Imperial College London

Department of Computing

Modeling Associative Memory and Strong Attractors with Restricted Boltzmann Machines

Author:
Julian Varghese

Supervisor:
Prof. Abbas Edalat

Submitted in partial fulfilment of the requirements for the **MSc Degree** in **Computing Science / Artificial Intelligence** of Imperial College London

September, 2013

Abstract

Childhood mental health disorders are becoming a growing issue in modern societies causing several psychosocial problems, not only during childhood, but also at adulthood without a proper treatment. A secure attachment type in developmental psychology is an integral component of an individual's psychological capability to cope successfully with stress and to retain psychological resilience. Hence, enforcing secure attachment types in children potentially reduces the risk of suffering from mental disorders. A new neurocomputationally based psychotherapeutical method, that deals with attachment theory is to retrain an individual's associative memory with a secure attachment type to reduce its associations with any potential insecure attachment types.
Several experiments have been carried over with Hopfield Networks to represent an individual's associative memory, to show neurodynamical effects of retraining the memory with different attachment types. In our report we represent the human's associative memory with a more elaborated network, the Restricted Boltzmann Machine (RBM). RBMs have several advantages over Hopfield Networks in terms of representational power as artificial neural networks that we will introduce. They consist of a visible layer and an additional hidden layer to represent a memory part, which is accessible by senses and another part which can be thought of a subconscious memory respectively. The RBM is trained with Hinton's training method, called Contrastive Divergence that approximates a gradient descent learning very efficiently.
As a first part of our experiments we want to obtain an ideal topology of visible and hidden units and an ideal Learning rate to obtain a network that is just capable of supervised pattern recognition and learns training examples in the fastest way with a sufficiently low error rate.
In the second and last part we will investigate the strengths of two competing Strong Attractors which are referring to attachment types. Strong Attractors are generated as binary pattern that are going to be learned multiple times. Generally, the neurodynamical effects in any kind of artificial neural network of learning so called Strong Attractors are surprisingly less frequently analyzed in published papers, though, from a neurocognitive point of view, repetitive learning happens in everyday life. We will analyze their effects by measuring their association strength in terms of **Average Recall Probability**, **Basin of Attraction** and **Categorization Strength** which makes use of the subconscious part of the memory and will introduce a new term, called **Investment Dip** that characterizes the dynamics of categorization with an RBM that is trained with two Strong Attractors, one representing a secure attachment type, the other one representing an insecure attachment type. As a conclusion of our experimental results we can show a successful dominance of the secure attachment type in all three measurements by retraining the network adequately.

Acknowledgments

I would like to give special thanks to:

- **Professor Abbas Edalat**, who supervised the whole progress of my project and provided useful advices not only on the title and the main direction of this project but also on how to solve particular problems we had to combat to get the results of the report that we aimed to achieve.
- My daughter **Evelyn**, the thought of her reminds me every day of the beautiful wonder of life, where every thought of her encourages me to carry my life on with hope, pride and satisfaction.
- My mother **Omana** and my father **Sabu**, my brother **Steve** and my sister **Lea** for their their faith in me and their support, both financially and emotionally.
- All the friends and acquaintances I have got to know during my stay in London. Their accompaniment, both during my working time and leisure time , provided a nice time which made it possible to enjoy the stay in a foreign country and to enrich my overall abroad experience.

Contents

1 Introduction **7**
1.1 Motivation . . . 7
1.2 Contribution . . . 11
1.3 Related Works . . . 12

2 Theoretical Background **14**
2.1 The McCulloch-Pitts neuron in the Hopfield-Model . . . 14
2.2 Representation of the Associative Memory . . . 14
2.3 Learning patterns . . . 15
2.4 Energy function and attractors . . . 15
2.4.1 Spurious States . . . 16
2.5 Strong Attractors . . . 17
2.6 Stochastic Hopfield-Networks . . . 17
2.7 Boltzmann Machines . . . 19
2.8 Restricted Boltzmann Machines . . . 21
2.8.1 Learning with Contrastive Divergence . . . 22

3 Experiment Design **23**
3.1 Overview . . . 23
3.2 Model and Implementation subject to analysis . . . 23
3.3 Input, Output and Parameters of the Model being evaluated . . . 24
3.4 Effect of strong attractors . . . 25
3.5 Software and Hardware Environment . . . 26

4 Implementation **27**

5 Experiment Results **33**
5.1 Finding an ideal Learning rate and an ideal ratio of visible and hidden units 33
5.2 Effect of strong attractors . . . 41
5.2.1 Recall Probability of two strong attractors with incremental learning 41
5.2.2 Recall probabilities, Learning and forgetting . . . 45
5.2.3 Basin of attraction . . . 47
5.2.4 Strength of categorization . . . 56

6 Evaluation **60**

7 Conclusion and Future Work **61**
7.1 Implications of the Experiment Results . . . 61
7.2 Future Work . . . 63

Index **64**

References **66**

1 Introduction

1.1 Motivation

Childhood mental health disorders are becoming a growing issue in modern societies with a suggested prevalence rate of around 10 % among children aged 5-16 in UK according to [Bradshaw, 2011] in 2004. This increases the risk of leading them into cheerlessness and a general reduction of quality of life at their young age and later at their adulthood leading to numerous long term absences from work without a curing or mitigating treatment. A secure attachment type in developmental psychology is an integral component of an indivual's psychological capability to succesfully cope with stress and retaining psychological resilience [Mikulincer M, 2007]. Therefore, enforcing secure attachment types in children potentially reduces the risk of suffering from mental disorders at a later time. A new psychotherapeutic approach based on attachment theory and with a neurocomputational justification based on modeling the associative memory of an individual is being developed under the direction of Prof. Abbas Edalat [1].
This project deals with a specific type of neural networks for modeling an individual's associative memory: The Restricted Boltzmann Machine (RBM) which is an extension of the Hopfield Network. Both Types of networks will be described in detail in the following chapter. The Hopfield Network is a commonly used type of network to represent the associative memory and it is an example of a wider class of dynamical physical systems that may be thought of instantiating "memories" as stable states associated with minima of a suitably defined system energy [Gurney, 1997]. The learning impact on Hopfield Networks with so called Strong Attractors is studied in the papers [**Edalat and Mancinelli, 2013**] and [**Tucker, 2013**]. Strong Attractors represent learned patterns which have been learned more than once, they are therefore described by a positive integer **d**, which stands for the **degree** [2] to indicate how many times that pattern had been learned. In the following subsection **Related Works** we give a brief description on the works of [Edalat and Mancinelli, 2013] and [Tucker, 2013] which indicate the success both mathematically and experimentally of retraining a Hopfield Network towards a secure attachment type, when it was initially trained as an insecure one. As a partial conclusion of our report results, we would like to show if we can also apply this sort of successful retraining using the more elaborate RBM.
Compared to the Hopfield Network, Boltzmann Machines consists of a one extra hidden layer. Visible units as the input/output layer interact with the environment. The underlying neurons are stochastic, that is, the firing states are calculated with a probability distribution which is inspired by models of statistical mechanics. In this way, the state of a neuron gets more uncertain the more a given temperature as an additional stress factor is increased. This creates a non-deterministic behavior and is a key difference of human brain-processes compared to deterministic computation. In this report we use the temperature of 1 which has no unit of measurement, which is a comparably low

[1]Department of Computing, Imperial College London, UK
[2]also called **multiplicity**

temperature to provide little background noise for the evaluation of the units [3].
The term *Restricted* Boltzmann Machines (RBM) refers to a specific Boltzmann Machine where the connections among the units are limited in the following way: There are only connections between visible and hidden units allowed what makes computation much more feasible since the number of connections between the units grow linearly instead of growing exponentially if we add more units to the system.

Hence, the Learning algorithm for stored patterns terminates within practical calculation time[4] and providing a network model elaborate enough to learn stored binary patterns accurately.

In this thesis, the practical application of this model is to represent learned patterns as attachment types. The term *attachment*[5] within the scope of developmental psychology was introduced by John Bowlby and it represents a mutual behavioral relationship of an infant and its primary caregiver. Given different ways of emerging relationships in childhood, children develop a certain type of attachment which can also be accompanied by some other minor attachments . Attachment is relevant to cognitive, as well as emotional, children's development, since secure attachment enforces independence, leading the child to explore its environment whereas insecure attachment rather suppresses this. Based on the *strange situation* research, conducted by Mary Ainsworth which evaluates the interaction between the infant, its caregiver and strangers, the infant's attachment(s) can be observed, categorizing attachments into a secure and an insecure one (Type B). The insecure one is divided into three different subtypes (Insecure-avoidant: Type A, Insecure resistant: Type C and Insecure disorganized: D) [Flanagan, 2008].
The Restricted Boltzmann Machine is the main model subject to the analysis of stored patterns for representing psychological attachment types in an associative memory. That is, we use this model for supervised pattern recognition with a training set. The visible units correspond to certain psychological attachment types which can be thought of patterns that have been learned during childhood. So called Strong Attractors, which are patterns that have been learned multiple times, stand for patterns learned with a higher learning-intensity or exposition-intensity. Hence, one can simulate the learning process of different evolving attachment types the whole network converges to by applying Restricted Boltzmann Machines and Strong attractors as the simulation framework. By learning a new Strong Attractor that represent a different attachment type we can analyze the effects of retraining the Network which can build the basis for Psychotherapy for people with a pathological or so called insecure attachment type.
The motivation to extend the simplistic Hopfield network is derived by representing the associative memory in a more sophisticated way with a higher neurophysiological fidelity, since multi-layered networks are also inspired by Psychoanalysis where one layer represents the basic Input and Output and another represents an important inner representation of learned Inputs, e.g in terms of a conscious and unconscious part [Wedemann et al., 2013]. A computationally relevant reason to extend the single

[3]Details in 2.6 and Equation 2.9
[4]using a an upper calculation time boundary of 30 seconds for learning a a whole training set
[5]see [Flanagan, 2008] for a more detailed explanation within development psychology

layer Hopfield network is the commonly known XOR-Problem in neural computation [Hertz et al., 1991] where higher correlations amid the Input cannot be derived with a single layer network.

We will also describe the so called spurious states in a Hopfield network. As a very easy way to make a Hopfield network recall false patterns is just to inverse the input bits of stably learned patterns. Applying a further hidden layer and extending the additive delta-rule from Hopfield network to the Contrastive Divergence Learning algorithm, inverse inputs will be handled as different inputs and will not be recalled[6].

An RBM with Contrastive Divergence-Learning[7] approximates gradient descent learning and acts a a practical learning algorithms with fast convergence. However there exists only less knowledge on how to set or obtain values of relevant parameters of the model. Those parameters are the learning-rate, the number hidden units given a number of visible units. Several systematic experimental simulations will determine ideal parameters for the learning process and the topology of the network which can serve as an ideal implementation model for future RBM-applications. Further simulations will illustrate the competition between two strong attractors which will represent two different attachment types.

By continuously increasing the multiplicity[8] of the second attractor we will show that after some time the second attractor gets dominant and supersedes the first strong attractor in terms of average probability of retrieval, basin of attraction in a supervised learning setting and categorization strength in an unsupervised learning setting. Thus, a theoretical basis of psychotherapy is established, aiming to retrain a patient's mind suffering from an insecure attachment type towards a secure attachment type.
The basin of attraction declares the maximum degree of perturbation of an original learned pattern that can be applied so that the network would still reconstruct original pattern correctly. Hence the basin of attraction also represents a stress or noise factor in the input-evaluation of one individuals memory which can be tolerated. Since it is known, that in average the levels of Dopamine are lower of an individual with an insecure attachment type [Strathearn, 2011] we can interpret the kind of noise caused by low Dopamine levels within our model either by increasing or descreasing the multiplicity of of an insecure attachment or secure attachment type respectively or by interpreting the low Dopamine level as a cause that induces perturbation in the secure attachment type and we can check by the basin of attraction of that secure attachment type whether an individual is still able to converge to the original secure attachment type.

The categorization strength of a Strong Attractor S is the number of patterns in the training set which will be classified in a unsupervised learning setting to the category of S. The category of a pattern is determined by taking a trained RBM, loading the input layer with that pattern and forward this pattern to the hidden layer, the values of the hidden units finally define the category. Since we use the hidden layer for the assigning of categories, categories can be thought of an inner representation of an individual's

[6]that means: mostly not recalled, depending on the error-bounds of our learning loop, described in 4
[7]see [Hinton, 2002]
[8]in our context, multiplicity of a pattern is the number times that pattern occurs in the training set, in some other literature also called degree or just d

associative memory that cannot be accessed directly to the environmental senses. The neuropsychologically inspired equivalent for this hidden part of the memory can be seen as a subconscious memory in our model, a memory which is not regarded in the single layer Hopfield Network.

1.2 Contribution

Following realizations or results haven been worked out within the project on our own:

- An Implementation of a Restricted Boltzmann Machine (RBM) learning with Contrastive Divergence[9], that is tested in a setting where we vary exhaustively the numbers of visible and hidden units and the Learning rate to obtain an ideal topology of a Restricted Boltzmann Machine and an ideal Learning rate. Ideal refers to a Network setting which terminates its learning successfully[10] as the fastest. An interesting result that we have obtained is that given a number of visible units[11] and an ideal Learning rate there exists an ideal number of hidden units, no matter how many examples we have in our training set. It turned out that for the two ideal Learning-rates that we have figured out, there exists an ideal ratio of the number of hidden units over the number of visible units for a wide range of 100 to 1000 visible units.

- We test RBMs to analyze the strengths of two competing strong attractors for binary pattern recognition and measure them in terms of:
  - **average recall probability**
  - **average basin of attraction**
  - **categorization strength**, with observing and explaining a neurodynamical phenomenon, that we call **Investment Dip**, which describes an initial loss of categorization strength of a second attractor, after which categorization strength increases and dominates the the categorization strength of the first strong attractor.

- Hopfield networks learning with standard Hebbian learning suffer from so called spurious inverse states leading them to achieve a maximum Basin of Attraction of 50%. We show that our RBM can reach **larger Basins of Attraction** for Strong Attractors which have a degree that is high enough and can be learned in a feasible time[12]

- **Overfitting in RBMs:** It is commonly known that artificial neural networks tend to overfit the training set if its structure gets to complicated by adding to many units or layers or by training the network with too many learning epochs. Neurodynamical overfitting is a phenomenon in which the artificial neural network fits the training data (almost) perfectly but get less accurate when it is applied

[9]This learning method is used for pattern recognition of binary patterns and was invented by [Hinton, 2002] and will be described in detail in our theoretical background chapter

[10]Successfully means, the training set which contains simple, random and binary patterns has to be recalled below a defined error-rate, details in 3

[11]Keep in mind that the number of visible units is depended on the input size

[12]For all our simulations we used a gradient descent approximating learning algorithm where a time limit has to be defined within learning must be terminated, we set it at 30 seconds for a customary laptop machine

on a test set that is not part of the training set. Within the analysis on the basin of attraction we also measured a behaviour of overfitting using RBMs with Contrastive Divergence Learning. Since our training patterns are all randomly generated and strong attractors are assigned randomly as well, we cannot provide a test set with real world data which contains the real classification of the data. But what we can do to get a test set is to take the training set and add random pertubations to the patterns of the training set and measure the basin of attraction. We will see in our experiments that if we use many more hidden units than visible units the basin of attraction decreases, the largest basin could be achieved by using almost as many hidden units as visible units. This interferes the rule that we have figured our in the first part, which stated that using around five times more hidden units than visible units makes the network most efficient in terms of learning time. Therefore, as a rule of thumb we stated that the ideal number of hidden units is slightly more than the number of visible units which provides a good compromise of aiming an efficiently learning network that is still resistant to random noise. It has to be noted that for a RBM which is used for categorization the number of hidden units is much less than visible units[13].

