachment theory evolved from initial work by John Bowlby [1]. Many of the concepts were defined by Mary Ainsworth in the 1960s and 70s [2]. It is based on the bond that develops between a child and its primary caregiver, usually the mother.

The attachment type a child develops can be described as either secure or insecure. The insecure type can be split into three sub-groups. For the purpose of our experiments, we are only concerned with differentiating between a secure and an insecure state (i.e. two reasonably different states).

2.1.1 Strange Situation Protocol

The strange situation protocol [2] is a test developed to try and identify the attachment type between a child and its caregiver. The key data for distinguishing between secure and insecure attachment is in how the caregiver and infant interact during reunion episodes (episodes 5 and 8), rather than the reaction from the infant upon the caregiver leaving. Other data, such as how the child interacts with the stranger, can be used to further support secure attachment or aid in identifying the type of insecure attachment.

The strange situation protocol consists of 8 separate, sequential episodes [2]:

1. Infant and caregiver are introduced to the room where the experiment will take place. The room contains a one-way mirror from which the researchers can observe actions and reactions during each episode of the experiment. (Caregiver, infant and a researcher are in the room.)
2. The researcher leaves the infant and caregiver in the room. (Caregiver and infant are in the room.)
3. A stranger enters the room. (Caregiver, infant and stranger are in the room.)
4. The caregiver leaves the room. (Infant and stranger are in the room.)
5. The caregiver enters the room while the stranger leaves. (Caregiver and infant are in the room.)
6. The caregiver leaves the room. (Infant is in the room.)
7. The stranger enters the room. (Infant and stranger are in the room.)

8. The caregiver enters the room while the stranger leaves. (Caregiver and infant are in the room.)

2.1.2 Secure Attachment

Securely attached infants have managed to form a strong bond with their primary caregiver. They are happy to explore a new environment in the presence of their caregiver, assured that the caregiver will respond to their needs appropriately.

The caregiver is able to reassure the infant during a reunion episode, allowing it to quickly resume play and exploration.

2.1.3 Insecure Attachment

There are three primary groups of insecure attachment in infants:

Avoidant

The primary caregiver doesn't respond, or gives little response, to the child when it is in distress. The child learns that the caregiver will not respond to its needs, and so treats the caregiver in the same manner as a stranger.

During the strange procedure protocol, the child may not appear distressed upon the departure of the caregiver, and ignores the caregiver once they return. The stranger and caregiver are treated in similar ways by the child.

Ambivalent/Resistant

The primary caregiver does not give consistent responses to the child when it is in distressed. For example, the caregiver may alternate between comforting the distressed child on one occasion, but then completely ignore it in a subsequent episode. As the child can not reasonably anticipate the response from the caregiver, it feels anxious and its behaviour is affected.

During the strange procedure protocol, the child becomes distressed upon the caregiver leaving. However, when reunited, the caregiver is unable to reassure the child, who is resisting and unwilling to return to exploration. The child is not easily reassured by the stranger either.

Disorganised

The primary caregiver responds to the child in distress with negative actions, such as behaviours that frighten or harm the infant.

During the strange situation protocol, the infant may react incoherently upon reunion with the caregiver. They may suddenly freeze or fall over, and if they approach the caregiver at all, they may do so in a strange manner, such as with their back turned.

This type of insecure attachment was observed in later experiments [15], but not in the original studies.

2.2 Artificial Neural Networks

The mathematical models for artificial neural networks were developed through studying the structure and behaviour of biological neural networks. However, they only served as inspiration. Biological neural networks are many, many times more complex than the artificial models, due to our current understanding in the field and the computational complexity needed to simulate even these 'simple' models.

The Hopfield network is one of the closest mathematical models we have to that of the human brain. It is still a simplification compared to the complexity of human cognition, however it is useful in investigating memory-associated tasks [12].

2.3 Hopfield Networks

Theory relevant to this investigation is described in [12] and explained below.

2.3.1 Phases

There are two main phases of a Hopfield network: the learning phase and the testing phase.

In the learning phase, the patterns to be recalled (retrieved) are stored in the network using one of the learning methods available to the system.

In the testing phase, the network is presented with a pattern to use as its initial state. From this initial state the network repeatedly updates itself until it assumes the state of one of the stored patterns or it is told to stop updating (i.e. an updating limit was set beforehand and then reached).

2.3.2 Modelling the Neuron and Network

A Hopfield network is an undirected graph consisting of N neurons, where each neuron can take on the value $+1$ (firing) or -1 (not firing). Each neuron is connected by an edge to every other neuron in the network (i.e. the graph is complete). Every edge has a weight, denoted w_{ij}, where i and j indicate the pair of neurons connected by the edge. The weight values are stored in an N by N matrix known as the weight matrix, in which initially all values are set to 0.

For quick reference, in the equations listed in this section:

- N is the number of neurons in the network.
- x_i is the value of neuron i. As stated above, $x_i = \pm 1$.
- p is the number of patterns to be stored or stored in the network.
- ϵ_i^μ means the value of neuron i in the pattern ϵ^μ.
- w_{ij} is the weight of the edge between neurons i and j.

2.3.3 Training the Network

There are several techniques that can be used to train the network. Three of the most common methods are described below:

Hebbian Rule

For a Hopfield network to learn p patterns using the Hebbian rule:

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{p} \epsilon_i^\mu \epsilon_j^\mu$$

If the neurons i and j in the pattern μ are equal in value (both positive or both negative), then their weight w_{ij} will be greater. However if i and j are different values (one positive and one negative) then the weight between them will be lower.

Pseudo-Inverse Rule

The Pseudo-Inverse rule makes use of an overlap matrix, which contains information on all patterns being stored in the system. As such, all patterns have to be introduced to the system at the same time and new patterns cannot be introduced later on. However, its storage capacity is much greater than that of the other two learning rules.

$$w_{ij} = \frac{1}{N} \sum_{\mu\nu} \epsilon_i^\mu (Q^{-1}) \epsilon_j^\nu$$

Where Q is the overlap matrix:

$$Q_{\mu\nu} = \sum_i \epsilon_i^\mu \epsilon_i^\nu$$

Storkey Rule

Defined by Storkey in 1997 [22], the Storkey rule is as follows:

To learn a new pattern ϵ^ν:

$$w_{ij}^\nu = w_{ij}^{\nu-1} + \frac{1}{N}\epsilon_i^\nu \epsilon_j^\nu - \frac{1}{N}\epsilon_i^\nu h_{ji}^\nu - \frac{1}{N} h_{ij}^\nu \epsilon_j^\nu$$

where

$$h_{ij}^\mu = \sum_{k=1, k\neq i,j}^{n} w_{ik}^{\mu-1} \epsilon_k^\mu$$

is known as a local field.

Comparison Between Learning Rules

Learning rules can have the following (desirable, for our purposes) properties:

- Local: Weights between neurons update based only on the information contained in the neurons either side of the edge being evaluated.
- Incremental: Learning a new pattern does not require using information from previously learned patterns.

A learning rule that is both local and incremental is more biologically plausible than a rule that lacks these one or both of these features.

The Hebbian and Storkey learning rules are local, as they update weights based on the information contained in the neurons each side of the connection being evaluated. The Pseudo-Inverse rule is not local as it makes use of an additional matrix (the overlap matrix).

The Hebbian and Storkey learning rules are incremental, as they do not require information on other patterns that have already been taught to the system. The Pseudo-Inverse is not incremental, again due to the use of the overlap matrix (which contains information on other patterns).

2.3.4 Updating

The following rule is used for updating neurons in the network:

$$x_i = sgn(\sum_j w_{ij}x_j + b_i)$$

Where b_i is the bias (also called threshold). However, we consider the bias to be 0 for the purpose of this project, and hence the rule becomes:

$$x_i = sgn(\sum_j w_{ij}x_j)$$

Neurons can be updated synchronously or asynchronously. In synchronous updating, all neurons are updated simultaneously during each time step. However in asynchronous updating, only one (randomly chosen) neuron gets updated during each unit of time.

Asynchronous updating is more suitable for modelling biological systems, as synchronous updating would require a global clock to keep all neurons in sync, which is something not found in systems such as the brain.

2.3.5 Basins of Attraction

A basin of attraction is the set of all (initial) states that will eventually converge towards a specific attractor state if the network is repeatedly updated.

The size of a basin of attraction can be measured using the Storkey-Valabregue technique [23], which can be calculated using the following steps:

1. Initially n = 1
2. Choose an initial fixed point corresponding to a stored pattern μ
3. Choose some initial normalised Hamming radius $r = r_0$
4. Let the set A be all the states Hamming-distant nr from the fixed point
5. Sample 100 states from A
6. Calculate how many of these states are attracted to the fixed point. Denote this number $t_\mu(r)$
7. Increment n by a suitable amount and repeat from (3)
8. Repeat for each stored pattern μ

2.3.6 Stability and Capacity

A neuron is in a stable state if applying the update rule to the neuron will not cause it to change state, i.e.:

$$sgn(h_i^\mu) = \epsilon_i^\mu$$

where

$$h_i^\mu = sgn(\sum_j w_{ij}\epsilon_j^\mu)$$

We can substitute the following:

$$w_{ij} = \sum_\nu \epsilon_i^\nu \epsilon_j^\nu$$

And then use the Hebbian rule to get the following equation:

$$h_i^\mu = \frac{1}{N}\sum_j \sum_\nu \epsilon_i^\nu \epsilon_j^\nu \epsilon_j^\mu$$

The contribution of neuron i is isolated and we get:

$$h_i^\mu = \epsilon_i^\nu + \frac{1}{N}\sum_j \sum_{\nu \neq \mu} \epsilon_i^\nu \epsilon_j^\nu \epsilon_j^\mu$$

2.3.7 Load Parameter

The load parameter of a network is defined as follows:

$$\alpha = \frac{p}{N}$$

Where p is the number of patterns stored in the network and N is the number of neurons in the network.

As derived and demonstrated in [12], the critical value for α, found by solving the capacity equations, is $\alpha_c \approx 0.138$. Non-trivial solutions disappear after this point.

Note that in this project when strong attractors are involved, p is interpreted to be the number of unique patterns in the network. So if a network was trained with 4 random patterns and a strong attractor of degree 8, $p = 5$.

2.3.8 Energy Landscape

To calculate the Lyapunov (energy function) of the Hopfield network, we use the following equation:

$$E = -\frac{1}{2}\sum_{i,j=1}^{N} w_{ij}x_i x_j$$

The value of the energy function decreases each time the update function is applied to the system, eventually reaching a local minimum (attractor).

2.3.9 Spurious States

A spurious state is a pattern that the network can converge to during the testing phase, despite it not being one of the patterns originally stored in the network during the testing phase. In contrast, a retrieval state is one of the patterns stored in the network during training.

There is a risk of a spurious state occurring when there is a linear combination of an odd number of patterns stored in the network. These are called mixture states. For example, for 3 stored patterns:

$$\epsilon_i^{mixture} = sgn(\epsilon_i^{\mu_1} + \epsilon_i^{\mu_2} + \epsilon_i^{\mu_3})$$

Another type of spurious state that can occur in a Hopfield network is a spin glass state [3]. Unlike mixture states, there is no obvious reasoning as to why they appear. Determining if a stable state is a spin glass state is done by a process of elimination (if it's stable and not a retrieval state, mixture state or retrieval state, then it's a spin glass state).

While the chance of ending up stuck in a spurious state is small, it is still an issue with this model. However, the stochastic Hopfield model (due to the introduction of the concept of temperature) is able to get out of these states.

It is also worth noting that the inverse of an attractor state is also stored in the network as a spurious state. These are sometimes referred to as the reversed states.

2.4 Strong Attractors

A strong attractor occurs when a Hopfield network is trained with multiple occurrences of the same pattern [10]. The number of times the network was trained with this specific pattern is referred to as the degree of the strong attractor. This is meant to mimic being exposed to an identical situation (pattern) multiple times, forming a 'strong' memory of the specific pattern.

Some studies mentioned in this project refer to strong attractors as super attractors. For clarification, they refer to the same concept.

2.5 Stochastic Hopfield Networks

Hopfield networks are mostly deterministic in operation, which is unlikely to be the case in a biological system. Therefore we now investigate the addition of stochastic behaviour to the Hopfield network.

2.5.1 Temperature

It is important to note that the temperature of the system bares no resemblance to the physical temperature that the brain or a physical system may have, rather it is an indication of how likely it is that the deterministic rule of the system will be broken.

We define the temperature T as:

$$\beta \equiv \frac{1}{T}$$

If $T = 0$, then this is the equivalent of the deterministic Hopfield network (as there is no temperature affecting the system).

2.5.2 Updating

The probability of a neuron i being in the firing state (+1) is given by:

$$P(x_i = +1) = \frac{1}{1 + e^{-2\beta z_i}}$$

Where z_i is the neuron's input.

2.6 Mean Field Theory

2.6.1 Classical

From mean field theory:

$$m = \tanh(\beta m)$$

But m can be rewritten as:

$$m = \frac{\langle x_i \rangle}{\epsilon_i^\mu} = \text{Prob(bit } i \text{ is correct)} - \text{Prob(bit } i \text{ is incorrect)}$$

Using m we can calculate the average number of bits in the pattern that are correct:

$$\langle N_{correct} \rangle = \frac{1}{2} N(1 + m)$$

The value of m varies with temperature (due to β). At zero or low temperature, we would expect $N_{correct} = N$ (i.e. all bits are correct). At high temperature we would expect $N_{correct} = \frac{1}{2}N$ (i.e. bits have randomly been assigned a binary value, so an individual bit is correct half of the time).

2.6.2 Strong Attractors

We now replace m with dm [8]:

$$\langle N_{correct} \rangle = \frac{1}{2} N(1 + dm)$$

$$dm = \frac{\langle x_i \rangle}{\epsilon_i^\mu} = \text{Prob(bit } i \text{ is correct)} - \text{Prob(bit } i \text{ is incorrect)}$$

It is proposed [8] that for a network containing one strong attractor pattern that this pattern can be retrieved up until:

$$\frac{p}{N} \approx 0.13 \times d^2$$

2.6.3 Phase Diagram

The phase diagram of the Hopfield model is demonstrated below:

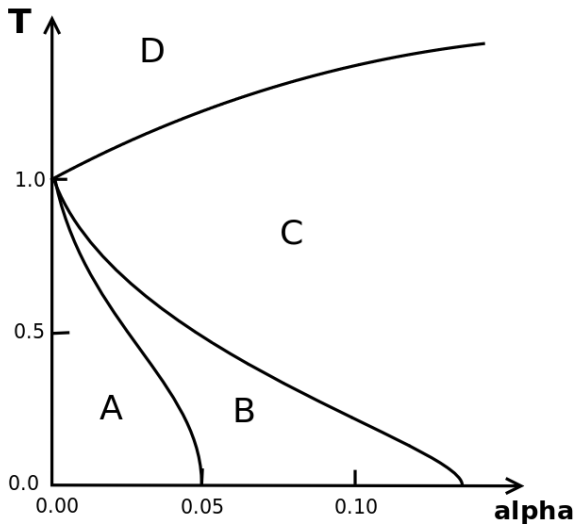

Figure 2.1: The phase diagram obtained in [6]

There are 4 regions in the alpha-temperature plain, regions A, B, C and D. In regions A and B the memory of the network is rather good, but in regions C and D it is not.

At zero temperature region A covers approximately $0.00 \leq \alpha \leq 0.05$ while region B covers approximately $0.05 < \alpha \leq \alpha_{critical}$ ($0.05 < \alpha \leq 0.138$).

[12] provides a 'highly idealised' illustration of the energy landscape in each of these regions. As we are interested primarily in regions A and B, we recreate them below:

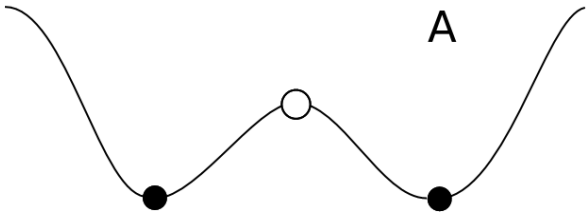

Figure 2.2: Landscape A as visualised in [12]

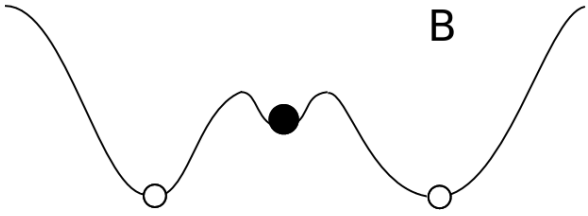

Figure 2.3: Landscape B as visualised in [12]

Where the black circles represent the retrieval states and the white circles represent the spin glass states.

2.7 Previous Studies of Hopfield Networks

2.7.1 Attachment Theory

Modelling Attachment Types with Hopfield Networks

Experiments performed by [16] form the basis for the research carried out in this project. He investigates the use of the Hopfield model and clusters of patterns to model Attachment Theory.

The three main findings were:

- A technique for encoding Gaussian distributed observations to a pattern format storable by a Hopfield network.
- A lower standard deviation usually results in fewer attractors, but with a larger size.
- Instability in one cluster can affect the stability of another cluster.

Attractor Neural Networks for Modelling Associative Memory

In their paper [13] the authors build on the work of Mancinelli [16] by introducing the use of super attractors in the Hopfield network. Again, this project builds on experiments undertaken during their research, particularly on the use of strong attractors.

Some of their findings were:

- A confirmation of the results obtained by Mancinelli through a reimplementation of the original experiments.
- Demonstration of the Hopfield network being capable of performing pattern recognition tasks.
- Extensions involving the implementation of Boltzmann Machines.
- Experiments involving super attractors, showing that they generally have a larger basin size, resulting in a greater chance of patterns converging towards them.

2.8 Other Related Works

2.8.1 Mental Health

Computer Simulations of Neural Information Processing and the Schizophrenia-Mania Dichotomy

In his paper [21], Hoffman uses the stochastic Hopfield network to investigate how neural functioning breaks down when its actions are restricted by certain perturbations. There is a particular focus on how the disturbances result in two states, the 'schizophreniclike' state and the 'maniclike' state, and how this relates back to studies of schizophrenia and mania in psychiatry.

2.8.2 Attachment Theory

Implementing a Theory of Attachment: A Simulation of the Strange Situation with Autonomous Agents

Dean Petters [19] describes and implements a number of different architectures that are used to represent the theories of infant attachment using autonomous agent techniques. The paper highlights the many limitations of the architectures, mainly in the ontology of the simulation, but nevertheless demonstrates how theories of attachment can be translated from written material to mathematical models.

