Weight Discretization due to Optical Constraints and its Influence on the Generalization Abilities of a Simple Perceptron

Inaugural-Dissertation

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

der Universität Mannheim

vorgelegt von
Diplom-Physikerin Maissa Aboukassem
aus Lattakia (Syrien)

Mannheim, 2000

Dekan:	**Professor Dr. Guido Moerkotte, Universität Mannheim**
Referent:	**Professor Dr. Reinhard Männer, Universität Mannheim**
Korreferent:	**Priv.-Doz. Dr. Reimer Kühn, Universität Heidelberg**

Tag der mündlichen Prüfung: 28. Juli 2000

Contents

1 Introduction to neural networks

1.1 Abstract

Historically, the interest in neural networks has two roots, first is the desire to understand the principles on which the human brain processes information, and the second is the wish to build machines capable of performing complex tasks for which the sequentially operating, programmable computers conceived by von Neumann are not well suited. Everyday observation shows that the modest brain of lower animals can perform tasks that are far beyond the range of even the largest and fastest modern electronic computer. A human can easily recognize faces, understand speech, or navigate a room; whereas a supercomputer is not able to perform such tasks.
Because of many features they are claimed for like parallelism, robustness, and the ability to learn from examples, neural networks promise to keep the dream of intelligent brain-like machines alive, and to scape the deaden predicted by computer scientist for conventionally programmed sequential computers.
The main problem which faces implementing such highly interconnected systems is the realization of the connections between the neurons. The high $3-D$ interconnection density , and the high storage capacity of optics suggests that it may be a suitable technology for implementing very large neural networks which are impractical to simulate or implement electronically. So we focused our investigation in this thesis on studying the effects of the optical constraints on the performance of the simplest neural network (the perceptron).
In this chapter we introduce briefly the nerurobiological background of Neural Networks (NN), their abstract Model, the properties and promises, and the advantages of optics to implement them. Finally, the motivation and structure of this thesis is outlined.

1.2 Neurobiological background

The key to design of both neural networks and neurocomputers is understanding the ways in which the brain uses biological neural systems for information processing. This is far from understood, but it is useful to survey briefly the current knowledge. The

human brain is composed of 10^{11} single, interconnected cellular units , the *neurons*. The detailed investigation of the internal structure of neural cells shows that all neurons are constructed from the same basic parts, independent of their size, shape and function. The neuron consists of an input part the *dendritic arbor*, a processing part the *soma* and a signal transmitting part the *axon*. The size of a typical neuron is about $10 - 18\mu m$, while dendrites and axons have a diameter of a few μm. Such neuron is illustrated in Fig.(1.1). The *neurons* communicate via *synapses*, which are the points along the axon of the $pre - synaptic$ neuron at which it can communicate the outcome of the computation that has been performed by the *soma* to the *dendriticarbor* or even directly to the soma of the $post - synaptic$ neuron. Each neuron receives some 10^4 synaptic inputs from the axon of the other neurons. Various signals are transmitted either electrically or chemically.
In the state of inactivity the interior of the neuron is negatively charged against the surrounding, the resting potential is about $-70mV$. This resting potential is caused by a deficiency of positive ions in the protoplasm which is supported by the impermeability of the cell membrane for positive ions.
Electrical transmission is based on an electrical discharge which starts at the cell body *soma* which acts as a kind of summing device that adds the depolarizing effects of its various input signals and then sends it down the axon to the various synaptic connection. Signals arriving from the synaptic connections causes a transient weakening, or *depolarisation* of the resting potential.
The chemical transmission takes place when the spike signal arrives at the presynaptic nerve terminal, where special substances called *neurotransmitter* are liberated in tiny amounts. The neurotransmitter molecules travels across the synaptic cleft reaching the post-synaptic neuron or muscle fiber within $0.5ms$. The Post-Synaptic Potential(PSP) defuses toward the soma where all the inputs from all the pre-synaptic neurons connected to the post-synaptic one are summed. This changes the permeability of the cell membrane of the post synaptic neuron so that it becomes permeable for positive ions. If the induced polarization potential is positive the synapse is termed *excitatory*. If it is negative, the synapse is called *inhibitory* . If the total sum of the PSP's arriving within a short period surpasses a certain threshold, which is the level at which the post synaptic potential becomes unstable against depolarizing ionic current flows, the probability for the emission of a spike becomes significant. This threshold is tens of millivolts and hence a number of inputs are required in order to produce a spike. After the emission of a spike, the neurons needs time to recover. There is a period of 1-2 milliseconds in which the neuron is not able to emit another spike, no matter how large the depolarizing potential may be. This period is the *refractoryperiod* of the neuron.
It sets the maximal spike frequency at 500-1000 per second. Yet the brain is capable of solving difficult problems of vision and language in about half a second (i.e. 500 milliseconds). This is particularly surprising given that the respond time of single neurons in the brain is in millisecond range. So the incredible complexity of the human central nervous system, rests not so much in the complexity, the high speed and diversity of single nerve cells, which is quite limited, as in the vast number of its constituent units,

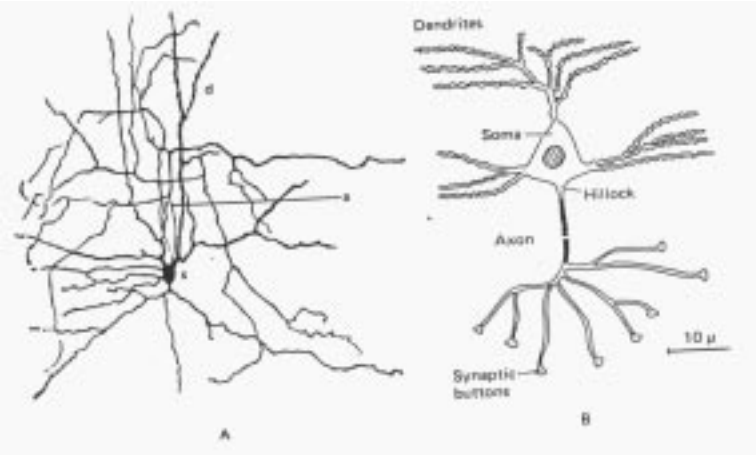

Figure 1.1: *Schematic representation of a neuron (After P. Peretto, unpublished)*

i.e. of the neurons and their mutual connections.

1.3 Artificial neural networks

Many concept and terms in the field of NN are motivated from neurobiology. However, the models considered by physicists, computer scientists, and statistical scientists are so much simpler than its biological example, that any serious comparison is misleading. Therefore they are often referred to as Artificial Neural Networks (ANN). Gross simplification are made with respect to the neurons, the number of different types of neurons, the manner the neurons are connected and the way they transmit signals, but a great deal of terminology that one encounters, comes from neurophysiology.

In mathematical terms a neural network is defined as a directed graph with the following properties.
The nodes of the graph are the processing units of the NN, the neurons. The links are represented by a number of input lines which connect the neurons to each other. In Fig.(1.2) the links are depicted with incoming arrows. Each input channel is a combination of a dendrites and a synapse. The input channels are activated by the signals they receive from other neurons. A state variable S_i is associated with each node i, which determine whether a neuron is active or not in the deterministic case, and gives the probability of its activation in the case of stochastic neurons.
A real valued weight w_{ik} is associated with each link (ik) between two nodes i and k. The numerical value of w_{ik} is the synaptic efficacy which determines the amount of post-synaptic potential that would be added to the neuron i if channel k were activated. The values of w_{ik} may be positive (excitatory) or negative (inhibitory), whereas w_{ik} equals zero, if the connecting channel between the neurons i and k is not activated. A real valued bias ϑ_i is also associated with each node i. A transfer function

$f_i(\{S_k\}, \{w_{ik}\}, \vartheta_i)$ is defined for each node i, which determines the state of the neuron as function of its bias, of the weights of its incoming links, and of the nodes connected to it by these links. The neuron operates as follows:

- At any given moment some of the logical inputs are activated.
- The soma receives an input PSP which is in the first order approximation the linear sum of the synaptic efficacies w_{ik} each multiplied by the corresponding state variable of those channels arriving at that neuron. In other words, denoting the PSP at neuron i by h_i, we can write

$$h_i = \sum_{j=1}^{N} w_{ij} S_j. \tag{1.1}$$

  where N is the number of pre-synaptic neurons.
- The PSP is compared to the threshold value of neuron i and the output channel is activated if it exceeds the threshold. Otherwise it is not.

At a given moment the actual state of the NN is made up of the states of all its neurons, and it changes either syncronically or assyncronically according to an update algorithm.
The topology (link map) of the graph, which defines which neurons are connected by a synapse, is called the *architecture* of the NN. The most common architectures are the fully-connected Hopfield model and the fully-connected Multilayer Perceptron (MLP). See Fig(1.2) for an illustration of these architectures. The Hopfield model contains cycles, whereas the MLP does not; for this reason, the former type is called *recurrent*, as opposed to *feed forward* to the latter.
We can distinguish the neurons according to their location in the graph. Some neurons are identified as *input neurons*, their states are the numerical representation of the actual input to be processed, e.g. the key pattern in the case of an associative memory. Some neurons are identified as *output neurons*, their states represent the output finally generated by the NN. Neurons that are neither output nor input neurons are called hidden neurons.
ANNs are being applied more and more for tackling numerous problems in another way than, or in combination with existing algorithms (see the literature on engineering with ANNs). The way that ANNs handle information is different from that of computers. In a von Neumann computer each byte of information is stored in one particular place an can be perfectly retrieved with the knowledge of its address. If one removes a small piece of information from the computer, all the information that this area contains is completely lost. The elementary processing time is of the order of Nano seconds, but only one operation (or few for parallel processors) can be done per time unit.
In a neural network, the information is spread over the whole network, or at least over

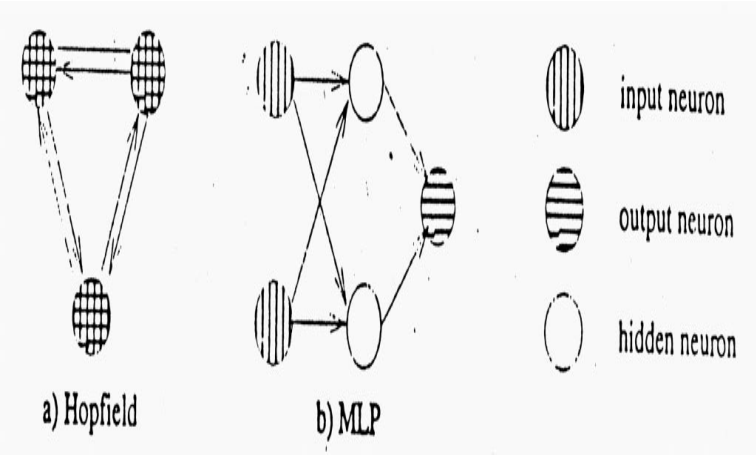

Figure 1.2: *Neural network architectures. (a) Hopfield model, (b) Multilayer perceptron (After [15])*

a large fraction of it. If one removes a small piece of information from the network (by dilution of some its weights for example) only an equally small piece of information is lost. In the real brain this happens constantly, because neurons die all the time, and nevertheless we are able to remember things. The information is retrieved (not always perfectly) by its content (this is called associative memory). This means that one has to present a part of the information to get back the whole information. The elementary processing time is of the order of milliseconds, but the processing is done massively parallel.
The associativity and the spreading of information over the neural network give it an enormous robustness against malfunctioning of some of its neurons, and the parallel processing makes it much faster than conventional computers for tasks like vision, motor control and decision on the basis of incomplete or noisy data. On the other hand, for high precision arithmetic and perfect storage and retrieval of information, the von Neumann computer is superior. So, ANNs are probably not going to replace computers, but are complementary devices that are applicable to other problems.

1.4 What do neural networks promise?

Neural networks exhibit many properties analogous to the brain: association, generalization, parallelism, learning, robustness, universality and flexibility.
Association ANNs are able to match input patterns to output patterns, as with a content-addressable memory.
Parallelism ANNs are inherently parallel, whereas most standard algorithm are not. NN fit better to existing general-purpose parallel hardware than standard algorithm do, and new special-purpose parallel hardware is easier to design for NN than for standard algorithm.

Learning Learning in ANNs occurs by adjusting the network parameters (the weights) so that the NN do well on the presented set of training patterns. Learning allows a neural network to discover patterns and their relationships, and to organize itself to perform associations. This capability has major implications: the ability to solve problems whose rules are difficult to formulate and to solve on von Neumann computer machines, and the ability to extract from large data sets with numerous variables both statistical models and knowledge-based rules. Learning in ANNs is performed by determining the network parameters (the weights) due to a learning algorithm after presenting a training set.
Universality It is proved that Neural Networks are universal in this sense: in principle for every boolean function there is neural network which emulate this function (McCulloch and Pitts 1943). ANNs are thus capable of computing any task a digital computer can, and more generally, ANNs can approximate any function arbitrarily accurate (Hecht-Nielson,1989, 1990)[47]. The latter result is based on Kolmogorov's theorem on the representation of functions of many variables by superposition of functions of one variable(Kolmogorov, 1957)[46].
Robustness Robustness in NN is a consequence of the well balanced distribution of the learnt information on all the weights of the NN, so that neither a single data item nor a single code word dominates the operation of the NN. A single error will decrease the performance only slightly. This is in sharp contrast to the operation of digital microprocessors, for which a single error (uncorrelated) bit error or a single code word will destroy the operation in most cases.
Flexibility In neural networks has a number of manifestations. Networks can adjust to changing environment through learning, this reduces the need for reprogramming.

1.4.1 The generalization ability of NN

The deep intention of training neural networks is not only to recall and store information, but also to process new inputs that were never learned. In other words we would like to make the neural network able to generalize or to extract information hidden in the training set. Experience has shown that NN, often succeed in this task, but not always, and sometime only to a certain degree. The response to the question, whether there is anything mystical about the ability of trained ANNs for generalization, is no. The generalization ability of NN is a direct consequence of the laws of statistics and probability. For a given amount of information fed into the network it will most likely choose the generalization that has the greatest probability of being realized.
Anshelevich et al. found by applying tools of information theory to investigate the behavior of NN that as a result of learning the correct response to $\mathcal{P}$ patterns, the network aquires $-\mathcal{P}N_0 \log_2 q$ bits of information, where q is the probability of correct response in any one of the N_0 output neurons.
In order to provide the network with sufficient information to learn any particular algorithm it is capable of performing, the network must be trained with at least

$$\mathcal{P}^* = \frac{1}{N_0} \log C. \tag{1.2}$$

input patterns where C the *capacity* of a neural network is the total number of different input-output mapping it can represent by appropriate selection of its weights.
The statistical approach to learning and generalization in NN has also been studied in detail by S. Solla and collaborators, Seung Sompolinsky [10], who have developed an explanation of the rather strange behavior of learning curves (the generalization error often drops suddenly to zero after a long period of slow progress) on the basis of thermodynamical concepts. There are many circumstances why ANNs may fail to generalize. If the new and previously unknown input deviate too much from the examples used in the learning phase the ANNs may not be able to generalize to the new input. The number and manner of presentation of the training set for a given task plays also an essential role on how fast the task can be learned. Generally the information needed to reach good generalization must not lie hidden to deep in the form of the learning example.
We must also refer to the fact that generally valid statements about the ability of neural networks to generalize are difficult because the generalization of a given set of examples is never unique. We should note, to understand generalization in the sense of the human ability to generalize is misleading.

1.5 Why optical neural networks?

The biggest problem which faces all implementation of neural networks and all other kinds of massively parallel systems is the realization of the connection channels. A Hopfield network with 1000 neurons needs a $5 \cdot 10^5$ connection channels. Simulation in software using sequential or parallel computation or dedicated special-purpose multiprocessor systems is the only way to implement such highly connected systems on electronic systems.
But there are a lot of limitations set by physics for electronic implementations of neural networks. These stem from the fact that electrons are the information carriers in such systems. Electrons are charged particles with a mass, thus they repel one another, and we need energy to accelerate them because of their mass. Due to the fact that they are *fermions*, they also can not be simultaneously in the same quantum state (for ex.they can not take simultaneously the same space). The effects of these properties on the highly integrated electronics is, electrons must be confined to wires to get them from one place to another reliably. They have lower speed compared to the speed of light, and its transport requires high energy consumption. Very high speed electronic machines use additional power to provide speed and have elements located very close to one another to limit the transmission time. In this case the technology for transferring heat out of the system is also a limiting factor.
These limits are not relevant for Photons as information carriers. Photons belong to a

class of particles called *bosons*. They are uncharged, massless particles. Consequently, light beams may pass through one another without distorting the information carried. Essentially all the strength and weakness of optics comes from this observation. Due to this facts it is possible to realize free space optical connections which allow to exhaust all the advantages of optics. Moreover free space optical connections are inherent parallel. Another advantage of optics over electronics is that optics can operate at an average energy per calculation that is less than the theoretical minimum for the most efficient electronic logic gate. Due to these properties of light, optical computers can feature massively parallel interconnections that are impossible physically and energetically with electronics. Optical approaches also show great promise for satisfying the performance requirement of large neural network models. The parallelism, the high $3-D$ interconnection density , and the high storage capacity of optics suggests that it may be a suitable technology for implementing very large neural networks which are impractical to simulate or implement electronically. On the other hand neural networks are the best example of all optical computing paradigm which uses the advantages of optics.

There are four main arithmetic operations in conventional neural networks: multiplication, addition, subtraction and nonlinear thresholding. The optics can provide analog multiplication and addition in real time, while nonlinear thresholding can be performed by an optical modulator. Implementation of subtraction in an optical neural network is a key issue. Coherent and incoherent techniques differ markedly.
In the recent years there has been considerable interest in the optics community in the optical implementation of the vector-matrix multiplication which is the most computing intensive operation in Neural networks. Incoherent optoelectronic implementations of matrix vector multipliers with nonlinear electrical feedback were used to demonstrate that imperfect analog hardware worked surprisingly well in the robust environment of a neural network. Goodman studied a first implementation of a vector-matrix multiplier, which was based on classical refractive optics [26]. Psalties and Farhat utilized the architecture of Goodman to implement an optical Hopfield network [25].
Holographic association with coherent light can be combined with optical nonlinearities with the nonlinear thresholding capabilities of an optical spatial light modulator to implement image association. Volume holograms can be repetively exposed to a number of Bragg angel multiplexed connectivity patterns to produce a holographic interconnection matrix.

1.6 Motivation and structure of the thesis

The study of neural networks with local constraints on the coupling strengths is well motivated from both biological and applications point of view. It is very implausible to assume a biological mechanism which preserves infinite precision of truly continuous

w_{ij}, and it is therefore interesting to study the effect of some coarse graining of synaptic efficacies, for example by encoding the information using a finite number of discrete values. Likewise, in hardware implementations it is simpler to realize networks wherein the couplings are restricted to discrete values such as $w_{ij} = 0{,}1$ (connected or not),$w_{ij} = \pm 1$, or more generally w_{ij} restricted to some set of discrete values. But above all this thesis is motivated by the attempts of the optic group at the department for computer science V, University of Mannheim, to implement a hybrid opto-electronical neural network. Within this project a holographic vector-matrix-multiplier (VMM) has been studied by (Dietrich, 1995)[3].
A vector-matrix-multiplication can be carried out completely in parallel by storing the matrix elements (weights of the NN) in holograms with a limited number of grey values (Goodman et al.,1978; Shamir et al. 1989) [50]. Due to the robustness of Neural networks discussed already, we expect this limitation to be not that severe. Robustness together with parallelism makes it possible for Neural networks to exploit the huge computing power of optical VMM. Despite general prospects being good, many questions must be investigated to design a hybrid optoelectronic neural network which successfully embeds the optical VMM: which NN architecture fits best to the optical hardware? Which training algorithm can be used, if we want to implement the training process optically? Which holographic material fit best to solve the problem of storing the weights in the case of off-line trained NN? Which (non-neural) pre- or post processing supplements the NN best? Which processing steps must be realized in hardware (optical or electronical) and which steps can be realized in software (on a host computer) in order to optimize the whole system? In this thesis we focus our investigation on one of these questions, namely on studying the dependence of the performance of a simple perceptron, measured in terms of generalization and training errors, on the number of allowed discrete values for the synaptic weights (there are 2^p allowed values for a bit precision of p), and on the training set size. The network we chose to look at is the simple perceptron, because it is the simplest architecture of neural networks to be implemented optically, and the simplest for the theoretical investigation by means of statistical mechanical tools. The perceptron is the simplest feed-forward network. This network consists of one input layer of N units that may take real or boolean values and a single boolean output. Since the simple perceptron embodies most of the general principles involved in this class of parallel architecture, results obtained by investigating this network should be generally applicable to other more complex networks. Our starting point is the teacher pupil paradigm and methods developed by using statistical mechanical tools to study complex spin systems.
Chapter 2 deals with the optical implementation of the simple arithmetic operations, and the vector matrix multiplier.
In **Chapter 3** we introduce the general statistical mechanical framework for systems with adjustable parameters exhibiting the ability of learning a rule from examples, the concept of defining a rule, and the formalism of learning a rule.
Some thermodynamical variables are introduced such as the free energy which serves as the generating function for various quantities characterizing training. In particular,

thermodynamic relations which link the training error, the volume in parameter space associated with a given error, and the generalization error are explained. The replica method which is a commonly used technique to perform the average over examples is briefly outlined.

Chapter 3 shows that statistical mechanics is a useful tool to analyze learning for different reasons. One is that many algorithms are stochastic and correspond exactly to Langevin or Glauber dynamics on a noisy energy landscape. Another reason for using statistical mechanics is that learning is essentially a problem of statistical inference and fits naturally into the same mathematical framework. A paradigm normally used for studying the generalization ability, the teacher pupil paradigm, has been introduced. The replica method, which is a technique commonly used to perform the average over the random examples is also introduced.
In **Chapter 4** we define the simplest neural network, the simple perceptron, which is the subject of our investigation. We summerize a number of sophisticated learning algorithms for the perceptron and explain some of the exactly solved models.
In **Chapter 5** the general framework of statistical mechanics is applied to the case of learning a rule at finite temperature. In our specific model a pupil perceptron, whose weights are restricted to discrete values, is trained to infer a rule presented by a teacher perceptron with continuous weights. The pupil is trained on $\mathcal{P} = \alpha N$ random input examples whose components take the values ± 1 with equal probability, and the generalization error is measured by the average disagreement between the respond of the pupil and that of the teacher on examples drawn from the same distribution as the training examples.
The replica trick and the saddle point method are used for evaluating the free energy. We find that the order parameters are the so called Edwards-Anderson parameter (this will be explained in Chapter 5), and a new quantity R the average overlap of the the ensemble of trained pupil perceptrons with the teacher perceptron. The replica symmetric ansatz is used and its stability is determined.
In the space of the relative number of examples, and the temperature the three important lines, the De Almeida-Thouless, the Gardner-Derrida, and the critical lines, have been calculated for the case of 1 bit and 5 bit synaptic depthes. The De Almeida-Thouless set the boundary for the area in the phases pace with stable solutions with respect to replica symmetry breaking. The Gardner-Derrida sets the boundary in the phase space where the saddle point equations become singular. The critical line sets the boundary for the existence of a numerical solution for the saddle point equations. The generalization error is also calculated, in the replica symmetric case, as a function of the relative number of examples α for different bit precisions and with a constant temperature which is a measure for the intensity of noise in the learning process. We studied also the evolution of the generalization curve as a function of the training temperature for a constant α. The dependence of the training error on the stability constant is investigated for the two extreme cases of 1 bit and 5bit.
In **Chapter 6** we are going to describe the simulated annealing algorithm and its implementation in the case of learning binary random input output patterns produced

by a teacher perceptron with continuous weights which has to be learnt by a pupil perceptron with discrete weights.
In **Chapter 7** The generalization as well as the training curves, obtained by the simulations, will be studied as a function of the training set size with a given allowed synaptic depth and compared to the results produced with tools of statistical mechanics.