1.3 Related Works

Two main works deal with Neural Computation on Hopfield Networks and the explicit analysis of Strong Attractors

- **Strong Attractors of Hopfield Neural Networks to Model Attachment Types and Behavioural Patterns** from [Edalat and Mancinelli, 2013]: One of the first papers that defines Strong Attractors and their learning impact on Hopfield Networks. Strong Attractors represent attachment types in developmental psychology and behavioral patterns in psychology and psychotherapy. Higher Stability of Strong Attractors in Hopfield Networks in terms of Error Probability and the size of the basin of attraction is proven mathematically and supported by various simulations.

- **Stochastic Hopfield Networks To Model Secure And Insecure Attachment Types With Noise** from [Tucker, 2013] is a B.Eng thesis which analyzes the basin of attraction and spurious states among other findings of two competing Strong Attractors in stochastic Hopfield Networks with increasing temperatures. The dominance of a second Strong Attractor using high degrees, is shown by various simulations which enforces the potential of success in psychotherapy based on stochastic Hopfield Networks if we model Strong Attractors as attachment types.

- Restricted Boltzmann machines (RBMs) have diverse applications in which they are successfully running as in classification, feature learning, collaborative filtering

[13]For the measuring of categorization strength of strong attractors in an RBM with 100 visible units, we could achieve the highest categorization strength with 34 hidden units for a given strong attractor with different multiplicities

and/or pattern recognition. In our report we focus on classification and pattern recognition with Strong Attractors. [Ruslan Salakhutdinov, 2007] describes a successful application of RBMs for collaborative filtering, such as representing user's rating of movies in a large data set of Netflix movies.

2 Theoretical Background

2.1 The McCulloch-Pitts neuron in the Hopfield-Model

Since we want to model neural networks inspired by neuroscience, where a neuron is a unit in a network of 10^{11} neurons [Hertz et al., 1991, Chapter two] he have to define a model of a neuron, that is its connections to other neurons (synapses) and the computation of input and output. A very simple approach is the neuron-representation of the **McCulloch-Pitts** neuron where a neuron is called a unit. In the following we use the mathematical terms adopted from [Hertz et al., 1991, Chapter two]. This neuron model is illustrated in Figure 2.1, the unit fires if the weighted sum $\mathbf{\sum_j w_{ij} n_j}$ of the inputs reaches or exceeds the threshold $\boldsymbol{\mu_i}$

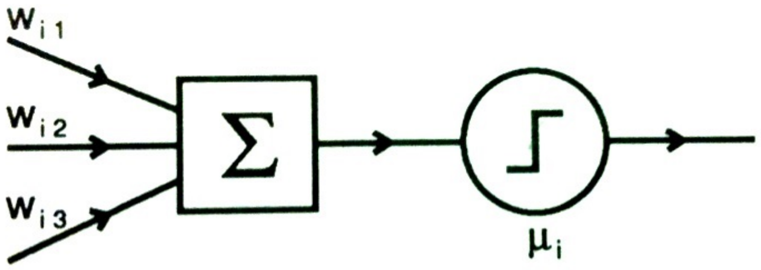


Figure 2.1: Schematic diagram of a McCulloch-Pitts neuron, from [Hertz et al., 1991, Chapter two]

The state of a unit is either -1 (not-firing) and +1 (firing), which will be denoted by S_i for the $i^t h$ unit. For mathematical convenience the the threshold is assumed to be 0. Thus S_i is defined as following ([Hertz et al., 1991, Chapter two]):

$$S_i := sgn(\sum_j w_{ij} S_j) \tag{2.1}$$

with sgn being the sign-function:

$$sgn(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{2.2}$$

Furthermore in the Hopfield-Model of this project there are no self-connections, which means $\mathbf{w_{ii} = 0}$

2.2 Representation of the Associative Memory

A set of patterns is defined by the expression ξ_i^μ, where every single point of one pattern is represented by one unit in the Hopfield-Model. The patterns are labelled by

$\mu = 1, 2, ..., p$, while the units of the network are labelled as i = 1,2,...,N. Hence, the Associative Memory problem can be described as retrieving a stored or learned pattern μ among all the other learned p patterns that most closely resembles an input-test-pattern C_i. Both the stored patterns given through ξ_i^μ and the test pattern C_i representing points, thus, they are 1 or -1 on each site i. *Maybe add some information about the distance-measurement, add the conceptual figure of attractors*

2.3 Learning patterns

For the simple case where we consider just to learn one pattern ξ_i, following condition must hold:

$$sgn(\sum_j w_{ij}S_j) = \xi_i \ \ (for\ all\ i) \tag{2.3}$$

Since $\xi_j^2 = 1$ rule 2.3 is true if we make

$$w_ij = \frac{1}{N}\xi_j\xi_j \tag{2.4}$$

where $\frac{1}{N}$ is used as a constant for proportionality.

A simple way to learn p different patterns is to to refine rule 2.4 according to the **Hebb rule**:

$$w_{ij} = \sum_{\mu=1}^{p} \frac{1}{N}\xi_j^\mu\xi_j^\mu \tag{2.5}$$

It has to be mentioned that this rules symmetric weights, $w_{ij} = w_{ji}$.

2.4 Energy function and attractors

A current state of a Hopfield network that is described by its current weights and different unit-states S_i can be defined by introducing the idea of an energy function H [14].:

$$H = -\frac{1}{2}\sum_{ij} w_{ij}S_iS_j \tag{2.6}$$

The most general term for H , from the theory of dynamical systems, is **Lyapunov function** or **Hamiltonian function** in statistical mechanics and for a neural network in general an energy function exists if the connection-weights are symmetric ([Hertz et al., 1991, Chapter two]) which they are in our model. If an updated S_i' remains the same, then the energy function does not change as well. If $S_i' = -S_i'$ then it easy to see from equation

[14]This is a simplified equation by omitting the terms for i=j since in our model $w_{ii} = 0$

2.6 that energy can only decrease. Thus the energy decreases or remains the same every time an S_i changes. If each state of a network, given by the values of all unit-states S_i (for all i(=), is assigned an energy-value according to 2.6 we will get an energy surface as shown in figure 2.2. After a Hopfield network has learned some patterns, the local minima of the energy surface represent these learned patterns. The dynamics can be thought as a motion of a partical (which represents a current state of the network) on this energy-surface. Under gravity-influence and friction, from any starting point of the particle, the particle slides down until it comes to rest at one of those local minima. Therefore, those local-minima (and thus the learned-patterns) are called **attractors**, where each attractor represents a learned pattern unless it is an attractor representing a spurious pattern which are defined in the next subsection.

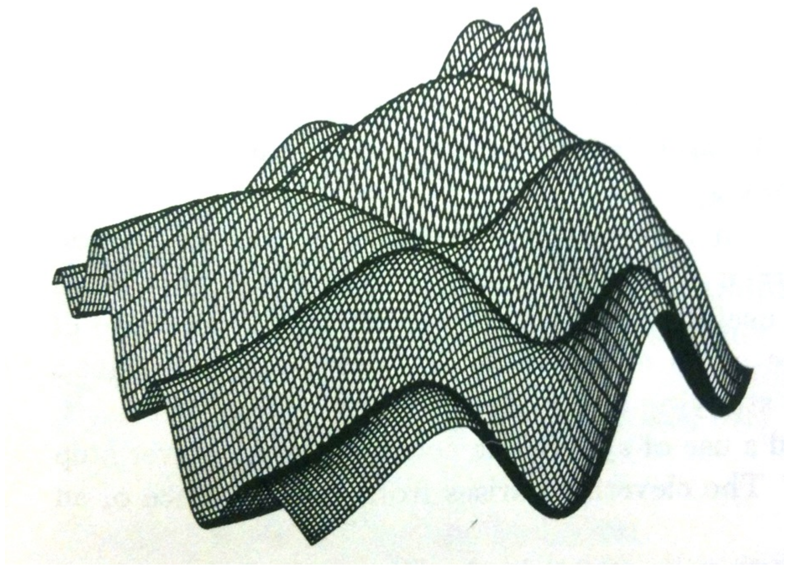


Figure 2.2: Imagining the energy as a landscape, the z-axis as the height axis is the energy and the 2^N corners (for all possible states S_i) are represented by the xy-plane, from [Hertz et al., 1991, Chapter two]

The **basins of attraction** are the valleys or catchment areas around each attractor. It is the Hamming-distance to the attractor. That is, if we have a basin of attraction of 10 for one attractor, we can flip in average 10 bits of the attractor-pattern and the network will successfully converge to the original attractor pattern.

2.4.1 Spurious States

With the Hebb-rule from equation 2.6 we get a dynamical system that has attractors. But the learned patterns are not the only attractors. Trivially the reverse states $-\xi_i^{\mu}$ of attractors lead to the same energy value which makes them as attractors as well. Then, there are also so called mixture states $-\xi_i^{mix}$ which are not equal to any learned patterns but it can be shown that they are linear combinations of an odd number of patterns [Amit et al., 1985a]. Lastly, it has to be mentioned, that for large p (the number of patterns to be learned) there are attractors that are not correlated with any finite number of the p learned patterns, which are called **spin-glass states** due to a close relation to spin glass models in statistical mechanics [Amit et al., 1985b].

2.5 Strong Attractors

Strong Attractors are learned patterns, that have been learned multiple times. Thus, it is defined by a parameter that we will call **d** throughout this thesis. The parameter **d** stores the number of times the pattern has been learned, it is also called the degree or multiplicity of the attractor. One of the main goals in this project is to show the strength of an attractor, that is with which probability the whole network recalls an attractor varying the multiplicity of an attractor. Thus, the multiplicity d represents the intensity a specific pattern has been learned which can simulate the way how adults experienced behaviors in their childhood which assigns them to an attachment type in terms of developmental psychology. By introducing new strong attractors that represent a different attachment type we can analyze the effects of retraining the Network which can represent the basis for Psychotherapy for people with a pathological or so called insecure attachment type.

2.6 Stochastic Hopfield-Networks

Inspired by statistical mechanics for magnetic systems influenced by temperature (for detailed description, see [Hertz et al., 1991, Chapter two]) we replace the previous dynamics of states in a Hopfield-network by a stochastic rule:

$$S_i = \begin{cases} +1 & \text{with probability } g(h_i) \\ -1 & \text{with probability } 1 - g(h_i) \end{cases} \tag{2.7}$$

This rule 2.7 is applied whenever unit-state S_i is updated. Function $g(h_i)$ takes h as an argument which represents the state of S_i depending on the summed up weighted contribution of the other units j described in the McCulloch-Pitts model, also called the **local field** of unit i [15]:

$$h_i = \sum_j w_{ij} S_j \tag{2.8}$$

The g-function represents a probability-value. Firstly, it should obviously increase if the value h_i, representing the votes of the other units, increases. Secondly, inspired by temperature dynamics in magnets, where thermal fluctuations tend to flip the spins of a magnet, the function g will also be influenced by adapting this phenomenon. That

[15]This is a simplified equation by omitting the threshold-value.

is, if the given parameter temperature increases, then the state of h_i should also tend to flip, making the temperature-parameter to a certain degree of noise in the states of the network. These two influences are mathematically described in the following way according to the **Ising-model with Glauber dynamics** [Glauber, 1963]to finally define the g-function as sigmoid shaped function:

$$g(h) = f_\beta(h) = \frac{1}{1 + exp(-2\beta h)} \tag{2.9}$$

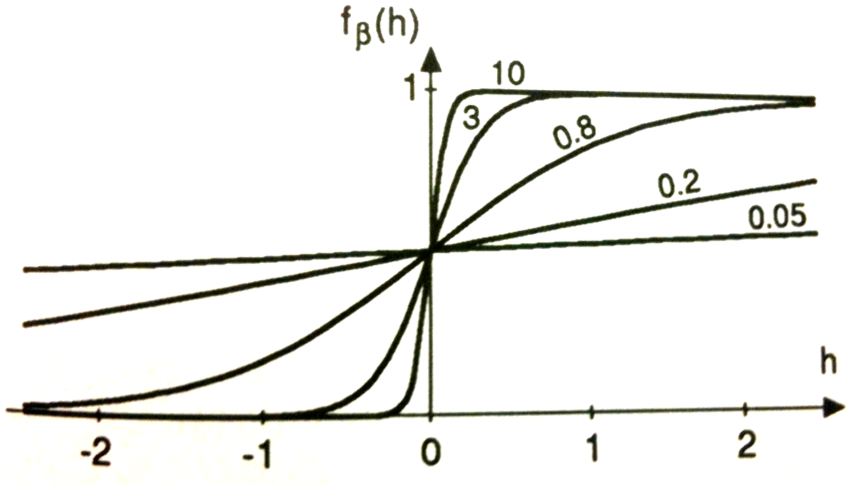


Figure 2.3: the sigmoid-shaped probability function for g(h) for several β-values, from [Hertz et al., 1991, Chapter two]

Figure 2.3 illustrates this sigmoid function by several values of β which is related to the absolute temperature in Kelvin. This sigmoid function is also called the **logistic function** in this context.

$$\beta = \frac{1}{k_B T} \tag{2.10}$$

where k_B is the Boltzmann constant, which will be assumed to be 1 in our model for mathematical convenience. Looking at the function-definition in 2.9 and the illustrated figure 2.3 it is noteworthy that the temperature T in Kelvin, whose minimum is 0, controls the steepness of the g-function near h = 0. With T getting very low, the g-function becomes a step-function representing the previous McCulloch-Pitts model. With incrementing T the sharp threshold-behavior is softened up in a probabilistic way. Since the g-function is symmetric (1-g(h) = g(-h)) we can express the equation 2.9 as follows:

$$Prob(S_i = \pm 1) = \frac{1}{1 + exp(\mp 2\beta h_i)} \tag{2.11}$$

2.7 Boltzmann Machines

[16] Boltzmann machines are an extension of the Hopfield-Networks, their units are divided into two layers, one with visible units and another with hidden units. They are called Boltzmann-machines because the probability of the states of all units is given by the Boltzmann distribution of statistical mechanics [Hertz et al., 1991, Chapter seven]. The visible layer can represent the Input and Output of the network, while the hidden layer is connected to the visible layer but has no connections to the outside world [Hertz et al., 1991, Chapter seven]. All units can be evaluated in the same stochastic way as we did in the Hopfield network with equations for the local field h_i(2.8) and the g-function (2.9) and the whole network in a given state can be assigned an energy-value H as defined in Equation 2.6. It has to be mentioned that the learning process described in this subsection aims to change the weights such that a given desired probability distribution over the input patterns will be learned. That is, after the process has successfully terminated, the average retrieval-probability for a given input-pattern approximates a given desired probability given through a probability distribution. However, using the complex topology of Boltzmann Machines and a possible learning process we will introduce appears to end in a very time-consuming application, which is the reason why this thesis focuses on Restricted Boltzmann Machines and a Learning Process, called Contrastive Divergence [Hinton, 2002]. As an additional theory-deepening a possible learning process for non-restricted Boltzmann Machines is provided in the following.