Strange Carers: Robots as Attachment Figures and Aids to Parenting

Petters, Schönbrodt and Waters [20] consider robots as carers and how attachments and bonds would play a role. One of their main conclusions is that human carers are more likely to form attachment or attachment-like bonds than a robot counterpart, and that while infants in the care of a robot will form expectations relating to their robot carer, it is the quality of the care obtained from human carers (e.g. responsiveness) that is more likely to make an impact on how the infant develops.

Chapter 3

Experiment Methodology

Models of the standard Hopfield network and stochastic Hopfield network were built in order to run the experiments. This chapter explains the concepts we wish to explore, the possible approaches to exploring the problems and the algorithms describing the final methodology chosen and used in our experiments.

3.1 Standard Hopfield Network Experiments

The methodology for experimenting with strong attractors in a standard Hopfield network is that used in [13] and repeated below for reference:

3.1.1 One Strong Attractor

1. Set the number of neurons in the network, N.
2. Generate a random pattern, p_{super}.
3. Generate a number of random patterns, $\vec{p}_{random}$, where the Hamming distance between p_{super} and each element in $\vec{p}_{random}$ is between 25% and 75%.
4. Set the degree, d, to be used this iteration. Then train the network using d instances of p_{super} ($\vec{p}^{\,d}_{super}$) and $\vec{p}_{random}$.
5. Measure the basin size for p_{super}.
6. Repeat from (2) for various values of d.

3.1.2 Two Strong Attractors

1. Set the number of neurons in the network, N.
2. Set the degree d_{origin} for p_{origin}.
3. Generate a random pattern, p_{origin}.
4. Generate a random pattern, p_{new}, where the Hamming distance between p_{origin} and p_{new} is between 25% and 75%.
5. Generate a number of random patterns, $\vec{p}_{random}$, where the Hamming distance between p_{origin} and each element in $\vec{p}_{random}$ is between 25% and 75%.

6. Set the degree d_{new} for p_{new}. Then train the network using $\vec{p}_{random}$, d_{origin} instances of p_{origin} and d_{new} instances of p_{new}.
7. Measure the basin size for p_{origin} and p_{new}.
8. Repeat from (3) for various values of d_{new}.

3.2 Stochastic Hopfield Network Experiments

3.2.1 Developing a Methodology for Investigating Strong Attractors

Unlike a deterministic Hopfield network, we will not be able to find stable states in the stochastic Hopfield network, especially as the temperature increases. Therefore we cannot use previous methodologies to measure the basin size of attractors in these networks.

We need to decide on how to measure how the state of the network changes as the update rule is applied. To do this we build a probability distribution based on applications of the update rule.

In order to build this probability distribution first we must choose which state to start the network in. We make use of two types of start location within our experiments: within 50% hamming distance of the strong attractor being measured, or within any valid state of the network.

Initially we apply the update rule for a set number of iterations that we ignore, in order to give the network a chance to converge towards a strong attractor state. Then we start building the probability distribution by continuing to apply the update rule to the network, but after every application of the update rule we increment a counter for each neuron that is firing.

For example, we initially apply 10000 iterations of the update rule that we ignore. We then apply an additional 5000 iterations of the update rule that we use to build the probability distribution. At the end of the computation, we observe that neuron 3 was firing for 3000 of 5000 iterations (it was firing 60% of the time).

Pseudocode detailing the computational steps for the one strong attractor experiment and the two strong attractors experiment is listed later on in this chapter.

Start Locations in the Stochastic Experiments

As mentioned above, we run the experiments with two differing types of start location, which are:

- S1: The start location is within 50% hamming distance from the strong attractor.
- S2: The start location is any valid network state.

Multiple Strong Attractors and How They Influence Each Other

We are interested in what happens in networks containing multiple strong attractors and hence we use a similar but adapted methodology to explore what happens in networks containing two strong attractors. Self-attachment therapy is based on the

theory of retraining to 'overpower' an old strong attractor (undesired attachment type) with a new strong attractor (desired attachment type), which is something we aim to mimic in the methodology for this experiment.

3.2.2 One Strong Attractor

1. Set the number of neurons in the network, `N`.
2. Set the number of random patterns in the network, `numberRandomPatterns`.
3. Set the strong attractor pattern, `P`.
4. Set degree of the strong attractor, `degree`.
5. Set the number of iterations, `iterations`.
6. Set the number of initial ignored iterations, `ignoreIterations`, to (10∗`iterations`).
7. Set `temperature` of the system to 0.
8. Initialise the training set, `trainingSet`.
9. Add `numberRandomPatterns` random patterns to the `trainingSet`, which all should be between 25% and 75% hamming distance from the strong attractor pattern `P`.
10. Add `degree` instances of `P` to the `trainingSet`.
11. Train the network using `trainingSet`.
12. Set the start location/pattern, `startLocation`, depending on whether this is an S1 or S2 experiment.
13. From the start location `startLocation` in the network, execute the converge function for `ignoreIterations`. The current state of the network is referred to as `currentState`.
14. Initialise an array, `firingArray`, the same size as `N`. This will be used to count which neurons are firing at the end of each measured iteration.
15. Continue executing the converge function `iterations` times. After each iteration, increment the appropriate values in `firingArray` depending on whether each neuron is firing or not firing. e.g. if `currentState[3]` is firing after the update rule has been applied, increment `firingArray[3]`.
16. Once completed, save any data needed for analysis.
17. Increment `temperature` by (`degree`/10), and continue again from step 12, until `temperature` is more than `degree` (i.e. last iteration has `temperature` = `degree`).

The experiment is then run several times and the results averaged, ready for analysis.

3.2.3 Two Strong Attractors

1. Set the number of neurons in the network, `N`.
2. Set the number of random patterns in the network, `numberRandomPatterns`.
3. Set the first strong attractor pattern, `POne`.
4. Set the second strong attractor pattern, `PTwo`. This should be between hamming distance 25% and 75% from the first strong attractor pattern `POne`.
5. Set the degree of the first strong attractor, `degreeOne`.
6. Set the degree of the second strong attractor, `degreeTwo`.
7. Set the number of iterations, `iterations`.
8. Set the number of initial ignored iterations, `ignoreIterations`, to (10∗`iterations`).
9. Set `temperature` of the system to 0.
10. Initialise the training set, `trainingSet`.
11. Add `numberRandomPatterns` random patterns to the `trainingSet`, which all should be between 25% and 75% hamming distance from the strong attractor pattern `POne`.
12. Add `degreeOne` instances of `POne` and `degreeTwo` instances of `PTwo` to the `trainingSet`.
13. Train the network using `trainingSet`.
14. Set the start location/pattern for each strong attactor, `startOne` and `startTwo`, depending on whether this is an S1 or S2 experiment. In the S1 experiments, `startOne` is within 50% hamming distance of `POne` and `startTwo` is within 50% hamming distance of `PTwo`.
15. Measure the firing distribution of each strong attractor seperately.
16. From the start location for the current strong attractor being measured, execute the converge function for `ignoreIterations`. The current state of the network is referred to as `currentState`.
17. Initialise an array, `firingArray`, the same size as `N`. This will be used to count which neurons are firing at the end of each measured iteration.
18. Continue executing the converge function `iterations` times. After each iteration, increment the appropriate values in `firingArray` depending on whether each neuron is firing or not firing. e.g. if `currentState[3]` is firing after the update rule has been applied, increment `firingArray[3]`.
19. Once completed, save any data needed for analysis, and go back to step 15 if there is another strong attractor to be measured.
20. Increment `temperature` by (`degree`/10), and continue again from step 14, until `temperature` is more than `degree` (i.e. last iteration has `temperature` = `degree`).

3.2.4 Measurement of Correlation Between Probability Distributions

Naive Measurement

The correlation between the attractor pattern and a probability distribution can be measured using the sum of squared differences.

First we calculate what the probability distribution would be if for all iterations the network remained in the attractor state. We call this probability distribution $P_{attractor}$.

Then we take a calculated probability distribution from the same experiment that we wish to compare to the attractor state. We call this probability distribution P_{new}.

The sum of squared differences is then calculated as so:

$$D = \sum_{i=0}^{n} (P_{new}(i) - P_{attractor}(i))^2$$

Where D is the sum of squared differences and n is the number of neurons in the network.

As the minimum probability a value in the probability distribution can take is 0 and the maximum value it can take is 1, we can conclude that the maximum difference between these distributions is equal to the number of neurons in the network.

Similarly, if in the probability distribution the probability of most neurons is roughly 0.5 (i.e. "there is equal chance of the neuron being in the firing state or in the off state"), then we would expect the value of D to be roughly $0.5^2 * n = 0.25n$

Measurement Based on Mean Field Theory

As explained in the background section on stochastic Hopfield networks:

$$\langle N_{correct} \rangle = \frac{1}{2} N(1 + m)$$

$$m = \frac{\langle x_i \rangle}{\epsilon_i^\mu} = \text{Prob(bit } i \text{ is correct)} - \text{Prob(bit } i \text{ is incorrect)}$$

From the probability distribution we plan to construct, we can calculate Prob(bit i is correct) and Prob(bit i is incorrect), so therefore we can calculate m and $\langle N_{correct} \rangle$.

As temperature in the network increases, we would expect the value of m and $\langle N_{correct} \rangle$ to decrease. m would eventually equal 0 at high temperature (assuming probability of bit being correct and bit being incorrect become equal), meaning that we have $\langle N_{correct} \rangle = \frac{1}{2} N$ at high temperature.

This measurement is referred to as NCorrect in this report.

3.3 Verification of Energy Landscapes

3.3.1 Mixture States

Mixture states occur when the network is trained with an odd number of patterns, and in regions A and B of the phase diagram the stable mixture states should have higher energy than the retrieval states [12].

In order to investigate this theory, we want to calculate the stable mixture states and their average energies for networks in various regions of the phase diagram, particularly regions A and B. By increasing or decreasing the number of patterns we use to train the network we can move between different areas of the phase diagram (at zero temperature). When investigating the energy landscape with strong attractors, the degree of the strong attractor is fixed but the number of random patterns in the network is varied in order to move between regions of the phase diagram.

We also want to check if the lowest minima in the network is an attractor state or a mixture state, as this also gives an indication of the structure of the energy landscape.

3.3.2 Mixture States Investigation Algorithm

1. Set the number of neurons in the network, `N`.
2. Set the strong attractor pattern, `P`.
3. Set the degree of the strong attractor, `degree`.
4. Set the number of random patterns, `numberRandomPatterns`, to 1 if `degree` is even, to 2 if `degree` is odd (so that the overall number of patterns the network will be trained with is an odd number and therefore mixture states occur).
5. Initialise a new training set for the network, `trainingSet`.
6. Add `degree` instances of `P` to the `trainingSet`.
7. Add `numberRandomPatterns` random patterns to the `trainingSet`, where each random pattern is hamming distance 25% to 75% away from `P`.
8. Train the network with the `trainingSet`.
9. Calculate all of the mixture states using the patterns (including duplicates) in `trainingSet`, save these mixture states in set `mixtureStates`.
10. Check each pattern in `mixtureStates` for stability. Remove patterns from the set that are not stable.
11. Calculate the average energy of all the patterns in `mixtureStates`, `avgEnergyMixture`.
12. Calculate the average energy of all the retrieval states (i.e. all the random patterns and one instance of `P`), `avgEnergyRetrieval`.
13. Calculate the energy of `P`, `energyStrong`.
14. Compare `avgEnergyMixture` to `avgEnergyRetrieval` and `energyStrong`.
15. Measure the energy of all retrieval states and mixture states, to see which set the lowest observed energy minima belongs to.

16. Increment `numberRandomPatterns` by 2 and go back to step 5, but stop looping once `numberRandomPatterns` is 17 or greater.

3.3.3 Spin Glass States

Spin glass states occur in a Hopfield network for high enough p, but it isn't known as to how exactly they appear. The theories state that in region A the spin glass states have higher energy than the retrieval states, but in region B (and C) the spin glass states have lower energy than the retrieval states [12].

2^N Method

Calculating all of the spin glass states in a Hopfield network has $2^{Neurons}$ complexity. To put this in perspective, running an experiment with $N + 1$ neurons could take up to twice as long as an experiment with N neurons.

Spin glass states only occur for large enough p. When running the experiments, they only appeared in initial implementations when $p \geq 4$. This would mean to explore even just the upper edge of region A ($0 \leq \alpha \leq 0.05$) we need at least:

$$\frac{4}{0.05} = neurons = 80$$

And to explore the upper edge of region B ($0.05 \leq \alpha \leq 0.138$) we need at least:

$$\frac{4}{0.138} = neurons = 29$$

If we wanted to investigate a network that contained mixture states as well as spin glass states we need to train the network with an odd number of patterns, which would make the minimum p required 5. This would increase the minimum neuron requirements to:

$$\text{Region A} = \frac{5}{0.05} = neurons = 100$$

$$\text{Region B} = \frac{5}{0.138} = neurons = 37$$

Although experiments were initially written using this method, it was impractical to run them at even the smallest scale needed for our experiments. Hence a heuristic method was developed.

Spin Glass States Investigation, Heuristic Method

The heuristic method does not find all spin glass states in the network, but finds a subset of them for us to investigate.

1. Set the number of neurons in the network, `N`.
2. Set the strong attractor pattern, `P`.
3. Set the degree of the strong attractor, `degree`.
4. Set the number of random patterns, `numberRandomPatterns`, to 1.
5. Initialise a new training set for the network, `trainingSet`.

6. Add `degree` instances of `P` to the `trainingSet`.
7. Add `numberRandomPatterns` random patterns to the `trainingSet`, where each random pattern is hamming distance 25% to 75% away from `P`.
8. Train the network with the `trainingSet`.
9. Calculate all of the mixture states using the patterns (including duplicates) in `trainingSet`, save these mixture states in set `mixtureStates`.
10. Calculate all of the reverse states in the network and save them in the set `reverseStates`.
11. Combine `trainingSet`, `reverseStates` and `mixtureStates` into the known potential stable attractor states set, `knownAttractors`.
12. Initialise a new set, `spinGlassStates`.
13. Generate $\texttt{N} * 200$ random patterns, `testSet`.
14. For each pattern in `testSet`, start the network in the state specified by the pattern being investigated, and apply the update rule until the network reaches a stable state. If this stable state is not in `knownAttractors`, add it to `spinGlassStates` (if not already in it).
15. Calculate the average energy of all the patterns in `spinGlassStates`, `avgEnergySpinGlass`.
16. Calculate the average energy of all the retrieval states (i.e. all the random patterns and one instance of `P`), `avgEnergyRetrieval`.
17. Calculate the energy of `P`, `energyStrong`.
18. Compare `avgEnergySpinGlass` to `avgEnergyRetrieval` and `energyStrong`.
19. Measure the energy of all retrieval states and spin glass states, to see which set the lowest observed energy minima belongs to.
20. Increment `numberRandomPatterns` by 2 and go back to step 5, but stop looping once `numberRandomPatterns` is 17 or greater.

3.4 Data Processing

The experiments produce a lot of data that we need to process at some stage. It is unrealistic to keep all of this in memory until the end of all the calculations due to the total processing time of all runs. Outside influences (e.g. power cut, computer reboot) can ruin an experiment that has been running for any significant amount of time. Also in order to run the experiments in parallel effectively (e.g. by using HTCondor) we need to be able to save the data so that we can retrieve results from multiple locations.

Saving data to the file system is also useful as running the experiment and processing the results can be done as separate functions or programs. If there are changes made to the way the data is processed but not to how it is stored then the experiment does not have to be run from the very beginning.

3.5 Predictions

We have a number of predictions relating to the energy landscape that we wish to investigate:

- Our first theory is that the quality of the landscapes should not change when implemented with strong attractors.
- Our second theory is that the attractors in landscapes A and B represent secure attachment states, while the ripples (spin glass states) represent the insecure attachment types.
- Our final theory is that landscape A represents a securely attached person, as by increasing temperature it is easy to get out of the undesirable spurious state (insecure attachment) and into the desired attractor state (secure attachment). In contrast, landscape B represents an insecurely attached person, as increasing temperature won't help much in getting out of the spurious state.

Chapter 4

Technologies

The various technological components used to build our implementation are briefly covered in this section.

4.1 Java

Java [17] was the programming language used to implement the models and methodologies for our experiments.

4.1.1 Advantages and Disadvantages

As Java programs are run within the Java Virtual Machine on a host computer, they can run slower and have more memory disadvantages when compared to native programs written in C/C++. However a Java program can be run on any computer with a JVM installed without rewriting or recompiling for a specific architecture. This is an advantage when using the Condor pool (see below) as we don't have to tweak or recompile for each type of architecture present in the pool of machines as we might have to if using C/C++ (faster, but would have to tweak/compile per computer architecture type).

4.2 HTCondor

HTCondor [7] is a software framework for distributed computation and there is a version of it deployed within the ICLDoC labs for academic use.

As the experiments needed to be run in batches of hundreds or thousands and then averaged, executing them sequentially is time consuming, potentially taking days to get any meaningful results. This is especially problematic if a bug or problem is detected and the experiments need to be recompiled and run from the beginning. Therefore we decided to make use of the Condor framework and took its use into account when writing the implementation.

4.3 Git

Git [11] is a distributed version control system. We used this to keep track of code changes and to keep changes synced between different machines (e.g. lab machines and home machines).

It is standard procedure in software development to create backups of a project in case something happens that makes the copy being edited unavailable (e.g. server downtime, file corruption, machine out of service) or if earlier code needs to be recovered (e.g. specifications reverted, changes made that breaks functionality that previously existed). Using Git meant that copies of the code base were stored on several machines, essentially serving as backups without having to specifically dedicate time to creating and maintaining backups.

4.4 JUnit

JUnit [14] is a unit testing framework for Java.

It is very important that we confirm the models and methodologies behave as we expect them to. As testing manually is a very time expensive process, we write unit tests to be able to quickly check that changes to the code have not broken any existing functionality.

Unit tests are very good for checking whether equations are evaluating to the correct answer given certain inputs, however unit testing cannot test everything so manual testing is still required.

Chapter 5

Experiment Results

In this chapter we list the results from the experiments we carried out. A more in depth discussion of these results is conducted in the Evaluation and Conclusion sections. Tables of results for the Hopfield and Stochastic experiments are included as at the end of this report as an appendix.