2 Optical neural networks

2.1 Introduction

The potential advantages of using optics in the implementation of neural networks are well known and stem, as we have mentioned in the last chapter, from the capability of optics for 3-D, high density interconnections and analog data storage, as well as rapid multiplication and addition of analog signals. These are, in addition to the nonlinear thresholding, the main arithmetic operations in a conventional neural network. The optics can provide analog multiplication and addition with a very high speed (real time), while nonlinear thresholding as well as the transfer function can be implemented by utilizing a nonlinear optical effect, or can be realized electronically. Such implementation which utilizes optical and electronical elements are referred to as hybrid NN. In principle, we can distinguish between two kinds of optical Neural networks implementations. Some optical neural networks combine the learning and recall (data processing) in the same hardware. These can be very powerful, but they are also very difficult to realize. In [27] Psaltis and Wagner presented a new approach to learning in multilayer optical neural network based on holographically interconnected nonlinear devices. The proposed network can learn the interconnection that form a distributed representation of a desired pattern transformation operation.
Other optical neural networks do the learning off-line in a digital computer and embody the learned weights in a hologram. These are simpler and therefore likely to be used sooner. Figure (2.1) shows schematically how such an implementation works. In the recall phase (in the case of a simple perceptron), the hologram will be illuminated by a light beam which is modulated corresponding to the activity of the input patterns. The input pattern is then multiplied by the appropriate values of weight matrix represented by the hologram. An anisotropic lens system at the output will then form the output vector representing the corresponding output, which is detected by a variable time CCD camera. The interaction between the hologram and the illuminating light beam represents the information processing of the NN. In the electronical implementation the output of the output neurons is calculated sequentially, whereas in the optical implementation the output is performed parallel because the light beams may pass through one another without distorting the information carried. For high-order neural networks, fully parallel implementations have been reported that use holographic lenslet arrays and spatial light modulators . Nevertheless, the limited diffraction ef-

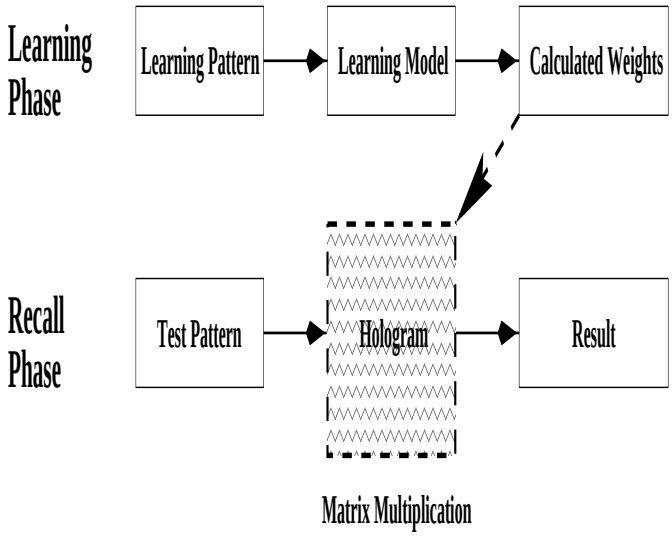

Figure 2.1: *Schemata of a hybrid neural network*

ficiency of the replicating optics and the lack of contrast and controllability of the spatial light modulators used seemed to prohibit the implementation of large scale optical connections by this method.

In this chapter we introduce, how can we implement the important arithmetical operations optically and we will review some of the methods to implement the vector matrix multiplication optically.

2.2 Optical numerical processing

An electromagnetic wave can be determined by three parameters its *amplitude*, its *phase*, and its 3-D *wavenumber*

$$\vec{A}e^{i(\vec{k}\vec{r}-\phi)}. \tag{2.1}$$

When a wave passes though a medium, all these parameter can be influenced. When light passes through a transparency, its *amplitude* is multiplied at each point by the transmittance of the transparency which is normally a real and positive number ≤ 1. The transmittance might be complex, i.e., it might also produce a phase change. Now it is clear that we are able to implement the **multiplication** by a number smaller than 1 optically.
It is also possible to realize the multiplication by numbers ≥ 1, but this needs a very sophisticated optics and an additional light source (because of the energy conservation law). But this is not necessary for neural networks, because common scale can be absorbed in scale of threshold, we can map all the weights on the interval [0,1]. **Addition** is accomplished by modulating two separate light beams, and then combining them with a beam splitter. A beam splitter is actually just a semi transparent mirror.

In this case what it does is to let one beam (modulated by f1) through, and break the other (f2). Thus the output contains both. If the light is coherent, the complex amplitudes are added up. This is another way to say the beams interfere. In the case of incoherent light, the intensities of the beams are summed.
Implementation of subtraction in optical neural networks is a key issue. A fully coherent optical system can **subtract** signals directly, using differences in the phase or path length, of the optical beams. The trade-off of this method is that the system must keep relative path lengths stable to less than a wavelength. In addition the phases of components in the system must be accurately controlled. An incoherent system is more robust in terms of stability, position accuracy requirements and noise immunity. Signals are typically encoded in light intensities. This provides only real, non-negative quantities. But it is also very important to be able to realize negative values of the weight matrix optically to perform a general vector- matrix multiplication optically. Two ways have been suggested to solve this problem. The first method was introduced by White and Gaulfield [53]. The weight matrix $\mathbf{w}$ can be written as a difference of two positive valued Matrix $\mathbf{w}^+$ and $\mathbf{w}^-$,

$$\mathbf{w} = \mathbf{w}^+ - \mathbf{w}^+. \tag{2.2}$$

In this case the vector matrix multiplication can be accomplished by performing two vector matrix multiplications with positive weights and then perform the difference electronically or optically.

$$\mathbf{w}y = \mathbf{w}^+y - \mathbf{w}^+y. \tag{2.3}$$

The second method is based on performing the sum $\mathbf{w}^*$ of the weight matrix $\mathbf{w}$ and a matrix $\mathbf{M}$, whose coefficient are built of the largest value of the weight matrix $m_{ij} = |w_{ij}|_{max}$. Then the vector matrix product $\mathbf{w}y$ can be calculated as a subtraction of a positive vector matrix product $\mathbf{w}^*y$ and the weighted sum $|w_{ij}|_{max} \sum y_i$ of the vector components.

$$\mathbf{w}y = \mathbf{w}^*y - |w_{ij}|_{max} \sum_i y_i. \tag{2.4}$$

2.3 Implementation of the vector-matrix multiplier

Matrix operations are the computationally most intensive and the simplest operation needed in a neural network to compute the weighted sum of the input vector for further processing. A lot of research has been done to develop special purpose architecture that will compute these operations more efficient than serial digital electronic computers. Special purpose optical systems are also being considered. The optical implementation of a parallel, real time vector matrix multiplier is straightforward , as shown in Fig(2.2) . The information content of an input vector is represented by a phase or amplitude modulated light signal obtained from an array of LEDs: so that the light intensity issuing from LED number i is proportional to the ith component of the input vector. Using the light intensity to represent data have the drawback to be limited

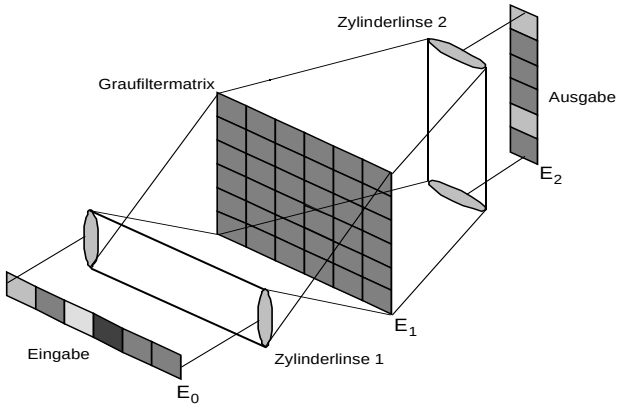

Figure 2.2: *Vektor-Matrix- Multiplier with Gray value matrix*

to data with positive real values. The matrix could be represented by a spatial light modulator (SLM) a hologram or a spatial light rebroadcaster (SLR). For the SLM implementation, the SLM is divided into N^2 small squares. The transmittance of square i, j is proportional to the matrix element a_{ij}. An astigmatic lens spreads the light from each LED onto the corresponding column of the gray values matrix represented by the transparency. Thus the intensity of the light passing through the square i, j is proportional to $a_{ij}x_j$. Another astigmatic lens collects the light from whole row, and focuses it onto the corresponding detector which could be a photodiode or a CCD-element. Here the conversion into an electronic signal could be carried out. The intensity of the light falling on the detector number i is then proportional to $\sum_j^N a_{ij}x_j$. Therefore the output of the detector array represents the product vector $(\mathbf{a} \cdot \mathbf{x})$. A striking feature of the above configuration is that all the N^2 scalar multiplications are done in parallel. This results in very high speed. It has been demonstrated that the multiplication of a 100 element vector by a 100×100 matrix may be done in $20ns$ [55]. The throughput, however may be limited if the matrix is to be changed.

By using holograms to store the matrix, on illumination of the hologram with a collimated beam, the light beam will be, dependently to the type of the first order diffraction, made to represent the matrix pattern. Later investigation have shown that holographic mapping elements are the most promising candidate for implementation of an interconnection matrix because of the large storage capacity possible in the volume of a crystal and the dynamic response possible with a photorefractive crystal. One of the most important impediments to holographic neural networks is holographic cross talk which arises from an effect known as *Bragg degeneracy*. Left uncompensated, this crosstalk results in large distortion in the effective interconnection weights.
The simple perceptron is the simplest neural network which can be implemented optically, here we need to record only one hologram to represent the weight matrix. Also

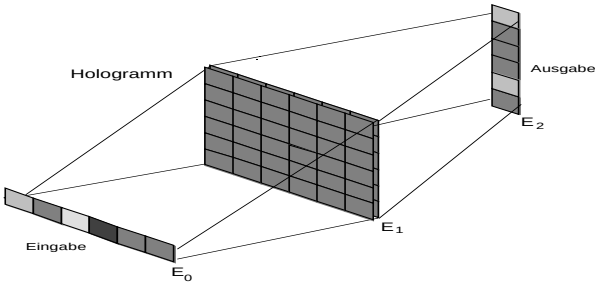

Figure 2.3: *Vektor - Matrix - Multiplier with Hologram as a storage for the weight vector*

multilayer perceptron could be realized with utilizing volume holograms or cascaded holograms.

3 Introduction to statistical mechanics

Neural networks have achieved many successes, but their underlying theory remains difficult to formalize. The theory of learning in neural networks has benefitted from an interplay of ideas that come from various scientific fields. This includes *Computer Science, Mathematical Statistics, Information Theory and Statistical Physics*. But even the most advanced statistical theory which has been applied, Vapnik-Chervonenkis theory, also known as VC theory, has been unable to do much more than place rigorous bounds on their success, and then only for the simplest networks.
Statistical mechanics has been found to be a good tool for studying neural networks. This stems from the analogy between a neural network and a system of atomic magnetic dipoles or *spins*, which has been pointed out by William Little. By denoting the two possible states of a neuron by the variables $s_i = +1$ (active neuron) and $s_i = -1$ (resting neuron) the analogy between spin systems and NN becomes obvious. This analogy became especially fruitful because of the advances achieved over the last decade in understanding the thermodynamic properties of disordered systems of spins, the so called *spin glasses*.
Hopfield (1982)[45] realized that the dynamic of the whole neural network can be investigated using statistical mechanic, if the states of neurons can take just two values, the weights are symmetrical ($w_{ij} = w_{ji}$), and the states of the neurons change asyncronically. Under these circumstances it is possible to define an energy for the network. The weights of the network can then be adjusted so that pre-chosen configurations of the network are fixed points of the networks dynamics. These fixed points would be related to memories in real, biological networks. A basic characteristics of memory networks is the limit of capacity, that is the maximal number of patterns they can store. It has been shown by using tools of statistical mechanics that, if the original Hebbian rule is applied to a system of N neurons, the capacity is asymptotically proportional to $N\log N$, or,with an error tolerance smaller than 1%, it was shown by Kühn *et.al.*[49] to be $\approx 0.138N$. On the other hand the optimal capacity has recently been found to be $2N$ for the case of directionally independent synapses, allowing asymmetric coupling parameters.
Learning and generalization has also been a subject of intensive interest. A statistical mechanics approach to learning from examples was first proposed by Carnevali and Patarnello [30], and further elaborated by Kinzel and Opper [32] followed by Tishby [10], Solla and Levin [33] Del Giudice, Franz,Virasoro, and Hansel and Sompolinsky

who applied spin glass theory to study perceptron learning of a classification task [34]. Instead of rigorous bounds, the new theory is able to make *precise* quantitative predictions for the typical behavior of specific learning models, and has solved problems which have been outstanding in the machine learning community for many years. More importantly, statistical mechanics gives insights into practical applications. New techniques for learning may be motivated by exploiting advanced statistical physics, particularly that of spin glasses.

3.1 The formalism of learning

The natural method of programming a conventional sequential computer is to write an algorithm, which consists of a list of instructions that have to be performed by the computer. This method implies severe limitations in the case of problems for which there is no exactly defined algorithm. However Neural networks can be used successfully to solve such problems, because they can learn from examples, that are pairs of questions and correct answer. The purpose of learning is to design a network such that if the states of the nodes in the input layer are set equal to a question, the state of the output neurons will become equal to the correct answer. This problem is quite different from that of storing memories, because answer and question are related to each other by a rule, and because we are trying to deduce the answer to *new* questions, rather than just recall the answer to old ones.
Learning a rule by Neural networks is achieved by adjusting their weights according to a *learning algorithm* such that it performs well on the *training set*. If the examples in the *training set* are not independent and comprises knowledge about some rule which relates inputs to outputs, the NN can infer this rule from the data. This capability of NN is called the *generalization ability*. To design a network just from question and answer pairs is called *supervised learning*, because it requires a *teacher*, knowing the rule which gives the correct answer to the example questions. In this case the *learning algorithm* has to minimize some cost function E, which is a measure of the discrepancy between the answer of the network and the desired answer.
We can illustrate the learning process geometrically as follows. Suppose that we have $\mathcal{P}$ example questions, for each we know the corresponding answer. The known question and answer examples form the *training set*. Each example places a constraint upon the places in rule space where the true rule $\mathcal{B}$ must lie (see Fig(3.1)), this region is called the *version space*. After seeing $\mathcal{P}$ examples, the region in which $\mathcal{B}$ must lie is reduced to a region in the rule space around the true rule.
Let the rule to be learned is defined by the weight vector $\mathcal{B}$. The four planes $\mathcal{D}_1, .., \mathcal{D}_4$ are constraints on $\mathcal{B}$ in Fig(3.1) from examples 1 to 4, and each forces the weight vector of the learning network to lie on the undashed side of the plane. Note that the plane $\mathcal{D}_5$, from some example 5, adds no new information since planes $\mathcal{D}_1, .., \mathcal{D}_4$ already constrain the weight vector to an area agreeing with example 5: example 5 has not reduced the version space.

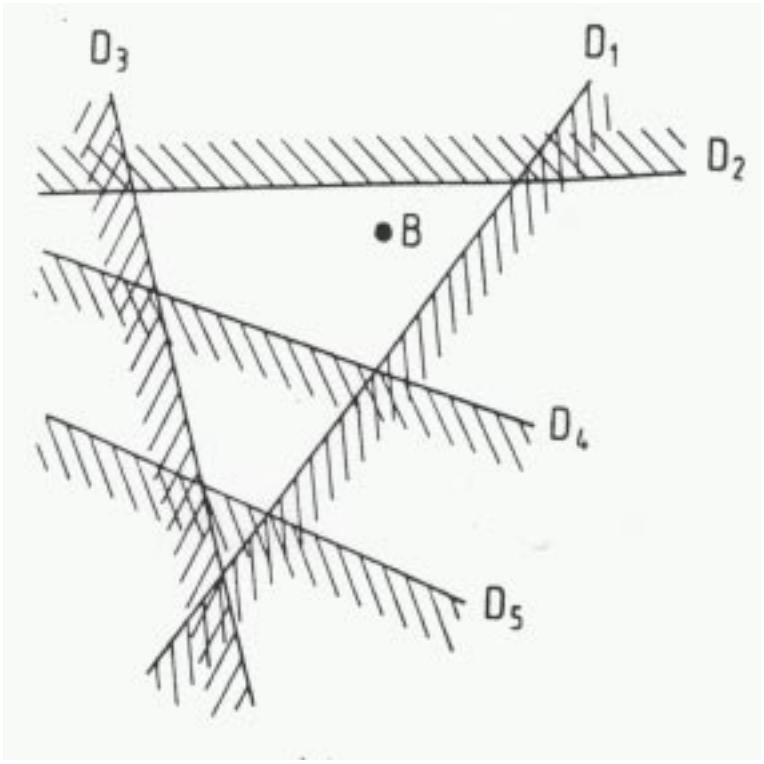

Figure 3.1: .

A rule for which a network in the learner space exists which will give the correct answer to all possible questions is called *learnable*. Conversely, a rule which no network in the learner space can learn exactly is called *unlearnable*. In this case, no such network exists, either because of architectural constraints or because of the presence of noise in the data. This case has been considered by several authors. Particularly relevant works are [28], which show that even with noisy data the underlying rule can be reproduced exactly.However, sometimes one would like also to find a classification of input patterns *without* knowing the answer of a teacher. In such a situation the input patterns must have a structure which the student network has to find out. This kind of learning is called *unsupervised learning*.

3.1.1 Definition of a rule

A rule defines a relationship between two variables X and Y. In the terminology of learning theory we call X the question, and it is a vector of (usually) high dimension; Y is called the answer, and it is a vector (usually of much smaller dimension). The most general way to describe the relationship between question and answer is a conditional probability law $P(y|x)$, which gives the relative frequency that the random variable Y takes the value y given that the random variable X takes the value$x$.
The rule can also be a deterministic law which associates an answer Y to the question

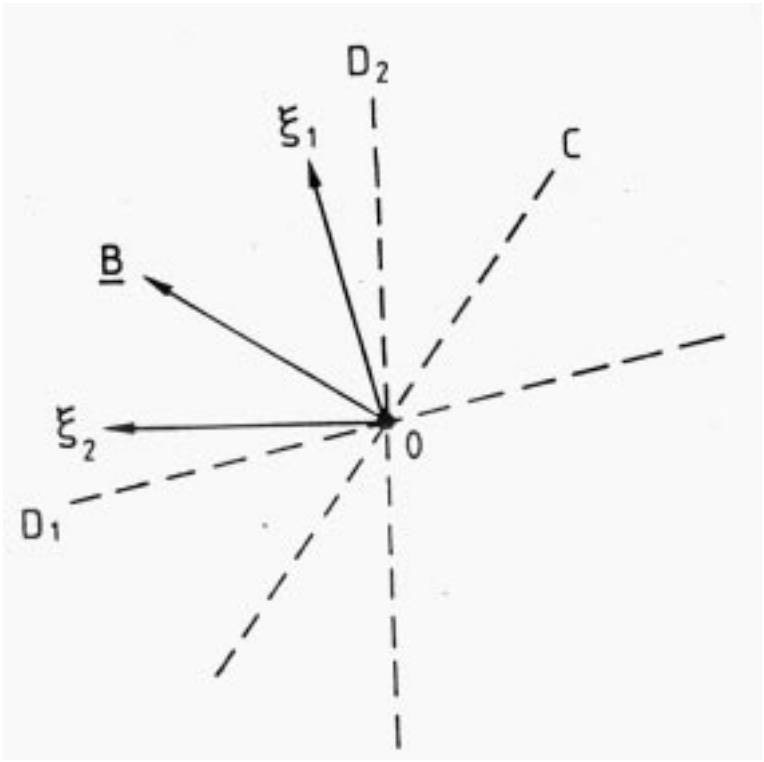

Figure 3.2: .

X so that.

$$Y = \mathcal{V}(X) \tag{3.1}$$

Thus the variable X is an element of the *question space*, Y of the *answer space* and $\mathcal{V}$ of the *rule space*. The rule may also be encoded in the *data sample*, which is a set of $\mathcal{P}$ independent *examples* $\{(\xi^1, \zeta_0^1), (\xi^2, \zeta_0^2), ..., (\xi^{\mathcal{P}}, \zeta_0^{\mathcal{P}})\}$, which comprises the empirical knowledge about the rule. Thus, each example consists of a question and its corresponding answer. In real-world problems the data is obtained by repeated measurements. The noise in the data represents some measure in the inaccuracy of these measurements. This accuracy together with the size of the training set determine how well the training set conveys knowledge about the underlying rule.
As an example for a deterministic rule we define a *linearly-separable* rule as one of the form

$$\mathcal{V}(X) = \text{sgn}(\frac{1}{\sqrt{N}}\mathcal{B}.X - \Phi) \tag{3.2}$$

Where $\mathcal{B}$ is any vector in the weight space. We show in Fig.(3.2) a section of the N dimensional *question space* containing vector $\mathcal{B}$, which is normal to plane C. Questions falling onto the same side of C as the positive direction of $\mathcal{B}$, such as ξ^1, ξ^2 will be answered by +1, and others which fall onto the other side of C will be answered by -1, so that the question space is divided by the plane C which is defined by the rule.

3.1.2 The teacher pupil problem

The teacher pupil method is a natural approach to study the properties of neural networks and especially their generalization ability, if we want to get general information about their behavior independent of a special training set.
For a given architecture, a *teacher network* is defined by fixing its weights according to some distribution. Many distributions are conceivable, but zero-mean symmetric distributions are often used in the first step.In order to represent typical cases of an ensemble of rules of a given kind, the values of the weights are fixed to random values. The teacher NN realizes a function, which represents the rule to be learned.
The training set is generated by feeding questions drawn from some other distribution as input into the teacher. The update algorithm is applied to compute the corresponding output. This output is the answer to be recorded together with the question, as one example of the training set.
A *pupil network* with architecture identical or a bit different from that of the *teacher network* is trained on the training set. In order to get results independent of the specific realization of the random variables in the previous steps *teacher's weights, questions in the training set, pupil's weights* averages over the underlying distributions are performed.
For theoretical work on the generalization ability of neural networks, the teacher-pupil problem is as essential as the problem of storing entirely random examples (random questions and random answers) for theoretical work on memorization. The teacher pupil problem is advantageous regarding numerical studies too.
For any kind of learner considered, a particularly important question to be answered is how many examples are required for training to achieve good generalization, i.e. successfully learning the rule and not the training sample. Therefore, the dependence of the training and generalization errors which are a measure of the deviation of the pupil's output from the teacher's output on the training examples and on new examples which have not been learnt before, on the training set size is in the focus of our study. To summarize established results, we sketch prior the theory involved.

3.1.3 Formalism of learning from examples

Let us characterize the output of a NN by the symbol $\sigma = \sigma(\mathbf{w}; \xi)$, with input vector ξ and the synaptic weights $\mathbf{w}$.
The goal of learning is to find a set of weights $\mathbf{w}^*$ such that $\sigma(\mathbf{w}^*; \xi)$ best approximates (or exactly realizes) some target rule $\sigma_\circ(\mathcal{B}; \xi)$ defined by a teacher with weight vector $\mathcal{B}$.
In order to measure the deviation of the network output from the teacher's output on a given input ξ, we introduce an error measure $\epsilon(\mathbf{w}, \mathcal{B}, \xi)$. It should be zero if pupil and teacher give the same answer to the question ξ, and positive everywhere else. The most obvious choice for ϵ is the binary measure $\epsilon \in \{0, 1\}$. The square error is also a

common choice for ϵ.

$$\epsilon(\mathbf{w}, \mathcal{B}, \xi) = (\sigma(\mathbf{w}, \xi) - \sigma_\circ(\mathcal{B}, \xi))^{\mathbf{2}} \tag{3.3}$$

as in nonlinear regression. We may define the *training energy* as

$$E_t(\mathbf{w}, \{\xi^\mu\}, \mathcal{B}) = \sum_{\mu=1}^{\mathcal{P}} \epsilon(\mathbf{w}, \mathcal{B}, \xi^\mu) \tag{3.4}$$

which is the sum of the errors on the whole training set of examples $(\xi^\mu, \sigma_\circ^\mu)$, with $\mu = 1, ..., \mathcal{P}$. The inputs ξ^μ are chosen at random from the input space according to some a priori normalized measure $d\mu(\xi)$. Learning is then accomplished by minimizing the energy, for example by gradient descent.

$$\frac{\partial \mathbf{w}}{\partial t} = -\nabla_{\mathbf{w}} E_t. \tag{3.5}$$

The partial derivative in Eq.(3.5) is with respect to the time.