All following mathematical equations in this section are (unless stated otherwise) from [Hertz et al., 1991, Chapter seven]. Thus, the Boltzmann-Gibbs distribution is given by:

$$P\{S_i\} = \frac{e^{-\beta H\{S_i\}}}{Z} \tag{2.12}$$

With $H\{S_i\}$ being the energy of the system (as defined before) in configuration S_i and Z being the normalization factor to get a probability value between 0 and 1:

$$Z = \sum_{\{S_i\}} e^{-\beta H\{S_i\}} \tag{2.13}$$

It has to be noted that the probability given through the Boltzmann-Gibbs distribution is the given probability after an equilibrium has reached. To get a learning rule to define the links involving the hidden units a new approach will be introduced, different from the Hebbian learning rule. Let the states of the visible units be labeled with an index α and the states of the hidden units be defined by the index β [17]. Thus the probability distribution $P_\alpha \beta$ can be calculated using equation 2.12. $P_a lpha$ is calculated by marginalizing over the states of β:

[16]This subsection deals as an additional chapter for non-restricted Boltzmann Machines. Since the actual neural model is implemented by restricted Boltzmann machines the reader may skip this subsection. The project analysis and results will not make any use of it. However, it is exemplary to know what the more general Boltzmann Machine is capable of if we derivate a learning rule and to understand that computation gets very costly compared to the Restricted Boltzmann Machine

[17]beta as a constant is also defined as the inverse of the temperature in our terms, but as an index it refers to the hidden units

$$P_\alpha = \sum_\beta e^{-\beta H_{\alpha\beta}} \tag{2.14}$$

Let now R_α be the desired probabilities for the α-states, thus we can define the error term E using the relative entropy of R_α and P_α:

$$E = \sum_\alpha R_\alpha log \frac{R_\alpha}{P_\alpha} \tag{2.15}$$

Having defined an error term, we can minimize the error with gradient-descent method as it is commonly known for learning in neural networks by adjusting the weights of the net as follows:

$$\Delta w_{ij} = -\eta \frac{\delta E}{\delta w_{ij}} = \eta \sum_\alpha \frac{R_\alpha}{P_\alpha} \frac{P_\alpha}{\delta w_{ij}} \tag{2.16}$$

where η is a constant defining the learning-rate. Using equation 2.15 for P_α we get out of equation 2.16 the following equation which is the central learning rule for Boltzmann-machines (details of the derivations are given in [Hertz et al., 1991, Chapter seven, p. 167]:

$$\Delta w_{ij} = -\eta\beta[\sum_{\alpha\beta} R_\alpha P_{\beta|\alpha} S_i^{\alpha\beta} S_j^{\alpha\beta} - < S_i S_j >] \tag{2.17}$$

where the conditional probability is given through Bayes-Law through:

$$P_{\alpha\beta} = P_{\beta|\alpha} P\alpha \tag{2.18}$$

If we look at the first term of the square brackets in equation 2.17, one can think of summed up value of $< S_i, S_j >$ when the visible units are bound or clamped in state α and averaged over different αs with weight of R_α. We therefore, rewrite the equation in a more convenient way:

$$\Delta w_{ij} = -\eta\beta[\overline{< S_i, S_j >_{clamped}} - < S_i S_j >_{free}] \tag{2.19}$$

The clamped term of equation 2.19 represents the **Hebbian learning** while the second term is called **Hebbian unlearning** with the system free running [Hertz et al., 1991, Chapter seven, p. 167]. The intuition behind this is, that if the clamped states $< S_i^{\alpha\beta} S_j \alpha\beta >$ are heterogeneous compared to the free states $< S_i, S_j >$ the weight w_{ij} will be non-zero in a way to reduce the error term in equation 2.16. Thus, the process of adjusting the weights converges when clamped unit correlations $< S_i, S_j >$ are equal to the free ones. For updating the weights one needs to determine the average values of $< S_i, S_j >$ for the clamped and the free term. A state flip of S_i is defined by following probability which enforces the system to tend to to lower-energy states:

$$Prob(S_i \rightarrow -S_i) = \frac{1}{1 + exp(\beta \Delta H_i)} \tag{2.20}$$

The procedure of updating the weights is very time consuming and the systems gets trapped into local minima of energy. A technique, called **simulated annealing** which initially updates the units according to 2.20 with a high temperature that incorporates more randomness and gradually decreases the temperature and then achieves equilibrium at one working temperature much sooner [Hertz et al., 1991, Chapter seven, p. 168].

2.8 Restricted Boltzmann Machines

Restricted Boltzmann Machines (RBMs) prohibit connections within the hidden layer and within the visible layer. Thus a visible unit can only be connected to a unit of the hidden layer and vice versa, therefore it is a **bipartite graph**. As mentioned in the introduction, the RBM is the neural model the whole experimental analysis and experimental results will be based on. Unless otherwise stated, all equation appearing in this subsection are taken from [Hinton, 2002].

Different from the Hopfield Network and the non-restricted Boltzmann Machines, units in the following RBM represent binary values: 1,0 [18]. Let **(v,h)** be a current configuration of an RBM, where v is a specific binary vector standing for the visible units and h is a specific binary vector standing for the hidden units. The energy E(v,h) of a given RBM is calculated as:

$$E(v,h) = \sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_j h_j w_{ij} \tag{2.21}$$

with v_i, h_j representing the states of visible unit i and hidden unit j respectively, a_i, b_j are their biases and w_{ij} is the weight between them.

Based on the energy function in 2.21 the probability distribution for an RBM in a state (v,h) with v as a specific pattern for the visible units and h for the hidden units is defined as:

$$p(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{2.22}$$

with Z being the normalizing partition function:

$$Z = \sum_{v,h} e^{-E(v,h)} \tag{2.23}$$

The probability of having the visible units in state v is obtained by marginalizing over the hidden units:

$$p(v) = \frac{1}{Z} \sum_{h} e^{-E(v,h)} \tag{2.24}$$

[18]This is mainly because recent applications and research ([Hinton, 2002]) dealing with Restricted Boltzmann machines use binary 0,1 inputs. The rest of the this thesis will keep using this convention

2.8.1 Learning with Contrastive Divergence

According to [Hinton, 2002] the derivative of the log-likelihood of training vector under the RBM with respect to a weight is:

$$\frac{\delta logp(v)}{\delta w_{ij}} =< v_i h_j >_{data} - < v_i h_j >_{model} \tag{2.25}$$

where $< v_i h_j >_{data}$ is equal to the expectation under the data distribution, this can be obtained simply by setting the input to the training vector and updating the hidden units according to the mentioned activation function and initial random weights. $< v_i h_j >_{model}$ is equal to the expectation under the model distribution and can be quickly approximated by reconstructing the visible units by updating the given states of hidden units from $< v_i h_j >_{data}$. The number of reconstruction, is annotated as k. It is proposed that using small number of k, or simply k=1, already approximates the equation above well enough to provide a good learning [Hinton, 2002]. For a higher precision level we apply a second reconstruction by updating again the hidden units and then updating the visible units another last time, that is k=2. However for the problem of using an RBM for the categorization of the input by reading out the hidden values, we have figured out that k has to be set higher for getting higher categorization-strength for strong attractors [19] Thus we obtain following weight-updating learning rule:

$$\Delta w_{ij} = \varepsilon < v_i h_j >_{data} - < v_i h_j >_{model} \tag{2.26}$$

where ε is the learning rate. Finding an ideal learning rate and the ideal number of hidden units given a number of input units is one of the main goals of our experiment results which will be presented in the next chapter. Furthermore, we will make use of following theorem, proven in [Roux and Bengio, 2007]:

Theorem 1. *Any distribution over* $0,^n$ *can be approximated arbitrarily well with an RBM with k + 1 hidden units where k is the number of input vectors whose probability is not 0.*

[19]see Experiment Design, under 3.4 for details

3 Experiment Design

3.1 Overview

Given a RBM that learns with Contrastive Divergence[20] which will be applied for binary pattern recognition to represent humans associative memory there are 4 main properties of the RBM that we would like to analyze and optimize:

- an ideal ratio of hidden units and visible units
- an ideal Learning rate
- effect of strong attractors in terms of recall probability and basin of attraction while increasing the strength, that is the multiplicity of a strong pattern.

3.2 Model and Implementation subject to analysis

The experiments presented here will evaluate several simulations of an implemented RBM that incorporates **Contrastive Divergence** as the learning algorithm which was introduced in the previous chapter. Every RBM being tested in this thesis consists of a visible layer and hidden layer where connections are first of all only existent between a visible unit and a hidden unit. Furthermore we will make use of one bias unit for visible layer and the hidden layer[21]. Bias units are generally in neural networks required for a universal approximation of the output function[22]. The RBM has been implemented in Matlab. Matlab provides a programming environment with a huge variety of mathematical tools and libraries being used by many research institutes with a large user base all over the world. Furthermore it provides very efficient operators for matrix multiplications which is the main calculation type being used in training a network and recalling a pattern from a network given a weight matrix. For the training process we initially start with a random weight matrix that has random values around 0, that is uniformly distributed between -1 and $+1$. The standard temperature is $+1$. For a better understanding of our applied model the implementation of the RBM will be listed in the next chapter 4 with detailed comments and explanations. Since the Implementation of our experiment scenarios is just trivially derivable from the following experiment description and we do not want the reader to get lost in to many pages of code-details the experimental tests and experiment results and the whole model are available in a compressed zip archive to download on the web: to download all available Matlab code and data from this link :

https://docs.google.com/file/d/0BxfnhHTIk8tqOWl5NTBNVFJWUmM/edit?usp=sharing

[20][Hinton, 2002]

[21]Details of Model and Implementation in 4

[22]An easy example to understand the importance of bias units is to imagine what would happen when all input units of a neural network are zero. In this case also the output will be zero

or accessible for markers under the folder *Matlab-Implementation* in the attachment of the submitted report.

The whole implementation code is listed in the folder *Matlab-Implementation*, the file **readme.txt** contains a brief description of all code files related to this report.

3.3 Input, Output and Parameters of the Model being evaluated

The input and output of the RBM is represented by the input layer of the RBM. We use randomly generated binary patterns as input vectors[23]. For the scope of this project an input pattern represents an attachment type of an individual and the state[24] of the currently trained RBM represents the current state of its associative memory. Initially all patterns are randomly generated and distinct. Hence every pattern has the same degree of d=1, which is called a simple pattern. Two relevant parameters of the model subject are subject to the main analysis:

- an ideal ratio of hidden units and visible units
- Learning rate

We will observe different performances of the RBM if we vary the number of hidden units given a number of visible units and the learning rate. Performance in this context only refers to the learning times within the network converges with a high accuracy rate of at least 95 percent. The accuracy will only be calculated over the training examples. There is no test set being involved in this scenario. Hence, we are only interested in the learning efficiency of explicitly given training examples. In the experiment setting described in the next subsection 3.4, we will see that the problem of overfitting[25] will appear if we choose too many hidden units, though they might lead the network to a better learning efficiency for the training examples. In the next subsection, we aim to find a balance of high learning efficiency and finding as few hidden units as possible, so that learned patterns with variable noise (basin of attraction) can be successfully recalled.

Simulations will be carried over with different numbers of examples to get an overview storage capacity of different networks. The time limit within the network has to converge is set to 30 seconds. This is considered to be an affordable time a user can wait for running an application, and during our simulations, it has been figured out to be a sufficiently long time to obtain relevant results to obtain ideal values for the two main parameters.

[23]also called as learning or training examples

[24]That is determined by currently learned weights

[25]Overfitting is a common term in Machine Learning and is a behavior of a classifier or a learning system in which the training examples will be perfectly classified but other examples being not part of the training set are increasingly falsely classified.

3.4 Effect of strong attractors

After having found an ideal learning rate and an applicable capacity of training examples we will focus the strength of Strong Attractors in an RBM of 100 visible units. Following measurements will be calculated given two strong attractors and the rest of simple patterns, where all patterns, both strong and simple patterns, are randomly generated.

- average recall probability of the two Strong Attractors of the RBM that is presented to a random pattern
- average basin size of two Strong Attractors
- categorization strength of two Atrong Attractors

We will investigate these three characteristics of Strong Attractors where the degree of the second Strong Attractor in a given training set will be continuously increased to see whether the second Strong Attractor that represents a secure attachment type can dominate the first strong attractor that represents the pathological insecure attachment type. As RBMs can also be used for unsupervised categorization of the input if we read out the states of the hidden units, we will measure the categorization strength of a Strong Attractor. That is we count the total number of patterns in the training set which are assigned to the category of the strong attractor. For determining the category we take a trained RBM and forward the input to the hidden layer just by updating the hidden layer. Using a large number of 100 visible units, even with very strong attractors [26] every pattern will have its own category. This is due to the fact, that, given a Strong Attractor, a simple pattern is in average 50 bits away from the strong one. Small pertubations of more than 20 bits will already lead a trained RBM likely to another inner representation in the hidden layer. However we have figured out that if the we update the hidden layers multiple times by backwarding and forwarding the input multiple times, which means we increase the k-value mentioned in 2.8 the hidden values converge to values which provide higher categorization strength for Strong Attractors. That is, a Strong Attractor is assigned to category where other simple patterns are also assigned or attracted to.

A Strong Attractor is implemented by increasing its occurence, we do this by adding the same pattern to the given training set. By doing this we repesent a model with continuous learning capability, since the training set increases[27].