5.1 Hopfield Experiments

5.1.1 One Strong Attractor

Configuration: 100 neurons, 16 random patterns, experiment run 800 times.

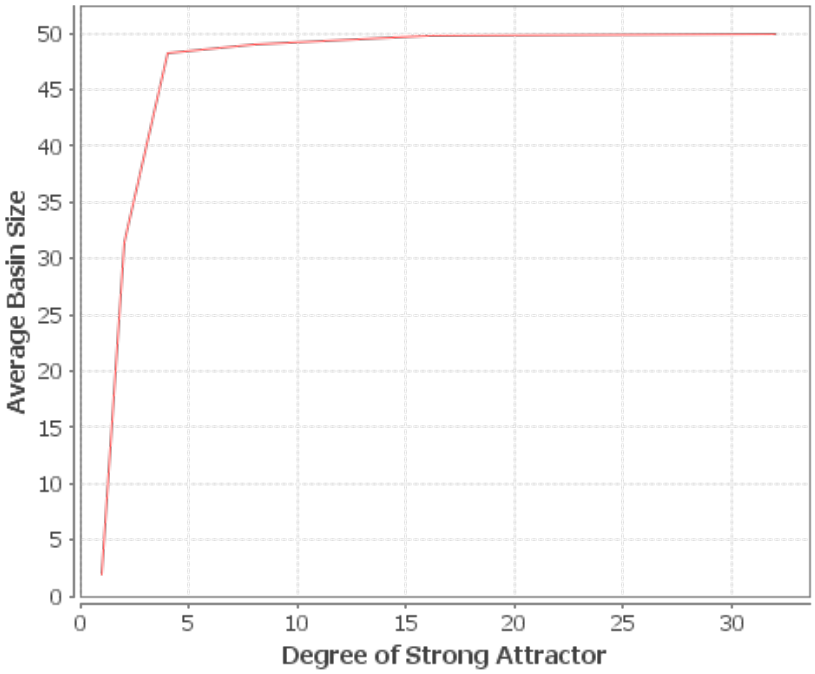

Figure 5.1: Average basin size VS degree of the strong attractor, N = 100

In this experiment we observe that as the degree of the strong attractor increases, the average basin size of the strong attractor also increases and appears to approach a value of 50% of the number of neurons in the network.

5.1.2 Two Strong Attractors

Configuration: 100 neurons, first strong attractor with fixed degree of 8, 8 random patterns, experiment run 800 times.

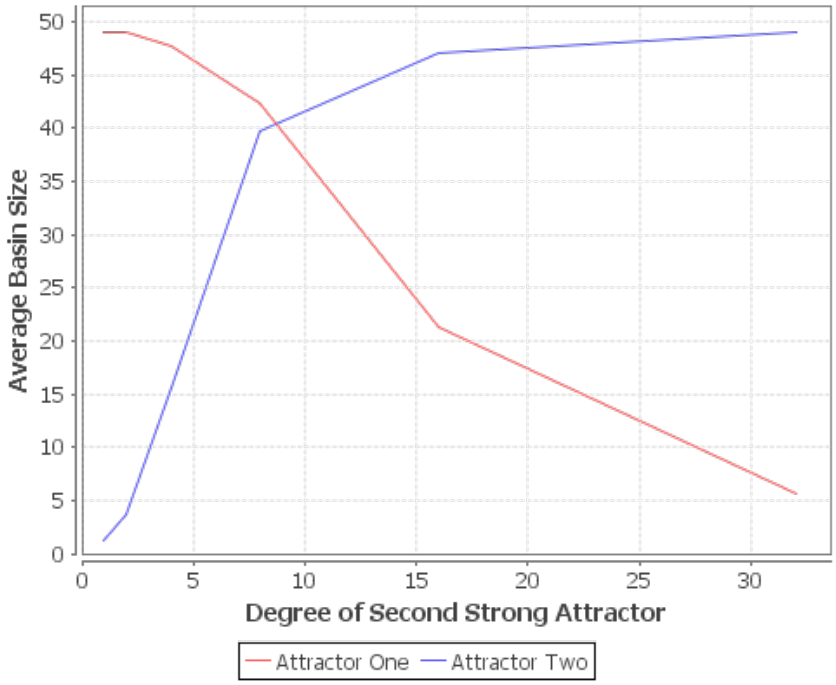

Figure 5.2: Average basin size VS degree of the second strong attractor, with first strong attractor having fixed degree 8, N = 100

In this experiment we observe that as the degree of the second strong attractor increases, the average basin size of the first strong attractor decreases and the average basin size of the second strong attractor increases. Note that when they are of equal degree the first strong attractor has a basin size that is slightly bigger than the basin size of the second strong attractor. This particular occurrence is explained later on in the evaluation.

5.2 Stochastic Experiments

5.2.1 One Strong Attractor, Start Location Type S1

Configuration: 800 neurons, first strong attractor with fixed degree of 32, 20 random patterns, experiment run 1000 times.

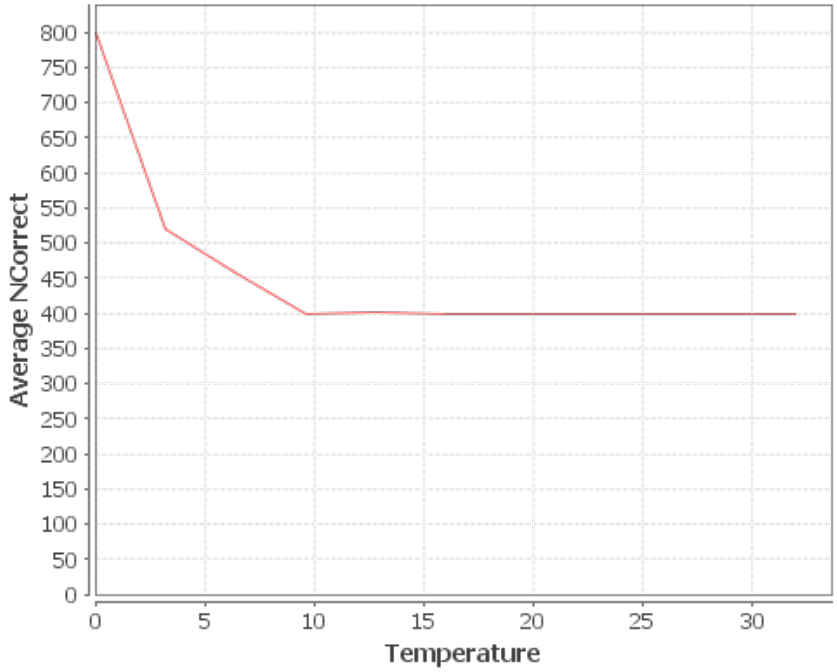

Figure 5.3: Average NCorrect VS Temperature, N = 800, degree = 32, 20 random patterns, Start location S1

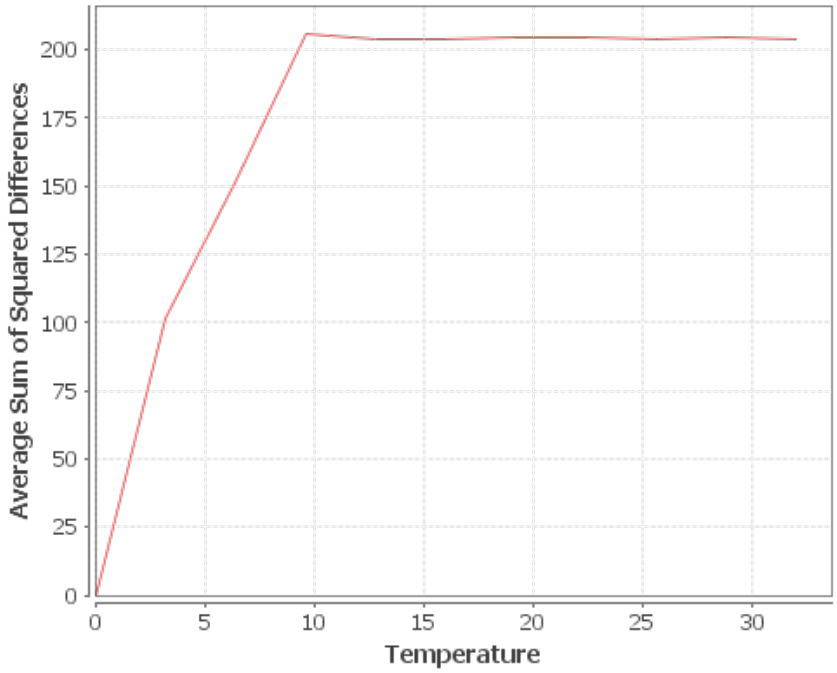

Figure 5.4: Average Sum of Differences Squared VS Temperature, N = 800, degree = 32, 20 random patterns, Start location S1

In this experiment we observe that as the temperature in the network increases, average NCorrect decreases and the average sum of square differences increases.

Configuration: 800 neurons, first strong attractor with fixed degree of 32, 80 random patterns, experiment run 1000 times.

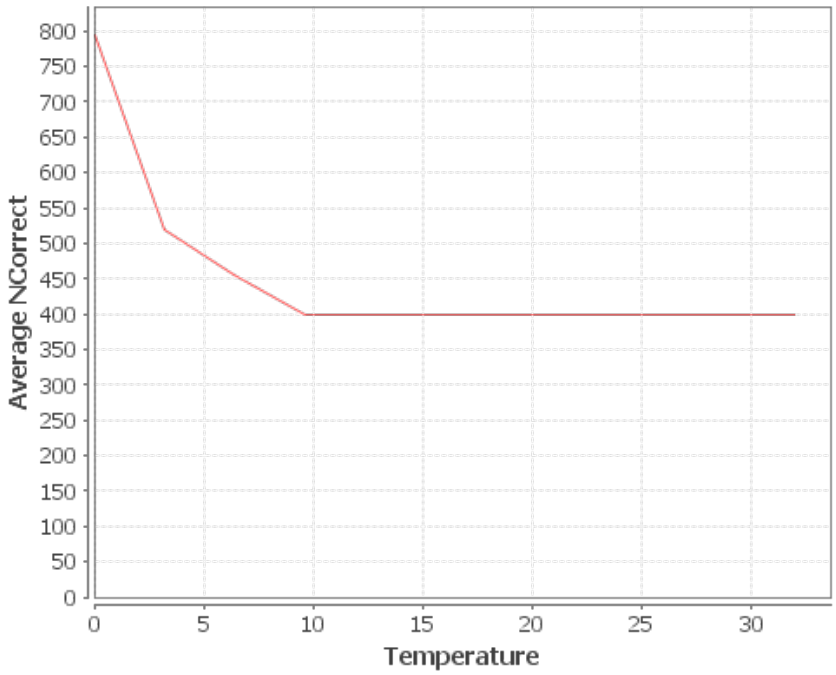

Figure 5.5: Average NCorrect VS Temperature, N = 800, degree = 32, 80 random patterns, Start location S1

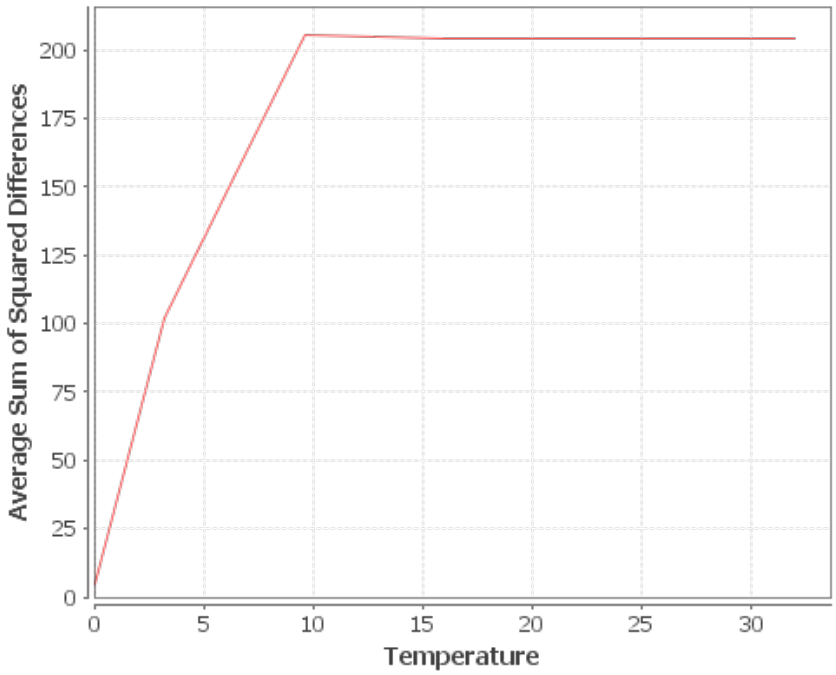

Figure 5.6: Average Sum of Differences Squared VS Temperature, N = 800, degree = 32, 80 random patterns, Start location S1

In this experiment we observe that as the temperature in the network increases, average NCorrect decreases and the average sum of square differences increases. Compared to the previous experiment this experiment has more random patterns present, but this doesn't appear to have made a difference to the overall results or behaviour.

Configuration: 800 neurons, 80 random patterns, one strong attractor with degrees {1,2,4,8,16,32}, experiment run 1000 times (for each degree).

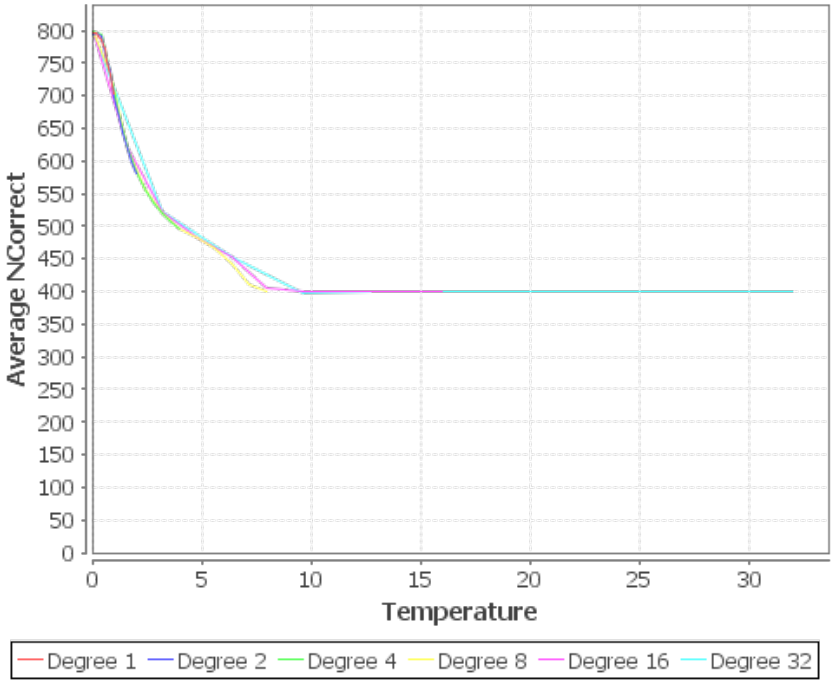

Figure 5.7: Average NCorrect VS Temperature, N = 800, degree = {1,2,4,8,16,32}, 80 random patterns, Start location S1

In this experiment we observe that as the temperature in the network increases, average NCorrect decreases. However the degree of the strong attractor doesn't appear to make much difference in how fast or slow average NCorrect changes with temperature.

5.2.2 One Strong Attractor, Start Location Type S2

Configuration: 800 neurons, 80 random patterns, one strong attractor with degrees {1,2,4,8,16,32}, experiment run 1000 times (for each degree).

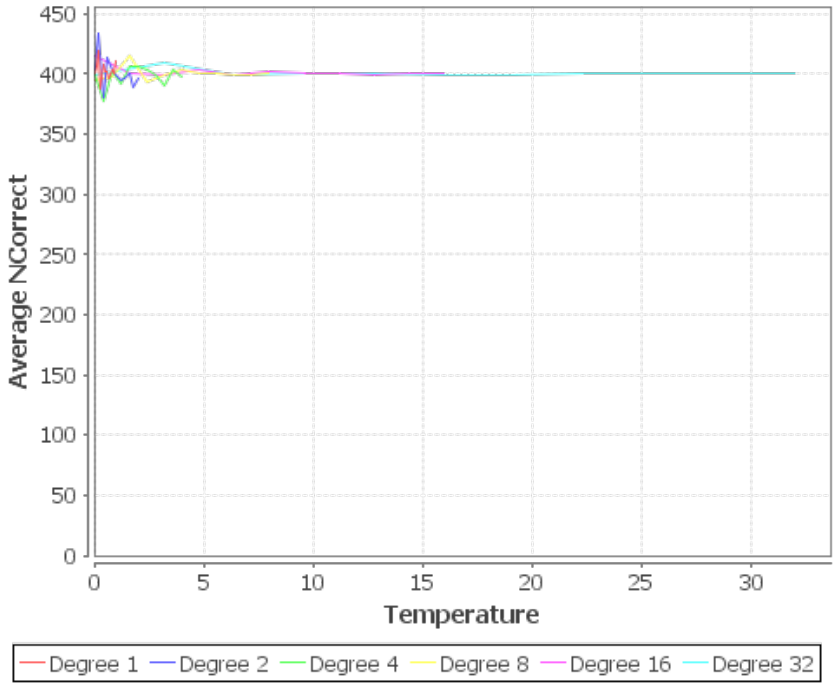

Figure 5.8: Average NCorrect VS Temperature, N = 800, degree = {1,2,4,8,16,32}, 80 random patterns, Start location S2

In this experiment we observe that behaviour is quite erratic at low temperature, but as temperature increases all degree values level off at 50% of neurons in the network being correct.

5.2.3 Two Strong Attractors, Start Location Type S1

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 1, experiment run 500 times.