Gradient descent schemes have been criticized because they tend to become trapped in local minima of the training energy. There is also another potential problem with the training energy: it measures the network's performance on a limited set of examples. The true goal of supervised learning is to find a student network which performs well on all the question space, not just on the training set. A good measure for the average disagreement between teacher's and the pupil's answers on the whole question space is the *generalization error*, which is defined as follows

$$\epsilon(\mathbf{w}, \mathcal{B}) = \int d\mu(\xi) \epsilon(\mathbf{w}, \mathcal{B}, \xi). \tag{3.6}$$

The integration over the normalized measure $d\mu(\xi)$ represents averaging over the distribution of the questions. $\epsilon(\mathbf{w}, \mathcal{B})$ is a measure of how good the pupil with weight vector $\mathbf{w}$ is in reproducing the teacher with weight vector $\mathcal{B}$.
For a given distribution of the training set questions ϵ is a function of the learning algorithm generating the pupil network $\mathbf{w}$, the number of examples introduced to the pupil to learn from and the distribution from which the questions have been taken. The average generalization error is then given by

$$\epsilon_g = \langle \langle \epsilon(\mathbf{w}, \mathcal{B}) \rangle_{\mathbf{w}} \rangle_{\mathcal{B}}. \tag{3.7}$$

where the bracket $\langle ... \rangle_{\mathcal{B}}$ refers to the average over the distribution from which the teacher has been chosen, and $\langle ... \rangle_{\mathbf{w}}$ indicates an average over the distribution of the pupils.
If almost any realization of the teacher and of the training set give the same result for ϵ_g is called *self-averaging*.
Like the gradient descent algorithm, any training algorithm determines some trajectory in the weight space. The performance of a training algorithm can be monitored by following the generalization and training errors as functions of time.

3.2 Learning at finite temperature

For physicists the learning proccess of a neural resembles the dynamic of a physical system evolving according to an energy. The space of weights can be explored by considering a stochastic learning process using any observable energy E. For continuous weights $\mathbf{w}$ and a differentiable energy function learning can be accomplished by allowing the weights to evolve according to a Langevin dynamic (Langevin 1908; Itzykson and Drouffe, 1989) which has the same form as the dynamics of a physical system in the limit of high viscosity

$$\frac{\partial \mathbf{w}}{\partial t} = -\nabla_{\mathbf{w}} E(\mathbf{w}, ...) - \nabla_{\mathbf{w}}(V(\mathbf{w})) + \eta(t). \tag{3.8}$$

where the dots in $E(\mathbf{w}, ...)$ stands for the dependence of the energy on the teacher's weight vector, the training set, or something else. Possible assumption about the a priori distribution and constraints on the range of the weights $\mathbf{w}$ can be included in the term $V(\mathbf{w})$which depends only on the weights and does not depend on the examples. $\eta(t)$ is a white noise term with variance

$$\langle \eta_i(t)\eta_j(\tilde{t})\rangle = 2T\delta_{ij}\delta(t-\tilde{t}). \tag{3.9}$$

T is a parameter which should denote a physical temperature at which the student network operates. As we can see the energy may increase occasionally due to the influence of this parameter on the dynamics of the NN. Whereas the first two terms in Eq. (3.8) tends to decrease the energy. At $T = 0$, the noise term drops out, leaving only a simple gradient descent equation. Gradient descent training algorithm are known for their drawback of becoming trapped in local minima of the energy surface, so that adding thermal noise is a successful way to prevent being trapped in local minima.
In simulated annealing algorithms for optimization problems the temperature is decreased slowly so that at $T \approx 0$ the system settles to a state with energy near the global energy minimum. Thermal noise could play the same role in the present training dynamics. In fact it may help to prevent the system from *overtraining*, namely finding an accurate fit to the training data at the expense of good generalization abilities. A training temperature is particularly important when trying to learn unlearnable rules.
For discrete $\mathbf{w}$ or a non differentiable energy function, one can use a discrete time Metropolis Monte Carlo dynamics, similar to that used in simulating Ising systems [39] .
One can now study the training process after the stochastic dynamic in Eq.(3.8) was applied for a long time t. Independently of the initial conditions of Eq. (3.8) at $t = 0$, for $t \to \infty$ Eq. (3.8) generates a Gibbs distribution over the space of solutions.

$$P(\mathbf{w})d\mathbf{w} = Z^{-1}e^{-\beta(E(\mathbf{w},...))}d\mu(\mathbf{w}). \tag{3.10}$$

$$d\mu(\mathbf{w}) = d\mathbf{w}e^{-\beta V(\mathbf{w})}. \tag{3.11}$$

The variance of the thermal noise in the training algorithm is considered in β which is the inverse of the temperature T of the Gibbs distribution T. β determines the extent to which we are allowed to explore the weight space during the training: $\beta \to \infty$ forces the system into the minimum of E, $\beta \to 0$ open the whole phase space. In general β determine the training error we tolerate.

$$T = \beta^{-1}. \tag{3.12}$$

The normalization factor Z is the *partition function*

$$Z = \int d\mu(\mathbf{w}) \exp\{-\beta E(\mathbf{w})\}. \tag{3.13}$$

If the energy E used in Eq.(3.13) is the training energy, then in the limit $\beta \to \infty$, Z is dominated by the $\mathbf{w}$ with minimal training error.
The contribution of $V(\mathbf{w})$ is incorporated into the normalized measure in weight space $d\mu(\mathbf{w})$. For a $\mathbf{w}$ which is equally likely to be in any direction and whose magnitude is fixed to one, the measure on the weight space is just $d\mu(\mathbf{w}) = d\mathbf{w}\delta(\mathbf{w}\cdot\mathbf{w} - 1)$. Similarly for Ising weights, if any combination of weights is a-priori equally likely, it is $d\mu(\mathbf{w}) = d\mathbf{w}\prod_i[\frac{1}{2}\delta(w_i - 1) + \frac{1}{2}\delta(w_i + 1)]$.
The powerful formalism of equilibrium statistical mechanics may now be applied to calculate thermal averages, i.e., averages with respect to the distribution of weights $P(\mathbf{w})$. They will be denoted by $\langle\cdots\rangle_T$. In the thermodynamic limit, such average quantities yield information about the typical performance of a network independent of the initial conditions of the learning dynamics. Thermal averages of other observables are performed if the logarithm of the partition function Eq.(3.13) is calculated, for example

$$E_t = \langle E(\mathbf{w})\rangle_T = -\frac{\partial \ln Z}{\partial \beta}. \tag{3.14}$$

By introducing an additional, auxiliary term into the energy

$$E(\mathbf{w}) \to E(\mathbf{w}) + h\epsilon_g(\mathbf{w}, \mathcal{B}). \tag{3.15}$$

where the auxiliary field h will be set later to zero, we could calculate the thermal average of the generalization energy

$$\langle\epsilon_g(\mathbf{w}, \mathcal{B})\rangle_T = -\frac{1}{\beta}\frac{\partial \ln Z}{\partial h}. \tag{3.16}$$

We could also calculate the expectation value of any quantity by changing the coefficient of h, provided we are able to evaluate that coefficient as a function of $\mathbf{w}$.
Even after the thermal average is done, there is still a dependence on the training set. Since the examples have been chosen randomly and then fixed, they represent a

quenched disorder. Thus to calculate the typical behavior we must perform a second average over the distribution of the training examples. This is denoted by $\langle\langle\cdot\rangle\rangle \equiv \int \prod_\nu d\mu(\xi^\nu)$.

The free energy F and entropy S of the network are then given by

$$F(\mathcal{P}, T) = -T\langle\langle \ln Z\rangle\rangle. \tag{3.17}$$

$$S(\mathcal{P}, T) = -\int d\mu(\mathbf{w})\langle\langle P(\mathbf{w}) \ln P(\mathbf{w})\rangle\rangle. \tag{3.18}$$

where $\mathcal{P}$ is the number of training examples. The entropy is an important *extensive* variable, it measures the number of states accessible by the dynamics for a given temperature in the discrete case; additionally it gives (except for an offset) the information content of the Gibbs distribution . The free energy F and the entropy are related by the identity:

$$F = E_t - TS. \tag{3.19}$$

Knowing F, E_t can be calculated as follows

$$E_t = \frac{\partial \beta F}{\partial \beta}. \tag{3.20}$$

and the entropy is given as a partial derivative of F:

$$S = -\frac{\partial F}{\partial T}. \tag{3.21}$$

The average generalization and training errors are given by

$$\epsilon_g = \langle\langle\langle \epsilon(\mathbf{w}, \mathcal{B})\rangle_T\rangle\rangle, . \tag{3.22}$$

$$\epsilon_t = \mathcal{P}^{-1}\langle\langle\langle E(\mathbf{w}, \mathcal{B})\rangle_T\rangle\rangle. \tag{3.23}$$

The above results will be exact in the thermodynamic limit, i.e. when the number of degrees of freedom, namely the total number of independently determined synaptic weights N of the network, approaches infinity. For the limit $N \to \infty$ to be well defined we note that the problem at the hand as well as the network architecture allow for a uniform scale up of N. However the results obtained in the thermodynamic limit should provide a good approximation of to the behavior of networks with a fixed large size.

As we have seen above, calculating the average of logarithms of the partition function is the key to calculate many other quantities, but this is not simple for any temperature and sample size. The *replica method* is the technique commonly used to perform such averages. In what follows we will introduce this method.

3.3 The replica method

As we have seen in the last section, the free energy is a key quantity which generates the moments of the Gibbs distribution. If this quantity is known for any realization of the training set, then the other thermodynamic quantities can be calculated from it. In order to perform the average of the free-energy over the random variables, the *replica method* is the technique commonly used. It was first developed for analyzing rubber elasticity by Edwards [35] and then applied to study highly frustrated magnetic systems called *spin glasses* by (Edwards and Anderson and many others). E.Gardner [37] reapplied it to analyze the student space of NN.
The replica trick is used in cases in which averaging of the partition function is easer to perform, but that of the free-energy, the logarithm of the partition function Z, is difficult. This would be typically the case if the random variables appear in the energy linearly or quadratically, or if they appear uncorrelated at different sites. The replica method is based on the representation of the logarithm in terms of a power function through the identity

$$\langle\langle \ln Z \rangle\rangle = \lim_{n\to 0} \frac{\ln\langle\langle Z^n \rangle\rangle}{n}. \tag{3.24}$$

It implies that in order to obtain the average of $\ln Z$ one can average Z^n. Now averaging over an integer power of Z is not much more complex than the task of averaging Z itself. But the hitch is in taking the limit. The average can be performed for any integer n, but not for $n = 0$. One must interpret the procedure to be a calculation of the average of the right hand side as a function of n, and then the limit is reached by analytic continuation. Provided that there is no phase transition for any finite n, this procedure should give the correct result [38]. The formal aspects of this analytic procedure leave much to be desired, but this does not render the technique any less useful. One notes that Z^n is itself like a partition function of n identical copies of the original system which, for any given set of the random variables, do not interact. These are called *replicas*. They come about in the following way. If the energy of the system is the training energy

$$E_t(\mathbf{w}, \{\xi^\mu\}, \mathcal{B}) = \sum_{\mu=\mathbf{1}}^{\mathcal{P}} \epsilon(\mathbf{w}, \xi^\mu, \mathcal{B}) \tag{3.25}$$

One can write

$$\langle\langle Z^n \rangle\rangle = \int (\prod_{\gamma=1}^{n} d\mu(\mathbf{w}^\gamma)) \langle\langle \exp\{-\beta \sum_{\mu,\gamma} \epsilon(\mathbf{w}^\gamma, \xi^\mu, \mathcal{B})\}\rangle\rangle \tag{3.26}$$

$$= \int (\prod_{\gamma=1}^{n} d\mu(\mathbf{w}^\gamma)) \exp\{-\beta \mathcal{H}(\{\mathbf{w}^\gamma\})\}. \tag{3.27}$$

where γ in Eq.(3.27) is the replica index, and $\mathcal{H}$ the effective Hamiltonian in the replicated space. It is given by

$$\mathcal{H}(\{\mathbf{w}^\gamma\}) = -\frac{1}{\beta}\ln\left[\int d\mu(\xi)\exp\{-\beta\sum_{\gamma\mu}\epsilon(\mathbf{w}^\gamma,\xi^\mu)\}\right]. \tag{3.28}$$

$d\mu(\xi)$ is the normalized distribution from which the questions are chosen. Once the average over the random variables has been carried out, the various replicas of the system begin to interact. In the cases of interest it will turn out that the Hamiltonian Eq.(3.28) depend on the replicated weights $\mathbf{w}^\gamma$ only through the order parameters of the form

$$q_{\gamma\gamma`} = \frac{1}{N}\mathbf{w}^\gamma\cdot\mathbf{w}^{\gamma`}. \tag{3.29}$$

$$R_\gamma = \frac{1}{N}\mathbf{w}^\gamma\mathcal{B}. \tag{3.30}$$

and the partition function can be written in terms of a multiple integral over these order parameters. The integral in Eq.(3.27) will be evaluated in the thermodynamical limit $N\to\infty$ by the *saddle-point method*, which requires to minimizing a free energy function (with respect to the order parameters of the system) derived from $\mathcal{H}(\{\mathbf{w}^\gamma\})$. In fact one must perform two limits $n\to 0$ (from the analytic continuation) and $N\to\infty$ (from the thermodynamic limit). In principle one should take $n\to 0$ first and then $N\to\infty$. Particularly, however, these limits can only be performed by reversing the order of the limits. Another point that may cause problems, is that one must make some assumption about the order parameters. By the symmetry of different replicas, one expects the values of different order parameters are invariant under permutations the replica; this assumption is called replica symmetry (RS). As we will see in the following chapters, for some thermodynamical systems this (RS) must be given up at least in some region of the α, T parameter space, and more complicated structure of the order parameter must be used to reach better results.

3.4 The high temperature limit

A simple and interesting limit of the learning theory is that of high temperature. Using a cummulant expansion we can write the free energy Eq.(3.17) as a power series in β, with coefficients that are functions of $\frac{\mathcal{P}}{N}\beta$. The zeroth order term of this expansion is the high temperature limit. This term represents the non random part of the free energy, which does not couple different replicas (see Sompolinsky *et.al.*[10],1990). In this limit the energy can simply be replaced by its average $\epsilon(\mathbf{w},\mathcal{B})$, and the fluctuations δE due to the finite sample of the training set can be ignored. Hence the equilibrium distribution of weights is given by

$$\mathcal{P}_0(\mathbf{w}) = Z_0^{-1}\exp\{-\beta\mathcal{P}\epsilon(\mathbf{w},\mathcal{B})\}. \tag{3.31}$$

and the partition function reduces to

$$Z_0 = \int d\mu(\mathbf{w}) \exp\{-N\alpha\beta\epsilon(\mathbf{w}, \mathcal{B})\}. \tag{3.32}$$

The expressions above show that for $\beta \to 0$ the limit is only defined if $\alpha \to \infty$ leaving $\alpha\beta$ constant. In this limit all thermodynamical quantities, including the average training and generalization errors, are functions of the effective temperature $\frac{T}{\alpha}$. It is known that by the high temperature limit in statistical mechanics all states of the system becomes equally likely, regardless of the energy. But here the simultaneous $\alpha \to \infty$ guarantees nontrivial behavior, so that as the effective temperature $\frac{T}{\alpha}$ decreases, the network approaches the optimal *ground state* weight vector $\mathbf{w}^*$ which minimizes $\epsilon(\mathbf{w})$. α here is given by $\mathcal{P} = \alpha N$. The properties of the system are determined by the dependence of the entropy on the generalization error. A very important feature of learning at high temperature is the lack of overtraining. One measures overtraining by the difference between the expected training and generalization errors, i.e., $\epsilon_g - \epsilon_t$. From Eq.(3.31) and the definition of ϵ_g and ϵ_t it follows that $\epsilon_g = \epsilon_t$ in the high T limit. Of course the price that one pays for learning at high temperature is the necessity of a large training set, as α must be at least of order T.

3.5 The annealed approximation

In the annealed approximation(AA), the average of the logarithm of the partition function$Z$ is replaced by the logarithm of the average of Z, resulting in the *annealed free energy.*

$$F_{an} = -T \ln\langle\langle Z \rangle\rangle. \tag{3.33}$$

Falk (1975)has pointed out that the annealed free energy is a lower bound for the correct free energy, this is due to the convexity of the logarithm function. But the minima of the annealed and quenched free energies are not always at the same place, thus they do not always lead to the same equilibrium results. Seung, Sompolinsky and Tishby [10] have pointed out that this approach is the exact theory for a dynamical process where both weights and the examples are updated according to a stochastic dynamic similar to Eq.(3.8), involving the same energy function. Because the AA reduces to the high-temperature limit for $T \to \infty$, it is valid for high temperature. It is a more powerful approximation than the high-temperature in that it predicts some of the effects of quenched disorder, but its predictions cannot be guaranteed to be exact at finite temperature generally, regarding both generalization and training errors. With respect to the generalization error, the AA correctly predicts the asymptotic behavior when $\alpha \to \infty$ if either the rule is realizable by the pupil or the pupil NN has a single boolean output [10].

4 The perceptron

4.1 Definition

The perceptron(Rosenblatt, 1962)[57] consists of N input nodes and one output node, and a transfer function between input and output.
A perceptron is defined through the fact that its output depends only on the local field h^μ. The output is given by

$$\sigma^o = f(h^\mu) \tag{4.1}$$

The local field is given as follows

$$h^\mu = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} \xi_i^\mu w_i - \psi \tag{4.2}$$

where ξ^μ represents the input vector, w the weight vector and, ψ is a constant *threshold*. The factor $\frac{1}{\sqrt{N}}$ appears in Eq.(4.2) to keep the local field of order one.
The sum in Eq.(4.2) is over all N values of i. The variables ψ and $\{w_i\}$ completely defines the perceptron's action.
If the state of the output is restricted to be ± 1, we might chose the function f so that

$$\sigma^o = \mathrm{sgn}(\frac{1}{\sqrt{N}} \sum_{i=1}^{N} \xi_i^\mu w_i - \psi) \tag{4.3}$$

The perceptron in which σ^o is restricted to be ± 1 is called a *binary perceptron*. Choosing $f(x) = x$ so σ^o may take any real value, makes the system a *linear perceptron* (Herz et al., 1991)[56].
If each component of the weight vector w_i can take any real value subject to the normalization condition $\sum_i {w_i}^2 = 1$, we call the perceptron *spherical*; if it is restricted to be ± 1, the resulting perceptron is called *Ising*. Thus for example an *Ising binary perceptron* has a ± 1 output and the components of the $\mathbf{w}$ are also ± 1. The spherical and Ising constraint are the upper and lower extremes of the $\mathbf{w}$-space. Of the two, the Ising case is the most instructive because in real networks engineering restrictions are likely to lead to quantized weights.
Notice that the output, σ^o, of the perceptron is invariant under the transformations

$$w_i \to \lambda w_i, \forall i \tag{4.4}$$

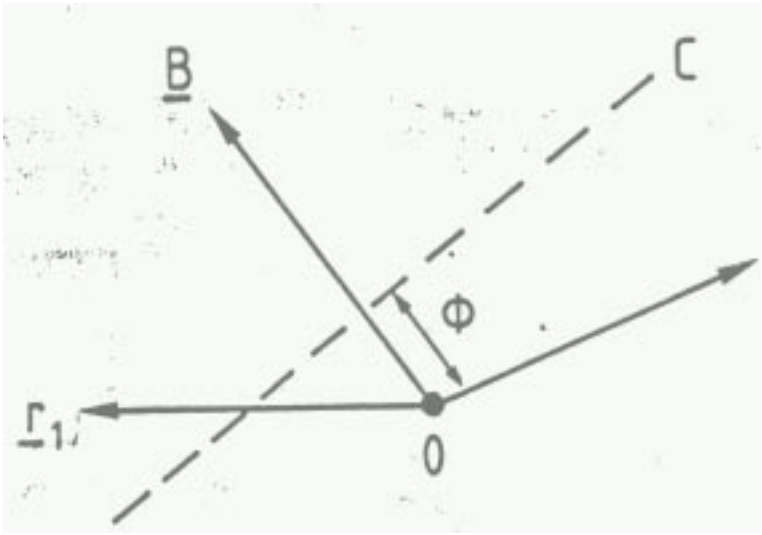

Figure 4.1:

$$\xi_i \to \nu\xi_i, \forall i \tag{4.5}$$

$$\psi \to \nu\lambda\psi \tag{4.6}$$

for any positive value of λ and ν. These are the *gauge freedom* of the percepton. Normally the λ gauge is fixed so that

$$\mathbf{w}.\mathbf{w} = N \tag{4.7}$$

It is clear that a perceptron can learn only linearly separable problems, Eq.(4.3).
The space of possible spherical $\mathbf{w}$ is the unit N-sphere. If the perceptron is Ising, then $\mathbf{w}$ lies on one of the corners of a unit N-cube.
An easy way to visualize the action of a perceptron is from a schematic diagram, Fig.(4.1) of the N-dimensional *input space*, which is all the possible configuration of the nodes in the input layer. The vector $\mathbf{B}$ is shown in this space, and is perpendicular to the $(N-1)$ dimensional hyperplane C, which is displaced from the origin by a distance Φ. Only input configurations, such as $\mathbf{r}_1$, which fall on the same side of C as the direction of $\mathbf{B}$ have a positive $\frac{1}{\sqrt{N}}\sum_{i=1}^{N}\mathbf{r}_i B_i - \Phi$ and will cause the output σ^o to be set to $+1$. Throughout the rest of this work the perceptron is further restricted to $\Phi = 0$, so that C passes through the origin.

4.2 Results for perceptron learning

4.2.1 The Boolean perceptron

For the Boolean perceptron, the transfer function is

$$f \equiv \text{sgn} \tag{4.8}$$

The generalization error for an isotropic distribution of the questions gets

$$\epsilon_g(\mathbf{w}) = \frac{1}{\pi}\arccos(R) \tag{4.9}$$

where R is the average overlap between teacher's and pupil's weight vectors. $\epsilon_g(\mathbf{w})$ is the probability that the teacher and pupil disagree. This relation gets evident (Hertz et al.,1991) by a geometrical argument, noting that $\arccos(R)$ is the angle between the teacher and pupil weight vector, if the pupil weights vector is exposed to the same spherical constraint as the teacher.

4.2.2 Boltzmann algorithm

The gradient descent algorithm, which leads to the Gibbs distribution after long time, is denoted *Boltzmann algorithm.*The asymptotic behavior of the generalization error for $\alpha \to \infty$, is obtained by the replica method [10](Seung et al., 1992). For a finite temperature, the approach to perfect learning is algebraic,

$$\epsilon_{Boltz} \sim \frac{1}{\alpha}, \qquad \alpha \to \infty \tag{4.10}$$

At zero temperature, the decay is (see [28])

$$\epsilon_{Boltz}(T=0) \sim \frac{0.652}{\alpha} + \mathcal{O}(\alpha^{-2}), \qquad \alpha \to \infty; \tag{4.11}$$

The annealed approximation yields the correct power law, but does not predict the pre factor correctly(1 instead of 0.625 for $T = 0$).

4.2.3 Maximum stability algorithm

The contribution to the training error from an example μ is 1 if, and only if the pupil gets the example wrong. However, it is possible to make this contribution more sophisticated by relating it to a physical quantity which measures the certainty with which the student gets the question right. With σ_t^μ the respond of the teacher and h^μ the local field of the pupil we can define the overlap

$$\Lambda^\mu = \sigma_t^\mu h^\mu \tag{4.12}$$

Λ^μ is called the stability. It measures the certainty with which the student gets the questions right. $\Lambda^\mu \geq 0, \forall \mu$ guarantees zero training error. Geometrically Λ^μ is the distance between the weight vector of the pupil perceptron $\mathbf{w}$ and the boundaries set by the example ξ^μ on the version space (this is a hyperplane $\mathcal{D}_\mu$ which is perpendicular

to ξ^μ).
It is plausible that requiring $\Lambda \geq \kappa$ with

$$\Lambda = \min_\mu \Lambda^\mu, \tag{4.13}$$

will be advantageous for the generalization ability, because this requirement pushes the weight vector $\mathbf{w}$ of the pupil into a reduced volume in the version space towards the weight vector of the teacher. Clearly as the value of κ rises, the volume available for $\mathbf{w}$ decreases until at some value of κ it shrinks to a point: the $\mathbf{w} - vector$ with the maximum stability, κ_{max}.This is the *maximum stability algorithm*, MSA. The asymptotic behavior of the MSA obeys the same power law as the Boltzmann algorithm, however with a slightly different pre factor. A Replica symmetric ansatz yields [32]

$$\epsilon_{MSA} \sim \frac{0.57}{\alpha}, \qquad \alpha \to \infty. \tag{4.14}$$

For known maximal stability $\kappa = \max_{\mathbf{w}} \min_\mu \lambda^\mu$, the MSA can be implemented through the Langevin dynamics (see chapter 3) at zero temperature by using the training energy

$$E_{MSA}(\mathbf{w}, \kappa) = \sum_\mu \Theta(\kappa - \sigma_t^\mu h^\mu). \tag{4.15}$$

This is equivalent in the zero temperature limit to analyze explicitly the fractional volume of the student space embedding every input-output configuration with a stability larger than κ. Meier and Fontanari [48] have recently extended this analysis to minimizing energies of the form

$$E_r = \sum_\mu (\kappa - \lambda^\mu)^r \Theta(\kappa - \lambda^\mu); \qquad r = 0, 1, 2 \tag{4.16}$$

For r=0 this reduces to E_{MSA}. They found that, always selecting the best value for κ, $\epsilon_g(\alpha)$ decreases faster as r is increased beyond 0.
Numerical solutions show that even the original (zero stability/ online) perceptron algorithm performs comparably well.