[26]we have simulated degrees of 20 ¡d¡100 in an initial training set of 20 patterns

[27]that is, the stronger one attractor gets the larger will training set be, but the training set will never exceed capacity of the network

3.5 Software and Hardware Environment

The experiments, described as above, were carried over on following two machine types with following specifications:

1) Lab Computer at the ICL DOC, machine: project-01, HP Compaq dc8200

- Processor: Intel Core2 Duo E7400 2.80GHz
- Ram: 8 Gb
- Operating System: Linux version 3.6.0, gcc version 4.6.3 (Ubuntu/Linaro 4.6.3 -1ubuntu5)
- Matlab Version: R 2012 A, 64-Bit

2) Private Laptop, Model: Samsung 350V5c:

- Processor: Intel Quadcore: i3-3110M CPU 2.4 GHz
- Ram: 6 Gb
- Operating System: Windows 8 64 Bit
- Matlab Version: R 2012 A, 64-Bit

The first environment was the main one where all experiments and results published in this thesis were run and obtained respectively. The second environment was used firstly as a further coding platform possibility to work at home and secondly as a control environment to check whether the result of the former environment are reproducible. For example, initially, we figured out that an ideal ratio of hidden units over visible units is around 5 for several different learning rates within a certain range of numbers of visible units on both Hardware environments. Different Hardware environments might lead to different learning times but the learning-time-ideal ratio of hidden units over visible units remains the same. Our ratios were calculated on two different Hardware Environments with the same result.

4 Implementation

The goal of the Implementation is to simulate a RBM that learns binary pattern with Contrastive Divergence Learning[28]. However, there are some model-relevant parameters that have to be defined and initialized results will be reproducible on other reimplementations. For the specifications of Matlab and the Hardware environment, see 3.5. All model-relevant parameters are listed in the following:

- Network model: Restricted Boltzmann Machine with temperature-linked statistical units[29]
- Network layers: Two layers: One layer as input and output and one hidden layer
- Learning-algorithm: Contrastive Divergence Learning with k=2 reconstructions[30]
- Learning rate:0.01, 0.05, 0.1, 0.2 **subject to optimization**
- Input/Output format: binary
- number of visible and hidden units: **subject to optimization**, testing range 15-3000 visible units, 10-100.000 hidden units
- Weight Matrix initialization: random uniform distribution in the interval [-1,+1]
- Learning timelimit: 30 seconds
- error-rate under which learning terminates: 0.05 (Hamming distance between training examples and generated output

According to the definition of a RBM, our model consists of a visible layer and hidden layer and connections in-between the units are only possible between a visible and a hidden unit. Furthermore we use a bias unit for the visible and hidden layer. Image 4.1 illustrate the connection-settings in a more detailed way.

[28][Hinton, 2002]
[29]see 2.6
[30]details will be explained in the following

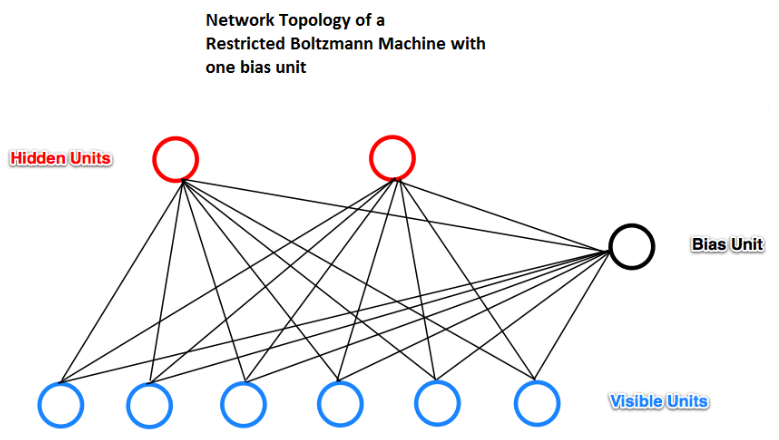


Figure 4.1: Connections in an RBM with two hidden units and six visible units with a bias unit connected to both layers, modified from [Chen, 2011]

Following code listings illustrate the initialization and the learning algorithm of the RBM. Every comment line begins with the character ′%′.

Listing 1: file RBM.m, initialization of an RBM

```
1  % Constructs a hidden and a visible layer for a
2  % Restricted Boltzmann Machine.
3
4
5  % Specify number visible (Nv) and hidden units (Nh)
6  Network.Nv=Nv+1;
7  Network.Nh=Nh+1;
8  Network.Temperature=1;
9
10 % Create Connectivity matrix that defines random
11 % connection weights (interval [−1,1], with a uniform distribution)
12 % between visbible and hidden layers, note that first visible and
13 % first hidden unit represents the bias unit
14 Network.WeightMatrix = −1 + (2).∗rand(Nv+1,Nh+1);
15 Network.visible = zeros(1,Network.Nv);
16 Network.visible(1)=1; %bias unit for visible layer
17 Network.hidden = zeros(1,Network.Nh);
18 Network.hidden(1)=1; %bias unit for hidden layer
```

Listing 1, lines 6-8: The number of hidden and visible units will be varied in our simulations where the variables Nv and Nh will be assigned real values. Our standard temperature being used is 1. For getting an ideal topology of the network which is the first main part of our simulations as mentioned in 3.3 the temperature will be fixed. As described in 2.6 a low temperature like 1 leads the whole RBM to a very deterministic behavior with less uncertainty.

Listing 2: file Learning2, Implementation of the learning algorithm

```
1  % This script is to learn all available training examples given in the
2  % matrix training by applying the method of
3  % contrastive divergence for Restricted Boltzmann Machines
4
5
6  %initialise
7  Rbm;
8
9  Learningrate=0.05;
10 lasterrors=0;
11 gooderrorrate=0;
12 epochnr=0;
```

```
timelimit_reached=false;

timelimit=30;

%start measuring time
tic;

%The Positive and Negative Term which will be summed up to adjust
%the weights and correspond to the data distribution and model distribution
%Positive=zeros(Network.Nv,Network.Nh);
%Negative=zeros(Network.Nv,Network.Nh);
converged=false;


[rows, columns]=size(training);
if Network.Nv ~= columns+1
    disp('training data does not match the input size of the Network')
else
    data =[ones(rows,1), training]; %add column of ones for the visible bias unit

    while(converged==false && timelimit_reached==false)

        % First update of hidden units by take the training data
                                % as the input
        hidden_activation = data * Network.WeightMatrix;
        hidden_probs = neuronstateprob_matrix(hidden_activation, Network.Temperature);
        hidden_states= neuronstates_matrix(hidden_activation, Network.Temperature);
        Positive = transpose(data)*hidden_probs;

        %Reconstruct the visible units
                                visible_activation = hidden_probs * transpose(Network.WeightMat
        visible_probs = neuronstateprob_matrix(visible_activation, Network.Temperature);
        visible_probs(:,1)=1;
        visible_states= neuronstates_matrix(visible_activation, Network.Temperature);
        visible_states(:,1)=1;

        %Reconstruct the hidden units a second time
                                hidden_activation = visible_probs *Network.WeightMatrix;
        hidden_probs = neuronstateprob_matrix(hidden_activation, Network.Temperature);
        hidden_states= neuronstates_matrix(hidden_activation, Network.Temperature);

        Negative = transpose(visible_probs)* hidden_probs;


        Network.WeightMatrix = Network.WeightMatrix +
                                                        Learningrate*((Positive - Negative)/rows);


        %Update a second time the visible layer to check calculate errors
        visible_activation = hidden_probs * transpose(Network.WeightMatrix);
        visible_states= neuronstates_matrix(visible_activation, Network.Temperature);
        visible_states(:,1)=1;

                                  %calculate error of network-learning
        error=0;
                                  for ex=1:rows
            error= error + (pdist([visible_states(ex,:);data(ex,:)],'hamming'))/rows;
        end

        % Stop converging process if error rate was 3 times smaller 0.05 in terms
                                  % of Hamming distance
        if (error < 0.05)
         gooderrorrate=gooderrorrate+1;
        end

        if gooderrorrate==3
            converged=true;
        end
        epochnr=epochnr+1;

        if(toc>=timelimit)
            timelimit_reached=true;
        end
    end
end
```

Listing 2, lines 27-57: These codelines incorporate the Contrastive Divergence learning algorithm described in 2.8.1. Training examples are listed row-wise in the matrix variable *training*. From then the hidden units will be reconstructed two times by using Matlabs fast matrix multiplication. The function call *neuronstateprob_matrix* will calculate the probabilities of a unit being on given it's activation function, described in 2.6 for the stochastical Hopfield network and the temperature of the model. Based on the probability a binary state will be determined by *neuronstateprob_matrix*. Both function calls are taking input matrices *hidden_activation* or *visible_activation* that contain activations of all units in the hidden or visible layer respectively. The output is a matrix as well with values for every unit of the appropriate layer row-wise for every training example. In **Line 57** we have the **central weight-updating learning rule**, described in equation 2.26, the two variables *Positive* and *Negative* correspond to the data distribution and model distribution respectively.

Line 33: Learning iteration will stop when the error[31] converges to a minimal error or a given timelimit, which was specified as 30 seconds according to 3.

Line 68-75: The error of the currently learned network is the current error measured in hamming distance between the training examples and the visible generated output. An acceptable minimum error rate was taken as 0.05, since this means, that at least 95% of the output units had the correct bit values. Furthermore we want to have the error under 0.05 for two further iterations to make sure that current calculated error was not calculated as a result of good luck.

[31]see following explanation on the error for code line 67

5 Experiment Results

5.1 Finding an ideal Learning rate and an ideal ratio of visible and hidden units

Given a number of visible units which defines the dimension of the input and output we can observe the network performance in terms of learning time and capacity while increasing systematically the number of hidden units. As an overall result we could figure out whatever visible layer size we used in the range of 15 and 3.000 units, the network performance (learning time and capacity) improved. However, there exists in ideal number of hidden units after which increasing the number of hidden units is not beneficial anymore.

The figures 5.1 5.2, 5.3, 5.4 illustrate this for 400, 600, 1000 neurons with learning rate 0.1[32] as a dip[33] within the 3D-Plot where the x-axis going left in the horizontal plane represents the number hidden units, y-axis going towards back in the horizontal plane represents the number of training examples and the z-axis going upwards vertically represents the learning time which determines the color on the surface plot just like on a heat map.

[32] We started our simulations with a learning rate of 0.1 which shows a stable behavior in terms of an ideal ratio of hidden units over visible units over a wide range of visible units. Details about simulation results with different learning rates will follow in this chapter

[33]the dip in the 3D surface plot is only slightly visible in the plot for 100 visible units as the darkest blue surface for 500 hidden units. It is not clearly visible, because 100 visible units provide a low calculation cost for for many hidden units, but the dip can be clearly deduced by comparing explicit values of its plotmatrix in the file 100files 100vuLR0.1100.mat under variable 'Plotmatrix'

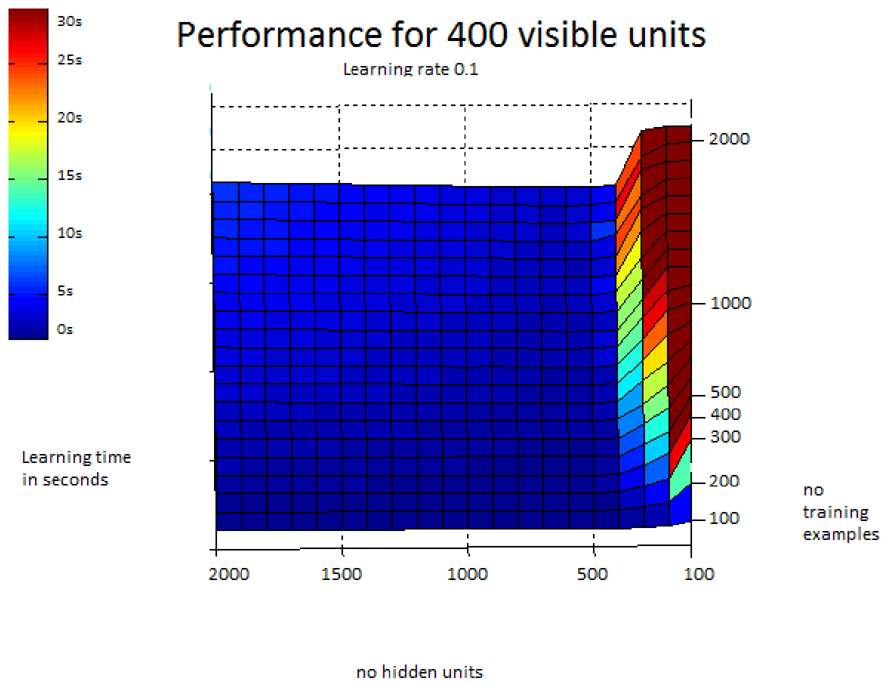


Figure 5.1: Learning Performance for 100 visible units, 100 visible units is the mainly used network input format for the further analysis of Strong Attractors, note that we still have learning times of less than 30 seconds using 100 hidden units and a training set of 300 random patterns, see 3.5 for Hardware environment details. Looking at the colours (dark blue marks lowest learning times) of the surface plot and by reading out the explicit values of the plotmatrix we can observe the fastest learning times for hidden units which is 5 times more than the used number of visible units. Explicit values of the plotmatrix are listed in the file 100files 100vuLR0.1100 .mat under the variable 'Plotmatrix'

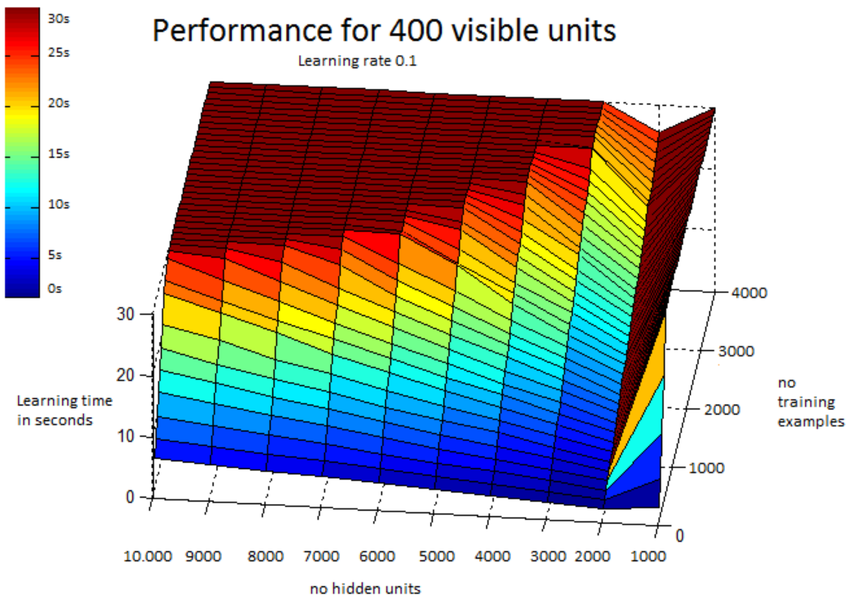


Figure 5.2: Learning Performance for 400 visible units, the so called 3D-Dip is clearly visible