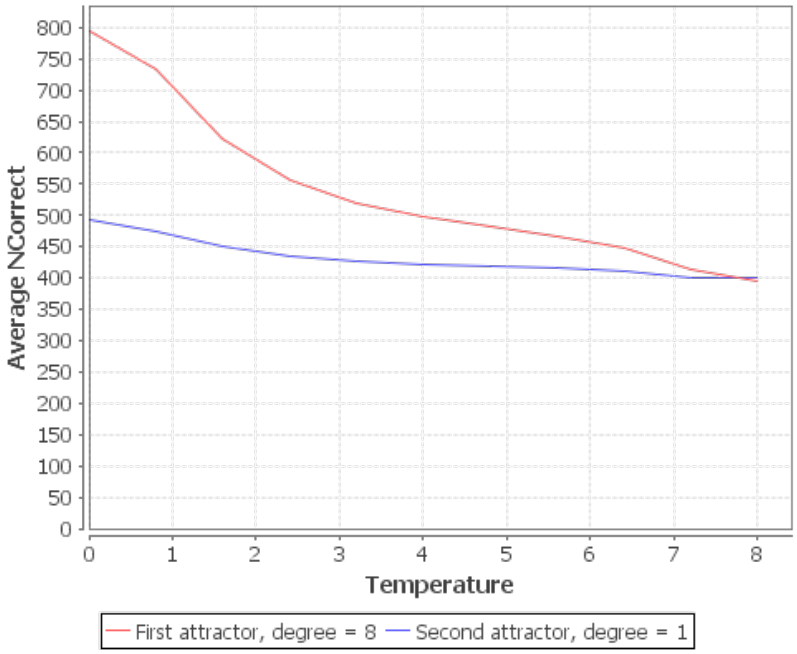

Figure 5.9: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 1, 80 random patterns, Start location S1

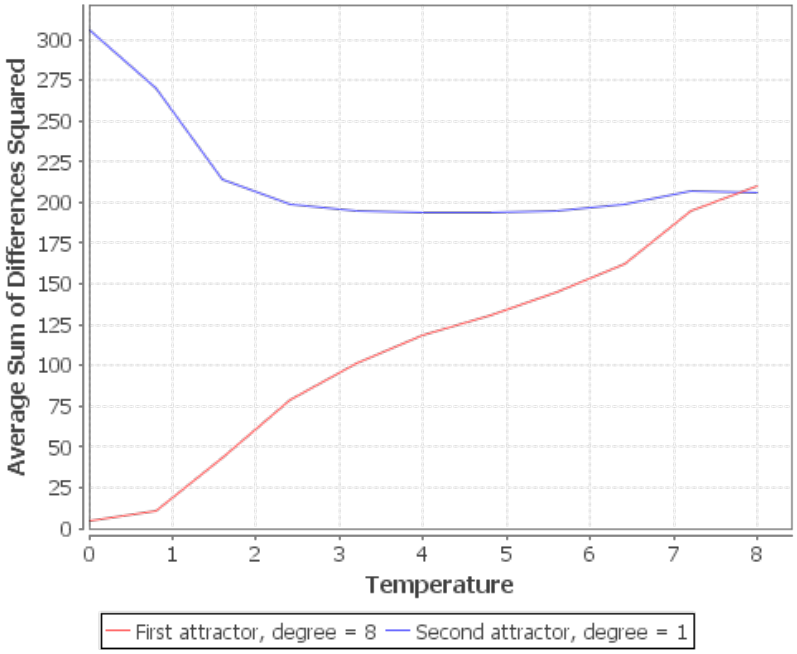

Figure 5.10: Average Sum of Differences Squared VS Temperature, N = 800, degreeOne = 8, degreeTwo = 1, 80 random patterns, Start location S1

In this experiment we observe that for both strong attractors, as temperature increases NCorrect decreases, approaching the 50% of neurons correct level. The first strong attractor has higher degree than the second strong attractor and the first strong attractor continues to be 'more correct' as temperature increases. It is only at high temperature that both strong attractors have a similar value for NCorrect.

In the sum of square differences graph, we observe that at no temperature the first strong attractor has very little error, while the second strong attractor has an error that is higher than $0.25 * neurons$. As temperature increases the sum of square differences for the first strong attractor increases while the sum of square differences for the second strong attractor decreases. This could be an indication of the strength the first strong attractor has in the network compared to the second strong attractor.

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 2, experiment run 500 times.

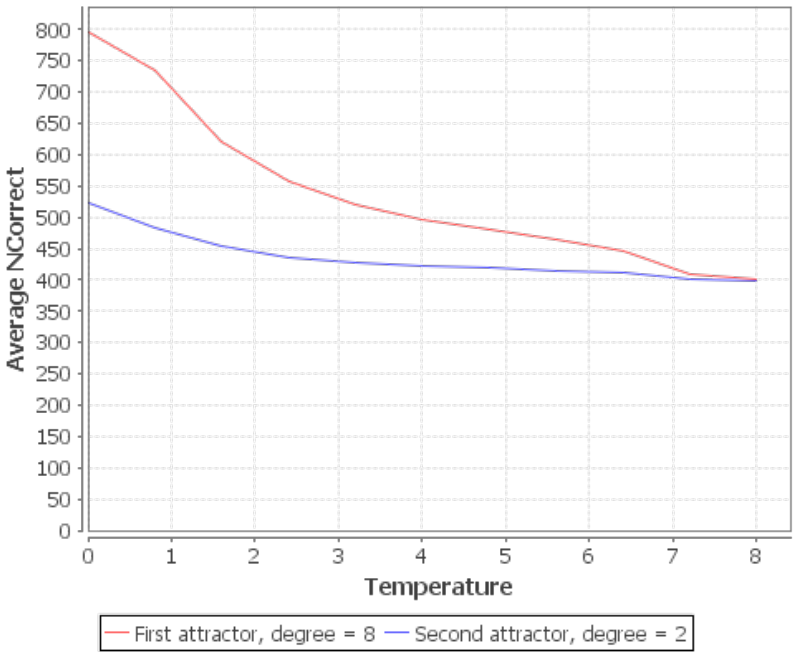

Figure 5.11: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 2, 80 random patterns, Start location S1

In this experiment we observe that for both strong attractors, as temperature increases NCorrect decreases, approaching the 50% of neurons correct level. The first strong attractor has higher degree than the second strong attractor and the first strong attractor continues to be 'more correct' as temperature increases. It is only at high temperature that both strong attractors have a similar value for NCorrect.

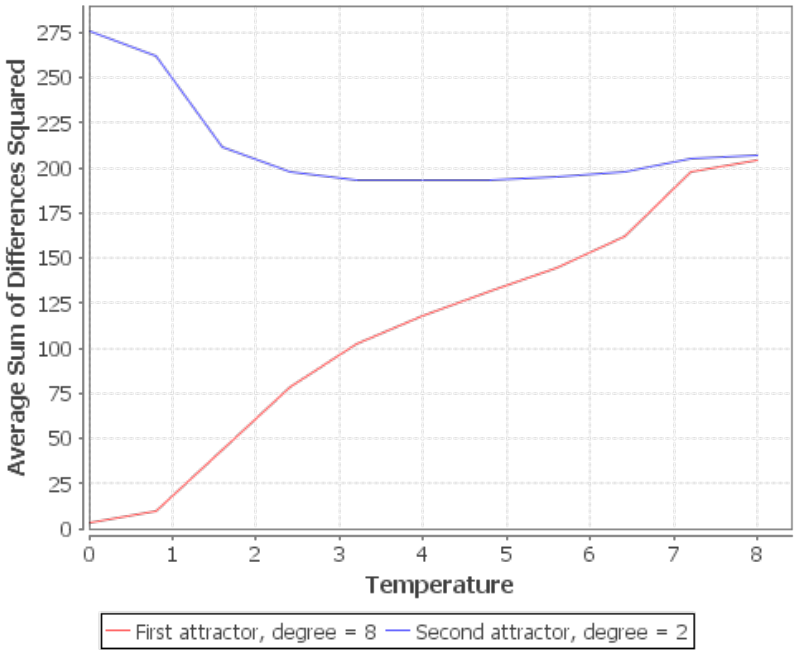

Figure 5.12: Average Sum of Differences Squared VS Temperature, N = 800, degreeOne = 8, degreeTwo = 2, 80 random patterns, Start location S1

In the sum of square differences graph, we observe that at no temperature the first strong attractor has very little error, while the second strong attractor has an error that is higher than $0.25 * neurons$. As temperature increases the sum of square differences for the first strong attractor increases while the sum of square differences for the second strong attractor decreases. This could be an indication of the strength the first strong attractor has in the network compared to the second strong attractor.

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 4, experiment run 500 times.

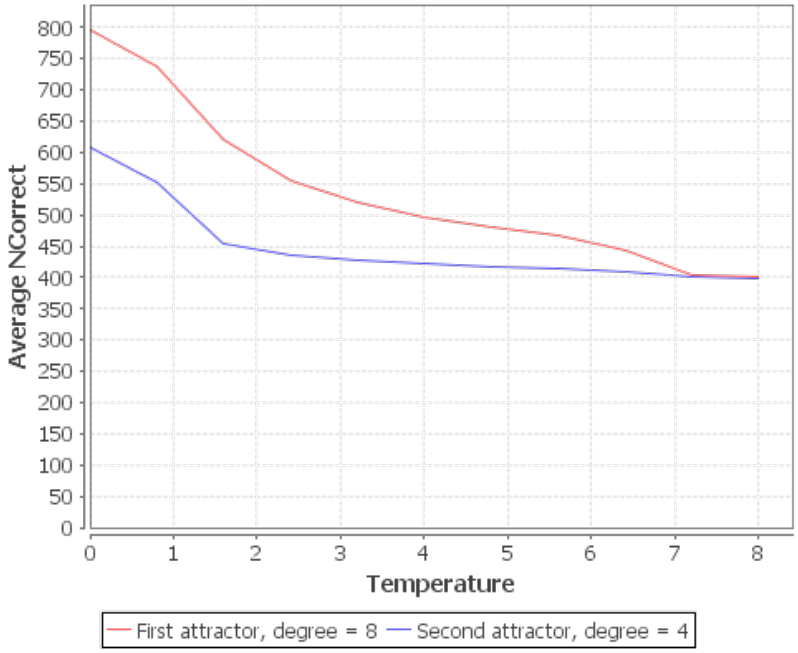

Figure 5.13: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 4, 80 random patterns, Start location S1

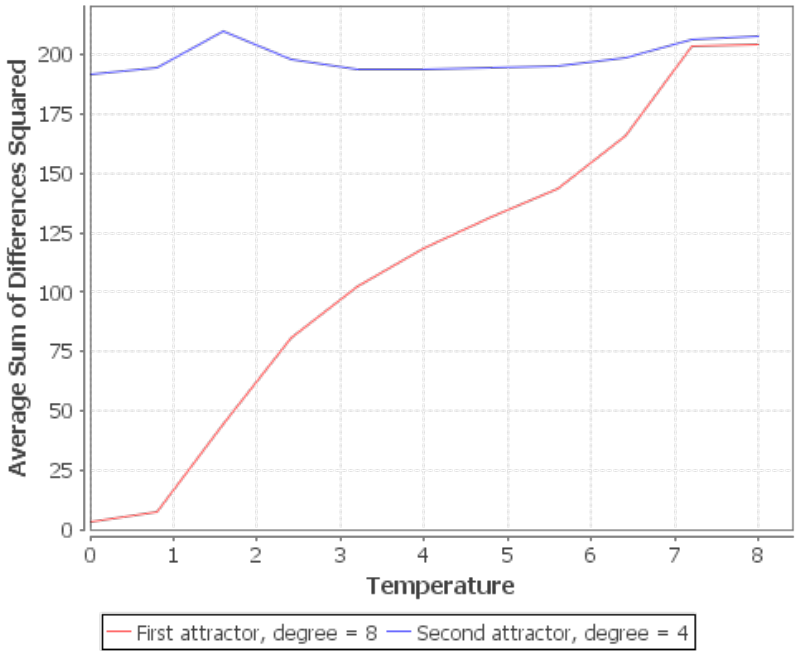

Figure 5.14: Average Sum of Differences Squared VS Temperature, N = 800, degreeOne = 8, degreeTwo = 4, 80 random patterns, Start location S1

In this experiment we observe that for both strong attractors, as temperature increases NCorrect decreases, approaching the 50% of neurons correct level. The first strong attractor has higher degree than the second strong attractor and the first strong attractor continues to be 'more correct' as temperature increases. It is only at high temperature that both strong attractors have a similar value for NCorrect.

In the sum of square differences graph, we observe that at no temperature the first strong attractor has very little error, while the second strong attractor has an error that is just below $0.25 * neurons$. The sum of square differences for the first strong attractor increases as temperature increases until it levels off around $0.25 * neurons$, while the sum of square differences for the second attractor oscilates around the $0.25 * neurons$ level regardless of temperature.

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 8, experiment run 500 times.

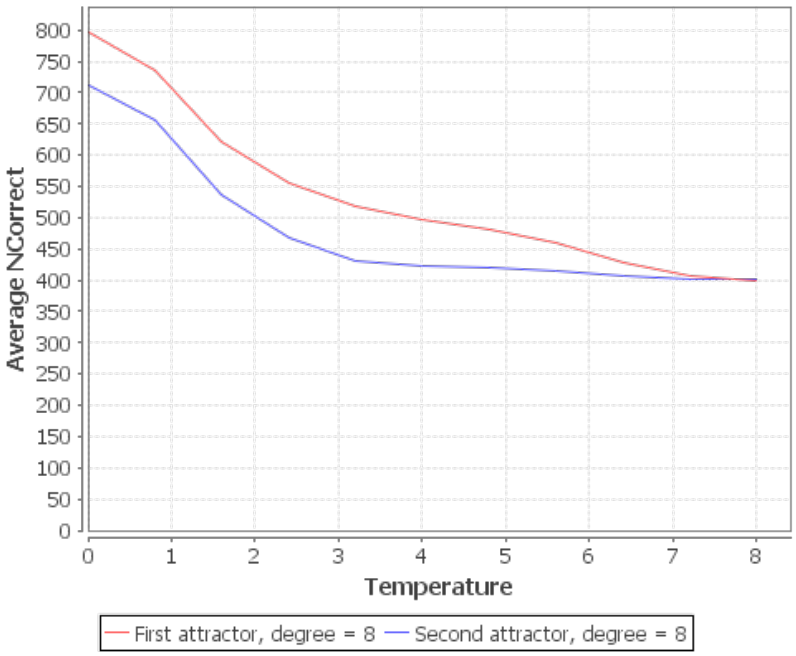

Figure 5.15: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 8, 80 random patterns, Start location S1

In this experiment we observe that for both strong attractors, as temperature increases NCorrect decreases, approaching the 50% of neurons correct level. The first strong attractor has the same degree as the second strong attractor but the first strong attractor continues to be 'more correct' as temperature increases. It is only at high temperature that both strong attractors have a similar value for NCorrect. This unusual behaviour is discussed and analysed in the evaluation.

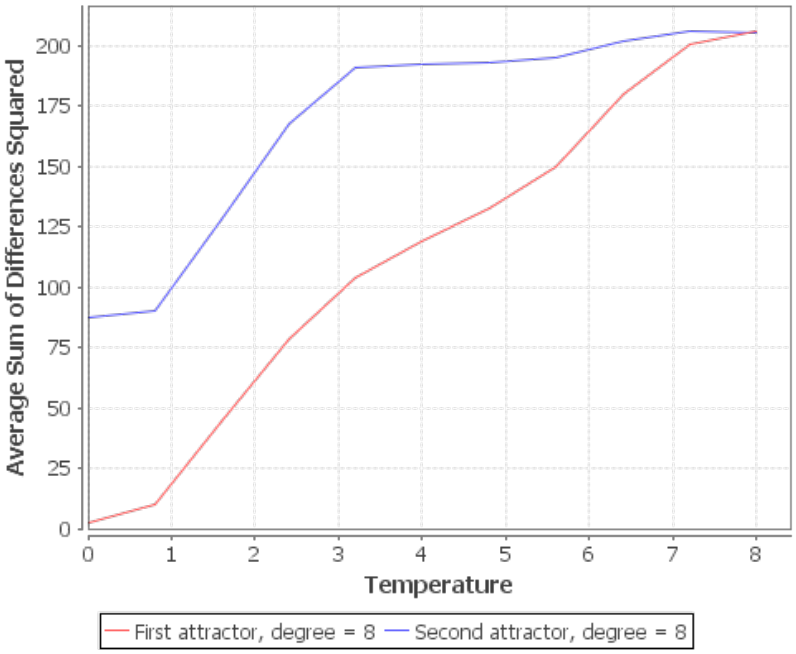

Figure 5.16: Average Sum of Differences Squared VS Temperature, N = 800, degreeOne = 8, degreeTwo = 8, 80 random patterns, Start location S1

In the sum of square differences graph, we observe that at no temperature the first strong attractor has very little error, while the second strong attractor has an error that is between 0 and the $0.25 * neurons$ level. The sum of square differences for the first strong attractor increases as temperature increases until it levels off around $0.25 * neurons$, and this behaviour is also observed in the second strong attractor.

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 16, experiment run 500 times.