4.2.4 The Bayes algorithm

There is an *optimal* information-theoretic prescription for predicting the answer to new questions from known examples. Using the Bayes theorem (see[20]), we can calculate the posterior probability of any given rule having generated the training set. The information-theoretic (Bayesian) prediction of the answer to new example is the one which maximizes the posterior probability.

The probability that the Bayes algorithm gives a wrong answers to a new example, $\epsilon_g^{Bayes}(\xi^\mu)$, is the probability that the true rule $\mathcal{B}$ lies in one of the regions of version

space with a different answer to the question. The average Bayesian generalization error, ϵ_g^{Bayes}, is the average of $\epsilon^{Bayes}(\xi^\mu)$ over the distribution of (ξ^μ). Regarding information theory, the Bayes algorithm is optimal in predicting answers to new examples based on known examples.
Opper and Haussler show that the Bayes algorithm can be approximated by a NN with one hidden layer of n (n odd) neurons. The hidden neurons are trained independently using the Boltzmann algorithm, and the output neuron computes the majority (i.e. the weights from the hidden neurons to the output neurons are fixed to 1), such an architecture is called committee machine. They prove that in the limit of large n, the average generalization error ϵ_n of this NN converges to the average error of the Bayes Algorithm, $\epsilon_n \to \epsilon_{Bayes}$ for $n \to \infty$. Note that $\epsilon_1 = \epsilon_{Boltz}$.
Assuming that the distribution of the rules $d\mu(\mathcal{B})$ and $d\mu(\xi)$ are such that the weighted inputs to the hidden neurons obey the central limit theorem, and further replica symmetry, Haussler and Opper found the following relation

$$\epsilon_{Bayes}(\alpha) = \frac{1}{\pi} \arccos \sqrt{\cos \frac{1}{\pi} \epsilon_{Boltz}(\alpha)}. \tag{4.17}$$

between the error of the Bayes algorithm and that of the Boltzmann algorithm. The asymptotic development for large α,i.e. small generalization error, is

$$\epsilon_{Bayes}(\alpha) \sim \frac{1}{\sqrt{2}} \epsilon_{Boltz}(\alpha), \qquad \alpha \to \infty \tag{4.18}$$

For $\alpha = 0$, $\epsilon_{Bayes}(0) = \epsilon_{Boltz} = 0.5$, i.e. random guessing as expected, and for small α the approach to 0.5 is

$$\epsilon_{Bayes}(\alpha) \sim 0.5 - \sqrt{\pi(0.5 - \epsilon_{Boltz}(\alpha))}, \qquad \alpha \to 0. \tag{4.19}$$

Using the results for the Boltzmann algorithm with uniform spherical distribution of both the questions and the teacher weights (Györgyi, 1990b), they finally obtain the asymptotic behavior of the learning curve to be

$$\epsilon_{Bayes} \sim \frac{0.44}{\alpha}, \alpha \to \infty. \tag{4.20}$$

Starting from random guessing at $\alpha = 0$, the Bayes algorithm improves with infinite slope $(0.5 - \epsilon_{Bayes}(\alpha) \sim \sqrt{(\frac{2\alpha}{\pi^3})}, \alpha \to 0)$ whereas for the Boltzmann algorithm this improvment is only linear in $\alpha (0.5 - \epsilon_{Bayes}(\alpha) \sim \frac{2}{\pi^3}\alpha, \alpha \to 0)$.

4.2.5 The Hebb algorithm

The Hebb algorithm is the simplest strategy to correlate the output of the teacher and that of the pupil. The Hebb algorithm constructs the weight vector of the pupil from the trainig examples so that

$$\mathbf{w}_i = \frac{1}{N}\sum_{\mu}^{\mathcal{P}} \sigma_0^{\mu}\xi_{\mathbf{i}}^{\mu} \tag{4.21}$$

It is clear that the components of examples in the direction of the teachers weight vector will add constructively to $\mathbf{w}$. For a specific form of the teacher (each value of the teacher's weights occurs with identical frequency) the generalization error is calculated to be [21]

$$\epsilon_{Hebb}(\alpha) = \frac{1}{\pi}\arctan\sqrt{\frac{\pi}{2\alpha}}. \tag{4.22}$$

The result does not depend on the specific form of the teacher (as long as the central limit theorem can be applied), as an alternative derivation based on geometrical signal-to-noise argument shows Watkin et al. [20].
Asymptotically, the generalization error of the Hebb algorithm is

$$\epsilon_{Hebb} \sim \frac{1}{\sqrt{2\pi\alpha}}, \alpha \to \infty; \tag{4.23}$$

The Hebb algorithm is simple, but the $\mathbf{w}$ it generates has a finite training error ϵ_t. It rises from zero to a peak at $\alpha \approx 4$, but then tends to ϵ_{Hebb}, and together they tend to zero as $\alpha \to \infty$ (when $\mathbf{w} \to \mathcal{B}$). The rise and fall of ϵ_t is not typical of other algorithm; typically ϵ_t either remains zero or rises monotonously. These results can be understood as follows, For a small α the training set is learned by memorization without understanding the rule. This memorization gets harder for large α because of the well known limitation of the storage capacity of the Hebb rule. For large α, the pupil gets more and more aligned with the teacher weight vector, i.e. the pupil understands the rule presented by the teacher. The training error approaches the generalization error, because the training set is too large to memorize, and both the training and generalization error approach perfect learning $\propto \frac{1}{\sqrt{2\pi\alpha}}$.

4.3 Learning a rule with an Ising perceptron

4.3.1 learning a realizable rule with an Ising perceptron

If there exist a weight vector in the pupil's weight space, which can learn the rule generated by the teacher perfectly, we call such a rule realizable. If both the weights of the teacher and that of the pupil are restricted to ± 1, the pupil is able to learn the rule produced by the teacher perfectly. At zero temperature and zero stability the annealed approximation and the replica symmetric approach were analyzed by Györgyi, Seung et. al [29]. We first report on the replica symmetric results. For $\alpha < \alpha_{th} \approx 1.245$ the replica symmetric free energy has two minima, a result first derived by Gardner and Derrida, one at an overlap with the teacher $R = 1$ and the other at $0 < R < 1$. The solution at $0 < R < 1$ has the lower free energy and is therefore the equilibrium

solution. For $\alpha < \alpha_{th}$ there are many pupils with zero training error, and most of them have an overlap with the teacher $0 < R < 1$. As α is increased beyond α_{th}, the solution at $R = 1$ becomes the absolute minimum, however the solution with $0 < R < 1$ persists until the spinodal point $\alpha_{sp}^{RS} \approx 1.49$ is reached.
For $\alpha_{th} < \alpha < \alpha_{sp}^{RS}$ stochastic algorithms which iterate from a random initial configuration ($R \approx 0$) will not converge to the global minimum of the free energy, but become stuck in one of the meta-stable states. This part of the phase diagram is interpreted as a spin glass phase [31], rich in meta-stable states. Regarding the analysis of the spin glass phase, the assumption of replica symmetry must be given up. This leads to replica symmetry breaking RSB, which results in a more complicated analysis and less transparent interpretation.
Seung , Sompolinsky and Tishby have analyzed the first step replica symmetry breaking [10] in accordance with the Parisi theory. For small enough temperature they found a meta-stable phase. The thermodynamic transition temperature is still at $\alpha_{th} \approx 1.245$, whereas the spinodal line indicating the disappearance of the spin-glass phase is shifted to $\alpha_{sp}^{RSB} \approx 1.63$.

4.3.2 Learning an unrealizable rule with an Ising perceptron

Seung, Sompolinsky and Tishby [10] have analyzed the case of an Ising perceptron learning a spherical perceptron. Here each of the components of the teachers weight vector $\mathcal{B}$ is drawn from a Gaussian distribution with variance one and zero mean, which means that on average $\mathcal{B}$ is constrained to lie on the N-dimensional unit sphere. The pupil's weights are taken to be $w_i = \pm 1$. For any fixed temperature, according to the replica symmetric theory, the asymptotic dependence of the generalization error on the number of examples is of the form $\epsilon_g - \epsilon_{min} \sim \frac{1}{\alpha}$ which is as slow as the Hebb learning of a learnable rules. The annealed approximation fails completely by predicting $\epsilon_g - \epsilon_{min} \sim \alpha^{-2}$ dependency.
Seung, Sompolinsky and Tishby showed [10] that in the case of using the training energy as the quantity to be minimized, ϵ_g does not have a monotonic dependence upon the training temperature. If one employs the optimal training temperature at each value of α, then learning becomes asymptotically faster, $\epsilon_{min} - \epsilon_g \sim \alpha^{-\frac{4}{5}}$.

5 Theoretical investigation

5.1 Introduction

This chapter deals with the analytical investigation of learning a linearly separable rule produced by a simple perceptron (a teacher) by another simple perceptron (pupil). The weights of the pupil are restricted to discrete values with some synaptic depth, also referred to as synaptic precision, whereas the weights of the teacher can take any real value satisfying the spherical constraint Eq.(5.2).
As we have mentioned in the last chapters, the range and precision of weights in hardware implementations may limit neural network performance and applications, when trained to learn a rule produced by a continuous weighted teacher. Thus we will specifically focus on the investigation of the effects of the precision of discrete weights of a simple perceptron on its performance. In particular we study the dependence of the learning curves of a simple pupil perceptron with discrete weights, on the size of the training set at a finite training temperature, on the temperature with a constant training set size, and on the number of allowed discrete values of the weights (there are 2^L allowed values for a synaptic precision of L.). The de'Almeida Thouless, stability line, and the Gardner Derrida line will be also studied.
By learning curves we refer to a plot of the average generalization and training errors on the size of the training set for a given number of allowed values of the weights.
A proper treatment of learning from examples requires the use of sophisticated techniques from statistical mechanics of disordered systems. Since we are dealing with the simplest neural network, this task can be analyzed analytically by extending Gardner's formalism (Gardner 1988, Gardner and Derrida 1988). Gardner showed that the critical storage capacity of a perceptron can be deduced when the fractional volume in phase space of the weights w_{ij}, within which the conditions for learning or storing the presented pattern are valid, shrinkes to zero.
Gardner's work effectively decouples the question of the existence of solutions from the problem of actually producing such solutions using a specific learning algorithm. Following Gardner we use the methods of statistical physics discussed in chapter 3, the replica trick and the saddle-point method, to calculate the learning curves of our model.
The microscopic states of our system are given by the weights $\mathbf{w}$. We assume that the examples presented to the learning network, to be referred to as the *pupil*, are drawn

from another single-layer perceptron, which is called the *teacher*. The training examples are the quenched variables which introduce the disorder in the system, and over these random variables the average will be taken. Another average must also be done. This is the average over the probability distribution of the teacher's weight vector $\mathcal{B}$, which is assumed to be Gaussian in our case.
Because we are interested in typical results independent of a special choice of the training examples, we focus on the case of independent uniformly distributed random binary $\{\pm 1\}$ input-output training examples.

5.2 The formalism

Throughout this thesis we will be concerned with a perceptron of N input neurons described by the N-dimensional vector ξ, with components $\xi_i \in \{\pm 1\}, i = 1, ..., N$. There is only a single output ζ_0 which is given by

$$\zeta_0^\mu = \mathrm{sgn}(\mathcal{B}.\xi^\mu) \tag{5.1}$$

where the scalar product $\mathcal{B}.\xi$ of the two N dimensional vectors is defined by $\mathcal{B}.\xi = \frac{1}{\sqrt{N}} \sum_i \mathcal{B}_i \xi_i$.
We assume the student will learn from a stream of $\mathcal{P} = \alpha N$ input output examples $(\xi^1, \zeta_0^1), (\xi^2, \zeta_0^2), ..., (\xi^\mathcal{P}, \zeta_0^\mathcal{P})$, where the inputs $\xi^\mu = (\xi_1^\mu, \xi_2^\mu, ..., \xi_N^\mu)$ are drawn at random. The outputs ζ_0^μ are assumed to be generated (computed) by a single-layer perceptron (the teacher). Each ξ_j^μ is chosen independently and randomly from the set $\{\pm 1\}$ with equal probability $prob(1) = prob(-1) = \frac{1}{2}$.
The vector $\mathcal{B}$ in Eq.(5.1) is the weight vector of the teacher. The components of $\mathcal{B}$ take their values from the real numbers. The weight vector $\mathcal{B}$ in our model will be chosen to satisfy the spherical constraint

$$\frac{1}{N} \sum_i \mathcal{B}_i^2 = 1. \tag{5.2}$$

The response of the student perceptron to an input ξ^μ is similarly given by

$$\zeta^\mu = \mathrm{sgn}(\gamma^\mu). \tag{5.3}$$

where γ^μ is referred to as the local field and is given by

$$\begin{aligned} \gamma^\mu &= \frac{1}{\sqrt{N}} \sum_i (\mathbf{w}_i.\xi_i^\mu) \\ \gamma_0^\mu &= \frac{1}{\sqrt{N}} \sum_i (\mathcal{B}_i.\xi_i^\mu) \end{aligned} \tag{5.4}$$

The weights $\mathbf{w}$ of the pupil in Eq.(5.3) take their values from the set

$$\mathbf{\Omega_w} \equiv \{w_0, ..., w_{L_w}\}. \tag{5.5}$$

Here L_w is the number of possible values and is sometimes referred to as the *synaptic depth* (continuous weights have infinite synaptic depth). For a precision of L bits there are 2^L allowed different values for the weights. If we include the zero in the allowed values we have then $2^L + 1$ allowed values.
To fix the weights $w_1, ..., w_{L_w}$ we divide the interval of allowed weight range $[-range, range]$ into equal pieces after the following rule

$$w_i = -range + \frac{i}{2^{L-1}}. \tag{5.6}$$

where i takes the values $i \in \{0, ..., 2^L\}$. Because of the scaling invariance of the perceptron, see chapter 4, we take the interval $[-range, range]$ equal to $[-1, 1]$.
From the examples we want to construct a weight vector $\mathbf{w}$ that will find the correct answer $\zeta_0^{\mu+1}$ to a random new question $\xi^{\mu+1}$. The chance that the answer will be correct is the generalization ability $\mathcal{G}$. Clearly a randomly chosen $\mathbf{w}$ will give $\mathcal{G} = \frac{1}{2}$, if $\gamma_0^{\mu+1}$ has an equal chance of being ± 1. The goal of the training process is to minimize the generalization error defined as $\epsilon_g = 1 - \mathcal{G}$, which, as we have pointed out in chapter 3, is not necessarily equivalent to minimizing the number of training examples that the perceptron gets wrong.
Fig(5.1) shows the projection of the N-dimensional space onto a two-dimensional hyperplane containing the weight vector of the pupil $\mathbf{w}$ and that of the teacher $\mathcal{B}$, which lie on the N-sphere centered at the origin and are perpendicular to planes A and C, respectively. The hyperplane D, composed of all N-vectors whose overlap with $\mathcal{B}$ is R, intersects the unit N-sphere on an N-1 dimensional sphere, whose projection into the plane of the diagram is the solid part of the line D. It is clear from the diagram that the radius of the small sphere is just $\sin(\theta)$. To within a constant the entropy of the spherical perceptron is the logarithm of the area of this surface. Random examples lie randomly on the surface of the N-sphere and it is clear that in the case of both Ising and spherical perceptrons, the generalization function $\mathcal{G}$, is the proportion of the area of the N-sphere in the regions E and H. Thus,

$$\mathcal{G}(R) = 1 - \frac{\theta}{\pi} = 1 - \frac{cos^{-1}(R)}{\pi}. \tag{5.7}$$

It has been pointed out that this geometrical formulation relies upon the components of $\mathcal{B}$ being chosen independently from distributions whose first moment is zero and the second moment is of order 1.
Another quantity of interest is the training error which characterizes the performance of the network with a weight vector $\mathbf{J}$ on examples from the training set, and is usually taken to be of the form

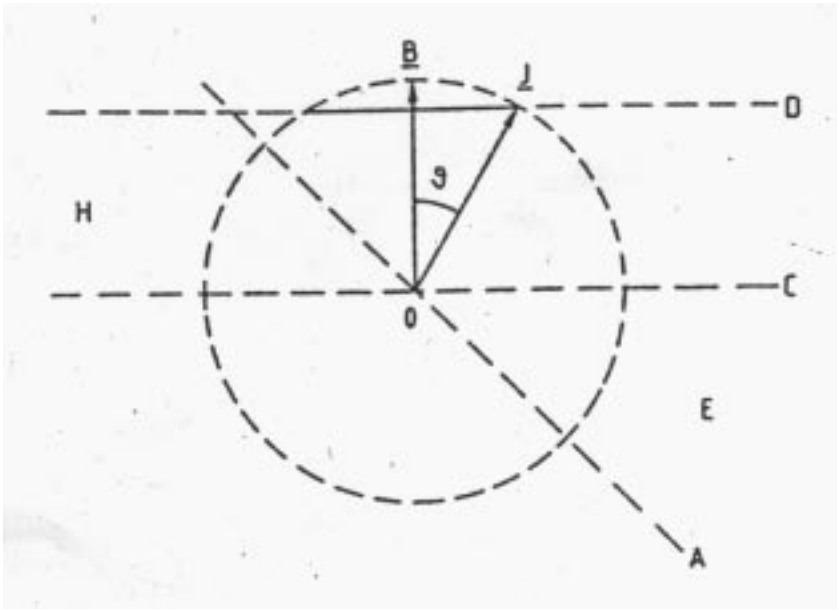

Figure 5.1: .

$$E(\mathbf{w}, \{\xi^\mu\}) = \sum_{\mu=1}^{\mathcal{P}} \epsilon(\mathbf{w}, \xi^\mu). \tag{5.8}$$

where the specific choice of the error function $\epsilon(\mathbf{w}, \xi^\mu)$ is up to the trainer. $\{\xi^\mu\}$ in Eq.(5.8) refer to the ensemble of training examples (from now on we are going to drop the dependence of the training energy on the training set $\{\xi^\mu\}$.
In order to describe the error function for a perceptron type architecture it is useful to define the variable

$$\Delta^\mu(\mathbf{w}, \zeta^\mu, \zeta_0^\mu) = \mathrm{sgn}(\gamma_0^\mu)\gamma^\mu \tag{5.9}$$

Analytical calculations have been done for the following error functions, which have been well known for many years. The first one is the Gardner and Derrida error measure,

$$\epsilon^{GD}(\Delta^\mu, \kappa) = \Theta(\kappa - \Delta^\mu). \tag{5.10}$$

Where $\Theta(x) = 0$ or 1, if $x \leq 0$ or $x \geq 0$, this error function does not give rise to a gradient-descent learning algorithm, as it is piecewise constant. Obviously the choice for $\kappa = 0$ correspond to counting the number of missclassified patterns. Moreover the GD error function can be minimized using simulated annealing.
The second function normally used in the letriture is the perceptron function which has also been studied by Meir and Fontanari [48]

$$\epsilon^P(\Delta^\mu, \kappa) = (\kappa - \Delta^\mu)\Theta(\kappa - \Delta^\mu) \tag{5.11}$$

while the third interesting one is the relaxation function

$$\epsilon(\Delta^\mu, \kappa) = (\kappa - \Delta^\mu)^2 \Theta(\kappa - \Delta^\mu) \tag{5.12}$$

The learning algorithm corresponding to the choice Eq.(5.11) is just the celebrated perceptron learning algorithm (performed in batch mode), while that corresponding to Eq.(5.12) is the relaxation algorithm, introduced by Agmon [40]as a method to solve a set of linear inequalities, and later generalized by May [41] to include the parameter κ. In fact these authors were concerned with the on-line version of the learning process, where the weight adaptation is made after each pattern presentation. The relaxation Algorithm has also been studied in the context of neural networks by Anlauf and Biehl [42]. Meir and Fontanari [48] found that as long as the number of training examples is not large, so that the training error is zero, all three functions produce identical results. The difference between them becomes apparent when the training error is nonzero. The error function of our model is defined as follow:

$$\epsilon(\Delta^\mu, \kappa) = |\kappa - \Delta^\mu| \Theta(\kappa - \Delta^\mu) \tag{5.13}$$

In contrast to Gardner's error measure, the error measure in Eq.(5.13) and Eq.(5.11)does not count only the misclassified patterns, but every misclassified training pattern contribute to the training error with a corresponding weight, in our case this weight is $|\kappa - \Delta|$.
Since the only information available to us is the training error we would like to consider the space of all networks of a given training error $E(\mathbf{w})$. As we have seen in chapter 3, the Langevin dynamics, after it has been applied for a long time, defines a probability distribution of networks over the space of solution. This distribution is given by the standard Gibbs distribution over the space of networks with temperature $T = \frac{1}{\beta}$.

$$P(\mathbf{w}) = Z^{-1} \exp\Big\{-\beta E(\mathbf{w})\Big\}. \tag{5.14}$$

where the partition function Z is given by

$$Z = \sum_{\{w_j\}} \exp\Big\{-\beta \sum_\mu \epsilon(\Delta^\mu, \kappa)\Big\}. \tag{5.15}$$

Note that at zero temperature, $\beta \to \infty$, only those networks with minimal training error contribute to the partition function. In contrary to the case of continuous pupil and continuous teacher we may have here many pupil perceptrons with minimal training error, because we restricted the weights to be discrete. Also, because we are dealing with a pupil percepton with discrete weights, the integral over the weight space in the definition of the partition function has been replaced by a sum over all allowed weight configurations of the pupil.
Now after defining the partition function Eq.(5.15) of our system we can set out to evaluate the average free-energy which is the key to calculate the thermodynamical quantities interesting for us, such as the training error, the generalization error,and the

entropy.
As explained previously, the training patterns are randomly drawn from the set $\{\pm 1\}$ and hence introduce a disorder into the system. As we are not interested in the calculation of the free energy for a specific set of training examples, statements must be derived which are typically valid on the average if a set of patterns is selected arbitrarily out of a huge ensemble of patterns. So the average with respect to the probability distribution of the random training examples must be calculated. The mathematical problem we are faced with at this stage is the calculation of the average free-energy density in the thermodynamic limit $N \to \infty$. Since the free-energy is related to the logarithm of the partition function, averaging over the training example is a very demanding task, complicated by the fact that the partition function Z stands in the argument of a logarithm. The replica trick, which has been discussed in chapter 3, will be used to calculate the average free energy. In what follows we will see on our specific model in the course of evaluation of the free energy, how one can gain information about the average generalization- training error, entropy and many other thermodynamical quantities.