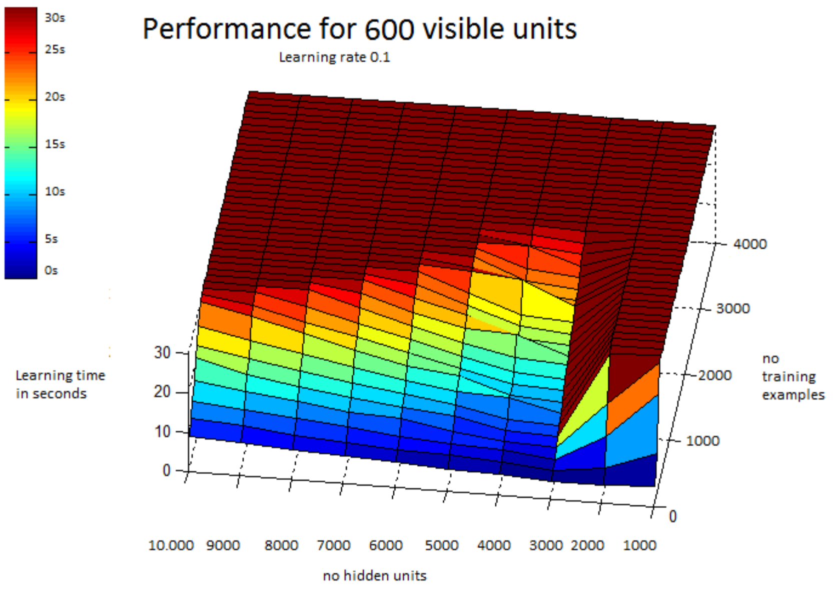


Figure 5.3: Learning Performance for 600 visible units

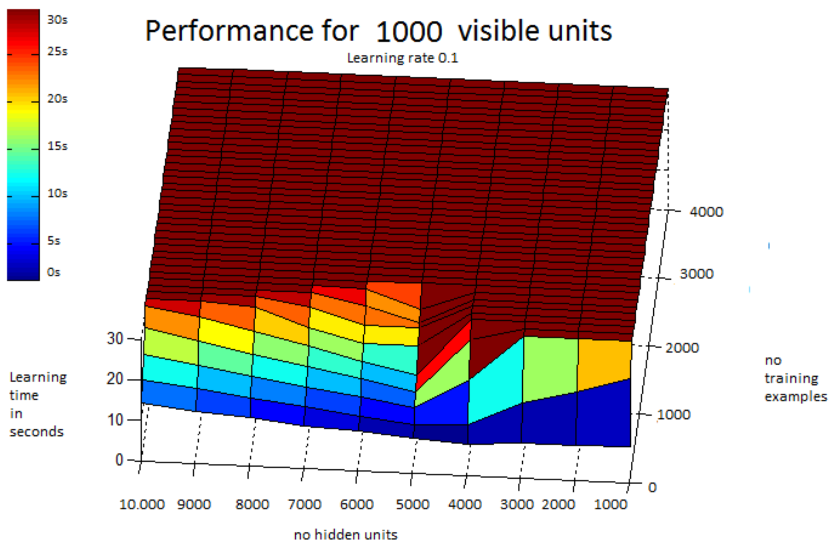


Figure 5.4: Learning Performance for 1000 visible units

Looking at the figures 5.2 to 5.4 we always see a single dip in the 3D surface plots, which means there is an ideal number of hidden units providing us with a minimum of learning time till learning terminates for throughout every number of training examples that have to be learned. A reason which accounts for this kind of phenomenon is to think that the more hidden units we add, the more is our model capable of learning more relations in between the input bits such that it fits the data-distribution in the input much better. However, we also have to take into account that by adding further hidden units the computational costs grow linearly. The dip in the plot represents a network size which is poised between having enough hidden units to approach a desired data distribution and having not to many hidden units which would slow down the construction of the output. It is also noteworthy, that the dark red areas of the 3D-Plot refer to times higher than the chosen time limit of 30 seconds. This means that those areas represent network setting[34] where the learning process had to be terminated without knowing whether the network would ever have converged.

For **400, 600** and **1000** units we can observe this dip at **2000**, **3000** and **5000** hidden units respectively.

A remarkable observation is that those ideal ratios also indicate an ideal number of hidden units providing **the highest capacity of training examples** that could be learned with a low error-rate within our given time limit[35], as one can see from the figures 5.2 to 5.4 that the dip also represents the longest color-bar with non-critical learning time values. E.g. in model with 400 visible units more than 4000 examples could be learned with a low error rate within our given time limit. That's more than 10 times more than visible input units we provide. Compared to Hopfield networks this is a huge number where a critical number of patterns to learn is less than 0.2 times the number of visible units [Hinton, 2002].

We repeated those simulations for visible units in the range of **15 to 3000** visible units and with 4 different learning rates: **0.01, 0.05, 0.1, 0.2**. The learning rate of **0.05 and 0.1** were proven to be the ones producing the fastest learning times throughout all network topologies, that is, whatever number of visible and hidden units and number training examples we have tested[36]. In the range of **15** and **500** visible units we obtain faster Learning times with **0.1** as the Learningrate, using more than 500 units leads to a faster Learning time if we set the Learning rate to **0.05**. These fastest Learning rates also provide us with a highest capacity of training examples that can be learned in the given time limit.[37] The more we differ from the Learning rate 0.05 and 0.1 we can observe longer learning times.

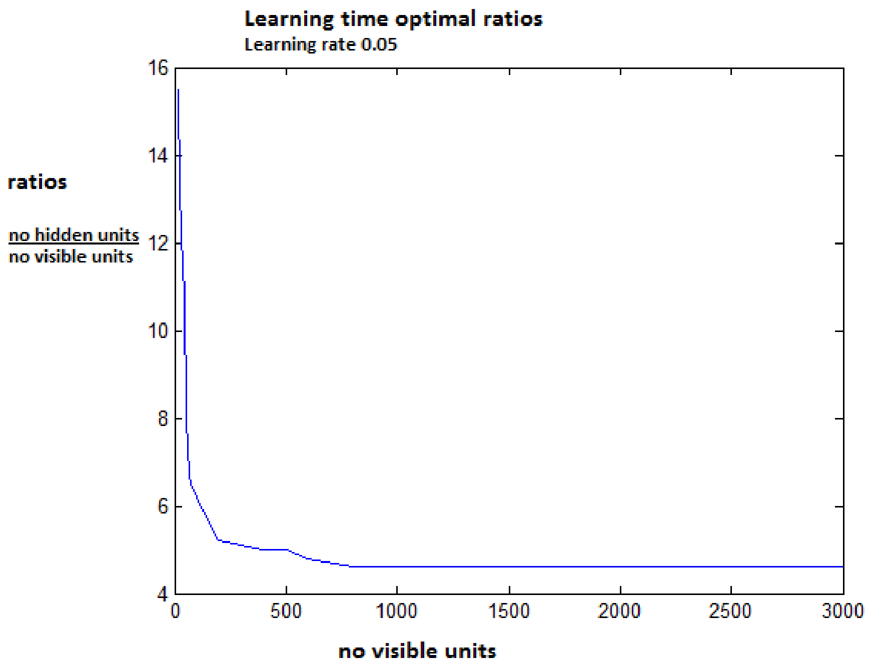


Figure 5.5: Learning time optimal ratios for different numbers of visible units, after 100 units the ratio plateaus at around 5

The next figure 5.5 illustrates a 2D-Plot which shows the ideal ratio, that is the factor of hidden units over visible units on the Y-axis versus the number of visible units in the model. Again, ideal refers to the ratio providing the fastest Learning process termination.

The ideal ratio starts to be high being 15.5 for 15 visible units, decreases quickly the more visible units we add to the model and reaches a plateau at 100+ units. From there we have a wide range with an ideal ratio of approximately 4.6 [38] The initial steep descent of the ideal ratio in the beginning range of 15-100 visible units can be explained by the fact, that with a few visible unitis we only have a small number of distinct patterns to learn. Hence, feasibility remains low when we add more hidden units. However the more visible visible we have in our model the number of distinct patterns increase exponentially and any increase of the number of hidden units has to be taken more carefully, that's why we achieve a plateau of around 4.6 as the ideal ratio after 100+ visible units.

[34]a setting with a given number of visible units, hidden units, examples to learn
[35]For recall: a low error rate is less than 5% in Hamming distance, time limit is 30 seconds
[36]see 4 for test settings and parameter-ranges
[37]See variable plotmatrix containing all the learning time values for different network topologies and different Learning rates in "DiffVu-LR-XX.mat" where XX refers to the Learning rate.
[38]Value rounded to the first decimal place

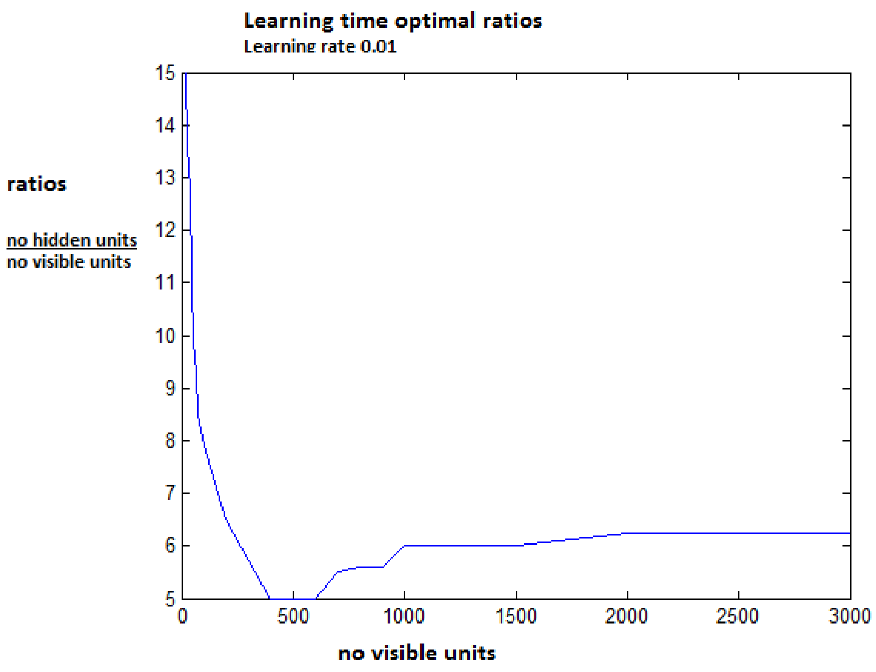


Figure 5.6: Learning time optimal ratios for different numbers of visible units, non-optimal Learning rate: 0.01

Bare in mind that for different Hardware environments we might obtain different Learning times but the ratio of hidden units over visible units remains the same. Our ratios were calculated on two different Hardware Environments (see 3.5) with the same result. The figures 5.6 to 5.7 the ideal ratio for the slower working learning rates 0.01 and 0.1:

The learning time dynamics for the Learning rate 0.01 (figure 5.6) and Learning rate 0.1 (figure 5.7) are different from the learning time-optimal Learning rate 0.05 (figure 5.5). For the learning rates 0.01 and 0.1 we observe the known behavior of a steep descent of the ideal ratios in the range of 15-100 visible units, but then we see a minimum ratio of around 5 just like in the case of the optimal learning rate 0.05 but after about 600 visible units (in case of learning rate 0.01) or 1000 visible units (in case of learning rate 0.1) we see beginning increase of the ideal ratio (more than 5) but which steepness is not as high as the initial descent in beginning range of 15-100 visible units.

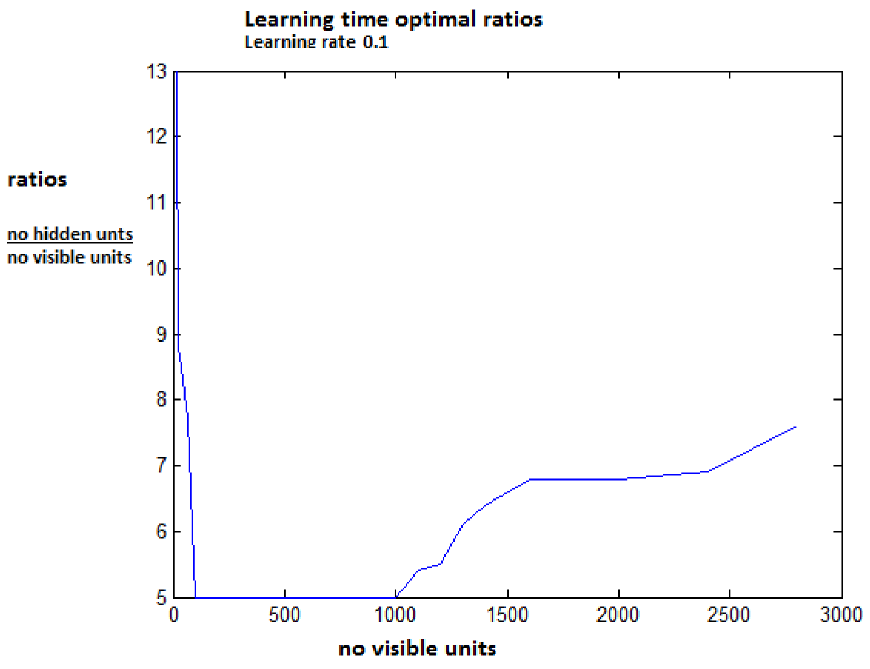


Figure 5.7: Learning time optimal ratios for different numbers of visible units, non-optimal Learning rate: 0.1

5.2 Effect of strong attractors

5.2.1 Recall Probability of two strong attractors with incremental learning

The recall probability of a pattern *pat* in a trained Rbm R is the average probability p that R will output *pat* when we set a random pattern as an input. Thus, an intuitive understanding of recall probability of *pat* is the strength with which pattern *pat* is associated in our model.

Given the definition of the probability distribution in equations 2.21 to 2.24 that is based on the energy of the RBM in the visible state that represents the pattern whose recall-probability we want to calculate. If we look at the partition function of the probability function one can easily see that we have to consider all possible binary states of visible and hidden units leading to infeasible probability calculation if we have an Rbm with many units.

In the following experiments we see the recall probabilities of two strong attractors in a test setting with an Rbm with 6 visible units and 6 hidden units learning 20 patterns. Two of the 20 patterns will become Strong Attractors, we will train the first Strong Attractor up to a multiplicity of d=5 and then add the second Strong Attractor and increase in every step its multiplicity one by one and observe the changes in recall probabilities of the two strong attractors to see whether the second Strong Attractor can dominate the first one. The second Strong Attractor's multiplicity will be increased until d=30. This is a learning scenario in which the RBM learns the pattern incrementally, that is, for making a pattern a strong one we just add an identical pattern to the training set without replacing any existing ones, and then retrain the whole network. Hence, the training set size increments one by one. In a neurocognitive way this learning scenario is inspired by the assumption that an individuals associative memory is amenable for learning new patterns, enabling it for continuous learning by retraining the network once we added a new pattern. Old patterns will not be replaced or "forgotten", the training set keeps on growing but within a practical capacity[39] of the network.