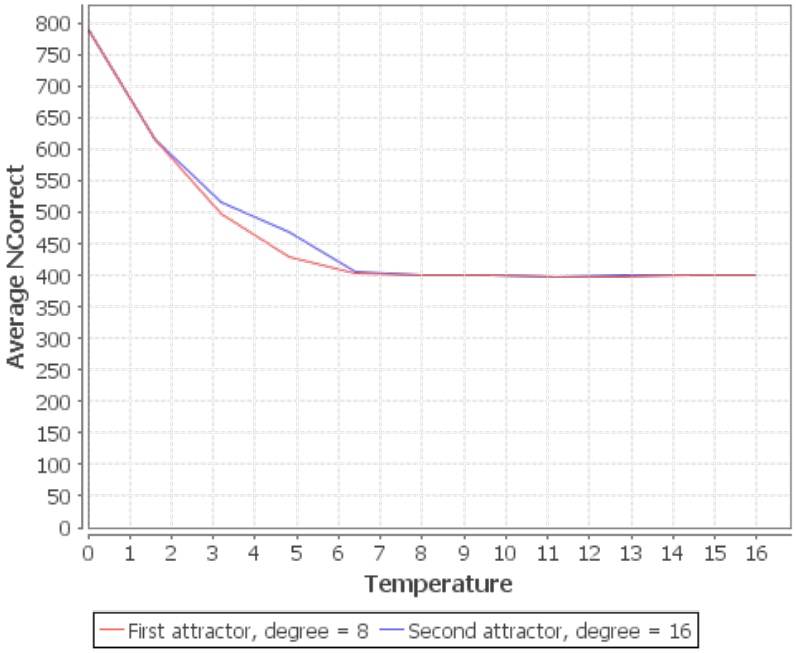

Figure 5.17: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 16, 80 random patterns, Start location S1

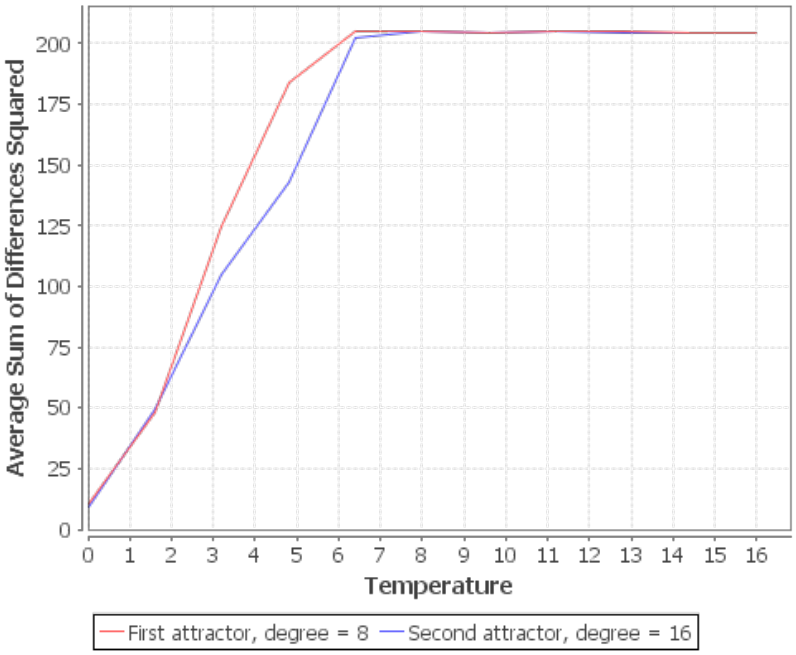

Figure 5.18: Average Sum of Differences Squared VS Temperature, N = 800, degreeOne = 8, degreeTwo = 16, 80 random patterns, Start location S1

In this experiment we observe that for both strong attractors, as temperature increases NCorrect decreases, approaching the 50% of neurons correct level. The first strong attractor has a lower degree than the second strong attractor but the second strong attractor is only slightly more correct as temperature increases than the first strong attractor. It is only at high temperature that both strong attractors have a similar value for NCorrect. This unusual behaviour is discussed and analysed in the evaluation.

In the sum of square differences graph, we observe that at no temperature the first strong attractor has very little error, while the second strong attractor also has little error. The sum of square differences for the first strong attractor increases as temperature increases until it levels off around $0.25 * neurons$, and this behaviour is also observed in the second strong attractor.

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 32, experiment run 500 times.

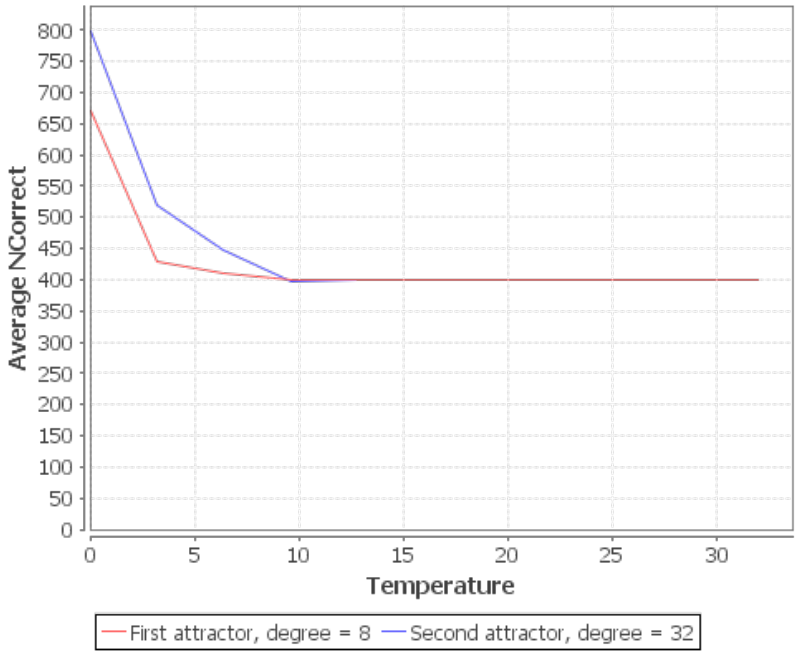

Figure 5.19: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 32, 80 random patterns, Start location S1

In this experiment we observe that for both strong attractors, as temperature increases NCorrect decreases, approaching the 50% of neurons correct level. The first strong attractor has a lower degree than the second strong attractor and the second strong attractor remains more correct as temperature increases than the first strong attractor. It is only at high temperature that both strong attractors have a similar value for NCorrect.

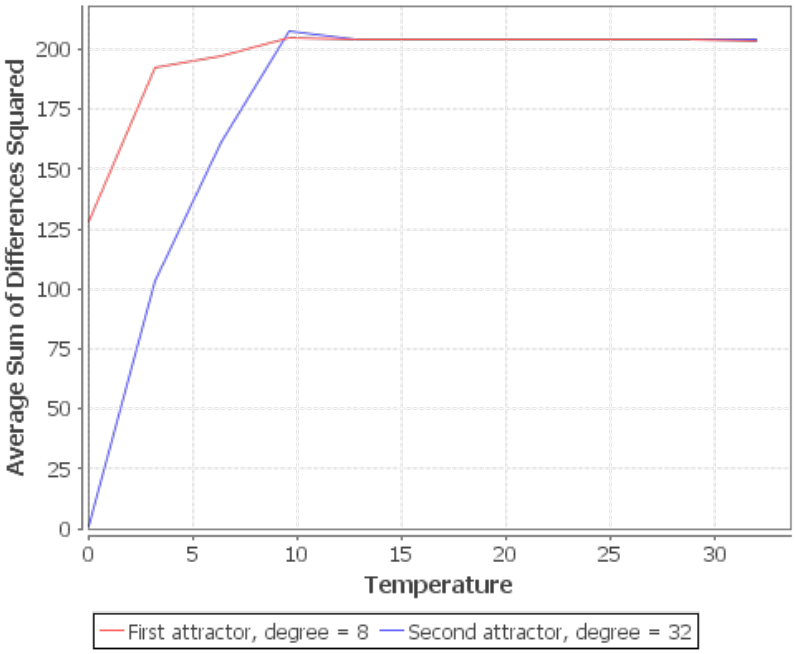

Figure 5.20: Average Sum of Differences Squared VS Temperature, N = 800, degreeOne = 8, degreeTwo = 32, 80 random patterns, Start location S1

In the sum of square differences graph, we observe that at no temperature the first strong attractor has an error that is higher than the second strong attractor but less than the $0.25 * neurons$ level, while the second strong attractor has little error. The sum of square differences for the first strong attractor increases as temperature increases until it levels off around $0.25 * neurons$, and this behaviour is also observed in the second strong attractor.

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 64, experiment run 500 times.

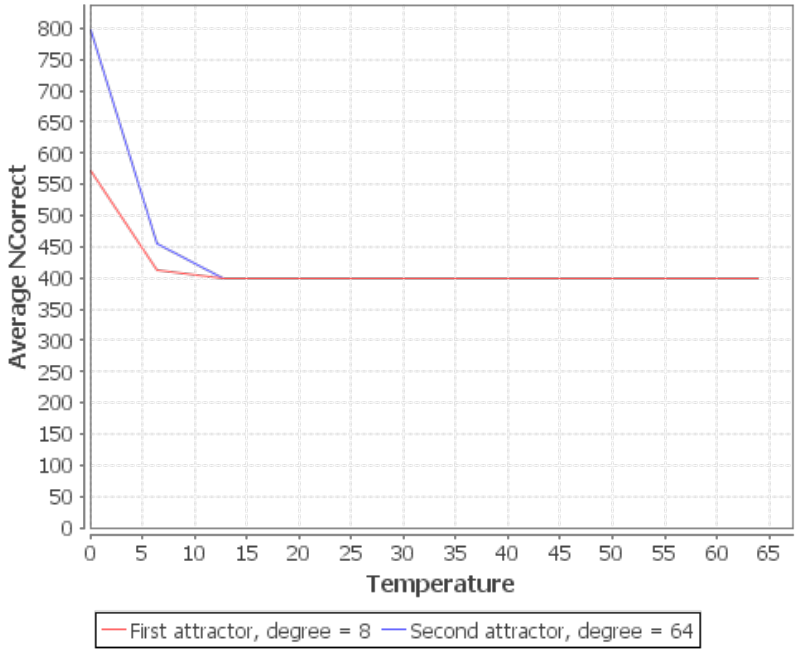

Figure 5.21: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 64, 80 random patterns, Start location S1

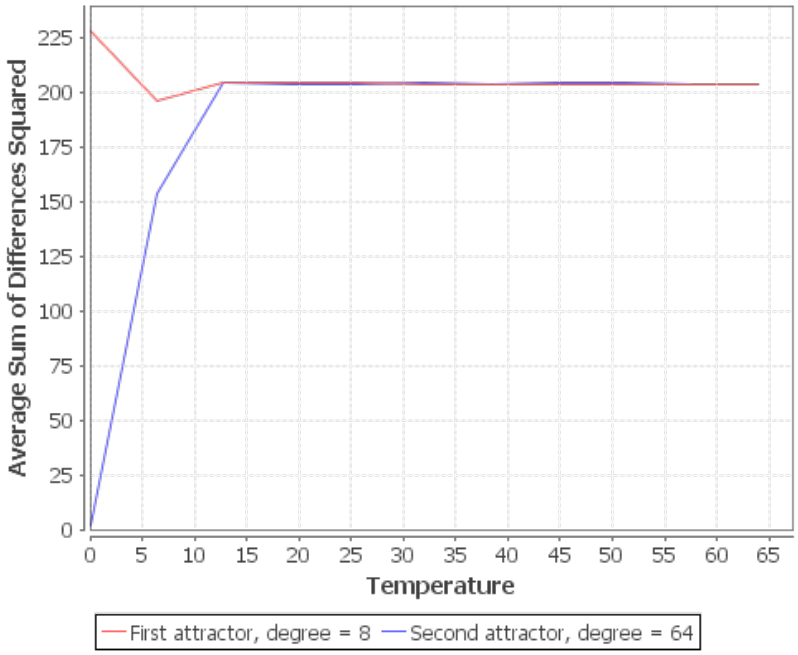

Figure 5.22: Average Sum of Differences Squared VS Temperature, N = 800, degreeOne = 8, degreeTwo = 64, 80 random patterns, Start location S1

In this experiment we observe that for both strong attractors, as temperature increases NCorrect decreases, approaching the 50% of neurons correct level. The first strong attractor has a lower degree than the second strong attractor and the second strong attractor remains more correct as temperature increases than the first strong attractor. It is only at high temperature that both strong attractors have a similar value for NCorrect.

In the sum of square differences graph, we observe that at no temperature the first strong attractor has an error that is higher than the $0.25 * neurons$ level, while the second strong attractor has little error. The sum of square differences for the second strong attractor increases as temperature increases until it levels off around $0.25 * neurons$, while the sum of square differences for the first strong attractor oscillates around the $0.25 * neurons$ level regardless of temperature.

5.2.4 Two Strong Attractors, Start Location Type S2

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 1, experiment run 1000 times.

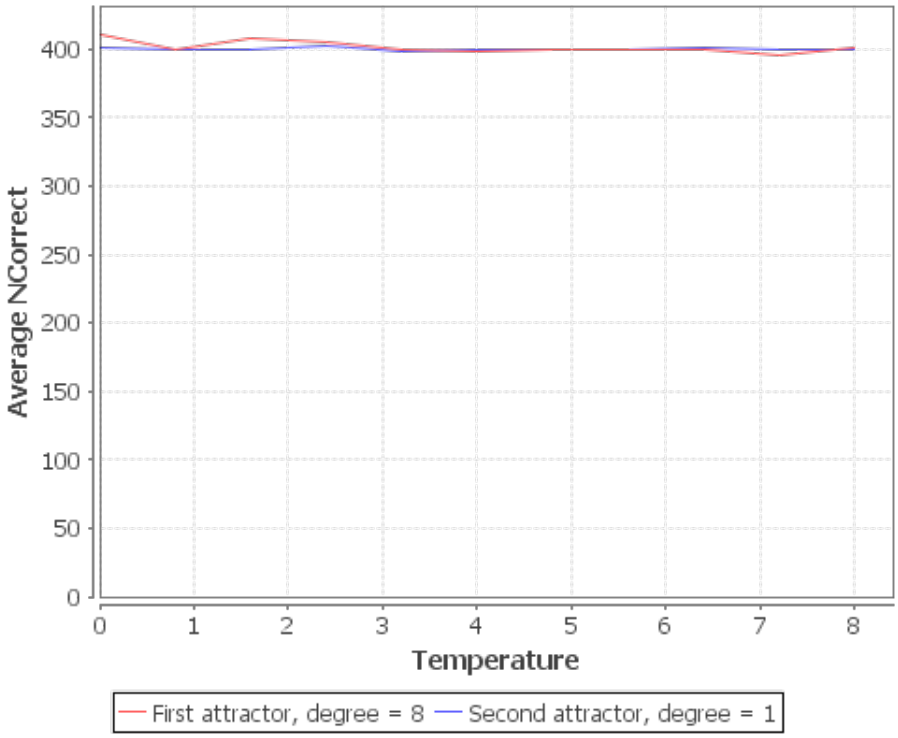

Figure 5.23: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 1, 80 random patterns, Start location S2

In this experiment we observe that even at low temperature, both strong attractors are only around the 50% correct level. The first strong attractor shows more instability than the second strong attractor as temperature changes.

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 2, experiment run 1000 times.

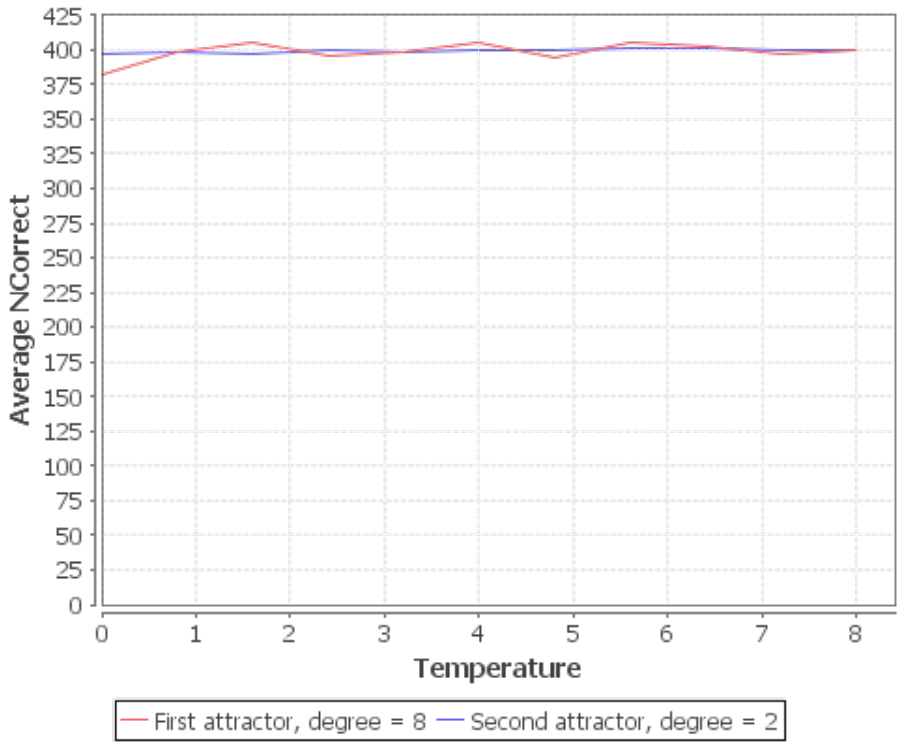

Figure 5.24: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 2, 80 random patterns, Start location S2

In this experiment we observe that even at low temperature, both strong attractors are only around the 50% correct level. The first strong attractor shows more instability than the second strong attractor as temperature changes.

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 4, experiment run 1000 times.

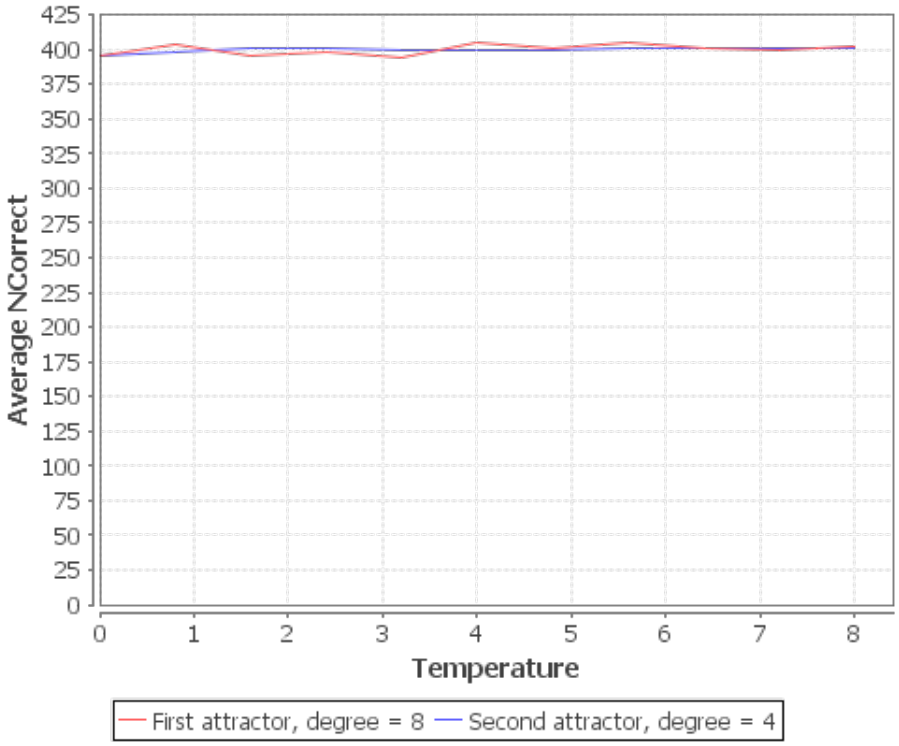

Figure 5.25: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 4, 80 random patterns, Start location S2

In this experiment we observe that even at low temperature, both strong attractors are only around the 50% correct level. The first strong attractor shows more instability than the second strong attractor as temperature changes.