5.3 The replica approach

As mentioned before the real problem is the evaluation of the quenched free energy, which requires the calculation of the logarithm of the partition function averaged over the distribution of the training examples.
To perform this calculation we use the replica trick discussed in chapter 3
Using the replica trick the problem of calculating the free-energy (logarithm of the partition function) is converted to averaging that of a power of the partition function as

$$\langle\langle \ln Z \rangle\rangle = \lim_{n\to 0} \frac{\ln\langle\langle Z^n \rangle\rangle}{n}. \tag{5.16}$$

By introducing the double brackets$\langle\langle ... \rangle\rangle$ we have indicated that we are interested in the average over an ensemble of training patterns $(\xi^1, \zeta_0^1), (\xi^2, \zeta_0^2), ..., (\xi^{\mathcal{P}}, \zeta_0^{\mathcal{P}})$. We have to average n copies of the partition function over the same set of patterns.
The expression for Z^n in Eq.(5.16) of course can be computed only for positive integer values of n. In order to employ Eq.(5.16) we have to evaluate for these values and then perform an analytic continuation to arbitrary real values. Finally the limit $n \to 0$ can be performed. In order to proceed we follow a common strategy in the physics literature, by first calculating the thermodynamical limit ($N \to \infty$) in Eq.(5.16) and then taking the limit $n \to 0$. In principle one should take the limit $n \to 0$ first and afterwards $N \to \infty$. Practically, however, it is often more convenient to reverse the order of the limits. While this interchange of limits has been shown to be valid for the spin-glass problem, we do not know of any arguments for its validity in general. But

it turns out that most of the times this should not be a source of troubles .
Now we can explicitly work out the evaluation of the average of the replicated partition function $\langle\langle Z^n\rangle\rangle$ over the patterns. The replicated partition function has the following form

$$\langle\langle Z^n\rangle\rangle = \langle\langle \sum_{\{w_i^a\}} exp\{-\beta \sum_{\mu,a}^{\mathcal{P}n} \epsilon(\Delta_a^\mu - \kappa)\}\rangle\rangle_\xi. \tag{5.17}$$

Where a is the replica index $a = 0, ..., n$, and energy function $\epsilon(x)$ is defined in Eq.(5.13). κ is called the stability parameter which controls the basin of attraction for the fixed points of the system. Thus κ biases the trainingalgorithm to favor networks less sensitive to effects of noise in the training data. Meir and Fontanari found that it is advantageous to set $\kappa \geq 0$ for small training set size, but keeping κ positive gives a poor asymptotic behavior. It is therefore interesting to chose an optimal $\kappa^{opt}(\alpha)$ in such a way that for every α the generalization error is minimized.
Since summation is a linear operation, it commutes with the averaging procedure, which acts directly on the summand leading to

$$\langle\langle Z^n\rangle\rangle = \sum_{\{w_i^a\}} \langle\langle \prod_{\mu a}^{\mathcal{P}n} exp\Big\{ -\beta\epsilon(\mathrm{sgn}(\gamma_0^\mu)\gamma_a^\mu)\Big\}\rangle\rangle_\xi. \tag{5.18}$$

The average over the distribution of the examples is equivalent to an average over the γ_a^μ. Because we are dealing with the case of independent uniformly distributed random binary input-output training examples, we can deduce by use of the central-limit theorem in the limit of an infinite system ($N \to \infty$) that the fields γ_a^μ are Gaussian variables. So the average over the patterns ξ_i^μ turns into a Gaussian average over γ_a^μ with a zero mean and correlation defined by the correlation matrix C. The probability distribution of γ_a^μ has the following form

$$P(\gamma_a^\mu) = \frac{1}{\sqrt{2\pi^n}}(detC)^{-\frac{1}{2}} exp\Big\{ -\frac{1}{2}\sum_{ab} \gamma_a^\mu {C_{ab}}^{-1} \gamma_b^\mu \Big\} \tag{5.19}$$

$$P(\gamma_a^\mu) = \int \prod_a (\frac{dx^a}{2\pi}) \exp\{-\frac{1}{2}(x, Cx) + i(\gamma, x)\} \tag{5.20}$$

where $(x, Cx) = \sum_{ab} x^a C_{ab} x^b, (\gamma, x) = \sum_a \gamma_a x^a$. The correlation matrix C is defined by the relations $\langle\gamma_a^\mu\rangle = 0$, $\langle\gamma_a^\mu \gamma_b^\nu\rangle = \delta_{\mu\nu} C_{ab}$. It is given as follows

$$\begin{aligned} C_{00} &= \frac{1}{N}\sum_j \mathcal{B}_j^2 = 1 \\ C_{0a} &= \frac{1}{N}\sum_j \mathcal{B}_j w_j^a = R_a \\ C_{ab} &= \frac{1}{N}\sum_j w_j^a w_j^b = q_{ab} \end{aligned} \tag{5.21}$$

Clearly R_a represents the overlap between the replicated weight vector of the pupil and that of the teacher. Because the weight vector of the teacher is a normalized vector, $\frac{R_a}{\sqrt{q_{aa}}}$ gives $cos(\Theta)$, the angle between teacher's and pupil's weight vectors of the replica with index a. The generalization error of the perceptron depends only on its overlap with the teacher, and it is related to R_a as follow

$$\epsilon_{g_a} = \frac{1}{\pi} \arccos(\frac{R_a}{\sqrt{q_{aa}}}). \tag{5.22}$$

where the index a refers to one of the n replica of the system. The generalization error goes to zero as the angle between the student and the teachers weight vectors vanishes. Perfect learning corresponds to an $\frac{R_a}{\sqrt{q_{aa}}} = 1$ which is equivalent to $\Theta = 0$. In the replica symmetric approximation, where the parameters q_{ab}, q_{aa}, R_a are assumed to be independent on the replica index, R has the simple meaning of being the expected value of the overlap with the teacher. We introduce also the set of $\frac{1}{2}n(n-1)$ parameters q_{ab} which are the mutual overlap between the weight vectors realized in the replica copies a and b (corresponding to the Edward-Anderson parameters in the spin-glass case). $\frac{q_{ab}}{\sqrt{q_{aa}}\sqrt{q_{bb}}}$ is $cos(\Phi_{ab})$ where Φ_{ab} the angle between the weight vectors $\mathbf{w}^a$ and $\mathbf{w}^b$ for different replicas a, b. q_{aa} is proportional to the length of the weight vector of replica a. The coupling between different replica comes from the fact that the same patterns put constraint on the local fields γ_a^μ. The latter are essentially Gaussian variables. This is the reason why only the second order correlation between the replicas enter the calculation and no coupling between three or higher orders in the $\mathbf{w}'s$ appear. For the same reason, only the first two moments of the probability distribution of the input pattern matter.
Due to the fact that all the patterns are chosen independently everything decouples in the patterns, and as we average over these patterns, we get in Eq.(5.18) $\mathcal{P}$ times the same term. We can replace in Eq.(5.18) $\langle\langle\prod_{\mu=1}^{\mathcal{P}} ...\rangle\rangle$ with $\prod_{\mu=1}^{\mathcal{P}}\langle\langle ...\rangle\rangle = \langle\langle ...\rangle\rangle^{\mathcal{P}}$ and drop the index μ.
The correct thermodynamic limit requires that the energy function and the entropy are extensive variables, i.e., proportional to N. From the correct scaling of the entropy and energy with N it turns out that the number of examples should scale as

$$\mathcal{P} = \alpha N \tag{5.23}$$

where the proportionality constant α remains finite as N grows. From now on we work with α instead of $\mathcal{P}$. Using these facts we can write the average in equation Eq.(5.18) in the following form

$$\langle\langle\prod_{\mu a} exp(-\beta\epsilon(\mathrm{sgn}(\gamma_0^\mu)\gamma_a^\mu))\rangle\rangle_\xi = I^{\alpha.N}. \tag{5.24}$$

With I

$$I = \langle\langle\prod_{a} exp(-\beta\epsilon(\mathrm{sgn}(\gamma_0)\gamma_a))\rangle\rangle_{\gamma_\mathbf{a}} \tag{5.25}$$

Using the distribution for γ_a defined in Eq.(5.19), and (5.20) we can write Eq.(5.25) as follows

$$I = \int \prod_{a=0}^{n} \frac{dx_a}{\sqrt{2\pi}} \int \prod_{a=0}^{n} \frac{d\gamma_a}{\sqrt{2\pi}} \exp\Big\{ -\frac{1}{2}(x, Cx) + i(\gamma, x) - \beta \sum_a \epsilon(\mathrm{sgn}(\gamma_0)\gamma_a) \Big\}. \quad (5.26)$$

$$I = \int_0^\infty \tfrac{d\gamma_0}{\sqrt{2\pi}} \int \prod_{a=1}^n \tfrac{d\gamma_a}{\sqrt{2\pi}} \int \prod_{a=0}^n \tfrac{dx_a}{\sqrt{2\pi}} exp\Big\{.\Big\} + \int_{-\infty}^0 \tfrac{d\gamma_0}{\sqrt{2\pi}} \int \prod_{a=1}^n \tfrac{d\gamma_a}{\sqrt{2\pi}} \int \prod_{a=0}^n \tfrac{dx_a}{\sqrt{2\pi}} exp\Big\{.\Big\}$$

$$exp\Big\{.\Big\} = exp\Big\{ -\frac{1}{2}x_0^2 - \sum_{a=1}^n R_a x_0 x_a - \frac{1}{2}\sum_{a,b} q_{ab} x_a x_b + i\gamma_0 x_0 + i\sum_a \gamma_a x_a - \beta \sum_a \epsilon(\mathrm{sgn}(\gamma_0)\gamma_a) \Big\} \quad (5.27)$$

Now we perform the integration over x_0 in Eq.(5.26); the linear contribution is $x_0(i\gamma_0 - \sum_a R_a x_a)$. After performing the integration over x_0 Eq.(5.26) takes the following form

$$\begin{aligned} I &= \int_{-\infty}^{\infty} \frac{\mathcal{D}\gamma_0}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \prod_{a=1} \frac{dx_a}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \prod_{a=1} \frac{d\gamma_a}{\sqrt{2\pi}} \exp\Big\{ -\frac{1}{2}\sum_{a,b}(q_{ab} - R_a R_b) x_a x_b \\ &\quad - i|\gamma_0| \sum_a x_a R_a + i\sum_a \gamma_a x_a - \beta \sum_a |\kappa - \gamma_a| \Theta(|\kappa - \gamma_a|) \Big\} \quad (5.28) \\ &= \int_{-\infty}^{\infty} \frac{\mathcal{D}\gamma_0}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \prod_{a=1} \frac{dx_a}{\sqrt{2\pi}} \Big[\int_{-\infty}^{\kappa} \prod_{a=1} \frac{d\gamma_a}{\sqrt{2\pi}} \exp\Big\{ -\frac{1}{2}\sum_{a,b}(q_{ab} - R_a R_b) x_a x_b - \\ &\quad i|\gamma_0| \sum_a x_a R_a + i\sum_a \gamma_a x_a - \beta \sum_a |\kappa - \gamma_a| \Big\} + \\ &\quad \int_{\kappa}^{-\infty} \prod_{a=1} \frac{d\gamma_a}{\sqrt{2\pi}} \exp\Big\{ -\frac{1}{2}\sum_{a,b}(q_{ab} - R_a R_b) x_a x_b - \\ &\quad i|\gamma_0| \sum_a x_a R_a + i\sum_a \gamma_a x_a \Big\} \Big] \end{aligned}$$

To impose the definition of the $\frac{1}{2}n(n-1)$ combinations $a \le b$ q_{ab}, the nq_{aa} and R_a in Eq.(5.29) we use following the representations of unity

$$\begin{aligned} 1 &= \int_{-\infty}^{\infty} dq_{ab} \delta(q_{ab} - \frac{1}{N}\sum_j w_j^a w_j^b) \\ 1 &= \int_{-\infty}^{\infty} dq_{aa} \delta(q_{aa} - \frac{1}{N}\sum_j w_j^{a\,2}) \quad (5.29) \\ 1 &= \int_{-\infty}^{\infty} dR_a \delta(R_a - \frac{1}{N}\sum_j \mathcal{B}_j w_j) \end{aligned}$$

(5.30)

To proceed further we introduce the conjugate variables $\hat{q}_{ab}, \hat{q}_{aa}$ and $\hat{R}_a$ and use the Fourier representation of the $\delta - function$

$$\delta(q_{ab} - \frac{1}{N}\sum_j w_j^a w_j^b) = \int_{-\infty}^{\infty} d\hat{q}_{ab} \frac{1}{2\pi i/N} \exp(N\hat{q}_{ab}(q_{ab} - \frac{1}{N}\sum_j w_j^a w_j^b))$$

$$\delta(q_{aa} - \frac{1}{N}\sum_j w_j^{a2}) = \int_{-\infty}^{\infty} d\hat{q}_{aa} \frac{1}{2\pi i/N} \hat{q}_{aa} \exp(N\hat{q}_{aa}(q_{aa} - \frac{1}{N}\sum_j w_j^{a2})) \tag{5.31}$$
$$\delta(R_a - \frac{1}{N}\sum_j \mathcal{B}_j w_j a) = \int_{-\infty}^{\infty} d\hat{R}_a \frac{1}{2\pi i/N} \exp(N\hat{R}_a(R_a - \frac{1}{N}\sum_j \mathcal{B}_j w_j^a))$$

This leads to

$$\begin{aligned}\langle\langle Z^n \rangle\rangle \;=\; & \int \prod_a \frac{dq_{aa} d\hat{q}_{aa}}{2\pi i/N} \int \prod_{a<b} \frac{dq_{ab} d\hat{q}_{ab}}{2\pi i/N} \int \prod_a \frac{dR_a d\hat{R}_a}{2\pi i/N} \\ & \exp\Big\{ -N\Big[\sum_a q_{aa}\hat{q}_{aa} - \sum_{a<b} q_{ab}\hat{q}_{ab} + \sum_a R_a \hat{R}_a\Big]\Big\} \\ & \times \sum_{w_j^a} \prod_j \exp\Big\{ \sum_a \hat{q}_{aa}(w_j^a)^2 + \sum_{a<b} \hat{q}_{ab} w_j^a w_j^b + \sum_a \hat{R}_a w_j^a \mathcal{B}_j \Big\} \\ & \times \exp\Big\{\alpha N \ln I\Big\}\end{aligned} \tag{5.32}$$

All the terms containing w_j^a in Eq.(5.32) factorize in the index j. As we evaluate the sum over the w_j^a's we get N times the same term. So we can replace $\prod_j \sum_{w_j^a}$ with $(\prod_a \sum_{w^a})^N$, this leads to the G_2 given in Eq.(5.36). Equation Eq.(5.32) can be written in a compact form as follows

$$\langle\langle Z^n \rangle\rangle = \int \prod_a \frac{dq_{aa} d\hat{q}_{aa}}{2\pi i/N} \int \prod_{a<b} \frac{dq_{ab} d\hat{q}_{ab}}{2\pi i/N} \int \prod_a \frac{dR_a d\hat{R}_a}{2\pi i/N} \exp\{N(G)\}\,. \tag{5.33}$$

Since our results are relevant only in the thermodynamical limit, we must consider our system in the limit $N \to \infty$. In this limit we can perform the saddle point integration to evaluate the integral over the order parameters in Eq.(5.32). The integral can be approximated with the value of the integrand in its maximum. So we have to look for a set of real order parameters $(q_{aa}, \hat{q}_{aa}, q_{ab}, \hat{q}_{ab}, R_a, \hat{R}_a)$ that maximizes G and then we get a set of equation by demanding the first derivative of G with respect to the order parameters to be zero.
G is defined as follows

$$G = G_1 + G_2 + G_3. \tag{5.34}$$

The functions $G_1, G_2,$ andG_3 are given by the following equations

$$G_1 = -\sum_a \hat{q}_{aa} q_{aa} - \sum_{a<b} \hat{q}_{ab} q_{ab} - \sum_a R_a \hat{R}_a. \tag{5.35}$$

$$G_2 = \left\langle \ln \sum_{w^a} \exp\left\{ \sum_a \hat{q}_{aa}(w^a)^2 + \sum_{a<b} \hat{q}_{ab} w^a w^b + \sum_a \hat{R}_a w^a \mathcal{B} \right\} \right\rangle_{\mathcal{B}}. \tag{5.36}$$

$$G_3 = \alpha \ln I. \tag{5.37}$$

The function G_2 is referred to as the weight term because it depends explicitly on the constraints on the weight vector of the pupil and on the distribution of the teachers

weights vector, but it depends weakly on the constraints on the local field through the saddle point values of the conjugate variables. As we have mentioned in the introduction to this chapter, in order to get typical results, we have to average not only over the distribution of the random training examples but also over the distribution of the teacher's weight vector which is assumed to be Gaussian. This average is denoted by $\langle ... \rangle_{\mathcal{B}}$ in Eq.(5.36).
G_3 contains only order parameters with a clear physical interpretation. The dependence on the temperature, the chosen error function, the number of training examples and constraint on the local fields, is contained in G_3 which is referred to as the field function. G_3 depends also implicitly through the saddle point values of the physical variables on the constraints on the weights.

5.4 The replica symmetric ansatz

To proceed further, it is required to obtain the saddle-point equations for general n and then take the limit $n \to 0$. This procedure is in general very complex. In order to make progress, however, it has been customary to make an ansatz concerning the solution of these equations. In particular the most commonly used *replica-symmetric assumption* is just that all the variables take on values invariant under permutation of the replica,i.e.,

$$\begin{aligned} q_{aa} &= q_0, & q_{ab} &= q \\ \hat{q}_{aa} &= E, & \hat{q}_{ab} &= F \\ R_a &= R, & \hat{R}_a &= \hat{R} \end{aligned} \tag{5.38}$$

This idea comes from the fact that all replicas are a priori totally equivalent, because they are copies of the same system. If the order parameters are self-averaging they should be the same for all replica indices. This holds for order parameters with one replica index. However if the phase space is not connected and convex the replica symmetric approximation is not correct for order parameters with two replica indices. In that case we have to consider different stages of replica symmetry breaking **RSB**. How this can be understood physically will be discussed at the end of this chapter. The saddle-point is defined by the following equations

$$\begin{aligned} \frac{\partial G}{\partial q_0} = 0, \quad \frac{\partial G}{\partial R} = 0, \quad \frac{\partial G}{\partial \hat{R}} = 0 \\ \frac{\partial G}{\partial E} = 0, \quad \frac{\partial G}{\partial F} = 0, \quad \frac{\partial G}{\partial q_1} = 0 \end{aligned} \tag{5.39}$$

In order to evaluate the derivatives we need explicit expressions for the functions G_1, G_2 and G_3 in the limit $n \to 0$.

In this limit and using the replica symmetric ansatz we can rewrite G_1 in the following form,

$$G_1 = -n\left\{Eq_0 - \frac{1}{2}Fq_1 + \hat{R}R\right\}. \tag{5.40}$$

To evaluate G_2, the sum $\sum_{a<b} w^a w^b$ in Eq.(5.36) can be rewritten as $\sum_{a<b} w^a w^b = \frac{1}{2}((\sum_a w^a)^2 - \sum_a w^{a2})$ this leads to

$$G_2 = \left\langle \ln \sum_{w^a} \exp\left\{(E - \frac{1}{2}F)\sum_a (w^a)^2 + \frac{1}{2}F(\sum_a w^a)^2 + B\hat{R}\sum_a w^a\right\}\right\rangle_{\mathcal{B}} \tag{5.41}$$

Now we can use the following Gaussian integration formula to linearize $(\sum_a w^a)^2$.

$$\exp(ax^2) = \int \frac{dz}{\sqrt{2\pi}} \exp(-\frac{z^2}{2} \pm \sqrt{2a}xz). \tag{5.42}$$

With $(x = \sum_a w^a)$ we can bring Eq.(5.36) in the following form

$$G_2 = \left\langle \ln \int \mathcal{D}z \sum_{w^a} \exp\left\{(E - \frac{1}{2}F)\sum_a (w^a)^2 + (\sqrt{F}z + \mathcal{B}\hat{R})\sum_a w^a\right\}\right\rangle_{\mathcal{B}} \tag{5.43}$$

$$\begin{aligned} &= \left\langle \ln \int \mathcal{D}z \left[\sum_w \exp\left\{(E - \frac{1}{2}F)(w)^2 + (\sqrt{F}z + \mathcal{B}\hat{R})w\right\}\right]^n \right\rangle_{\mathcal{B}} \\ &\overset{n\to 0}{\approx} n\left\langle \ln \int \mathcal{D}z \sum_w \exp\left\{(E - \frac{1}{2}F)(w)^2 + (\sqrt{F}z + \mathcal{B}\hat{R})w\right\}\right\rangle_{\mathcal{B}}. \end{aligned} \tag{5.44}$$

For Gaussian distributed $\mathcal{B}$ and z, $(\sqrt{F}z + \mathcal{B}\hat{R})$ is also a Gaussian distributed variable with $\sigma^2 = F + (\hat{R})^2$ Using this fact we can write Eq.(5.43) as follows

$$G_2 = n\int \mathcal{D}u \ln \sum_w \exp\left\{(E - \frac{1}{22}F)w^2 + \sqrt{(F + \hat{R}^2)}uw\right\}. \tag{5.45}$$

To fix the values of q_1, q_0 we have to evaluate the function G_3 defined in Eq.(5.37) which is somewhat laborious. G_3 contains an n-dimensional Gaussian integration. Substituting the replica symmetric order parameters in Eq.(5.37) leads to

$$\begin{aligned} G_3 \;=\; & \alpha \ln \int \mathcal{D}\gamma_0 \int \prod_a \left[\frac{dx_a}{\sqrt{2\pi}}\left[\int_{-\infty}^{\kappa} \frac{d\gamma_a}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}(q_0 - q_1)\sum_a x_a^2 - \right.\right.\right. \\ & \left.\frac{1}{2}(q_1 - R^2)(\sum_a x_a)^2 - i\sum_a x_a(|\gamma_0|R - \gamma_a) - \beta|\kappa - \gamma_a|\right\} + \\ & \int_{\kappa}^{\infty} \frac{d\gamma_a}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}(q_0 - q_1)\sum_a x_a^2 - \frac{1}{2}(q_1 - R^2)(\sum_a x_a)^2 - \right. \\ & \left.\left.\left. i\sum_a x_a(|(\gamma_0)|R - \gamma_a)\right\}\right]\right] \end{aligned} \tag{5.46}$$

Here we used the identity $\sum_{a<b} x_a x_b = \frac{1}{2}((\sum_a x_a)^2 - \sum_a x_a^2)$ as in the case of G_2. The integration over x_a is also unpleasant because, owing to the squared sum term in the exponent, the integrals do not decouple. However, with the introduction of an auxiliary variable z and the Gaussian linearization according to Eq.(5.42), we can get rid of the offending term and factorize the x_a integrals. This leads to the following form for G_3

$$\begin{aligned} G_3 &= \alpha \ln \int \mathcal{D}\gamma_0 \int \mathcal{D}z \left[\int \frac{dx}{\sqrt{2\pi}} \left[\int_{-\infty}^{\kappa} \frac{d\gamma}{\sqrt{2\pi}} \right.\right. \\ &\exp\left\{ -\frac{1}{2}(q_0 - q_1)x^2 + i(\sqrt{q_1 - R^2}z + \gamma - |\gamma_0|R)x - \beta|(\kappa - \gamma)| \right\} + \\ &\left.\left. \int_{\kappa}^{\infty} \frac{d\gamma}{\sqrt{2\pi}} \exp\left\{ -\frac{1}{2}(q_0 - q_1)x^2 + i(\sqrt{(q_1 - R^2)}z + \gamma - abs(\gamma_0 R))x \right\} \right] \right]^n \end{aligned} \tag{5.47}$$

In the limit $n \to 0$ G_3 can be further simplified using

$$\begin{aligned} \ln \int_{-\infty}^{\infty} \mathcal{D}z \Phi^n(z) &= \ln \int_{-\infty}^{\infty} \mathcal{D}z e^{n \ln \Phi} \approx \ln \int_{-\infty}^{\infty} \mathcal{D}z (1 + n \ln \Phi) \\ &= \ln(1 + n \int_{-\infty}^{\infty} \mathcal{D}z \ln \Phi) \approx n \int_{-\infty}^{\infty} \mathcal{D}z \ln \Phi) \end{aligned} \tag{5.48}$$

Now we can perform the integration over x which is a gaussian one. The integral over the gaussian measure is $\int_{-\infty}^{\infty} \mathcal{D}z = 1$ so G_3 takes the form

$$\begin{aligned} G_3 &= n\alpha \int \mathcal{D}u \int \mathcal{D}z \ln \left[\int_{-\infty}^{\kappa} \frac{d\gamma}{\sqrt{2\pi(q_0 - q_1)}} \exp\left\{ -\frac{1}{2} \left(\frac{\gamma - |u|R + \sqrt{(q_1 - R^2)}}{\sqrt{(q_0 - q_1)}} \right)^2 - \right. \right. \\ &\left. \left. \beta|\kappa - \gamma| \right\} + \int_{\kappa}^{\infty} \frac{d\gamma}{\sqrt{2\pi(q_0 - q_1)}} \exp\left\{ -\frac{1}{2} \left(\frac{\gamma - |u|R + \sqrt{(q_1 - R^2)}}{\sqrt{(q_0 - q_1)}} \right)^2 \right\} \right] \end{aligned} \tag{5.49}$$