[39]a practical capacity is a training set size with which the network's learning successfully terminates in an applicable time as it was described in section 5.1. An RBM of 6 visible and 6 hidden units 100 examples could be learned in an acceptable learning time and error rate

Following setting description will give a detailed overview of the simulation scenario.
Test setting:

- Network: RBM with 6 visible units, 6 hidden units
- Initial training set: 20 simple patterns (every pattern has multiplicity d = 1)
- Training mode of Strong Attractors: **Increasing multiplicity by simple adding into the training set**
  - : Generate 20 samples to calculate average recall probabilities, every sample will be generated as following:
    - Select randomly two different patterns out of the 20 which are meannt to be Strong Attractors, calculate their Recall probabilities
    - Add multiplicity of first Strong Attractor up to d=5 by adding 4 more identical examples of the Strong Attractor, after every adding, calculate Recall probability of this Strong Attractor
    - Now Add the multiplicity of second strong attractor up to d=30 and calculate Recall probabilities of both Strong Attractors.

Figure 5.8 shows the Recall probabilities of the two competing Strong Attractors.

Looking at figure 5.8 we see that the first Strong Attractor obtains high probabilities as we increase its multiplicity step by step up to 5. Then Second strong attractor increases its multiplicity, when both have multiplicity of 5 both share a similar recall probability of around 0.1. From there we keep on increasing only the multiplicity of the second Strong Attractor and we observe higher probabilities of the second attractor, while the recall probabilities of the first one slowly decreases though its multiplicity of 5 stays the same. This is due to the increasing number of training examples, since we increase the multiplicity of the second attractor, so that the multiplicity of 5 gets relatively [40] smaller the with every step we increase the multiplicity of the second attractor. At the very right hand of the plot the second attractor has a multiplicity of 30, first one has still 5 and we have the initially 18 simple patterns, hence the training set consists of 53 examples in the end. As an overall result derived from the plot one can see that the second attractor dominates in terms of recall probability the first one once it reaches at least the same multiplicity, furthermore it also weakens the first one by reducing its recall probability. Figure 5.8 shows the results if we continue increasing the multiplicity of the second attractor up to d=100. It shows that the recall probabilities, although not linearly, keep on increasing as well while those of the first strong attractor decrease. The progress of increasing and decreasing probabilities for both strong attractors gets slower in the end[41].

[40]the relative multiplicity is thought to be the current multiplicity of a pattern divided by the number of patterns in the training set, keep in mind that the training set can contain duplicates

[41]E.g. we have an increase of 40 % for the second attractor in the multiplicity range from 1-30, this increase slows down while rising the probabilities towards 100%

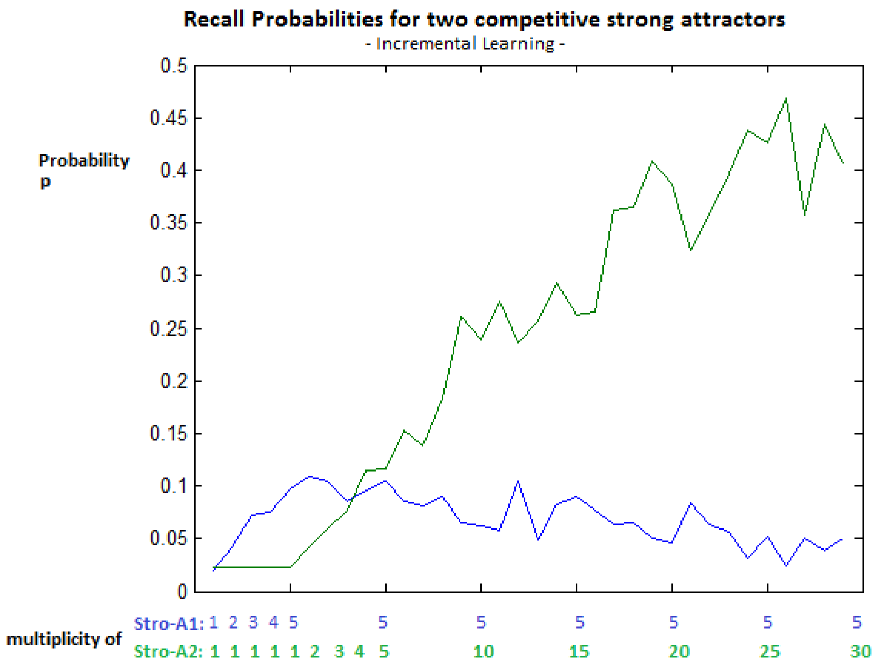


Figure 5.8: Recall probabilities in a competitive setting of two Strong Attractors, training mode: Increasing multiplicity by simple adding, see the text for details

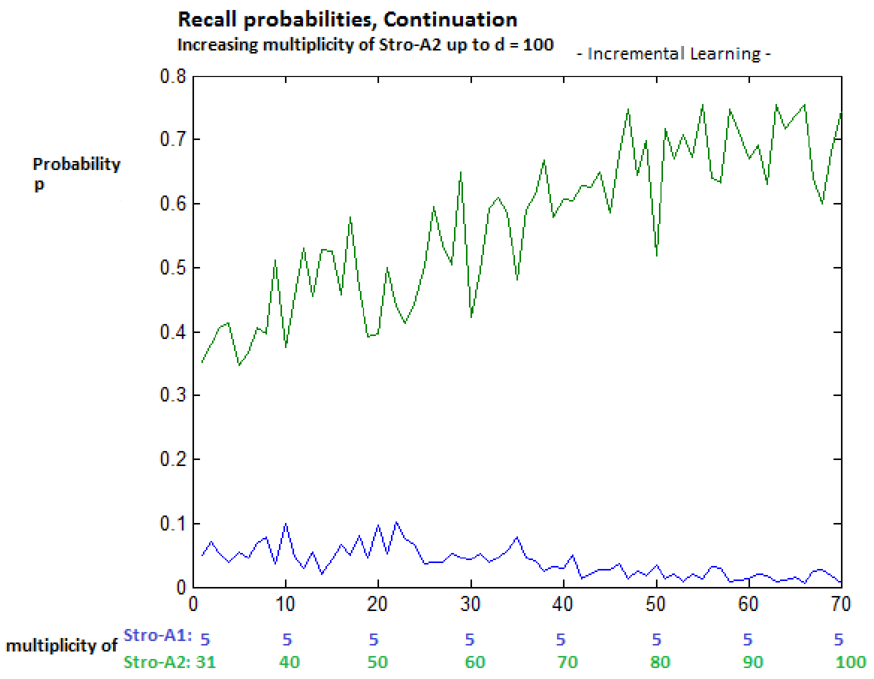


Figure 5.9: continuation of figure 5.8 see the text for details

5.2.2 Recall probabilities, Learning and forgetting

In the following test setting we change the retraining procedure in a way that we will keep the size of the training set steadily at 20. Whenever we add an identical pattern to increase the multiplicity of a strong attractor we delete randomly an existing simple pattern. This learning scenario is different from the previous one and brings in the feature of forgetting simple patterns. That is an individual completely forgets a pattern if it's degree was only one, which generally means that patterns which are only less frequently exposed to the individual tend to be forgotten completely.
Test setting:

- Network: RBM with 6 visible units, 6 hidden unitis
- Initial training set: 20 simple patterns (every pattern has multiplicity d = 1)
- Training mode of Strong attractors: **Learning and forgetting**
  - : Generate 20 samples to calculate average recall probabilities, every sample will be generated as following:
    - Select randomly two different patterns out of the 20, calculate their Recall probabilities
    - Add multiplicity of first strong attractor up to d=5 by adding 4 more identical examples of the strong attractor, after every adding calculate Recall probability. With every adding a random simple pattern has to be removed
    - Now Add the multiplicity of second strong attractor up to d=15. For every increase a simple pattern has to be removed to keep the training set size at 20.

Figure 5.10 shows the recall probabilities in *Learning and forgetting* training mode. Just like in the previous mode we see that the second Strong Attractor rules out the first one but we see different details in the the dynamics of rising and falling probabilities. Compared to the previous scenario with *incremental learning* the second Strong Attractor reaches already the peaking probability of around 45 % very early at a multiplicity of 10. Also the overall steepness of recall probabilities versus multiplicity is higher than in the previous scenario, while decreasing steepness of the first Strong Attractor is very low once the second one is involved. The dynamics is easy to understand since with every adding of the second Strong Pattern another simple pattern is deleted, leading the relative multiplicity [42] of the second one to grow more quickly than in the previous scenario. Since we do not delete any of the first Strong Attractor the relative multiplicity of the first Strong Attractor remains the same [43] which explains why the recall probability of the first Strong attractor remains almost the same once the Second Strong attractor is involved. It is remarkable that the recall probability decreases at all though the relative multiplicity stays the same. This behavior can be explained by imagining that the whole energy surface of the RBM is attracted more and more to the second Strong Attractor while it's relative multiplicity increases which also effects, although slowly, the energy surfaces of attractors whose relative multiplicity stay the same.

[42] again: the relative multiplicity is the current multiplicity of a pattern divided by the number of patterns in the training set, keep in mind that the training set can contain duplicates
[43] That is $5/20 = 0.25$

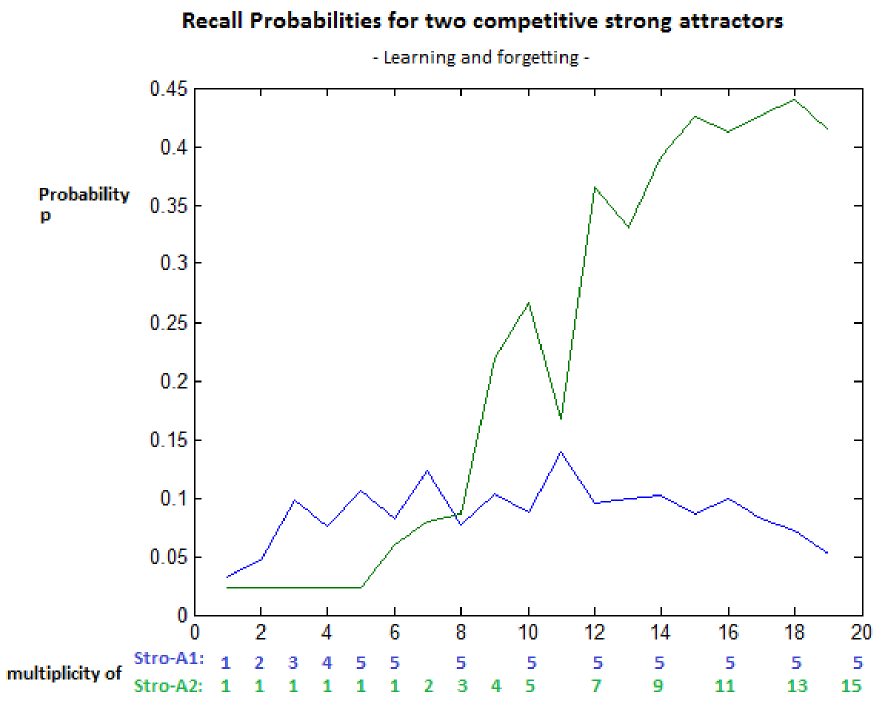


Figure 5.10: Recall probabilities in a competitive setting of two strong attractors, training mode: Learning and forgetting, see the text for details

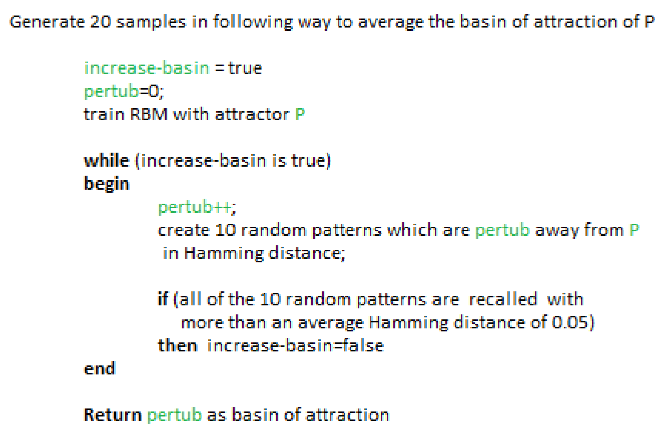
For the next experiments we want to stick to the Learning model of Incremental Learning, since it provides continuous learning since we have not a fixed example set size and the neurodynamics of ruling out the first strong attractor as it was shown in Figure 5.8 seems more reasonable because with higher multiplicities of the second Strong Attractor the first Strong Attractor's recall probability was more quickly reduced. To put it in a more neurobiological and cognitive way: If a pattern P is recently learned multiple times more then old patterns, then P should dominate the recall probabilities of all learned patterns, that is, it increases its own recall probability while reducing recall probabilities of other patterns. This kind of reducing is an implicit representation of forgetting learned patterns which are not learned frequently enough, so that our incremental learning model incorporates both, the dominance of Strong Attractors and the weakening of non-dominant patterns which can be seen as "forgetting" them.

5.2.3 Basin of attraction

For the analysis of the basin of attraction we can use a larger amount of visible units than in the previous section, since we do not need to consider all possible unit state as in equations 2.21 to 2.24. We will choose 100 visible units and will determine the number of hidden units with a large basin of attraction [44] to make the network resistant to a certain degree of noise.

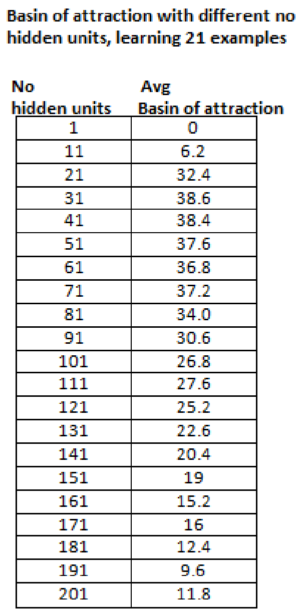
Since we use patterns with 100 bits the generation of possible similar patterns in terms of Hamming distance can lead to infeasible calculation of the basin of attraction. Therefore we have worked out an algorithm that generates randomly different patterns of a certain Hamming distance to the original attractor pattern, we sample this process many times to calculate the average basin. Following listing-figure 5.11 shows the pseudo code that we have worked out and applied to calculate the average basin of an attractor pattern P:

[44] Description of basin of attraction is listed in 2, the basin of attraction is measured in Hamming-distance. That is, if we have a basin of attraction of 5 for one attractor, we can flip in average 5 bits of the attractor-pattern and the network will successfully recall the original attractor pattern.



```
Generate 20 samples in following way to average the basin of attraction of P

        increase-basin = true
        pertub=0;
        train RBM with attractor P

        while (increase-basin is true)
        begin
                pertub++;
                create 10 random patterns which are pertub away from P
                 in Hamming distance;

                if (all of the 10 random patterns are  recalled  with
                    more than an average Hamming distance of 0.05)
                then  increase-basin=false
        end

        Return pertub as basin of attraction
```

Figure 5.11: calculating the average basin of attraction



Basin of attraction with different no hidden units, learning 21 examples

No hidden units	Avg Basin of attraction
1	0
11	6.2
21	32.4
31	38.6
41	38.4
51	37.6
61	36.8
71	37.2
81	34.0
91	30.6
101	26.8
111	27.6
121	25.2
131	22.6
141	20.4
151	19
161	15.2
171	16
181	12.4
191	9.6
201	11.8

Figure 5.12: Basins of attraction of a strong attractor with d=5, with a training set size of 21 examples

The following Experiment results show the basins of attractions if we vary for our 100-visible-units RBM the number of hidden units in a range of 21 and 201 examples.[45]

The figure 5.12 shows the average basins of attraction of a Strong Attractor with a multiplicity of **d=5**[46] for **21-201** hidden units with a training set of **21** examples. There, the ideal number of hidden units leading to a maximum basin is **31**.