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 8, experiment run 1000 times.

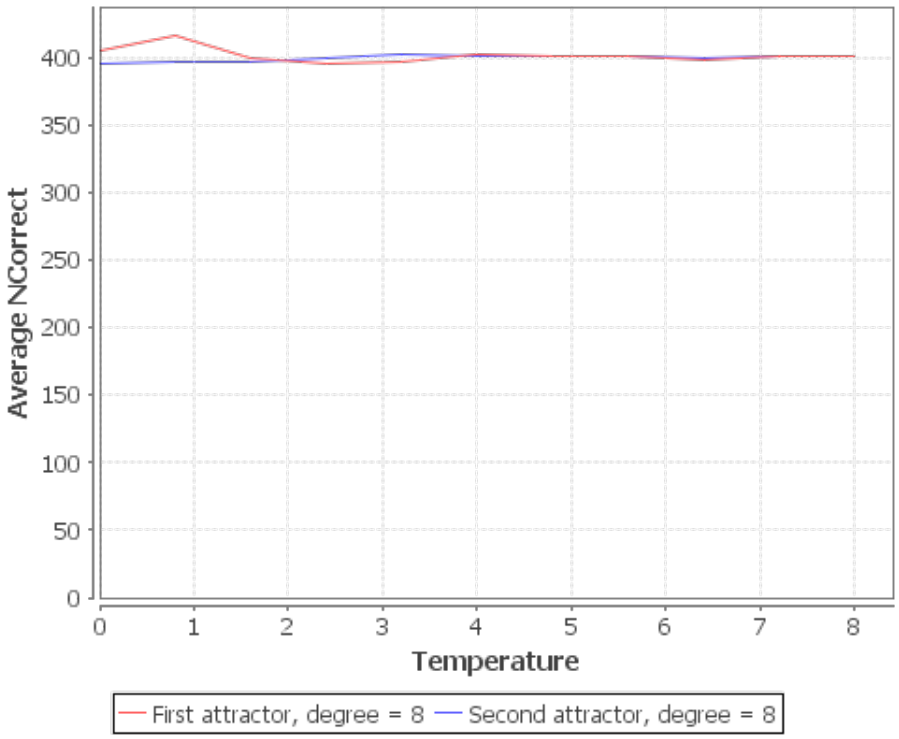

Figure 5.26: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 8, 80 random patterns, Start location S2

In this experiment we observe that even at low temperature, both strong attractors are only around the 50% correct level. The first strong attractor shows more instability than the second strong attractor as temperature changes.

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 16, experiment run 1000 times.

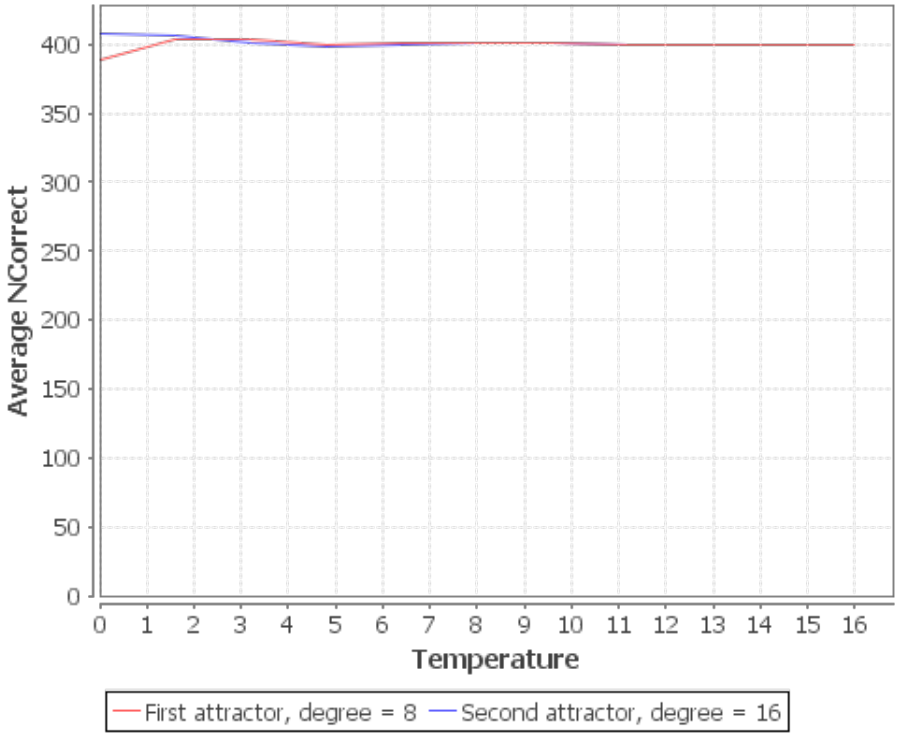

Figure 5.27: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 16, 80 random patterns, Start location S2

In this experiment we observe that even at low temperature, both strong attractors are only around the 50% correct level. Both attractors are about the same stability.

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 32, experiment run 1000 times.

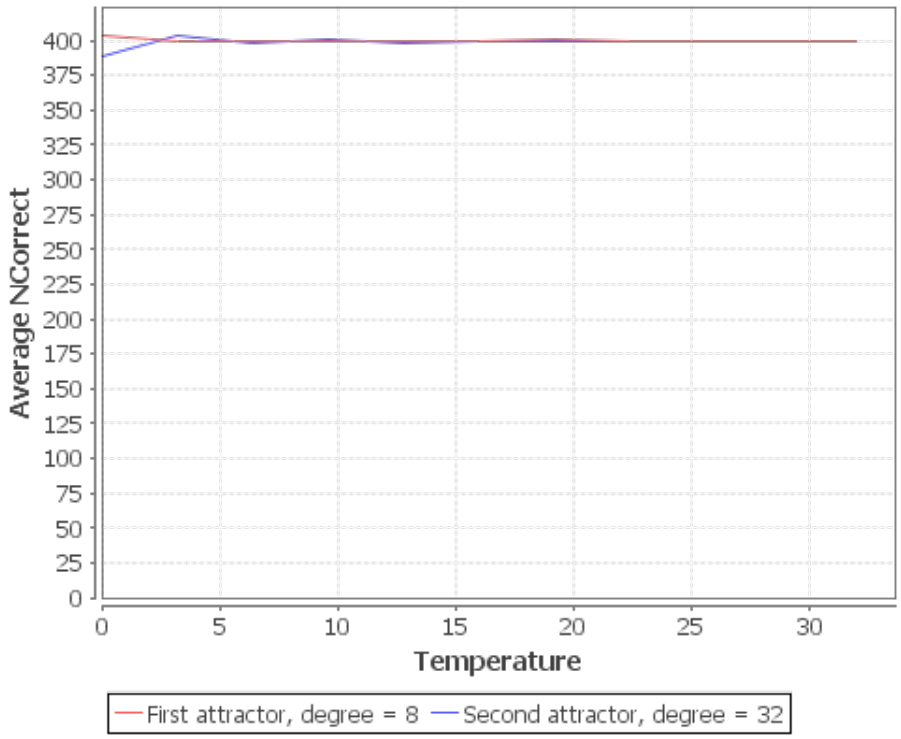

Figure 5.28: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 32, 80 random patterns, Start location S2

In this experiment we observe that even at low temperature, both strong attractors are only around the 50% correct level. The second strong attractor shows more instability than the first strong attractor as temperature changes.

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 64, experiment run 1000 times.

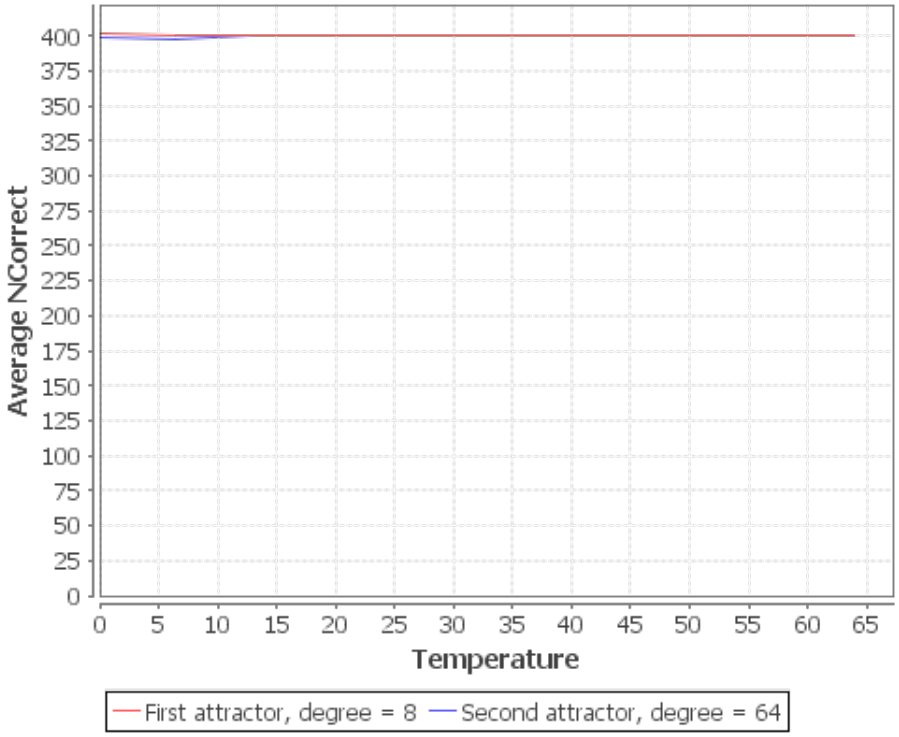

Figure 5.29: Average NCorrect VS Temperature, N = 800, degreeOne = 8, degreeTwo = 64, 80 random patterns, Start location S2

In this experiment we observe that even at low temperature, both strong attractors are only around the 50% correct level. The second strong attractor shows more instability than the first strong attractor as temperature changes.

5.3 Verification of Energy Landscapes

Table header notation used in this section:

`HigherThanAvg` = Number of experiments where the average energy of the stable mixture states was higher than the average energy of the retrieval states.

`HigherThanStr` = Number of experiments where the average energy of the stable mixture states was higher than the energy of the strong attractor.

`NoMixtures` = Number of experiments where there were no stable mixture states present in the network when observed.

`Accuracy` = Probability that if the network contains stable mixture states, that the average energy of the stable mixture states is higher than the energy of the strong attractor (calculation: `HigherThanStrong`/(Times Experiment Was Run−`NoMixtures`)).

`Lowest` = Number of experiments where a stable mixture state was the lowest observed energy minima in the network.

5.3.1 Mixture States Investigation, Classical Theory

Configuration: 100 neurons, experiment run 1000 times.

alpha	`HigherThanAvg`	`NoMixtures`	`Lowest`
0.03	1000	0	0
0.05	1000	0	0
0.07	997	3	0
0.09	985	15	3
0.11	965	35	4
0.13	975	25	10
0.15	992	4	24
0.17	999	0	69

Table 5.1: Results from the classical mixture states investigation run with 100 neurons.

In this experiment we observe that when alpha is in region A of the phase diagram, the network always contains stable mixture states and these stable mixture states have higher average energy than the average energy of the retrieval states (patterns the network was trained with).

However, once we enter region B, as alpha increases it becomes more likely that the network will not contain any stable mixture states, until after $alpha = 0.11$ when the likelihood starts decreasing again. However in networks that contain stable mixture states the average mixture state energy is still higher than the average retrieval state energy.

Also beginning in region B, as alpha increases it becomes more likely that the lowest energy minima in the network observed is a mixture state, rather than a retrieval state.

5.3.2 Mixture States Investigation, Strong Attractors Theory

Configuration: 100 neurons, degree 1, experiment run 500 times.

alpha	HigherThanAvg	HigherThanStr	NoMixtures	Accuracy	Lowest
0.03	500	500	0	1.000	0
0.05	499	499	1	1.000	0
0.07	498	498	2	1.000	1
0.09	494	494	6	1.000	3
0.11	489	489	10	0.998	2
0.13	490	489	9	0.996	7
0.15	494	495	2	0.994	6
0.17	499	499	0	0.998	36

Table 5.2: Results from the strong attractors mixture states investigation run with 100 neurons and degree 1.

In this experiment we observe that when alpha is in region A of the phase diagram, the network nearly always contains stable mixture states and these stable mixture states have higher average energy than the average energy of the retrieval states (patterns the network was trained with).

However, once we enter region B, as alpha increases it becomes more likely that the network will not contain any stable mixture states, until after $alpha = 0.11$ when the likelihood starts decreasing again. However in networks that contain stable mixture states the average mixture state energy is still higher than the average retrieval state energy, apart from a few cases around the border between regions B and C. However the Hopfield network is known to have slight error (i.e. the memory is not perfect).

Also beginning in region B, as alpha increases it becomes more likely that the lowest energy minima in the network observed is a mixture state, rather than a retrieval state.

Configuration: 100 neurons, degree 2, experiment run 500 times.

alpha	HigherThanAvg	HigherThanStr	NoMixtures	Accuracy	Lowest
0.02	500	500	0	1.000	0
0.04	500	500	0	1.000	0
0.06	493	493	7	1.000	0
0.08	477	477	23	1.000	0
0.10	436	436	64	1.000	0
0.12	435	435	65	1.000	0
0.14	460	461	69	1.000	0
0.16	495	496	4	1.000	0
0.18	500	500	0	1.000	0

Table 5.3: Results from the strong attractors mixture states investigation run with 100 neurons and degree 2.

In this experiment we immediately notice that at no point was a mixture state the lowest observed energy minima in any network.

In region A, we observe that there are no occurrences where a network has no stable mixture states. However once we enter region B, as alpha increases we observe more and more cases where the network does not contain any stable mixture states, until entering region C, where it becomes less likely again.

In all networks that contained stable mixture states, the average stable mixture state energy was higher than that of the strong attractor energy.

Configuration: 100 neurons, degree 4, experiment run 500 times.

alpha	HigherThanAvg	HigherThanStr	NoMixtures	Accuracy	Lowest
0.02	0	495	0	0.9900	0
0.04	384	492	0	0.9840	0
0.06	434	443	9	0.9022	0
0.08	384	385	90	0.9390	0
0.10	316	316	179	0.9844	0
0.12	295	295	204	0.9966	0
0.14	340	340	159	0.9971	0
0.16	411	411	88	0.9978	0
0.18	485	485	15	1.0000	0

Table 5.4: Results from the strong attractors mixture states investigation run with 100 neurons and degree 4.

In this experiment we again notice that at no point was a mixture state the lowest observed energy minima in any network.

In region A, we observe that there are no occurrences where a network has no stable mixture states. However once we enter region B, as alpha increases we observe more and more cases where the network does not contain any stable mixture states, until entering region C, where it becomes less likely again.

In the networks that contained stable mixture states, the average stable mixture state energy was higher than that of the strong attractor energy in most cases, however we observe that there a few occurrences where the opposite is true, especially within region B.

Configuration: 100 neurons, degree 8, experiment run 250 times.

alpha	HigherThanAvg	HigherThanStr	NoMixtures	Accuracy	Lowest
0.02	0	244	0	0.9760	0
0.04	0	216	0	0.8640	0
0.06	0	214	0	0.8560	0
0.08	0	197	0	0.7880	0
0.10	0	180	0	0.7200	0
0.12	0	170	0	0.6800	0
0.14	0	190	0	0.7600	0
0.16	0	205	0	0.8200	0
0.18	0	238	0	0.9520	0

Table 5.5: Results from the strong attractors mixture states investigation run with 100 neurons and degree 8.

In this experiment we again notice that at no point was a mixture state the lowest observed energy minima in any network. We also notice that in this case all networks measured had stable mixture states, unlike previous experiments of this type.

The average stable mixture state energy was higher than that of the strong attractor energy in most cases, however we observe that there more occurrences where the opposite is true than in previous experiments of this type, especially within region B.

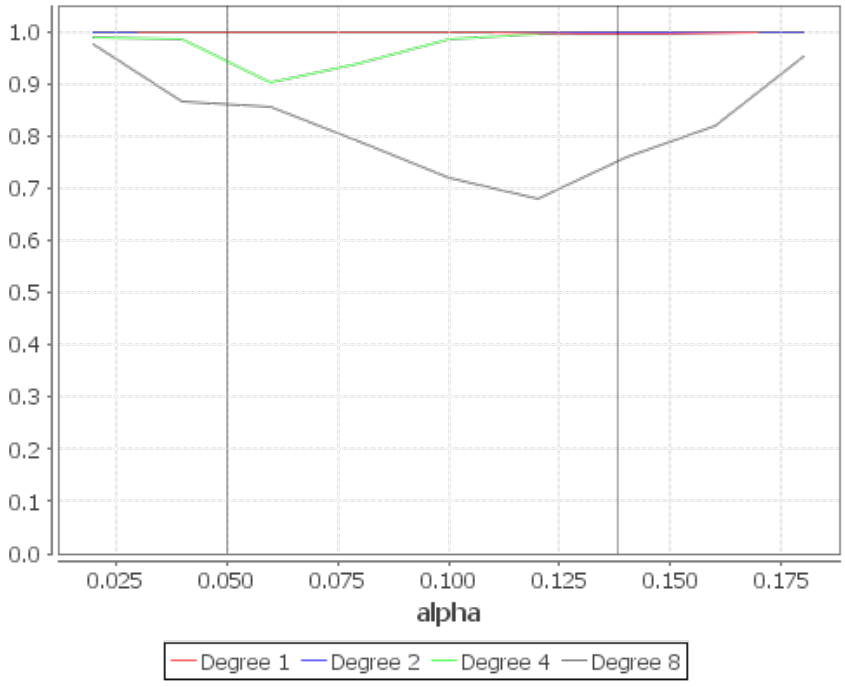

Figure 5.30: `Accuracy` VS alpha for the strong attractors mixture state investigation with 100 neurons and degrees {1,2,4,8}.

5.3.3 Spin Glass States Investigation, Classical Theory

Table header notation used in this section:

`LowerThanAvg` = Number of experiments where the average energy of the spin glass states was lower than the average energy of the retrieval states.

`LowerThanStr` = Number of experiments where the average energy of the spin glass states was lower than the energy of the strong attractor.

`NoSpinGlass` = Number of experiments where there were no spin glass states present in the network when observed.