In a compact form we can write Eq.(5.49) as follows

$$G_3 = n\alpha \int \mathcal{D}u \int \mathcal{D}z \ln \{I_1 + I_2\} \tag{5.50}$$

where I_1, I_2 are given by the following equations

$$I_1 = \int_{\kappa}^{\infty} \frac{d\gamma}{\sqrt{2\pi(q_0 - q_1)}} \exp\left\{ -\frac{1}{2} \left(\frac{\gamma - |u|R + \sqrt{(q_1 - R^2)}z}{\sqrt{(q_0 - q_1)}} \right)^2 \right\} \tag{5.51}$$

$$I_2 = \int_{-\infty}^{\kappa} \frac{d\gamma}{\sqrt{2\pi(q_0 - q_1)}} \exp\left\{ -\frac{1}{2} \left(\frac{\gamma - |u|R + \sqrt{(q_1 - R^2)}}{\sqrt{(q_0 - q_1)}} \right)^2 - \beta|\kappa - \gamma| \right\} \tag{5.52}$$

The remaining integration in Eq.(5.52) and Eq.(5.51) over γ is also an unpleasant one since the boundary does not extend to infinity. Using the following definition $H(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty \exp(-\frac{1}{2}\frac{t^2}{2})$, which is related to the complementary error function $\mathrm{erfc}(x) = 1-\mathrm{erf}(x)$ by $H(x) = \frac{1}{2}\mathrm{erfc}(\frac{x}{\sqrt{2}})$, we can write Eq.(5.52) Eq.(5.51) in the form

$$I_1 = H\left(\frac{\kappa - |u|R + \sqrt{(q_1 - R^2)}}{\sqrt{(q_0 - q_1)}}\right) \tag{5.53}$$

$$I_2 = \exp\left\{\frac{1}{2}\beta^2(q_0 - q_1) - \beta\sqrt{(q_0 - q_1)}.t^*\right\}(1 - H(t^* - \beta\sqrt{(q_0 - q_1)})) \tag{5.54}$$

t^* in Eq.(5.54 is given by

$$t^* = \frac{\kappa - |u|R + \sqrt{(q_1 - R^2)}}{\sqrt{(q_0 - q_1)}} \tag{5.55}$$

Demanding the first derivative of G with respect to the order parameters and their conjugate parameters to be zero we can obtain the saddle point equations:

$$\frac{\partial G}{\partial E} = 0, \Longrightarrow q_0 = \langle [w^2] \rangle \tag{5.56}$$

$$\frac{\partial G}{\partial F} = 0, \Longrightarrow -\frac{1}{2}q_1 = -\frac{1}{2}\langle [w^2] \rangle + \frac{1}{2}\frac{1}{\sqrt{(F + \hat{R}^2)}}\langle [w] \rangle = 2\langle [w]^2 \rangle \tag{5.57}$$

$$\frac{\partial G}{\partial \hat{R}} = 0, \Longrightarrow R = \frac{\hat{R}}{\sqrt{(F + \hat{R}^2)}}\langle u[w] \rangle = \hat{R}(q_0 - q_1) \tag{5.58}$$

In Eq.(5.56), (5.57), and (5.58) $[x]$ is an abbreviation for

$$[x] = \frac{\sum_w x \exp\left\{(E - \frac{1}{2}F)w^2 + \sqrt{(F + \hat{R}^2)}uw\right\}}{\sum_w \exp\left\{(E - \frac{1}{2}F)w^2 + \sqrt{(F + \hat{R}^2)}uw\right\}}, \tag{5.59}$$

and $\langle\rangle$ stand for

$$\langle x \rangle = \int \mathcal{D}u x \tag{5.60}$$

The saddle point equation for F, E and R are given as follows

$$\begin{aligned}
\frac{\partial G}{\partial q_1} &= 0 \\
F &= -2\alpha \int \mathcal{D}u \int \mathcal{D}z[...]^{-1} \left\{ \exp\left\{ \frac{1}{2}\beta^2(q_0-q_1) - \beta\sqrt{(q_0-q_1)}t^* \right\} \right. \\
&\quad (1 - H(t^* - \beta\sqrt{(q_0-q_1)}))(-\frac{1}{2}\beta^2 - \frac{1}{2}\beta\frac{1}{\sqrt{(q_1-R^2)}}) + \\
&\quad \exp\left\{ \frac{1}{2}\beta^2(q_0-q_1) - \beta\sqrt{(q_0-q_1)}t^* \right\} \frac{\exp -\frac{1}{2}(t^* - \beta\sqrt{(q_0-q_1)})^2}{\sqrt{2\pi}} \\
&\quad \times \left(\frac{t^*}{2(q_0-q_1)} + \frac{1}{2}\frac{z}{\sqrt{(q_0-q1)}\sqrt{(q_1-R^2)}} + \frac{1}{2}\frac{\beta}{\sqrt{(q_0-q1)}} \right) \\
&\quad \left. -\frac{1}{2}\frac{1}{\sqrt{2\pi}} \exp -\frac{1}{2}t^{*2} \left(\frac{t^*}{q_0-q_1} + \frac{z}{\sqrt{(q_0-q1)}\sqrt{(q_1-R^2)}} \right) \right\}
\end{aligned} \tag{5.61}$$

$$\begin{aligned}
\frac{\partial G}{\partial q_0} &= 0 \\
E &= \alpha \int \mathcal{D}u \int \mathcal{D}z[...]^{-1} \left\{ \exp\left\{ \frac{1}{2}\beta^2(q_0-q_1) - \beta\sqrt{(q_0-q_1)}t^* \right\} \right. \\
&\quad \times (1 - H(t^* - \beta\sqrt{(q_0-q_1)}))(\frac{1}{2}\beta^2) + \exp\left\{ \frac{1}{2}\beta^2(q_0-q_1) - \beta\sqrt{(q_0-q_1)}t^* \right\} \\
&\quad \frac{\exp -\{\frac{1}{2}(t^* - \beta\sqrt{(q_0-q_1)})^2\}}{\sqrt{2\pi}}(-\frac{t^*}{2(q_0-q_1)} - \frac{1}{2}\frac{\beta}{\sqrt{(q_0-q1)}}) \\
&\quad \left. -\frac{1}{\sqrt{2\pi}} \exp\{-\frac{1}{2}t^{*2}\}(-\frac{1}{2}\frac{t^*}{(q_0-q_1)}) \right\}
\end{aligned} \tag{5.62}$$

finally

$$\begin{aligned}
\frac{\partial G}{\partial R} &= 0 \\
\hat{R} &= \alpha \int \mathcal{D}u \int \mathcal{D}z[...]^{-1}(|u| + \frac{Rz}{\sqrt{q_1-R^2}}) \left\{ \beta \exp\left\{ \frac{1}{2}\beta^2(q_0-q_1) - \beta\sqrt{(q_0-q_1)}t^* \right\} \right. \\
&\quad (1 - H(t^* - \beta\sqrt{(q_0-q_1)})) - \frac{1}{\sqrt{2\pi(q_0-q_1)}} \exp\{-\frac{1}{2}(t^* - \beta\sqrt{(q_0-q_1)})^2\} + \\
&\quad \left. \frac{1}{\sqrt{2\pi(q_0-q1)}} \exp(-\frac{1}{2}t^{*2}) \right\}.
\end{aligned} \tag{5.63}$$

In Eqs. (5.61), (5.62), and (5.63) [...] stands for the argument of the logarithm ln[...] in G_3 Eq.(5.49). Equations Eq.(5.56), (5.57), (5.58) (5.61), and Eq.(5.62) are solved

numerically. The stability of the solutions against replica symmetry breaking will be discussed in the next section. As we can see, these equations become singular for $q_1 = q_0$. So we have determined the boundary in the α, T space where this happens. There is also another boundary in the α, T space, where we could not get numerical solutions for these equations, because $q_0 - q_1$ becomes too small so that some terms in the equations diverge. By determining the value of R at the saddle point we can calculate ϵ_g as a function of the training set size α, the temperature T, and the synaptic depth L_w.

5.5 Results within the replica symmetric assumption

5.5.1 The three lines of interest

The AT line and the replica symmetry breaking
As we have mentioned above, the calculation of the integral in Eq. (5.32) in the limit $N \to \infty$ was performed by the saddle point method. Since we are performing a saddle point integration over $q_{ab}, R_a, \ldots$, we are looking for the overall maximum of G. This requires that the fluctuations of G with respect to small local variations of q_{ab}, q_{aa}, R_a and their conjugate order parameters must be negative around a stable solution (valid solution).
Often one finds that the RS solution becomes unstable at low temperature, and hence invalid. In the phase space (α, T), the line at which the instability appears is known as the Almeida Thouless line (AT-line). This instability implies the onset of replica symmetry breaking RSB. To find solutions beyond this line we have to break the replica symmetry. The onset of RSB is characterized by a change of sign in at least one of the eigenvalues of the stability matrix, the second derivative matrix of G with respect to $q_{ab}, q_{aa}, R_a, \hat{q}_{ab}, \hat{q}_a, \hat{R}_a$. The stability matrix can be represented schematically, in view of the saddle point equations, in block form as follows

$$\mathbf{H} = \begin{pmatrix} \mathbf{A} & \chi \\ \chi^T & \mathbf{B} \end{pmatrix} \tag{5.64}$$

The upper left block contains the matrix $\mathbf{A}$ of the second derivatives of G with respect to q_{ab}, q_a, R_a, the lower right block $\mathbf{B}$ contains those with respect to the conjugate parameters $\hat{q}_{ab}, \hat{q}_{aa}, \hat{R}_a$, while χ contains derivatives with respect to the order parameters as well as to their conjugates. The Matrix $\mathbf{A}$, $\mathbf{B}$, χ are given in Appendix A.
Since the symmetry properties of the second derivative matrix are similar to those in the mean-field model of the Ising spin glass, the stability analysis here follows that of de Almeida and Thouless for the spin glass model [43]. Following [44] we can determine the eigensystems of $\mathbf{A}$ and $\mathbf{B}$ by considering three classes of eigenvectors

- eigenvector invariant under permutation of all replicas
- eigenvector invariant under permutation of all replicas but 1,
- eigenvector invariant under permutation of all replicas but 2,

After investigating the eigensystem of $\mathbf{A}$ we found:
There are two eigenvectors which are invariant under permutation of all replicas, these are given by

$$\lambda_0^l \to \vec{x}_0{}^l \equiv \begin{pmatrix} \vec{\eta}_0{}^l \\ \vec{\epsilon}_0{}^l \end{pmatrix}, \qquad (l = 1, 2) \tag{5.65}$$

we denote by λ_0^l the corresponding eigenvalues. The eigenvalue equations for the λ_0^l are given in Appendix A.
The eigenvectors, which are invariant under permutation of all replicas but one, are given by the following equation

$$\lambda_1^l(r) \to \vec{x}_1{}^l(r) \equiv \begin{pmatrix} \vec{\eta}_1{}^l(r) \\ \vec{\epsilon}_1{}^l(r) \end{pmatrix}, \qquad (l = 1, 2), \qquad r = (1, ...n-1) \tag{5.66}$$

The corresponding eigenvalues $\lambda_1^l(r)$ are with degeneracy $(n-1)$ (n is the number of replica). The eigenvalue equations are also given in Appendix A. The third kind of eigenvectors are those which are invariant under permutation of all replicas but 2. They are given by

$$\lambda_2(t) \to \vec{x}_2(t) \equiv \begin{pmatrix} \vec{\eta}_2(t) \\ \vec{0} \end{pmatrix}, \qquad (t = 1, .., \frac{1}{2}n(n-3)) \tag{5.67}$$

The eigenvalue $\lambda_2(t)$ is $\frac{1}{2}n(n-3)$ times degenerate. The equation for determining $\lambda_2(t)$ is given in Appendix A.
Similarly the eigensystem of $\mathbf{B}$ is $\frac{1}{2}n(n-1)+2n$ dimensional and $\mathbf{B}$ can be diagonalized with the following eigen vectors

$$\rho_0^k \to \vec{y}_0^k \equiv \begin{pmatrix} \vec{\chi}_0^k \\ \vec{\delta}_0^k \end{pmatrix}, \qquad (k = 1, 2) \tag{5.68}$$

$$\rho_1^k(r) \to \vec{y}_1^k(r) \equiv \begin{pmatrix} \vec{\chi}_1^k(r) \\ \vec{\delta}_1^k(r) \end{pmatrix}, \qquad (k = 1, 2), \qquad r = (1, ...n-1) \tag{5.69}$$

$$\rho_2(t) \to \vec{y}_2(t) \equiv \begin{pmatrix} \vec{\chi}_2(t) \\ \vec{0} \end{pmatrix}, \qquad (t = 1, .., \frac{1}{2}n(n-3)) \tag{5.70}$$

The eigensystem of **B** has the same structure as that of **A**. The $\rho_0^k, \rho_1^k(r), \rho_2(t)$ are the corresponding eigenvalue system to the eigenvector system $\vec{y}_0^k, \vec{y}_1^k(r), \vec{y}_2(t)$. For a better understanding please read appendix A. If we look at the eigenvalue equations in Appendix A we can see that in the limit $n \to 0$ the eigenvectors $\vec{y}_1^k(r), \vec{x}_1^l(r)$ and the corresponding eigenvalue equations reduce to that of $\vec{y}_0^k(r), \vec{x}_0^l(r)$ and ρ_0^k, λ_0^l respectively. Since these describe fluctuations within RS, the only eigenvalues involved in the stability against RSB are ρ_2, λ_2. As shown in appendix A, it suffices to look at the sign of the so called replicon eigenvalue λ_R to determine the stability of the RS solution with respect to RSB-fluctuations. λ_R is given as follows

$$\lambda_R = \rho_2 \lambda_2 - 1 \tag{5.71}$$

This follows from the properties of the matrix χ, χ^T. For a stable solution λ_R must be negative. The limit of stability is given by $\lambda_R = 0$, this condition defines a function in the parameter space $T_{AT}(\alpha)$, which is the AT-line. In Figs.(5.3), (5.2) we show the AT-line for the case of 5 bit. As we can observe for a fixed α and $T > T_{AT}(\alpha)$ the RS-solutions are stable with respect to RSB, which means that the high temperature phase indeed possesses the assumed replica symmetry. However, as the temperature is decreased leaving α fixed, the RS solutions become unstable below $T_{AT}(\alpha)$. The fact that the replica symmetric solution is unstable with respect to fluctuations towards RSB indicates that the replica assumption on the independence of the order parameters and their conjugates on their replica index is too simple. The assumption comes from the idea that all replicas are a-priori equivalent, because they are copies of the same system, i.e. every replica represents a randomly picked solution out of the weight space. If the order parameters are self-averaging they should be the same for all replica indices. This seems to hold for the order parameters with one replica index . However if the weight space is not connected and convex, the replica symmetric approximation is certainly not correct for order parameters with two replica indices.
The onset of the RSB signals the occurrence of a spin glass phase. Formally, the spin glass phase is characterized by a non trivial dependence of $q_{ab}, \hat{q}_{ab}$ on the replica indices. Physically, the spin glass phase is signaled by the existence of many degenerate ground states of the energy (or free energy) which are well separated in the configuration space. The different values of $q_{ab}, \hat{q}_{ab}$ represent the distribution of overlaps among pairs of these ground states. Furthermore, these degenerate ground states occupy disconnected regions in the configuration space that are separated by energy barriers that diverge with N. Such behavior leads to anomalously slow learning dynamics. Thus RSB is undesirable with respect to the existence of a good algorithm to find the optimal weights.
These results differ from the results found for the continuous perceptron where the RS solution is stable for every α and T [29]. Therefore the RS solution is thought to be exact in this case. RSB is a known phenomenon in the case of discrete weights, here the possible weights are restricted to a discrete set of points. RSB occurs because in the version space there are disconnected clusters of these allowed points of the pupil's weight vector. Seung Sompolinsky and Tishby [10] have observed similar effects of

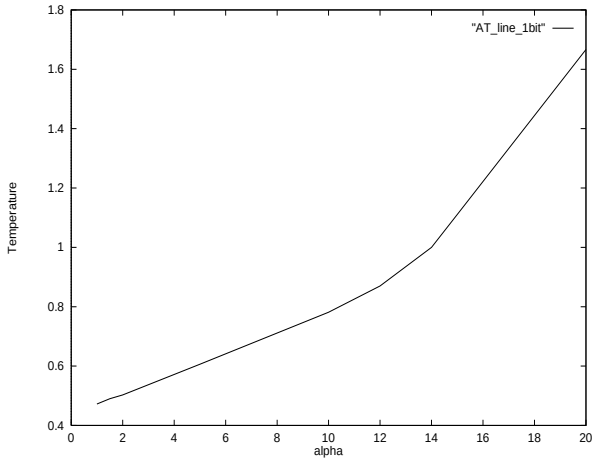

Figure 5.2: *The AT-line for the case of 1bit*

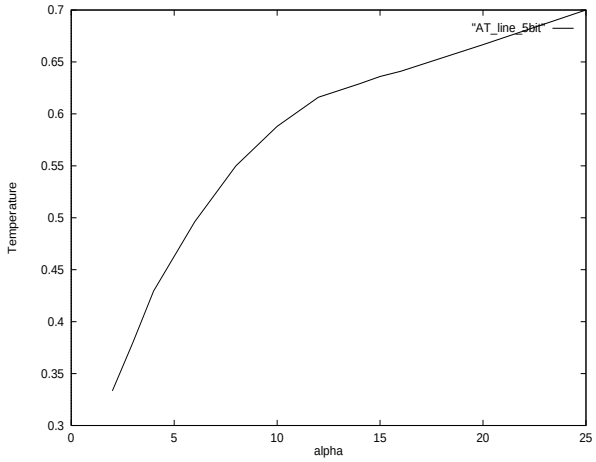

Figure 5.3: *The AT-line for the case of 5bit*

RSB by investigating the case of both the boolean and the linear binary perceptons learning a rule produced by a continuous weighted teacher.

The GD line: Solving the saddle point equations resulting from setting the derivatives of G with respect to the order parameters and their conjugates to zero, we find that for a fixed α and $T > T_{GD}(\alpha)$ the equations possess solutions with $q_1 < q_0$. For a fixed α, $q_1 \to q_0$ as $T \to T_{GD}(\alpha)$. At a temperature $T_n(\alpha) > T_{GD}(\alpha)$, $q_0 - q_1$ becomes too small that terms in the saddle point equations which contain $q_1 - q_0$ in the denominator become too large. For $T < T_n(\alpha)$, numerical solutions with available double precision arithmetic are no longer feasible.
At $T_{GD}(\alpha)$, $q_1 - q_0$ becomes zero, and the saddle point equations become singular. The fact that $q_0 - q_1 = 0$ at $T_{GD}(\alpha)$ can be understood as follows: The subspace of solutions shrinks until it contains only one solution, so that the overlap between two different

replicas becomes the same as the length of the weight vector of the optimal solution. The same happens for the spherical pupil perceptron with continuous weights when $q_1 \to 1$ (in this case the length of the weight vector of the optimal solution is equal to the length of the teacher's weight vector, which is set to be $\sqrt{N}$). In this case, we are sure that at the end of this shrinking we remain with a pupil that learnt the rule, because the subspace of solutions is continuous and convex and is embedded in a continuous and unbounded surroundings of candidate matrices for the weights. However, for a perceptron with discrete weights, when this shrinking happens we are no longer confident that at the end of this shrinking process we remain with a weight vector which has learnt the training examples and satisfies the constraints on the weights.

In Figs.(5.4),(5.5) the upper line represents the $Tn(\alpha)$. Above this line it is still possible to find numerical solution for the saddle point equations. The accuracy in determining this line depends on the algorithm used to determine the solution of the saddle point equations and on the numerical accuracy used to find the solutions. We utilized the Broydn Algorithm which is normally used to find the solutions of nonlinear equation systems. The calculations has been performed with double precision arithmetic. The line below presented in Fig.(5.4),(5.4) is the Gardner Derrida line which supply the boundary in the α, T space where our replica symmetric equations become singular, and thus determine the lower boundary of meaningful phase space.
The GD-line has been determined by solving the saddle-point equations numerically for a fixed α at high training temperature where $(q_0 - q_0 > 0$, and then repeating the procedure for lower temperature which leads to decreasing values of $(q_0 - q_1)$, this allows extrapolation for $(q_0 - q_1) \to 0$, thereby determining the temperature for a fixed α at which the equations become singular $(T_{GD}(\alpha))$.
This behavior has been already observed for systems with continuous weights by Gardner and Derrida [36] in the context of random mapping and by Györgi and Tishby [29]in the context of learning from examples with noisy input patterns. Because in both cases they dealt with continuous weights this transition has been observed when $q_1 \to 1$.
As we can see in Fig.(5.2), the GD-line is below the AT-line for the case of 1 bit synaptic depth, this means that the GD-line is not stable with respect to RSB. But in the case of 5Bit (see Fig.(5.2)) the GD-line remains below the AT-line until the point $(T = 0.68, \alpha = 24.5)$ is reached. At $(T = 0.68, \alpha = 24.5)$ the AT-line intersects the GD-line. For $T > 0.68$ and $\alpha > 24.5$ the RS solution become stable for all the points on the $T_{GD}(\alpha)$. This suggests that the effects of RSB become less severe as α becomes larger. Fig.(5.6) show the case of 1Bit, here the $T_{GD}(\alpha)$- and the $T_{AT}(\alpha)$-line diverge as $\alpha \to \infty$, this implies that, in contrast to the case of 5 bit, for any fixed temperature the GD-line remain unstable. This means the system never escapes from spin glass phase for $T = T_{GD}$ even as $\alpha \to \infty$. This suggests that the fluctuations in the training energy do not necessarily shrinks as α increases.
To find the asymptotic behavior of the GD-line analytically with respect to large α, we must develop the saddle point equations for large α, but the equations are too complex

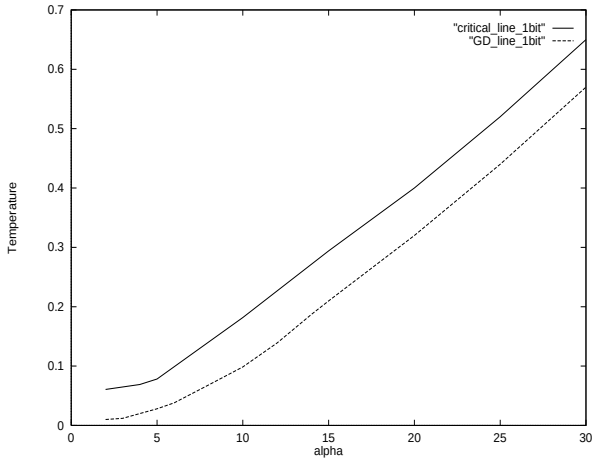

Figure 5.4: *Phase diagram in the (α, T) plane for the case of 1bit. The upper line marks $T_n(\alpha)$ line below which no numerical solution could be found. The line below marks the Gardner Derrida line*

to find a simple asymptotic dependence of the GD-line with respect to large α.
The ZE-line: In the previous subsection we presented the AT line as a boundary for accepting the replica symmetric solutions of the saddle-point equations. However, solutions with negative entropy can not represent a physical system, thus they can not be accepted as good solutions. From the definition of the entropy, positive entropy indicates that the number of solution that obey the constraints imposed on the weights is exponential in N, while negative entropy implies that there will be no valid solutions in the thermodynamic limit. Thus we can accept only solutions above the stability boundary, the AT line, and with entropy> 0. The line $T_{ZE}(\alpha)$, referred to as the zero entropy line, sets the boundary for physical solutions. Since the zero entropy line is easier to calculate than the AT-line, one can (in cases where the AT-line is difficult to estimate) rely on it to find the location of the RSB region. We found that for all studied synaptic precisions the entropie of the found solutions is a positive quantity. The entropy has also been found to be positive on the $T_n(\alpha)$ line. But on the T_{GD} the entropy must be negative, because on the T_{GD} line the volume of solutions of the saddle point equations shrinks to zero. So we can conclude that the $T_{ZE}(\alpha)$ must be in the region between $T_n(\alpha)$ and $T_{GD}(\alpha)$.