[45] A wider range was simulated but the relevant results are distributed in the range of 21-201 we are focusing on

[46] we repeated our simulations also for different degrees d=10,15,20 with the same results which are presented for d=5 concerning the ideal number of hidden units

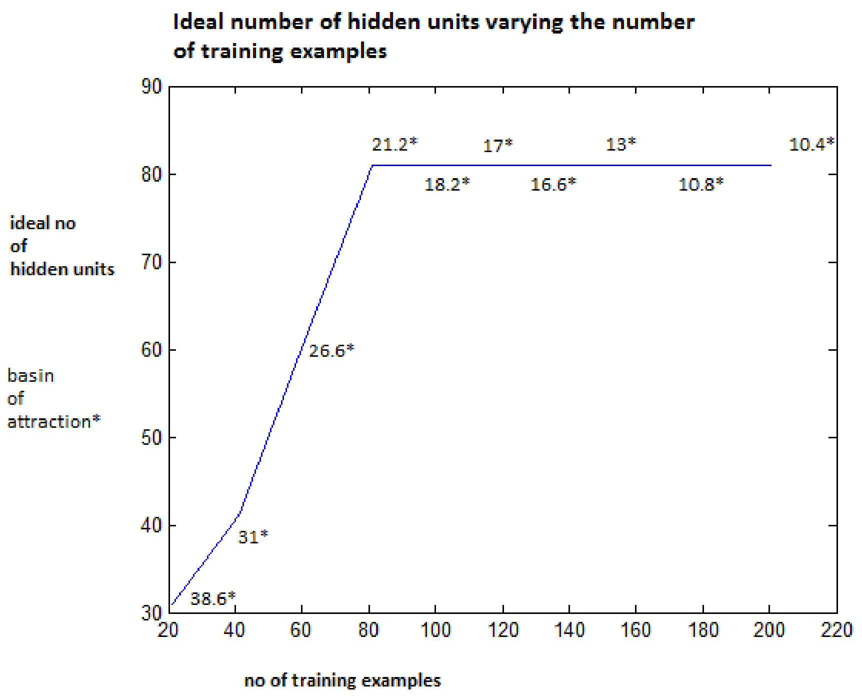


Figure 5.13: Ideal number of hidden units vs number of training examples, ideal means a number of hidden units leading to the largest basin of attraction

Since the basin of attraction strongly depends on the size of the training set, we repeated our simulations over different number of examples. Therefore, figure 5.13 shows the ideal number of hidden units (for the largest basin) varying the number of examples in the range of 21 - 201 [47].

[47]Again, a wider range was simulated but the relevant results are distributed in the specified range

Referring to figure 5.13 we can see that for 21-81 examples the ideal[48] number of hidden units is around the number of training examples, but when we add more then 81 examples the ideal number of hidden units remains at 81. Also, the actual basin of attraction (values annotated with a '*' in figure 5.13) decreases the more examples we add. As an overall result we notice that the ideal number of hidden units is always smaller then the number of visible units. According to 5.1 the ideal number of hidden units referring to Learning time performance would be around 460. But according to our simulation results we have stated that using more than 81 hidden units leads to a lower basin of attraction, that is a lower resistance to noise. This reveals a classic and common phenomenon in artificial neural networks, where the network tends to **overfit** the training examples after a certain number of hidden units and/or hidden layers is added. As we can see in table in figure 5.12 we see how the basin of attraction reduces when we add more then 31 hidden units.

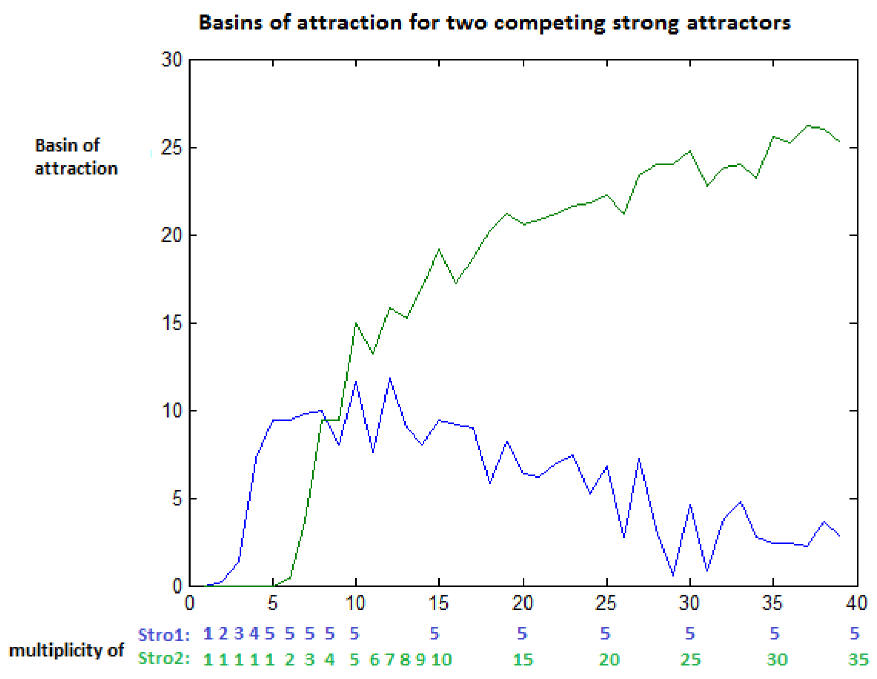


Figure 5.14: basin of attraction of two competing Strong Attractors, see Test setting for details

However, using 100 visible units, a sufficient number of hidden units , at least 101, is needed to make the RBM flexible enough to be capable of learning any arbitrary probability distribution according to theorem 1. Using more than 101 hidden units leads the network to recently observed overfitting behavior, with decreasing basins of attraction. So using 100 visible, 101 hidden units and training set size of 100 patterns, we still get an average basin of attraction of 16.2 for a strong attractor with multiplicity d=5, which is still large enough. We will stick to this network topology (100 visible, 101 hidden units) in the following simulations to observe the basins of attraction in a competitive setting of two Strong Attractors.
Test setting:

- Network: RBM with 100 visible units, 101 hidden units
- Initial training set: 20 simple patterns (every pattern has multiplicity d = 1)
- Training mode of Strong Attractors: **Increasing multiplicity by simple adding into training set**
  - : Generate 20 samples to calculate average recall probabilities, every sample will be generated as follows:
    - Select randomly two different patterns out of the 20 which are meant to be the Strong Attractors, calculate their **basins of attraction**
    - Add multiplicity of first Strong Attractor up to d=5 by adding 4 more identical examples of the Strong Attractor, after every adding recalculate its basin of attraction
    - Add the multiplicity of second strong attractor up to d=35 and recalculate the basins of attraction of both Strong Attractors in every step where multiplicity is added

Figure 5.14 shows the evolving basins of the two Strong Attractors during the simulation of the previously described Test setting.

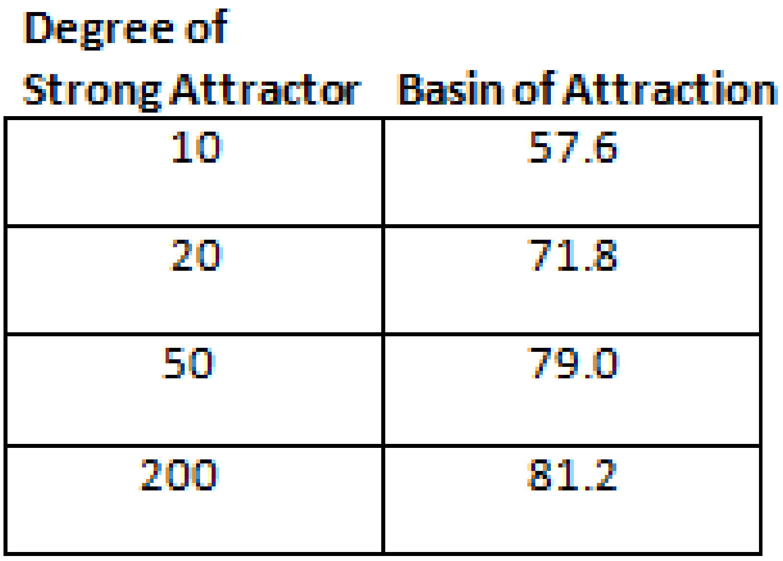
[48]remember, ideal refers to the number of hidden units leading to the largest basin of attraction, see figure 5.12 how we obtained the larges basin

As in the experiments on the recall probabilities we see in figure 5.14 a similar behavior where the second Strong Attractor overrules the first one once second one gets a stronger multiplicity. An interesting phenomenon in this simulation is that the increase and decrease of the basin of attraction of the two competing Strong Attractors is this time nearly symmetric in the range of the multiplicities we have tested.

Maximum Basin of attraction

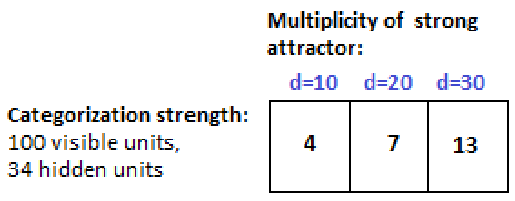
In Hopfield Networks with Hebbian Learning we have **50%** in Hamming distance as the maximum basin size of a Strong Attractor if we train this Strong Attractor with a degree or multiplicity high enough. This is due to the fact, that the Hopfield Network with Hebbian Learning suffers from the so called spurious inverse states that we have mentioned in the introduction. Hence, any higher pertubation than 50% would let the network converge towards the inverse pattern. Since our RBM with contrastive divergence does not suffer from inverse states we would like to know whether it is basically and practically possible in our simulations to get a larger basin than 50%. We take our RBM with 100 visible units with 101 hidden units which provided in our simulations a good resistance to noise, and as an extreme case to enforce a large basin, we will take an initial training set with just two examples, one of them will be considered as the Strong Attractor whose multiplicity we will largely increase and basin of attraction we will measure.

Furthermore we will increase the k-value to k=10 which was defined in 2.8 as the number of reconstructions the layers are updated by forwarding and backwarding the input through the layers, which makes recalling of the RBM with gradient descent more accurately. Apart from encountering the maximum basin of attraction and calculating the categorization strength in the next section, we use k=1 which is still mentioned as a good approximation of gradient descent according to [Hinton, 2002], higher k-values make calculation less feasible. But just for the purpose of trying to calculate the maximum basin of attraction of one Strong Attractor and for a few different degrees, we could increase the k-value up to 10 experiencing reasonable simulations times.
Figure 5.15 shows the Basins of attractions for strong attractors for 4 different degrees, we could observe basins larger than 50 beginning with degrees of 10. It has to be noted, that we can not simply increase the degree of the strong attractor endlessly since the practical training set capacity is limited regarding figure 5.1 which shows the learning times for different hidden units and training set sizes.



Degree of Strong Attractor	Basin of Attraction
10	57.6
20	71.8
50	79.0
200	81.2

Figure 5.15: Basin of one strong attractors with different degrees, and k=10 reconstructions of the layers.



	Multiplicity of strong attractor: d=10	d=20	d=30
Categorization strength: 100 visible units, 34 hidden units	4	7	13

Figure 5.16: Categorization strength for an RBM with 100 visible units, 34 hidden units scoring the strongest categorization strength within a training set of 20 examples

5.2.4 Strength of categorization

As mentioned in 3 the categorization strength of pattern P that has been assigned to category Cat is the number of patterns of the training set that fall into that category Cat.

For using the RBM as a tool which assigns every input to a category, which is a learned representation given by the hidden units, we again need to find the ideal number of hidden units, which provides a good degree of categorization. That is we would like the RBM to be capable of learning few categories for a larger set of training examples, and the categorization strength for a strong pattern should be obviously higher than that of a simple pattern.

Knowing, that fewer hidden units than visible units are needed for categorization we vary the number of hidden units in a range of 1-120[49] to obtain an ideal number of hidden units that provides the strongest categorization strength for a given strong pattern, d=5.

Test setting:

- Network: RBM with 100 visible units, number hidden units are varied in range of 1-120
- Initial training set: 20 simple patterns (every pattern has multiplicity d = 1)
- Training mode of one Strong Attractor with multiplicities 10,20,40: **Incremental Learning**
- Show the categorization strength for different numbers of hidden units in the above mentioned range.

Since we do not want to overload this report with to many tables of values [50] we want to present the results briefly with the table in figure 5.16.

[49]though we noted, that we need fewer hidden units than visible units, we would like to confirm this in our simulations, that's why we take a bigger range up to more than 100 hidden units

[50]Categorization strengths are given in the file "Categories-multi-100-20.mat" under the variable "resultsdiffhu " using 100 visible units and 20 training patterns

The highest categorization strength was achieved by using around **34 hidden units**, using more or less hidden units strives the RBM to lower categorization strengths. Hence, we take an RBM with 100 visible units and 34 hidden units as our learning model for the analysis of of competing categorization strengths of two Strong Attractors.

Test setting:

- Network: RBM with 100 visible units, 34 hidden units
- Initial training set: 20 simple patterns (every pattern has multiplicity d = 1)
- Training mode of one Strong Attractor with multiplicities 10,20,40: **Incremental Learning**
  - : Generate 20 samples to calculate average categorization strengths of the Strong Attractors, every sample will be generated as following:
    - Select randomly two different patterns which are meant to be the strong patterns
    - Add the multiplicity of the first Strong Attractor up to d=10 by adding 9 more identical examples of the strong attractor, after every adding calculate categorization strength
    - Now Add the multiplicity of second strong attractor up to d=35. After every adding, calculate categorization strength of both two Strong Attractors.