`Accuracy` = Probability that if the network contains spin glass states, that the average energy of the spin glass states is lower than the energy of the strong attractor (calculation: `LowerThanStrong`/(Times Experiment Was Run − `NoSpinGlass`)).

`Lowest` = Number of experiments where a spin glass state was the lowest observed energy minima in the network.

Configuration: 100 neurons, degree 1, experiment run 500 times.

alpha	`LowerThanAvg`	`LowerThanStr`	`NoSpinGlass`	`Accuracy`	`Lowest`
0.02	0	0	0	0.000	0
0.04	0	0	0	0.000	0
0.06	0	0	0	0.000	16
0.08	0	0	0	0.000	75
0.10	2	15	0	0.004	192
0.12	21	57	0	0.042	355
0.14	76	125	0	0.152	454
0.16	230	236	0	0.460	485

Table 5.6: Results from the spin glass states investigation run with 100 neurons and degree 1. Note that because degree is one (i.e. classical theory) `Accuracy` is calculated using `LowerThanAvg` rather than `LowerThanStr`.

In this experiment we first observe that all networks measured contained stable spin glass states.

In region A, no spin glass states are observed as being the lowest energy minima in a network. However once we enter region B, as alpha increases it becomes more and more likely that the lowest minima observed in a network is a stable spin glass state. At the upper edge of region B, the lowest minima is a stable spin glass state in over two thirds of networks measured. In region C nearly all networks have a spin glass state as their lowest energy minima.

Part way into region B, it starts to become more likely that the average energy of the spin glass states in a network has lower energy than the average energy of the retrieval states in the network.

5.3.4 Spin Glass States Investigation, Strong Attractors Theory

Configuration: 100 neurons, degree 2, experiment run 500 times.

alpha	LowerThanAvg	LowerThanStr	NoSpinGlass	Accuracy	Lowest
0.02	0	0	500	0.000	0
0.04	0	0	0	0.000	0
0.06	0	0	0	0.000	0
0.08	0	0	0	0.000	0
0.10	1	0	0	0.000	0
0.12	11	0	0	0.000	0
0.14	44	0	0	0.000	0
0.16	131	0	0	0.000	0

Table 5.7: Results from the spin glass states investigation run with 100 neurons and degree 2.

In this experiment we observe that no networks measured had a spin glass state as their lowest minima, even at high alpha. Also there were no instances where the average energy of the spin glass states was lower than the energy of the strong attractor. In region A at $alpha = 0.02$ we notice that all networks measured had no spin glass states.

Configuration: 100 neurons, degree 4, experiment run 500 times.

alpha	LowerThanAvg	LowerThanStr	NoSpinGlass	Accuracy	Lowest
0.02	0	0	500	0.000	0
0.04	0	0	137	0.000	0
0.06	28	0	3	0.000	0
0.08	209	0	2	0.000	0
0.10	237	0	0	0.000	0
0.12	212	0	0	0.000	0
0.14	187	0	0	0.000	0
0.16	197	0	0	0.000	0

Table 5.8: Results from the spin glass states investigation run with 100 neurons and degree 4.

In this experiment we observe that no networks measured had a spin glass state as their lowest minima, even at high alpha. Also there were no instances where the average energy of the spin glass states was lower than the energy of the strong attractor.

In region A at $alpha = 0.02$ we notice that all networks measured had no spin glass states. We also notice that in the rest of region A and in the lower section of region B there are also cases where the networks had no spin glass states.

Configuration: 100 neurons, degree 8, experiment run 500 times.

alpha	LowerThanAvg	LowerThanStr	NoSpinGlass	Accuracy	Lowest
0.02	0	0	500	0.000	0
0.04	0	0	500	0.000	0
0.06	0	0	494	0.000	0
0.08	0	0	493	0.000	0
0.10	0	0	493	0.000	0
0.12	0	0	486	0.000	0
0.14	0	0	493	0.000	0
0.16	0	0	493	0.000	0

Table 5.9: Results from the spin glass states investigation run with 100 neurons and degree 8.

In this experiment we observe that no networks measured had a spin glass state as their lowest minima, even at high alpha. Also there were no instances where the average energy of the spin glass states was lower than the energy of the strong attractor.

In region A we notice that all networks measured had no spin glass states. In regions B and C most networks had no spin glass states.

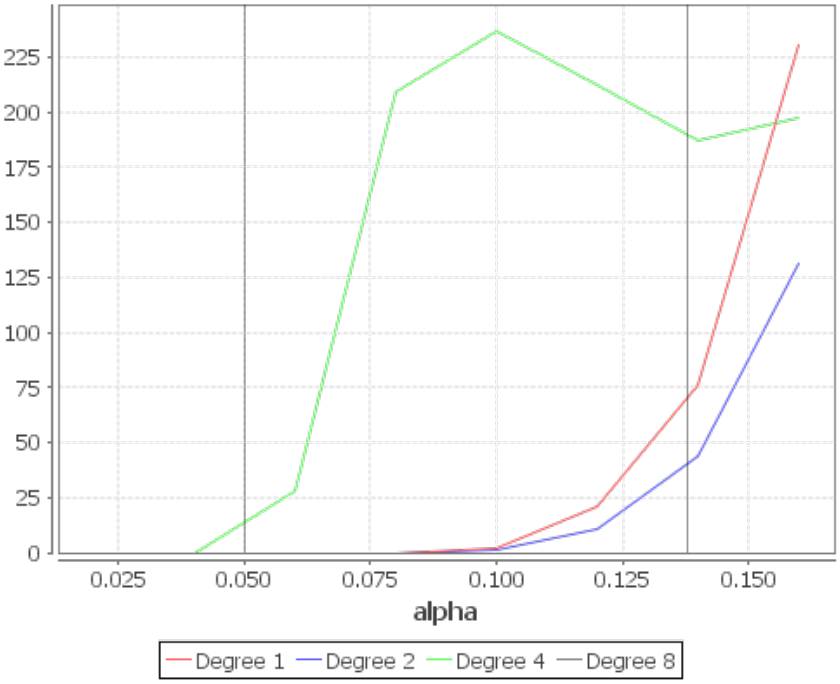

Figure 5.31: `LowerThanAvg` VS alpha for the spin glass states investigation with 100 neurons and degrees {1,2,4,8}.

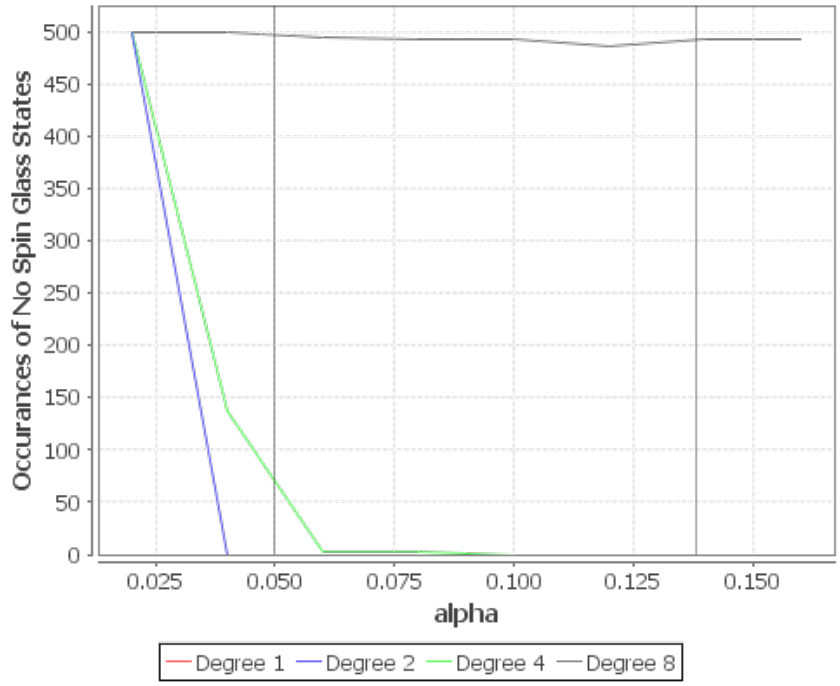

Figure 5.32: `NoSpinGlass` VS alpha for the spin glass states investigation with 100 neurons and degrees {1,2,4,8}.

Chapter 6

Evaluation

6.1 Analysis of Results

6.1.1 Hopfield Experiments

Our implementation reconfirms the results obtained in the previous studies by [13] and [16]. That is as the degree of a strong attractor is increased its basin size also increases, and for large enough degree the basin size tends towards 0.5 * neurons. In a Hopfield network containing two strong attractors, by fixing the degree of the first strong attractor and increasing the degree of the second strong attractor in stages, it is demonstrated that the basin size of the first strong attractor decreases whilst the basin size of the second strong attractor increases.

6.1.2 Stochastic Experiments

One Strong Attractor, Start Location S1

The stochastic experiments containing one strong attractor show that at zero temperature the strong attractor is stable, but as temperature increases it become more and more unstable/incorrect until it reaches a point where it is only getting about half of the strong attractor pattern correct (shown in their values for NCorrect).

Two Strong Attractors, Start Location S1

The stochastic experiments containing two strong attractors show that is is possible to overpower an undesired (first) strong attractor with another strong attractor, given that the more desirable (second) strong attractor is of bigger degree than the undesirable attractor, and that the network is at low temperature. Again as the temperature in the network increases both attractors become less stable, eventually both reaching a stage where they are only getting half of their respective attractor patterns correct (shown in their values for NCorrect).

An interesting observation in the two strong attractor experiment is the sum of square differences values for the strong attractors when they are of very different degrees (e.g. 8 and 2, 8 and 64). At low temperature, the weaker attractor has a sum of square differences that is higher than the 0.25N limit that is expected if all neurons have equal chance of being in the firing state. Also given that the strong attractors are 25% to 75% hamming distance apart, this suggests that there is a pull towards the stronger attractor in the network.

One and Two Strong Attractors, Start Location S2

We know from the Hopfield experiments that the basin size, for a large enough degree of strong attractor, tends towards 50% of the number of neurons in the network. In the start location S2 experiments the network was started in a random state. We would therefore expect that half of the experiments started within 50% hamming distance of the strong attractor (and therefore have a fairly good chance of converging towards the strong attractor) and that the other half did not (and therefore have little chance of converging towards the strong attractor).

Our experiments show that there tends to be more instability in NCorrect at low temperature, centred around NCorrect = 0.5 * neurons. At high temperature NCorrect was fairly stable at roughly NCorrect = 0.5 * neurons. These observations could be used to support the idea of being able to 'bounce' out of undesired states when there is temperature that is not too high.

6.1.3 Energy Landscape Experiments

Mixture States

The results show that for the classical theory the mixture states have higher average energy than the attractors in most cases for alpha in regions A or B. There is a chance at higher alpha (but still within regions B) that the average energy of the mixture states may be lower than that of the average energy of the attractor states, however this is very small. This could be interpreted as another small error in the model (the memory is known to not perfect). We observe that as alpha increases, there is more chance of the lowest minima being a mixture state rather than an attractor state. We also notice that in region B there is a small chance of the network no having any stable mixture states.

When a strong attractor is added to the network, the observations start to change. Even when the strong attractor has a degree of only two we no longer see mixture states being the lowest observed minima in the network at high alpha. As the degree of the strong attractor is increased, it becomes more likely that region B will have no stable mixture states, until it reaches a point where region B always has stable mixture states again. We also see that as the degree of the strong attractor is increased the probability of the mixture states having higher energy than the strong attractor becomes less and less.

Spin Glass States

For the classical theory, we observe that until $alpha = 0.1$ the average of the spin glass states doesn't have lower average energy than that of the attractors. However, from the moment we get into region B, we start observing spin glass states as the lowest observed stable state in the network, and this becomes more likely as alpha is increased. Also all networks in the classical theory were observed to contain spin glass states.

When we introduce a strong attractor to the network, we observe that even at a low degree value the spin glass states stop becoming the lowest stable minima in the network and that the spin glass states also never have lower energy than that of the strong attractor, but may at high alpha have lower average energy than the average

of all trained attractors in the network. Also, as degree is increased, it becomes more and more likely that the lower the alpha of the network, the less likely it is that the network will not contain any spin glass states.

6.2 Critical Analysis of Implementation

6.2.1 Stochastic Two Experiment, Behaviour When DegreeOne = DegreeTwo

We observe that the first strong attractor is still more accurate than the second when both strong attractors are of even degree. This behaviour occurs even when more runs of the experiment are carried out. We also observe that in the Hopfield network experiment containing two strong attractors, the first strong attractor has a bigger basin of attraction than the second strong attractor when they are of equal degree.

To investigate this issue the experiment was run again using the S1 starting position, the attractors having equal degree of 8 and with no random patterns in the network. 1000 iterations of the experiment were performed and the results are detailed below:

Temperature	Attractor One Avg. NCorrect	Attractor Two Avg. NCorrect
0.0	793.41	787.38
0.8	727.09	727.87
1.6	615.55	616.27
2.4	553.90	553.17
3.2	516.99	516.68
4.0	487.24	487.33
4.8	456.60	457.23
5.6	425.15	423.45
6.4	405.02	405.78
7.2	399.96	399.06
8.0	400.90	400.54

Table 6.1: Results for Stochastic-Two experiment run with 100 neurons, degreeOne = degreeTwo = 8 and no random patterns

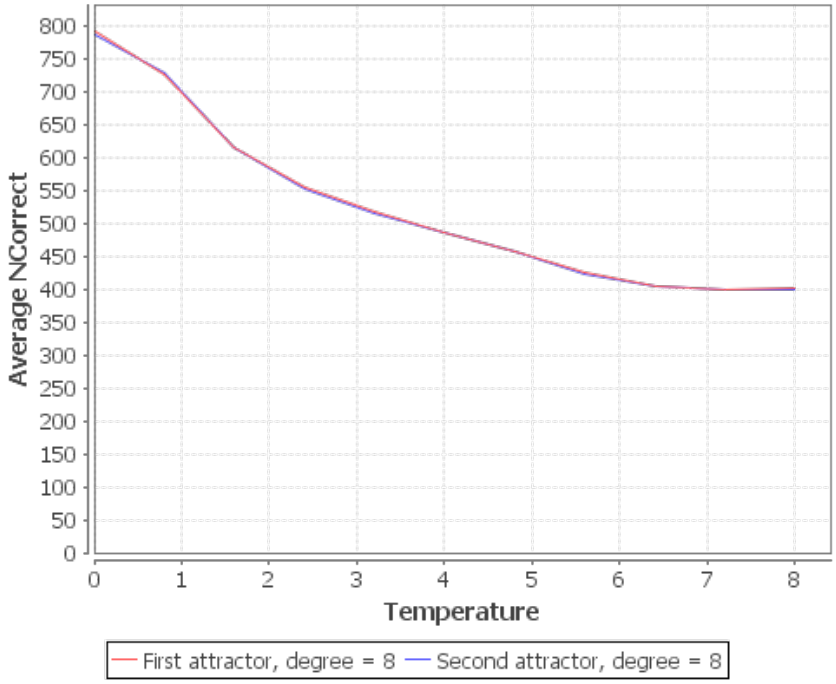

Figure 6.1: Average NCorrect VS temperature for Stochastic-Two experiment run with 100 neurons, degreeOne = degreeTwo = 8 and no random patterns

This suggests there may be an issue with the technique used for generating random noise patterns for the network, as by removing noise patterns we observe that for equal degree the strong attractors have almost identical behaviour. Random patterns are distributed in the area 25% to 75% hamming distance away from the first strong attractor. This area is also where the second strong attractor is located.

6.2.2 Mixture States Methodology

To investigate mixture states in a network containing strong attractors, an alternative method should have been used based on the theorems in [8], described below:

Algorithm 1 Mixture States Investigation

```
Set number of neurons in network, N
for various values of degree do
    Generate strong attractor pattern, P
    Choose small f = 3, 5...
    Initialise trainingSet
    for i = 0; i < degree; i++ do
        Generate new pattern by flipping f bits in P
        Add this pattern to the trainingSet
    end for
    Train network with trainingSet
    Generate new pattern by flipping f bits in P
    Check for stability in new pattern
    At high degree the new pattern should be a stable mixture state
end for
```

We did not have time during this project to implement this methodology.

6.2.3 Loss of Precision in Java double Data Type

The Java double data type is not precise due to the way it is represented in binary. The BigDecimal data type is more precise than the double data type, but not as intuitive when writing code for mathematical operations. Double and BigDecimal are used interchangeably in the implementation.

While the lack of precision for double should not make an impact on the results, it nonetheless exists and should be acknowledged.

6.2.4 Test Coverage

Although a test suite was created to confirm that various functions in the implementation behaved as expected, the robustness of this test suite is debatable.

During the process of creating the test cases, calculations needed to be done by hand for the inputs to obtain an expected answer for a function. For some functions it was highly impractical and time consuming to perform these calculations by hand for networks containing more than 10 neurons. Our experiments are run with a minimum of 100 neurons, hence the test suite doesn't properly cover the implementation for the parameters that are used.

6.2.5 Reuse of Implementation

Reuse or future adaptation of this implementation would be limited for those who can not program or do not know Java, as there is not a 'user friendly' option for performing experiments e.g. GUI based interface.

Chapter 7

Conclusion and Future Work

7.1 Implication of Results

In this project we have continued previous research into associative memory and the Hopfield model by investigating the stochastic Hopfield model, an adaptation of the Hopfield model that includes the concept of (non-physical) temperature and is therefore more likely to resemble what happens in a biological system. We show that as temperature increases in a stochastic network with two strong attractors, the attractor that performs better at zero temperature will continue to be the better of the two as temperature increases but there is a point at which both attractors are only half correct and increasing temperature doesn't cause their behaviour to change.

Linking this back to previous work in the field and to reattachment theory, we show that it is still possible to retrain a network with a new strong attractor to reduce the effects of a previously added strong attractor, despite the presence of temperature in the system. This mimics the retraining process being developed in reattachment therapy, where the aim is to reduce the effect of the initially acquired insecure attachment (the undesired strong attractor) by helping the patient to form a secure bond with their inner child (the new desired strong attractor).

We have also investigated several predictions related to the energy landscape of the Hopfield network and how this changes between the classical theories and the strong attractor theories, as described below.

7.2 Comparison With Our Predictions

Our first theory is that the quality of the landscapes should not change when implemented with strong attractors.