5.5.2 The learning curves

The performance of the pupil during training is usually summarized in so called learning curves. These curves describe the evolution of the generalization and training errors either as a function of the size of the training set size α or of the noise level T of the training process. Obviously all curves will depend parametrically on the synaptic depth L_w.
Because of the weight mismatch between the teacher and the pupil the optimal weight vector $\mathbf{w}^*$ is reached only in the limit $\alpha \to \infty$. This is true at all temperatures. Thus

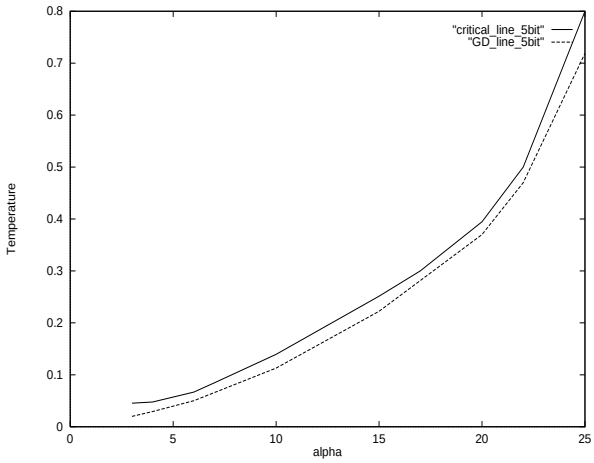

Figure 5.5: *Phase diagram in the (α, T) plane for the case of 5bits. The upper line marks the $T_n(\alpha)$ line below which no numerical solution could be found. The line below marks the Gardner Derrida line*

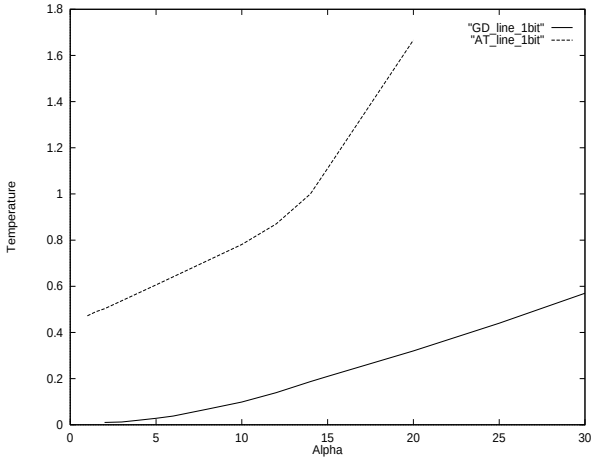

Figure 5.6: *Phase diagram in the (α, T) plane for the case of1bits.The upper line marks the marks the AT-line.The line below marks the Gardner Derrida line*

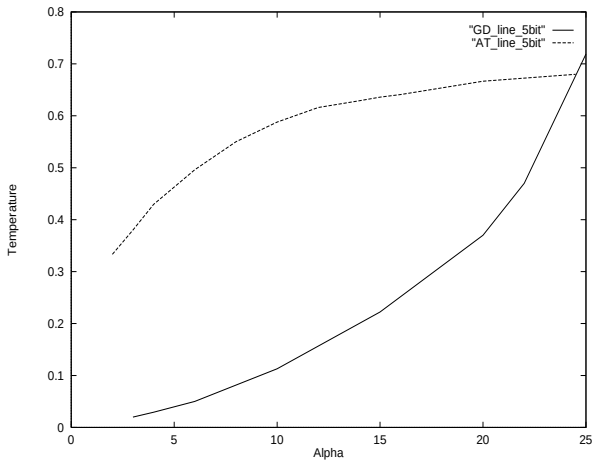

Figure 5.7: *Phase diagram in the (α, T) plane for the case of5bits.The upper line marks the marks the AT-line.The line below marks the Gardner Derrida line*

the minimal generalization error is achieved in the limit $\alpha \to \infty$.
Using the definitions of the training error function ϵ_t and that of the generalization error ϵ_g we can show that $\epsilon_t(T,\alpha) \leq \epsilon_g(T,\alpha)$ for all T and α (see [10]). As the number of examples increases for fixed T the difference $\epsilon_t(T,\alpha) - \epsilon_g(T,\alpha)$ decreases and goes towards zero, which implies that $\epsilon_g \to \epsilon_{min}$ and $\epsilon_t \to \epsilon_{min}$ as $\alpha \to \infty$.
The minimal generalization error achieved for the boolean perceptron in the limit $\alpha \to \infty$, is

$$\epsilon_{min} = \frac{1}{\pi} \arccos(R_\infty) \tag{5.72}$$

where R_∞ is the maximal overlap of teacher's and pupils weight vectors. For each allowed synaptic depth, R_∞ is given by,

$$\begin{aligned} R_\infty &= \int d\mathcal{B} P(\mathcal{B}) \sup_{\mathbf{w}_i} \sum_i \mathcal{B}_i \mathbf{w}_i \\ &= \int d\mathcal{B} P(\mathcal{B}) \sum_{\mathbf{i}} \mathbf{w}_\mathbf{i}^* \mathcal{B}_\mathbf{i} \end{aligned} \tag{5.73}$$

where $P(\mathcal{B})$ is the distribution of the teacher's weights, and $\sup_{\mathbf{w}_i}$ denotes the supremum of $\sum_i \mathcal{B}_i \mathbf{w}_i$ over all allowed values for the weights.
Since the aim of learning is to minimize the generalization error it appears necessary to explore parts of the parameter space other than the one with the minimal training error. That is achieved in a statistical sense, if the temperature is kept finite. As we have seen in chapter 3 the Gibbs distribution is a stationary state of the Langevin dynamics, with the training error as the drift potential. The temperature in that case is proportional to the variance of the noise. One of the most important facts is that learning at finite temperature is possible and sometimes advantageous (see [10]). Sometimes, even when the generalization error increases with T it may be profitable in certain circumstances to train the system at finite T because convergence time may

be prohibitively long at $T = 0$. This is particularly true in our highly nonlinear model with discrete weights.
By solving the saddle point equations for a fixed temperature, we have found that for any finite T, the generalization error is a monotonically decreasing function of the training set size α. In Fig.(5.8) we plotted the generalization error against α at $T = 0.4$, and $= 1$ for a synaptic depth of 1, and 5 bit. Surprisingly at small α, i.e. for $\alpha < \alpha_t$, the generalization error at fixed T is an increasing function of the synaptic depth, whereas at $\alpha > \alpha_t$ it becomes a decreasing function of the synaptic depth, as expected.
As we can observe in Fig.(5.8) the generalization curves end at an $\alpha_n(T, L_w)$ which represents the point where no numerical solutions could be found with the used numerical tools. So to find the asymptotic behavior of the generalization curve at a fixed T when $\alpha \to \infty$, we have to find a good fit for the generalization curves in Fig.(5.8). As we can see in Fig.(5.9) for fixed T and synaptic depth, the generalization error goes as $\frac{1}{\sqrt{\alpha}}$ towards the optimal value $\epsilon_g^{opt}(T, L_w)$ (this represents the limit of the fit curves as $\alpha \to \infty$). $\epsilon_g^{opt}(T, L_w)$ is a monotonically decreasing function of the temperature T and the synaptic depth L_w. So to achieve a better generalization ability, it is always advantageous to train our perceptron at a law temperature, but at the cost of the training time.
For a fixed α the generalization error falls monotonically with T as can be seen in Fig.(5.10). Here too, we have the points $\frac{1}{T_n}(\alpha, L_w)$ where no numerical solutions can be found. So to find the asymptotic behavior of the generalization curve at fixed α when $T \to 0$, we have to find a good fit for the generalization curves. By fitting the generalization curves at fixed α, we found that the generalization error goes as $\sqrt{T}$ towards $\epsilon_g^{opt}(\alpha, L_w)$ as $T \to 0$ see Fig.(5.10). In Fig.(5.12) we show the generalization curve for the case of 1 bit and 5 bit. These curves have been calculated along the $T_n(\alpha)$ line. The generalization error along the $T_n(\alpha)$ line has been found to be too high compared with the optimal generalization error plotted in Fig.(5.11) for different synaptic depth. The optimal generalization curves in Fig.(5.11) have been estimated by plotting $\epsilon_g^{opt}(\alpha, L_w)$ as a function of α for the corresponding synaptic depth. Because of the complex form of the saddle point equations, it is very difficult to find the asymptotic behavior of the generalization error with respect to the number of training examples analytically. Motivated by the results found for the continuous perceptron which predict $\frac{1}{\alpha}$ dependence for the generalization curve, we found also inverse power fit for the optimal generalization curves calculated for the cases of 2, 4, and 5 bit see Fig.(5.11).

In Fig.(5.15) we plotted the training error against α.There is a critical value α_c which marks the loading capacity, i.e., the point up to the examples are memorized perfectly. Above α_c the training error increases, and approaches from below the same limit as the generalization error, i.e. ϵ_t approach ϵ_{min}, as expected, from below as $\alpha \to \infty$.
In Fig.(5.14)we plot the dependence of the generalization error on the number of allowed weight values. This curve has been obtained by estimating the asymptotic limit of $\epsilon_g^{opt}(\alpha, L_w)$ for $\alpha \to \infty$ and then plotting this limit against L_w, the number of allowed

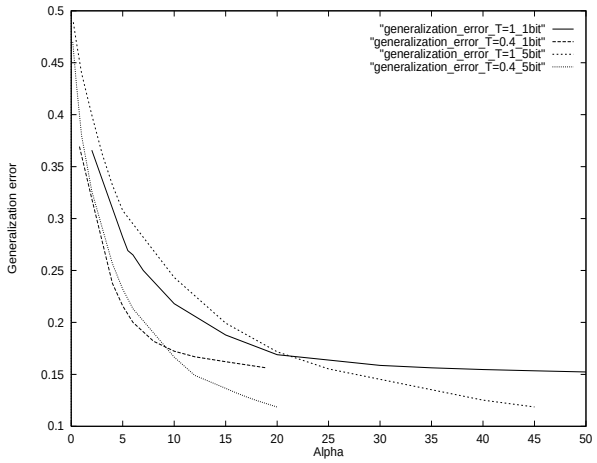

Figure 5.8: *Generalization error vs. α for 1bit and 5bit, T=0.4, 1*

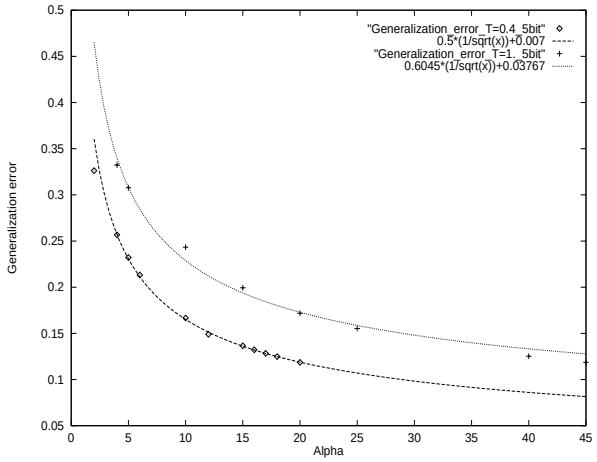

Figure 5.9: *Fit curves for the Generalization error vs. α for the case of 5bits, at T=0.4, T=1*

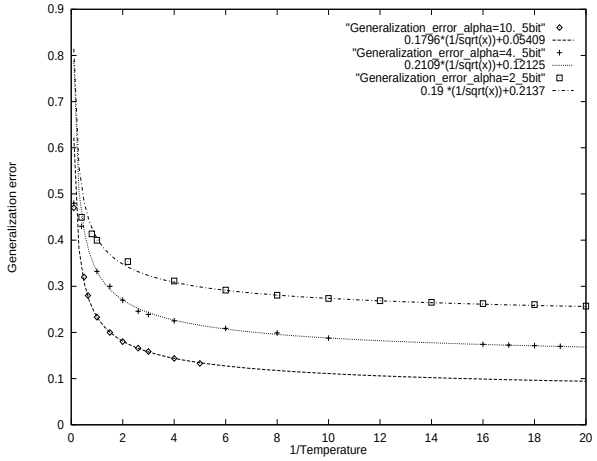

Figure 5.10: *Fit curves for the Generalization error vs. $1/temperature$ for the case of 5bits, at α=2, 4, 10*

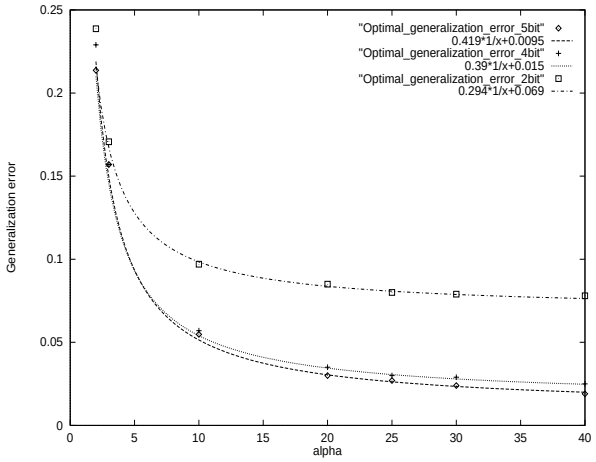

Figure 5.11: *Optimal generalization error vs. α for the synaptic depthes of 2,4, and 5 bit*

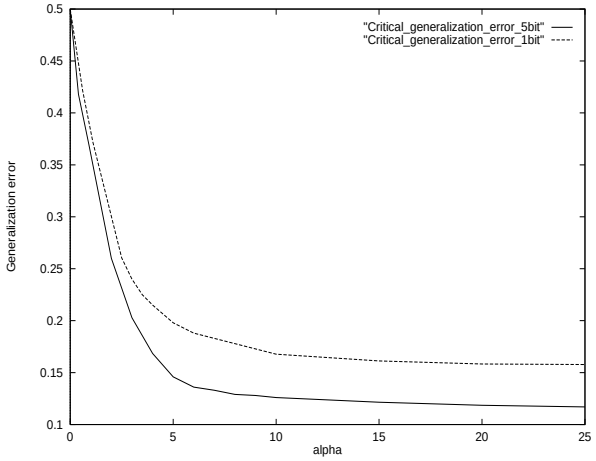

Figure 5.12: *Generalization error along the $Tn(\alpha)$ line for a synaptic depth of 1bit and 5bits*

weights. $\epsilon_g^{opt}(\alpha, L_w) \sim 0.3033\frac{1}{L_w}$ has been found to be a good fit for the dependency of generalization error on L_w. If we used 8 bit to represent the weight matrix by the optical implementation of the vector matrix multiplier, the generalization error is only about 8.10^3 smaller than generalization error in the case of 5 bit.

5.5.3 Calculating α_c

In order to calculate α_c, we have to determine the volume in the pupil's weight space for which the training examples are learnt without error. This volume is given by

$$V = \prod_{\mu} \sum_{w_i} \Theta(\kappa - \zeta_0^{\mu} \frac{1}{N} \sum_i w_i \xi_i^{\mu}) \tag{5.74}$$

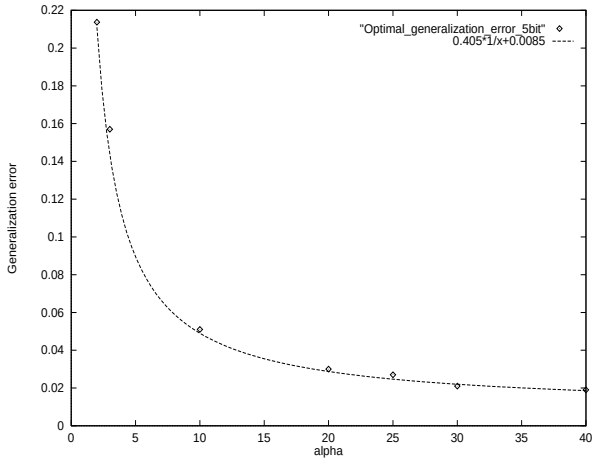

Figure 5.13: *The optimal generalization error for a synaptic depth of 5bit*

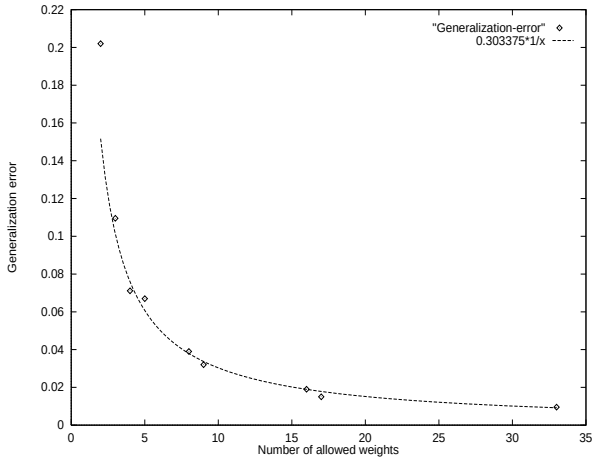

Figure 5.14: *The optimal generalization error vs. the number of allowed weight values*

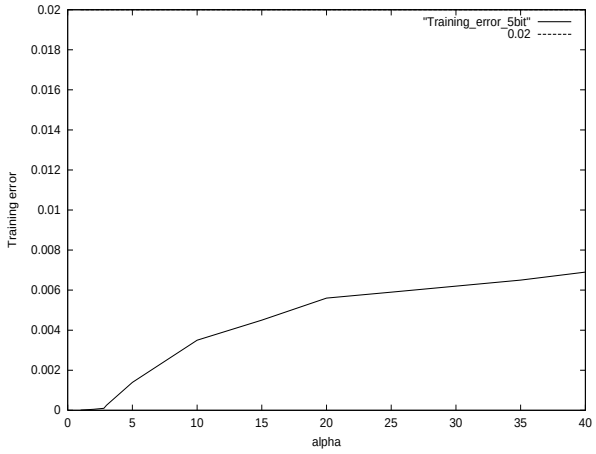

Figure 5.15: *The optimal training error for a synaptic depth of 5bit*

The typical volume is calculated as before by averaging over the distribution of the disorder produced by the examples. The saddle point equations for q_0, q_1 and R does not change when we set $T = 0$ because G1 and G2 are only implicitly dependent on the temperature. In the limit $T = 0$ the equations for $\hat{R}, F$ and E are given as follows

$$\hat{R} = -\frac{\alpha}{\sqrt{2\pi}} \int \mathcal{D}u\mathcal{D}z\{-\frac{|u|}{\sqrt{q_0 - q_1}} - \frac{Rz}{sqrtq_0 - q_1\sqrt{q_1 - R^2}}\}\frac{\exp\frac{-A^2}{2}}{H(A)} \tag{5.75}$$

$$F = \frac{\alpha}{\sqrt{2\pi}} \int \mathcal{D}u\mathcal{D}z \left\{\frac{z\sqrt{q_0 - q_1} + A\sqrt{q_1 - R^2}}{(q_0 - q_1)\sqrt{q_1 - R^2}}\right\}\frac{\exp\frac{-A^2}{2}}{H(A)} \tag{5.76}$$

$$E = \frac{\alpha}{2\sqrt{2\pi}} \int \mathcal{D}u\mathcal{D}z \left\{\frac{A}{(q_0 - q_1)}\right\}\frac{\exp\frac{-A^2}{2}}{H(A)} \tag{5.77}$$

By solving the saddle point equations we found:
There is a critical α_c which marks the loading capacity. As $\alpha \to \alpha_c$ the volume of solutions with zero training error shrinks to zero. For $\alpha > \alpha_c$ no solutions exists for the saddle point equations. So the training error begins to increase beyond zero for $\alpha > \alpha_c$. Although this α_c does not mark a transition to optimal generalization it give a reasonable scale for the decrease of ϵ_g. has been found to be $\alpha_c = 2.7, 2.3, 1.6, 1.1, 0.7$ for $5, 4, 3, 2, 1$ bit respectively.

5.5.4 Training with non zero stability κ

In the error measure we used, we have introduced the stability parameter κ which biases the training algorithm to favor networks less sensitive to the effects of noise in the training data. This helps to absorb the performance fluctuations of the optical elements representing the patterns and the weights. Tuning of κ allows us to find an optimal κ so as to minimize the generalization error for fixed α and T. In Figs.(5.16), (5.17) we plot the generalization error against κ for fixed values of T and α for the case of 1bit and 5bits. As can be seen in the figures the generalization error is not a monotonically decreasing function of κ. For every α and T there is an optimal value κ^{opt} which minimizes the generalization error. For $\kappa > \kappa^{opt}$ the generalization error begins to increase. At a constant temperature, the optimal value of κ is a decreasing function of the training set size α. In Fig.(5.18) we plot the optimal κ against α at $T = 1$ for the case of 1bit.

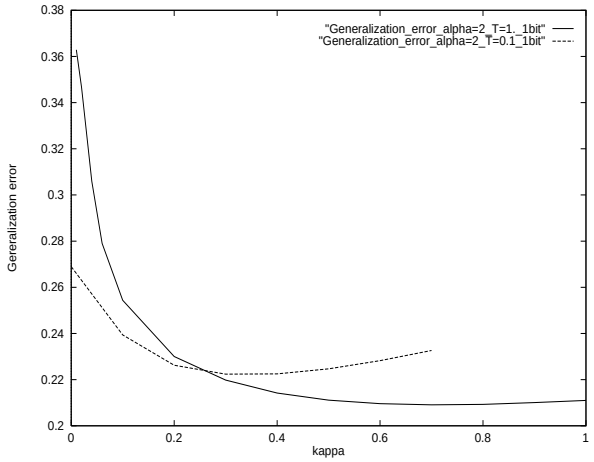

Figure 5.16: *Generalization error vs. κ for 1bit, T=0.1 and T=1., α=2*

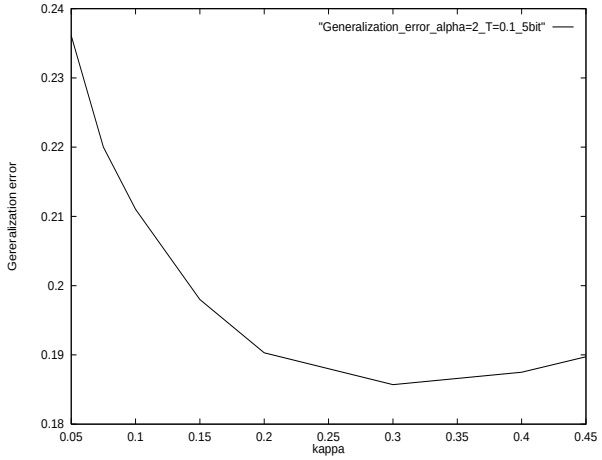

Figure 5.17: *Generalization error vs. κ for 5bit, T=0.1, α=2*

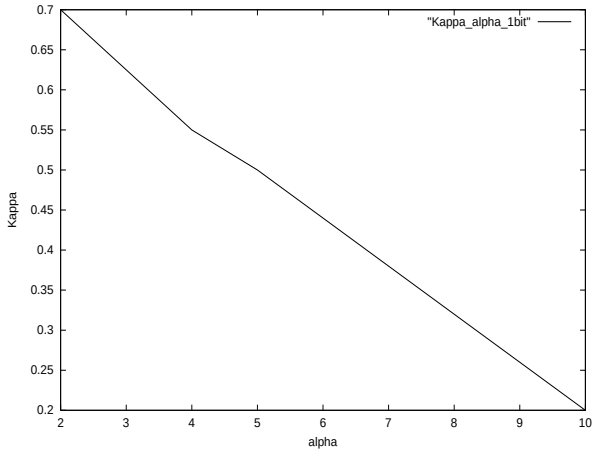

Figure 5.18: *Stability parameter κ vs α at $T = 1$, and a synaptic depth of 1bit*

6 Simulated annealing as a training strategy for discrete NN

Training a neural network with discrete weights belongs to the class of hard optimization problems referred to as NP-complete problems. NP stands for nondeterministic polynomial. This implies that the computational efforts grow exponentially with the size of the system. The discrete perceptron is a benchmark problem for the development and study of this kind of hard optimization problems, because it is simple in formulation and an objective comparison with theory is possible. A variety of learning rules have already been applied to the problem of training discrete Neural networks, this include the clipping of the weights of a continuous-valued perceptron, techniques from classical (deterministic) non-linear optimization theory, gradient descent, simulated annealing and an adaptive genetic algorithm. Of these the last two show the most potential. The adaptive genetic algorithm of Köhler consists of two parts. The first part, finds the best partial solutions and ranks them according to a simple optimization procedure and represents these solutions by 'genes'. The second part, the genetic section, fabricates new genes by splicing and combining the fittest successive genes from the old gene pool. These operations are repeated a prescribed number of times until perfect solutions are found or it takes $\mathcal{O}(N^3)$ computations to find an optimal solution. Although the adaptive genetic algorithm has the advantage of being able to mix information from different partial solutions, it still has a distinct drawback in that there is no straightforward method which guarantees to reach the optimal solution. Simulated annealing is now a well established statistical technique for tackling such hard optimization problems, and unlike other heuristic methods it is not problem specific. In what follows we are going to describe the simulated annealing algorithm and its implementation in the case of learning binary random input output patterns produced by a teacher perceptron with continuous weights which has to be learnt by a pupil perceptron with discrete weights.