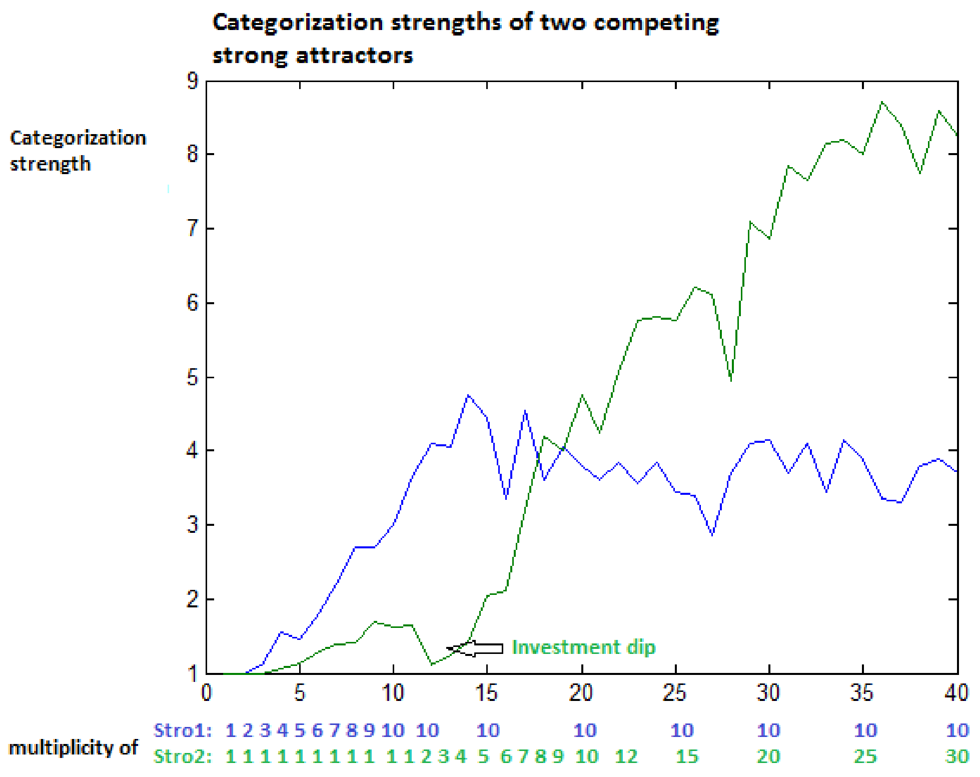


Figure 5.17: Categorization strength of two competing Strong Attractors, note the characteristic Investment Dip for the second attractor once its multiplicity is getting increased, this dip is not a random dip, it is observed in every other repeated simulation as well.

Figure 5.17 shows the results of the simulations from the previously described test setting. Two characteristics referring to the development of the categorization strength of the two Strong Attractors can be observed. Firstly, different from the increasing and decreasing process [51] of the first attractor that we have observed for the recall probability and basin of attraction, is that the categorization strength remains at the same level once the second attractor is also added. Secondly, a phenomenon that could only be clearly figured out after several repetitions of the simulations and a close look at the plotted values, is that the categorization strength of the Second Attractor reduces always once its multiplicity is increased from d=1 to d=2. After d=3,4... the categorization strength tends to increase as expected . This is not a event of randomness, since all repeated simulations showed the same result. The reason of the initial dip was figured out when we looked at the explicit categories assigned to the second attractor during its increase of multiplicity. In the state of d=1, where the second strong attractor is a simple pattern, it was in some samples assigned to a category of the first Strong Attractor which is already strong, in other words it was attracted to the category of the first one because of the strength of the First Attractor. As this happened only in some cases within the 20 samples, the average categorization strength is increased initially to a value between 1 and 2. Once we increase the multiplicity of the second Strong Attractor to d= 2, the second attractor **aims** to be assigned to **an own** category, which starts initially with a low categorization strength and then slowly increases as we see it in the simulation plot in figure 5.17. We call the dip of the categorization strength of the second Strong Attractor as an **Investment Dip**, since it has to invest the cost of loosing categorization strength first before it increases and pays off.

[51]remember, that the decrease of recall probability and basin of attraction happened to start once Second Attractor was involved

6 Evaluation

We can confirm similar effects of Strong Attractors in terms of pattern retrieval probability and the basin of attraction mentioned in the related works of [Edalat and Mancinelli, 2013] and [Tucker, 2013] in the more elaborated RBM model. That is, for higher degrees of patterns we could calculate in our simulations higher recall probabilities striving towards 1 and larger basins of attraction.

As shown in [Tucker, 2013] for stochastic Hopfield Networks, we see a similar behavior of two competing Strong Attractors in the RBM model. That is, the second Strong Attractor dominates a previously learned Strong Attractor as soon its degree exceeds the degree of the previous Strong Attractor. As a main difference we see that the basin of attraction continues to grow within a practical capacity for the training set and it can also grow larger than 50% towards 100% in terms of Hamming distance, a fact which is not possible via Hopfield Networks which suffers from misclassifying inverse states. From a neurocognitive perspective it makes sense to provide larger basins than 50%, that is to say, the more frequent an individual is exposed to the one and only input A, the more his association is attracted to A, even if we pertubate input A with more than 50% of the bits if the degree of A is high enough.

Since the the RBM uses an extra layer we could extend our understanding of an associative memory in such a way, that we see the visible layer as a part of a conscious memory that is connected to senses from the environment and a hidden layer which is not directly connected to the environment which makes it a representing candidate for the subconscious part of associative memory. A part of our memory which cannot be modeled by any kind of Hopfield Networks, since they only consist of one layer.
In a unsupervised learning setting the hidden layer is used to classify input patterns to model a subconscious understanding or an inner representation of the patterns. We introduced the term of categorization strength and could imply from our simulation again the dominance of the second Strong Attractor over the first one as soon as the degree exceeded the degree of the first Strong Attractor just as in the measurements for recall probability and basin of attraction mentioned before, but with one conspicuous difference. As soon as we increase the degree of the second Strong Attractor, an initial loss of categorization strength is inevitable before it increases and finally exceeds the categorization strength of the first Strong Attractor. This initial loss is the cost or investment for aiming a separate category or inner representation for the second Strong Attractor itself, a phenomenon that we call: **Investment Dip**.

7 Conclusion and Future Work

7.1 Implications of the Experiment Results

In the first part, we have implemented a Restricted Boltzmann machine that incorporates the efficient gradient descent learning algorithm of Hinton, pubblished in [Hinton, 2002]. Through several simulations, measuring the Learning time, we could figure out that we achieve the fastest learning RBMs as a pattern recognition tool by using many more hidden units than given visible units. For the two fastest Learning rate 0.1 and 0.05, we have a range between 100 and 1000 visible units where there is an ideal ratio of the number of hidden units over the number of visible units. This ideal ratio is around 5 and is independent from the size of the training set, as long the size is within the practical network capacity[52].

In the second and last part of the report we analyzed the strengths of two competing Strong Attractors. In terms of average recall probability, basin of attraction and categorization strength we figured out that the second Strong Attractor started to dominate the first Strong Attractor when the multiplicity of the second one went bigger than the multiplicity of the first one.

Due to the RBM's resistance of inverse states, which are misclassified in the Hopfield NMtwork, we have shown in our simulations that an RBM can be trained with a Strong Attractor that has a basin of attraction larger than 50%, what is already the maximum for Hopfield Networks with Hebbian Learning.

The generally known phenomenon of overfitting in artificial neural networks could also be shown for the RBM through the analysis of the basin of attraction of a Strong Attractor. We figured out that using almost as many hidden units as visible units results in a network providing the largest basin of attraction. This interferes with the rule that we have figured out in the first part, which stated that using 5 times more hidden units than visible units makes the network most efficient in terms of learning time. Therefore, as a rule of thumb we stated that the ideal number of hidden units is slightly more than the number of visible units which provides a good compromise of aiming an efficiently learning network on the one hand and that is still resistant to random noise on the other hand.

For the categorization strength, where we made use of reading out hidden unit values that represent the subconscious memory, we could first of all see that RBMs with Contrastive Divergence Learning can also be applied for categorization of a given input. There, we observed a so called Investment Dip for the second Strong Attractor, that is an initial loss of categorization strength which is the cost for aiming an own category. After increasing the multiplicity continuously, the categorization strength starts to increase as expected. From a neurophysiological point of view this kind of Dip can be explained by an initial confusion of an individual that tries to learn something new after it was exposed to a single dominant Strong Attractor. However, sticking to continue learning the Second Strong attractor, the individual learns to categorize the second Strong Attractor with a higher strength than the first Strong Attractor.

[52]See chapter 3 for the description of practical network capacity

Our experiments indicate the effectivity for a psychotherapeutical treatment derived from a neurocomputational basis. Thus, an individual needs to be retrained towards a secure attachment type at least with the same time or intensity as it was exposed to his environment that turned him to the insecure attachment type at the first place. Therefore, it is strongly recommended to initiate a very early psychotherapy to get rid of the insecure attachment type. It is a neurophysiological fact that in the early childhood the learning process, that is the establishment of synapses in the child's brain, is much faster. This timely related neurodynamics was not considered in our Learning model in which Learning is modeled as a constant and continious process with a constant Learning rate. This is enforces furthermore the indication of a early psychotherapeutic intervention striving for a secure attachment type.

7.2 Future Work

All our analysis of our experiments were carried over using an RBM using a fixed temperature of 1 which only provided low noise or uncertainty for the evaluation of a unit via the logistic function defined in equation 2.9. For a future analysis we are interested in the neurodynamics of our experiments with increasing temperatures. In the work of [Tucker, 2013] the effect of two competing Strong Attractors in a stochastic Network under increasing temperatues were shown, it confirms the idea of temperature as a noise factor which steadily increases uncertainty in the system the more we increase the temperature. However, the learning rule of the Hopfield Network uses a simple additive Hebbian Learning rule, while we incorparated Contrastive Divergence of [Hinton, 2002] which approximates gradient descent learning.

That is, if we increase the temperature in a Hopfield Model the evaluation of every unit gets more indeterministic, at some point it will be assigned a firing state probability of 0.5. The same will happen to the evaluation of the units in our model but it also effects our error term we have implemented in 4, leading the error to random fluctuation. This means our Learning needs more learning epochs, to let the error converge under the given limit, if it converges. But after learning is successfully terminated, an RBM ran under higher temperature meets the same error requirements as any other RBM that has finished its learning under a lower temperature, but the way how the weights are updated is different, that is an RBM trained with higher temperature might fit the training data better or not.

As another future work we are interested in, is the derivation of some mathematically proved equations concerning the degree of a Strong Attractor and its unit state probability and basin of attraction, which was worked out in [Edalat and Mancinelli, 2013] for the Hopfield Network at a fixed temperature of zero. A mathematical expression would not only confirm our experimental results it would clearly define the exact relations of a Strong Attractor's degree and its effects on other simple patterns in an RBM. However any approach into that direction will strongly depend on the learning mode of the RBM and will be complex since we have to take an additional layer into account.

Index

A

Advantages of RBMs compared to Hopfield Networks . . . 3
Attachment Types . . . 8

B

Basin of attraction . . . 16, 48
Boltzmann Machines . . . 19
Boltzmann-Gibbs Distribution . . . 19

C

Categorization in RBMs . . . 22
Categorization Strength . . . 10
Contrastive Divergence Learning . . . 21

E

Energy in Hopfield Networks . . . 15
Energy in RBMs . . . 21

G

G-Function . . . 17
Gradient Descent Learning . . . 21

H

Hardware Environment . . . 25
Hopfield Learning . . . 15
Hopfield Model . . . 14

I

Implementation of RBM with Contrastive Divergence Learning . . . 27
Investment Dip . . . 11
Ising Model . . . 18

L

Logistic Function . . . 18

M

Machines used for experiments . . . 25
Maximum Basin of Attraction in Hopfield Networks . . . 51
McCulloch-Pitts neuron . . . 14

O

Overfitting in RBMs . . . 11

P

Probability Distribution in Boltzmann Machines . . . 19

R

RBM . . . 21
Recall Probability . . . 19, 21
Recall Probability of Strong attractors 41
Restricted Boltzmann Machines . . . 21

S

Stochastic Hopfield Networks . . . 17
Strong Attractors . . . 17

T

Temperature . . . 18
Testing Environment . . . 25

References

[Amit et al., 1985a] Amit, D., H., G., and Sompolinsky, H. (1985a). Spin-glass models of neural networks. *Physical Review*, A(32):1007–1018.

[Amit et al., 1985b] Amit, D., H., G., and Sompolinsky, H. (1985b). Storing infinite numbers of patterns in a spin-glass model of neural networks. *Physical Review Letters*, Letter no(55):1530–1533.

[Bradshaw, 2011] Bradshaw, J. (2011). *The Well-being of Children in the UK 3rd ed.* The Policy Press, Bristol.

[Chen, 2011] Chen, E. (2011). Website: Introduction to restricted boltzmann machines, url: http://blog.echen.me/2011/07/18/introduction-to-restricted-boltzmann-machines/.

[Edalat and Mancinelli, 2013] Edalat, A. and Mancinelli, F. (2013). Strong attractors of hopfield neural networks to model attachment types and behavioural patterns. *Proceedings of the International Joint Conference on Neural Networks (IJCNN).*

[Flanagan, 2008] Flanagan, C. (2008). *AQA Psychology, book-pages 52-58.* Letts Educational.

[Glauber, 1963] Glauber, R. (1963). Time-dependent statistics of the ising model. *Mathematical Physics*, 4:294–307.

[Gurney, 1997] Gurney, K. (1997). *An Introduction to Neural Networks, book-page 93.* UCL Press.

[Hertz et al., 1991] Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation.* Addison-Wesley Publishing Company.

[Hinton, 2002] Hinton, R. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1711–1800.

[Mikulincer M, 2007] Mikulincer M, S. P. (2007). *Attachment in adulthood: structure, dynamics, and change.* Guilford Press.

[Roux and Bengio, 2007] Roux, N. L. and Bengio, Y. (2007). Representational power of restricted boltzmann machines and deep belief networks. Technical report, Dept. IRO, Universite de Montreal.

[Ruslan Salakhutdinov, 2007] Ruslan Salakhutdinov, Andriy Mnih, G. H. (2007). Restricted boltzmann machines for collaborative filtering. *Proceedings of the 24th International Conference on Machine Learning, Corvallis, OR, 2007.*

[Strathearn, 2011] Strathearn, L. (2011). Maternal neglect: Oxytocin, dopamine and the neurobiology of attachment. *J Neuroendocrinol.*, 23(11):1054–1065.

[Tucker, 2013] Tucker, R. (2013). *Stochastic Hopfield Networks To Model Secure And Insecure Attachment Types With Noise.* B.Eng Thesis, Imperial College London, Department of Computing.

[Wedemann et al., 2013] Wedemann, R. S., Donangelo, R., and de Carvalho, L. A. V. (2013). Strong attractors of hopfield neural networks to model attachment types and behavioural patterns. *Proceedings of the International Joint Conference on Neural Networks (IJCNN).*