If we look at the energy landscape experiments with spin glass states as an example, we notice that the landscape does change when a strong attractor is implemented in the network. When the energy landscape is implemented with a strong attractor spin glass states stop being the lowest energy minima in the network, and as the degree of the strong attractor is increased it becomes more likely at small alpha that there will be no observed spin glass states in the network (compared to classical theory where for large enough p, spin glass states were always observed).

However we only investigated a sub set of the spin glass states in all the networks evaluated as finding every single spin glass state in a network would take far too long to calculate (2^N complexity). This shouldn't have too big an impact on the evaluation of the energy landscape but should nonetheless be acknowledged.

Our second theory is that the attractors in landscapes A and B represent secure attachment states, while the ripples (spin glass states) represent the insecure attachment types.

In our experiments investigating spin glass states, we only investigated the energy landscape for classical Hopfield theory and a Hopfield network containing one strong attractor. We did not investigate landscapes containing multiple strong attractors, which could cause more or different changes to the energy landscape. Introducing strong attractors to the Hopfield network appears to remove or reduce the spin glass states in the network, so this theory may not hold when strong attractors are involved.

Our final theory is that landscape A represents a securely attached person, as by increasing temperature it is easy to get out of the undesirable spurious state (insecure attachment) and into the desired attractor state (secure attachment). In contrast, landscape B represents an insecurely attached person, as increasing temperature won't help much in getting out of the spurious state.

While we haven't been able to directly prove or disprove the latter statements of this theory in our experiments, we have acquired evidence of there being some type of instability in region B of the network, compared with region A and C, through our experiments relating to the energy landscape. The experiments show that region B was more "error prone" (e.g. occurrences of spin glass states as the lowest energy minima in a network or spin glass states having lower average energy than the average energy of the retrieval states).

7.3 Future Work In This Area

The following issues highlighted in this investigation could be considered as starting points for further research in this field:

- An alternative method for noise pattern generation in the deterministic and stochastic experiments. The current implementation, while it avoids clusters of attractors from forming, causes some irregularities in the results when two strong attractors are of equal degree.
- Develop other heuristic methods for investigating spin glass states. While the method we have demonstrated in this report is adequate, it would be interesting to compare the results between different methodologies to check that the same behaviours are observed.
- The mixture state investigation should be redone using the methodology suggested in the evaluation chapter, as it has more resemblance to theoretical studies on the topic.
- The energy landscape experiments should be run for networks containing multiple strong attractors, as this investigation only covered classical Hopfield network theory and networks containing one strong attractor. It would be useful to know

the effect that multiple strong attractors have on the energy landscape of a network as it may be different.

- It should be checked to see if introducing strong attractors to a Hopfield network creates new types of spurious states that are not observed in classical Hopfield theory. This investigation focuses on the types of spurious states we know about from classical Hopfield theory, but doesn't consider the possibility that the landscape changes because the spurious states have changed.

Bibliography

[1] *Attachment: Volume One of the Attachment and Loss Trilogy*. Pimlico.

[2] *Patterns of attachment: A psychological study of the strange situation*. Hillsdale NJ, 1978.

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

[4] Daniel J. Amit, Hanoch Gutfreund, and H. Sompolinsky. Storing infinite numbers of patterns in a spin-glass model of neural networks. *Phys. Rev. Lett.*, 55:1530–1533, Sep 1985.

[5] Daniel J. Amit, Hanoch Gutfreund, and H. Sompolinsky. Information storage in neural networks with low levels of activity. *Phys. Rev. A*, 35:2293–2303, Mar 1987.

[6] Daniel J Amit, Hanoch Gutfreund, and Haim Sompolinsky. Statistical mechanics of neural networks near saturation. *Annals of Physics*, 173(1):30–67, 1987.

[7] cs.wisc.edu. Htcondor documentation. `http://research.cs.wisc.edu/htcondor/manual/`. Accessed: June 2013.

[8] Abbas Edalat. Capacity of strong attractor patterns to model behavioural and cognitive prototypes.

[9] Abbas Edalat. Neural networks, attachment and therapy. Lecture at Imperial College, November 2012.

[10] Abbas Edalat and Federico Mancinelli. Strong attractors of hopfield neural networks to model attachment types and behavioural patterns. In *Proceedings of IJCNN 2013*, 2013.

[11] git scm.com. Git documentation. `http://git-scm.com/documentation`. Accessed: June 2013.

[12] John Hertz, Anders Krogh, and Richard G. Palme. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

[13] Wael Al Jishi, Niklas Hambuechen, Razvan Marinescu, Mihaela Rosca, and Lukasz Severyn. Attractor neural networks for modelling associative memory, January 2013.

[14] junit team.github. Junit documentation. `https://github.com/junit-team/junit/wiki`. Accessed: June 2013.

[15] Mary Main and Judith Solomon. Discovery of an insecure-disorganized/disoriented attachment pattern. 1986.

[16] Federico Mancinelli. Modelling attachment types with hopfield networks. Master's thesis, Imperial College London, 2012.

[17] Oracle. Java programming language documentation. `http://docs.oracle.com/javase/6/docs/`. Accessed: June 2013.

[18] World Health Organisation. The world health report 2001 - mental health: New understanding, new hope. `http://www.who.int/whr/2001/en/whr01_en.pdf`. Accessed: June 2013.

[19] Dean Petters. Implementing a theory of attachment: A simulation of the strange situation with autonomous agents. In *Proceedings of the Seventh International Conference on Cognitive Modelling*, volume 7, pages 226–231. Citeseer, 2006.

[20] Dean Petters, Everett Waters, and Felix Schönbrodt. Strange carers: Robots as attachment figures and aids to parenting. *Interaction Studies*, 11(2):246, 2010.

[21] Hoffman RE. Computer simulations of neural information processing and the schizophrenia-mania dichotomy. *Archives of General Psychiatry*, 44(2):178–188, 1987.

[22] Amos J Storkey and Romain Valabregue. The basins of attraction of a new hopfield learning rule. *Neural Networks*, 12(6):869–876, 1999.

[23] Amos J Storkey and Romain Valabregue. The basins of attraction of a new hopfield learning rule. *Neural Networks*, 12(6):869–876, 1999.

[24] Wikipedia. Attachment theory. `https://en.wikipedia.org/wiki/Attachment_theory`. Accessed: February 2013.

Appendix A

Tables of Results

A.1 Hopfield Experiments

A.1.1 One Strong Attractor

Configuration: 100 neurons, 16 random patterns, experiment run 800 times.

Degree	Average Basin Size (to 3 d.p.)
1	1.858
2	31.361
4	48.209
8	49.010
16	49.786
32	50.000

Table A.1: Results from the Hopfield-One experiment, run with 100 neurons and 16 random patterns (averages from 800 experiments).

A.1.2 Two Strong Attractors

Configuration: 100 neurons, first strong attractor with fixed degree of 8, 8 random patterns, experiment run 800 times.

Degree	Average Basin Size, First Attractor (to 3 d.p.)	Average Basin Size, Second Attractor (to 3 d.p.)
1	49.075	1.178
2	49.008	3.620
4	47.720	15.595
8	42.369	39.759
16	21.214	47.004
32	5.669	49.054

Table A.2: Results from the Hopfield-Two experiment, run with 100 neurons, first strong attractor with fixed degree 8 and 8 random patterns (averages from 800 experiments).

A.2 Stochastic Experiments

Note: For these experiments, figures for average NCorrect (Avg. NCorrect) and average sum of square differences (Avg. SoSD) are given to 2 decimial places.

A.2.1 One Strong Attractor, Start Location Type S1

Configuration: 800 neurons, first strong attractor with fixed degree of 32, 20 random patterns, experiment run 1000 times.

Temperature	Avg. NCorrect	Avg. SoSD
0.0	800.00	0.00
3.2	520.84	101.16
6.4	457.68	150.93
9.6	399.43	205.94
12.8	400.54	203.71
16.0	400.00	204.06
19.2	399.69	204.33
22.4	399.64	204.35
25.6	399.88	204.10
28.8	399.64	204.34
32.0	399.93	204.05

Table A.3: Results from the Stochastic-One experiment run with 800 neurons, strong attractor with fixed degree 32 and 20 random patterns from start location S1 (averages from 1000 experiments).

Configuration: 800 neurons, first strong attractor with fixed degree of 32, 80 random patterns, experiment run 1000 times.

Temperature	Avg. NCorrect	Avg. SoSD
0.0	795.20	4.80
3.2	520.27	101.70
6.4	455.08	153.58
9.6	399.71	205.60
12.8	399.40	204.87
16.0	399.98	204.07
19.2	400.13	203.89
22.4	400.11	203.89
25.6	400.02	203.96
28.8	399.74	204.24
32.0	399.86	204.10

Table A.4: Results from the Stochastic-One experiment run with 800 neurons, strong attractor with fixed degree 32 and 80 random patterns from start location S1 (averages from 1000 experiments).

A.2.2 Two Strong Attractors, Start Location Type S1

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 1, experiment run 500 times.

	Attractor One		Attractor Two	
Temperature	Avg. NCorrect	Avg. SoSD	Avg. NCorrect	Avg. SoSD
0.0	795.20	4.80	493.85	306.15
0.8	733.91	11.14	475.36	269.74
1.6	621.38	43.11	450.54	214.04
2.4	556.20	78.38	435.49	199.02
3.2	520.29	101.68	427.26	194.75
4.0	497.07	118.63	421.83	193.88
4.8	481.25	130.99	418.23	193.99
5.6	465.29	144.79	415.25	194.84
6.4	446.60	162.06	409.72	198.94
7.2	412.79	194.65	400.94	206.54
8.0	395.88	210.56	400.37	206.12

Table A.5: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 1 and 80 random patterns from start location S1 (averages from 500 experiments).

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 2, experiment run 500 times.

	Attractor One		Attractor Two	
Temperature	Avg. NCorrect	Avg. SoSD	Avg. NCorrect	Avg. SoSD
0.0	796.80	3.20	523.70	276.30
0.8	735.26	9.79	483.10	261.86
1.6	620.50	43.99	453.31	211.19
2.4	556.22	78.38	436.62	197.89
3.2	519.78	102.19	428.30	193.68
4.0	497.51	118.19	422.30	193.43
4.8	480.42	131.79	418.61	193.62
5.6	465.07	145.02	414.78	195.31
6.4	446.49	162.08	410.92	197.68
7.2	409.27	198.21	402.22	205.32
8.0	401.83	204.65	399.53	206.81

Table A.6: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 2 and 80 random patterns from start location S1 (averages from 500 experiments).

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 4, experiment run 500 times.

	Attractor One		Attractor Two	
Temperature	Avg. NCorrect	Avg. SoSD	Avg. NCorrect	Avg. SoSD
0.0	796.80	3.2	607.95	192.05
0.8	737.96	7.07	550.72	194.33
1.6	620.35	44.07	454.18	210.19
2.4	554.03	80.45	436.16	198.32
3.2	519.67	102.26	428.20	193.78
4.0	497.56	118.15	421.97	193.74
4.8	480.52	131.71	417.41	194.83
5.6	466.51	143.58	415.07	195.00
6.4	443.05	165.57	409.77	198.83
7.2	403.88	203.56	400.82	206.57
8.0	401.82	204.58	399.04	207.44

Table A.7: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 4 and 80 random patterns from start location S1 (averages from 500 experiments).

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 8, experiment run 500 times.

	Attractor One		Attractor Two	
Temperature	Avg. NCorrect	Avg. SoSD	Avg. NCorrect	Avg. SoSD
0.0	797.66	2.34	712.93	87.07
0.8	734.86	10.18	654.69	90.38
1.6	620.29	44.09	536.63	127.76
2.4	555.99	78.50	467.17	167.25
3.2	518.37	103.61	431.41	190.52
4.0	497.08	118.61	423.51	192.17
4.8	479.85	132.39	419.47	192.72
5.6	460.29	149.75	415.06	195.00
6.4	428.46	180.12	406.73	201.77
7.2	406.57	200.65	401.27	205.96
8.0	399.95	206.18	401.20	204.95

Table A.8: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 8 and 80 random patterns from start location S1 (averages from 500 experiments).

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 16, experiment run 500 times.

	Attractor One		Attractor Two	
Temperature	Avg. NCorrect	Avg. SoSD	Avg. NCorrect	Avg. SoSD
0.0	789.74	10.26	790.788	9.21
1.6	616.63	47.89	615.48	49.20
3.2	497.58	124.31	517.50	104.50
4.8	428.31	183.61	468.89	143.08
6.4	402.46	205.00	405.22	202.30
8.0	400.48	204.73	399.91	205.35
9.6	400.31	204.15	399.86	204.65
11.2	399.40	204.80	399.24	204.96
12.8	399.29	204.80	400.02	204.09
14.4	399.66	204.39	399.90	204.16
16.0	399.12	204.10	399.68	204.35

Table A.9: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 16 and 80 random patterns from start location S1 (averages from 500 experiments).

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 32, experiment run 500 times.

	Attractor One		Attractor Two	
Temperature	Avg. NCorrect	Avg. SoSD	Avg. NCorrect	Avg. SoSD
0.0	671.61	128.39	799.23	0.77
3.2	429.59	192.39	518.82	103.15
6.4	411.12	197.51	447.57	161.02
9.6	400.15	204.86	397.23	207.90
12.8	400.05	204.15	399.77	204.42
16.0	399.99	204.06	399.92	204.13
19.2	399.93	204.07	400.20	203.81
22.4	399.99	204.00	399.92	204.08
25.6	399.80	204.18	399.97	204.03
28.8	400.00	203.97	399.98	203.99
32.0	400.21	203.76	400.05	203.92

Table A.10: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 32 and 80 random patterns from start location S1 (averages from 500 experiments).

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 64, experiment run 500 times.

	Attractor One		Attractor Two	
Temperature	Avg. NCorrect	Avg. SoSD	Avg. NCorrect	Avg. SoSD
0.0	571.75	228.25	798.40	1.60
6.4	412.84	195.79	455.25	153.39
12.8	399.92	204.31	400.21	204.07
19.2	399.94	204.06	400.36	203.65
25.6	399.95	204.03	399.98	204.00
32.0	400.05	203.92	399.78	204.17
38.4	400.02	203.96	399.99	203.98
44.8	400.06	203.90	399.84	204.13
51.2	400.16	203.79	399.82	204.15
57.6	399.95	204.01	399.99	203.96
64.0	400.13	203.84	400.07	203.89

Table A.11: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 64 and 80 random patterns from start location S1 (averages from 500 experiments).

A.2.3 Two Strong Attractors, Start Location Type S2

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 1, experiment run 1000 times.

Temperature	Attractor One Avg. NCorrect	Attractor Two Avg. NCorrect
0.0	411.20	401.47
0.8	399.36	399.71
1.6	408.45	399.62
2.4	405.51	402.89
3.2	399.29	398.73
4.0	397.76	400.31
4.8	399.60	399.73
5.6	400.14	399.90
6.4	399.73	400.36
7.2	395.73	399.56
8.0	400.43	400.16

Table A.12: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 1 and 80 random patterns from start location S2 (averages from 1000 experiments).

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 2, experiment run 1000 times.

Temperature	Attractor One Avg. NCorrect	Attractor Two Avg. NCorrect
0.0	382.40	396.51
0.8	398.00	399.08
1.6	405.82	397.75
2.4	396.29	400.47
3.2	398.29	399.13
4.0	404.70	399.91
4.8	394.27	399.87
5.6	404.62	401.16
6.4	401.93	400.53
7.2	397.30	399.72
8.0	400.42	399.61

Table A.13: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 2 and 80 random patterns from start location S2 (averages from 1000 experiments).

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 4, experiment run 1000 times.

Temperature	Attractor One Avg. NCorrect	Attractor Two Avg. NCorrect
0.0	395.20	396.11
0.8	402.97	398.65
1.6	395.73	400.91
2.4	397.68	400.88
3.2	393.61	398.98
4.0	404.28	399.67
4.8	400.85	399.99
5.6	405.48	400.24
6.4	400.30	400.52
7.2	399.60	400.35
8.0	401.54	400.38

Table A.14: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 4 and 80 random patterns from start location S2 (averages from 1000 experiments).

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 8, experiment run 1000 times.

Temperature	Attractor One Avg. NCorrect	Attractor Two Avg. NCorrect
0.0	404.42	395.51
0.8	416.54	396.72
1.6	399.95	397.05
2.4	395.13	399.59
3.2	397.15	402.32
4.0	401.79	400.20
4.8	501.17	400.56
5.6	400.28	400.35
6.4	397.25	399.11
7.2	400.92	400.14
8.0	400.34	400.41

Table A.15: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 8 and 80 random patterns from start location S2 (averages from 1000 experiments).

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 16, experiment run 1000 times.

Temperature	Attractor One Avg. NCorrect	Attractor Two Avg. NCorrect
0.0	389.12	408.48
1.6	403.91	406.41
3.2	403.30	400.91
4.8	399.62	398.24
6.4	400.82	399.27
8.0	400.69	400.60
9.6	400.56	400.75
11.2	400.04	399.64
12.8	399.96	400.07
14.4	399.82	399.81
16.0	400.01	399.92

Table A.16: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 16 and 80 random patterns from start location S2 (averages from 1000 experiments).

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 32, experiment run 1000 times.

Temperature	Attractor One Avg. NCorrect	Attractor Two Avg. NCorrect
0.0	404.00	388.06
3.2	399.56	404.11
6.4	399.35	397.91
9.6	399.83	401.11
12.8	400.13	398.63
16.0	399.97	399.83
19.2	400.15	399.97
22.4	399.96	399.66
25.6	399.97	399.86
28.8	400.12	399.93
32.0	400.03	400.01

Table A.17: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 32 and 80 random patterns from start location S2 (averages from 1000 experiments).

Configuration: 800 neurons, first strong attractor with degree of 8, 80 random patterns, second strong attractor with degree of 64, experiment run 1000 times.

Temperature	Attractor One Avg. NCorrect	Attractor Two Avg. NCorrect
0.0	402.32	398.50
6.4	401.01	397.48
12.8	400.22	400.37
19.2	400.10	400.14
25.6	400.12	399.90
32.0	399.86	399.93
38.4	399.87	400.06
44.8	399.92	399.99
51.2	399.93	400.15
57.6	400.04	400.03
64.0	400.09	400.08

Table A.18: Results from the Stochastic-Two experiment run with 800 neurons, first strong attractor with fixed degree 8, second strong attractor with fixed degree 64 and 80 random patterns from start location S2 (averages from 1000 experiments).