6.1 Implementation of simulated annealing

A strategy, which has been applied to the solution of combinatorial optimization problems with great success, is the gradual *simulated annealing*. For example the method

of simulated annealing has been used successfully to find the optimal arrangement of integrated electronic circuits on semiconductor chips.
The concept of annealing is derived from material science, where it is used to describe the process of eliminating lattice defects in crystals by a procedure of heating, followed by slow cooling to room temperature.
If some material are cooled rapidly from the molten phase, its atoms are often captured in energetically unfavorable locations in the lattice. Once the temperature has dropped far below the melting point, these defects survive forever, since any local rearrangement of atoms costs more energy than is available in thermal fluctuations. The atomic lattice thus remains caught in a local energy minimum. However, the thermal fluctuations can be enhanced if the material is reheated until energy consuming local rearrangements occur at a reasonable rate. The lattice imperfection starts then to move and annihilate, until the atomic lattice is free of defects except for those caused by thermal fluctuations. These can happen gradually if the temperature is lowered so slowly that the thermal equilibrium is maintained at all times during the cooling process.
More precisely, in a mathematical language, the problem of optimizing by simulated annealing consists in minimizing a cost function defined on some discrete configuration space. The configuration space of the system is endowed with transformation rules which correspond to a class of local energy changes, $E \rightarrow E + \Delta E$. These transformations allow a controlled random walk in the configuration space which samples it according to the Gibbs probability distribution; the walk in the configuration space is usually generated by the Metropolis Algorithm. Metropolis *et.al.* in the early days of scientific computing introduced a simple algorithm that can be used to provide an efficient simulation of a collection of atoms in equilibrium at a given temperature. In each step of this algorithm, an atom is given a small random displacement and the resulting change in the energy, ΔE is computed. If $\Delta E \leq 0$, the displacement is accepted, and the configuration with the displaced atom is used as a starting point for the next step. The case where $\Delta E \geq 0$ is treated probabilistically that the configuration is accepted with the probability $P(\Delta E) = \exp(-\frac{\Delta E}{T})$. Like most iterative improvement schemes, the Metropolis algorithm proceeds in small steps from one configuration to the next, but the temperature keeps the algorithm from getting stuck by permitting uphill moves. The essential question by simulated annealing is to introduce a cooling schedule, which starts from some initial temperature, T_0, and progressively cooling through a sequence of partial equilibration, $T_0 \geq T_1 \geq T_2 ...$, until an optimal solution has been reached or until the system freezes. Although in outline this seems to go straight-forward, in practice performance of the annealing algorithm is sensitive to the particular choice of the cost function , cooling schedule, the way of transformation from a system state to the next and other adjustments. Indeed given the freedom in possible parameterizations, devising a good annealing algorithm is somewhat of an art. The numerical efforts of this method grow like a moderate power of the number dimension of the problem to be optimized, so that problems with quite large values of N can be treated successfully. The question of practical consequence concerns - what is the

most effective strategy given finite computing time? How can we utilize these ideas on the problem of learning a rule by a Neural network? As we have seen in chapter 3, the problem of learning a rule can be established by exposing the learning network to a stream of of $\mathcal{P} = \alpha\mathcal{N}$ learning input output examples. The outputs are produced by a network called the teacher. The learning network (the pupil) has to adapt its weights according to a training algorithm so that it gives the right output to a given input. This is equivalent to a walk in the weight space until the system settles in a minimum of a cost function defined on the configuration space which measures the rate of incorrect combination of a training example and its corresponding output.
In the language of simulated annealing the cost function is identified with the energy function E, and the configuration space is identified with the microstate of a statistical mechanical system which are defined by the different weight vectors the system can take.
In the present context a lattice defect corresponds to the incorrect combination of an training example and the corresponding output.

6.2 The Metropolis algorithm

We implemented the simulated annealing using the metropolis method of [2]. First the costs (energy) of a network with randomly initialized weights $\mathbf{w}$ is the training energy given in chapter 4. It is calculated using the following equation:

$$E = \sum_{\mu}^{\alpha N} |\kappa - \Delta_\mu| \Theta(\kappa - \Delta_\mu) \tag{6.1}$$

The Θ-function gives 1 as a result for positive arguments and 0 else. Weight changes are applied to find the network configuration with the lowest cost. Changes that lower the costs are always accepted whereas those that raise them are accepted with a certain probability dependent on the temperature . The acceptance probability is given by the following equation

$$\text{acceptance probability} = \exp(\frac{-\Delta E}{T}). \tag{6.2}$$

where $\Delta E = (E_{aft} - E_{bef})$ describes the costs differences between the new and the old weight configuration. By accepting weight configurations with higher costs the system is given the chance to leave local minima. By lowering the temperature the probability of their acceptance decreases causing the system to strive for a minimum.
The following parameters of the simulated annealing algorithm have to be adjusted:

- parameter n of cost function
- initial temperature
- cooling schedule

- step size of temperature changes
- number of weight changes per connection and temperature step
- mechanism of weight changes
- termination criteria

The parameter n of the cost function From the examined values of the parameter n (n=0,1,2) n=1,2 were found to make sense, whereas n=0 leads the system to freeze at all temperatures [7] The results represented in this thesis have been obtained by setting $n = 1$ in the cost function of equation Eq.(6.1).
The initial temperature To define an appropriate initial temperature we started with a fairly high value and observed the acceptance rate during the annealing process. The acceptance rate is calculated as a ratio between the number of accepted Monte Carlo steps and the whole number of Monte Carlo steps. Both the number of steps which minimize the cost function and are accepted with probability P=1, and the number of steps which increase the cost function and are accepted with a smaller probability than one contribute to the number of accepted MCS. A usual starting temperature of $T_0 = 10$ has been proven to be too high for our simple network. Steps in the simulated annealing algorithm which increase the cost function have a high probability at this temperature so that every state of the system is accessible. Preliminary examinations have shown that the cost function could not be reduced at this temperature. A continuously movement towards a minimum of the energy function could not be observed. This behavior of the system gave a sign for the good choice of the starting temperature. The choice of the parameter depends on the problem to be optimized. For a starting temperature near $T_0 = 0.3$ the cost function for the simple perceptron showed a constant movement towards a minimum despite temporary increases. We inferred from these results that the starting temperature could be chosen as low as $T_0 = 0.3$ resulting in an acceptance rate of 0.1. The number of Monte Carlo steps per weight shows us how many times every weight has been changed at some temperature, thus it is a measure for the time needed by the system at this temperature to reach thermal equilibrium. The number of MCS per weight is very intrinsic for the quality of the results. The size of temperature changes in the cooling schedule depends on the number of MCS per weight. If we have a large number of MCS per weight we can take bigger steps in the cooling schedule. Some investigations for $\Delta T = 0.01$ and $\Delta T = 0.005$ and 10 MCS showed that $\Delta T = 0.01$ is enough for reaching optimal results. For $\Delta T = 0.005$, despite of the high computing time, we have reached no improvement in the performance.
cooling schedule Regarding actual cooling schedule, there is until now no consensus on how to determine the optimal form. At any rate there seems to be some measure of qualitative agreement(for hard-optimization problems) that the schedule should have rapid cooling at high temperature followed by a slow gradual decrease at lower temperatures. A scheme based on the criterion of constant entropy production has been

proposed by Nulton and Solomon. However it has the drawback that the density of states must be known beforehand or somehow approximated. Previous investigations showed that using a reciprocal schedule

$$\frac{1}{T} \leftarrow \frac{1}{T} + a \tag{6.3}$$

resulted only in more computational expenditure compared to the linear schedule

$$T \leftarrow T - \Delta T \tag{6.4}$$

with $\Delta T = 0.01$ used in our simulations.

The weights The weights of the pupil are real numbers and their values were calculated by equally dividing the interval [-1,1] by the number of approved discrete weight values $n = 2^L$ which varies with the synaptic depth L. The allowed weights can be calculated as a function of n as follows

$$w_i = -1 + \frac{i}{(n-1)} \tag{6.5}$$

After initializing the connection strengths with random values drawn from the pool of allowed weight values, changes were applied in cyclic order. The mechanism to determine the new values for the weights is different from that for binary perceptron. In contrast to the case of a binary perceptron we have for $n \geq 2$ many allowed values for the weights so that the weight changes cannot be established only by flipping the weights from -1 to 1 or vice versa. The weight changes can be established in two ways, the first is to choose a random index for a component of the vector where we have stored the allowed weights, this component is then the new weight. The second way is to take the nearest allowed value to the value of the weight which has to be changed at a certain cycle.

The number of Monte Carlo steps per weight The number of Monte Carlo Steps (MCS) denotes the number of weight changes per temperature and connection. Its value depends on the number of input nodes and was fixed to 20MCS times 100 input nodes.

Termination criteria The algorithm terminated when

- the problem has been learned with zero training error.
- zero temperature has been reached.
- the acceptance rate dropped below 0.01.

The training set The binary learning patterns were generated by creating a real valued random number $\in [0, 1]$ using a random number generator. If this number were

smaller than 0.5 the value were set to -1 otherwise to 1.
To exclude influence of coincidence we averaged over 50 randomly chosen pupil configurations and newly initialized training patterns for every pupil. We additionally established a pocket mechanism where the best temporary solution were stored. The costs of the final solution were compared to those of the pocket solution and the cheaper one stated the actual solution.

7 Simulation results

In order to demonstrate the validity of our results we compare the theoretical predictions with computer simulations. In the simulations we investigated the case of pupil perceptrons with $N = 100$, $N = 200$, and synaptic depth of $L = 1, 2, 3, 4, 5, 8$. The components of the weight vector of the teacher are continuous valued numbers. They have been chosen randomly from the interval $[-1, +1]$. The simulations were performed by averaging over 50 pupils.
The patterns used for the training are independent uniformly distributed random binary $\{\pm 1\}$ input-output training examples. The dependence of the generalization and training errors on α has been studied for a given synaptic depth. We used simulated annealing as a training algorithm because training a neural network with discrete weights belongs to the class of hard optimization problems referred to as NP-complete problems.
Because the simulated annealing Algorithm is a very computing intensive algorithm, we tried to reply the question: whether we could obtain comparable results with Monte-Carlo simulations at a constant temperature $T = 0$. The simulated annealing results will be compared with results obtained by another known training algorithm for discrete weighted neural networks, namely training with direct clipping the components of the teacher's weight vector to the nearest allowed discrete weight value. To be able to compare results obtained by the analytical investigation, which are valid only in the thermodynamical limit $N \to \infty$, with the simulation results, we studied the influence of the network size on the generalization curve for perceptrons with 100, and 200 input neurons. [7]

7.1 Discrete perceptron - simulated annealing

7.1.1 Learning and generalization errors

Increasing the number of allowed weight values gives the pupil's weight vector the possibility to have a greater overlap with the continuous weight vector of the teacher, and hence to have a smaller generalization error. This behavior is observed for the

[7] The simulations have been done by Steffen Schwember in the frame of a diploma thesis which has been supervised by me as a part of my thesis.

Synaptic depth	Fit function	Asymptotic limit
1 bit	$0.181 + \frac{0.155}{\alpha}$	0.181 (0.18)
2 bit	$0.072 + \frac{0.246}{\alpha}$	0.072 (0.09)
3 bit	$0.034 + \frac{0.398}{\alpha}$	0.034 (0.044)
4 bit	$0.016 + \frac{0.515}{\alpha}$	0.016 (0.022)
5 bit	$0.012 + \frac{0.541}{\alpha}$	0.012 (0.01)

Table 7.1: *Functional dependence of the generalization error on α for different synaptic depth; Fit with a power-function after including only the last 30 values; between brackets we listed the generalization error obtained by directly discretizing the weights.*

generalization curves in figures (7.4), (7.5), (7.6), (7.7), (7.8). The generalization error decreases as expected when the bit precision of the pupil increases. To find the functional dependence of the learning curves on the number of training patterns for different bit precisions, we fitted the learning curves with an inverse power function of α. This has been motivated from some results found for different training algorithms in the case of a teacher-pupil problem, where both perceptrons (teacher and pupil) have continuous weights. A dependence of the form $\epsilon \sim \frac{1}{\alpha}$ has been found for many of these algorithms (see [10]). The found functional dependence of the generalization error for different bit precisions are listed in the table [7.1]. In table [7.1] we listed the limit of the fit functions for $\alpha \to \infty$. The generalization error estimated by directly discretizing the weights is also listed between brackets for the corresponding bit precision in table [7.1].
With the method of directly discretizing the weights of the teacher, namely setting every component of the teacher's weight vector equal to the nearest allowed discrete value in the interval $[-1, +1]$, we can find a good pupil with discrete weights. One might suspect the pupil found by this method is the optimal discrete pupil. This is true for the binary pupil perceptron. But for higher synaptic depths, directly discretizing the weights determines a pupil's weight vector with a similar length to that of the teacher, but not the one with the largest overlap R, which is crucial for a good generalization. (see fig (7.1),(7.2),(7.3)).

The approximation of the generalization curve with an inverse power function is bad,

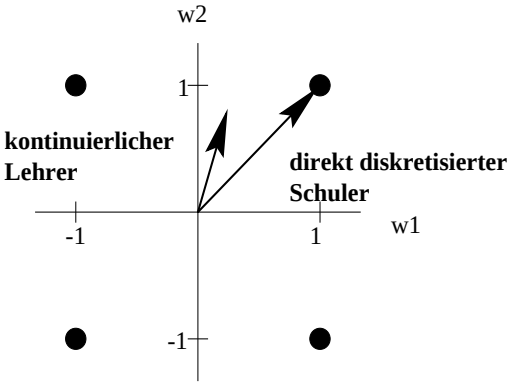

Figure 7.1: *In the case of a binary pupil the directly discretized pupil is the optimal one.*

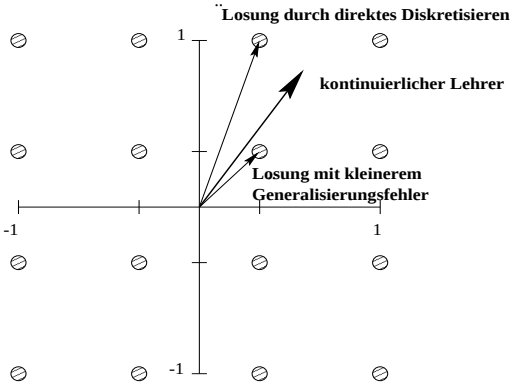

Figure 7.2: *In the case of higher synaptic depth there exists pupils weight vectors with a smaller generalization error than the one found by directly discretizing the weight vector of the teacher(case A).*

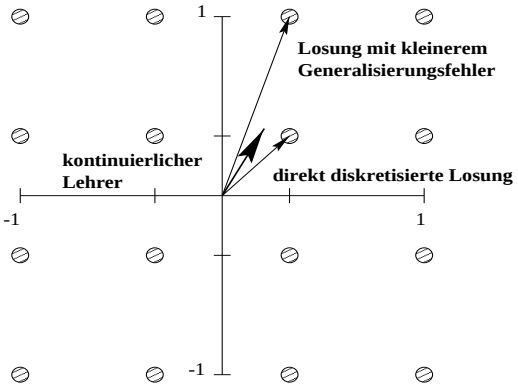

Figure 7.3: *In the case of higher synaptic depth there exists pupils weight vectors with a smaller generalization error than the one found by directly discretizing the weight vector of the teacher (case B).*

if we take all the calculated points. But if we take only the last 30 points we get a better approximation for the asymptotic behavior. This is very important for the comparison with theoretical results, because these are correct for large α (in the case of $L \geq 1$ the RS solutions are stable for large α). The asymptotic behaviour in the case of a teacher-pupil with continuous weights which predict a $\frac{1}{\alpha}$ dependence has been also confirmed in our results.

As we can see in figure (7.4), the fit curve approximates the behavior of the whole generalization curve in the case of 1bit very well. while for higher synaptic depth, the fit with an inverse power function is not so good in approximating the behavior of the generalization curve for small α.
As we can see in figures(7.4), (7.7), (7.8)the generalization curves for 1,4,5 bit goes asymptotically toward the generalization error calculated by directly discretizing the weight of a pupil with continuous weights. But in the case of 2 and 3 bits (Fig.(7.5),(7.6)) the generalization curves fall a little bit below that calculated by directly discretizing the weight vector of the teacher. This confirms the expectation that there exists some pupil weight vectors which have a greater overlap with the weight vector of the teacher than that obtained in the direct discretized case, and that ϵ_{min}, the asymptotic limit, is smaller than the generalization error calculated by direct discretizing the weights of the pupil. This tendency can be observed for the synaptic precisions of 1,4,5, if we look at the asymptotic limits of the fit curves listed in table[7.1]. For all synaptic precisions listed in table [7.1] the asymptotic limit is smaller than that for the direct discretized case (the values between brackets).
Simulated annealing is able to find these pupils with better generalization ability than the direct discretization method, if we use a large enough training set and a large enough number of MCS per weight. And thus it is better than the directly discretizing algorithm. But if the training set is smaller than $\alpha N = 10N$ the direct discretizing algorithm leads to pupils with smaller generalization error than the simulated annealing algorithm.
In the case of a continuous pupil-teacher problem we can get, with a suitable training algorithm, a training error equal to zero for every α. In our case where the pupil is constrained to discrete weights, it is not possible to find a pupil with zero training error as α increases (because the rule the pupil has to learn is not perfectly learnable with discrete weights). This means with a growing number of training examples it becomes difficult to find a hyperplane which separates the example space linearly with the allowed discrete weights. Thus the training error begins to increase at a critical α_c and and is expected to move towards ϵ_{min}.
To show that the training error increases with increasing α we calculated for $N = 100$ belated additional points for $\alpha = 20, 30$, and 50. The obtained training error support our hypotheses that the training error for large α increases. However, from the results listed in table (7.2) which summerizes the functional dependence found for the training error as a function of α, one would expect that the training and generalization errors go asymptotically to different ϵ_{min} as α increases. The asymptotic limit found for the training curves for large α is always smaller than the limit found for the generalization

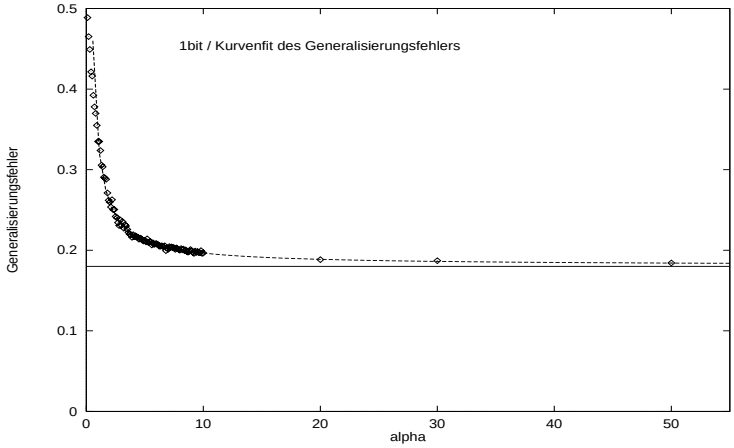

Figure 7.4: *Fit curve of the generalization error for 1bit, N=100; Fit with a power-function after including only the last 30 points.*

curves for the corresponding synaptic depth. The reasons for this discrepancy could be the small number of calculated points which makes it difficult to find a good fit.
The fit of the training error curves with an inverse power function are shown for the different synaptic depth in Figures (7.9), (7.10),(7.11). The fits become worse with increasing synaptic depth for small α (this is expected because the training error vanishes for $\alpha < \alpha_c$). For the case of 5 bit, we could not find a good fit.
Because the theoretical results obtained with the tools of statistical mechanics are relevant only in the thermodynamical limit where the number of input neurons goes to infinity, $N \to \infty$, we have investigated the effect of the number of the input neurons by studying a pupil with 200 input neurons. No significant difference has been found for the learning curves in comparison with the case of 100 input neurons. Fig.(7.12) supports this statement. We have not investigated networks with more than than 200 input neurons, because a very high computing time is required to investigate larger networks.

7.1.2 Monte-Carlo simulations at T=0

The simulated annealing algorithm is very computing intensive algorithm because of the Monte-Carlo simulations at different temperatures. The questions which arises is whether we could obtain comparable results with Monte-Carlo simulations at a constant temperature $T = 0$. Here we use a higher number of Monte-Carlo-Steps (MCS) per weight than in the case of simulated annealing. We used Monte-Carlo-simulations at $T = 0$ with 700 MCS per weight. The weights are changed cyclically and are given new values from the setof allowed weights values corresponding to the allowed synaptic depth. From the Figs.(7.13),(7.14), (7.15), we can observe that the generalization error obtained with simulated annealing for the same number of training patterns is a bit smaller than that of Monte-Carlo simulation at zero temperature $T = 0$. Monte-

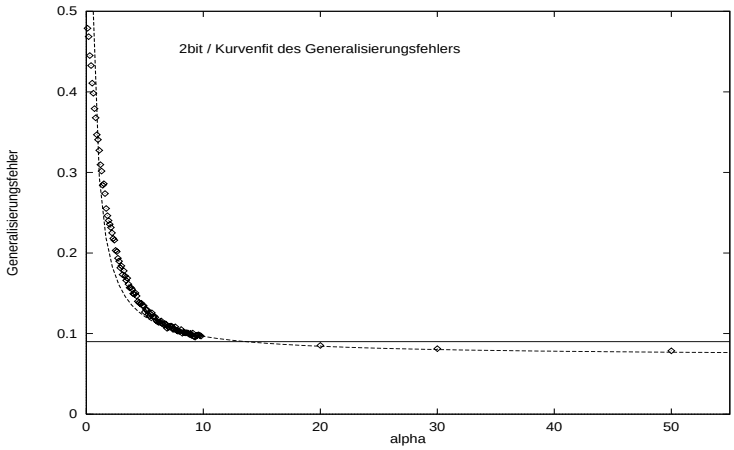

Figure 7.5: *Fit curve of the generalization error for 2bit, N=100; Fit with a power-function after including only the last 30 points.*

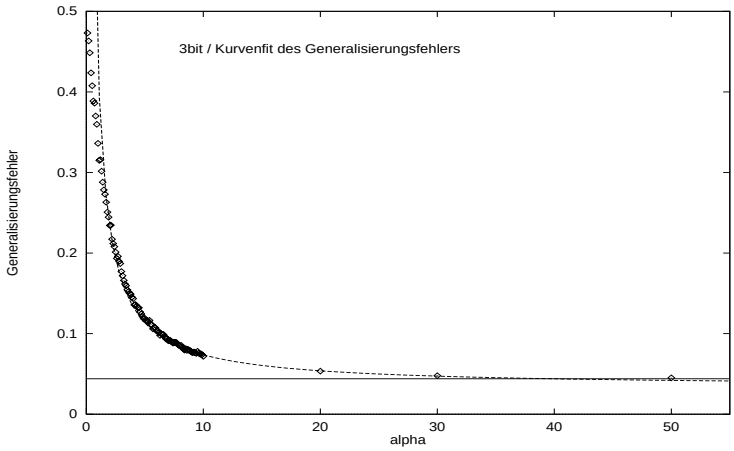

Figure 7.6: *Fit curve of the generalization error for 3bit, N=100; Fit with a power-function after including only the last 30 points.*

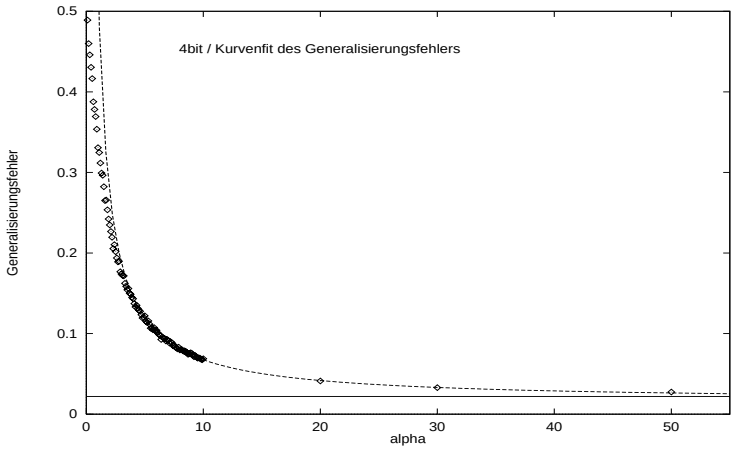

Figure 7.7: *Fit curve of the generalization error for 4bit, N=100; Fit with a power-function after including only the last 30 points.*

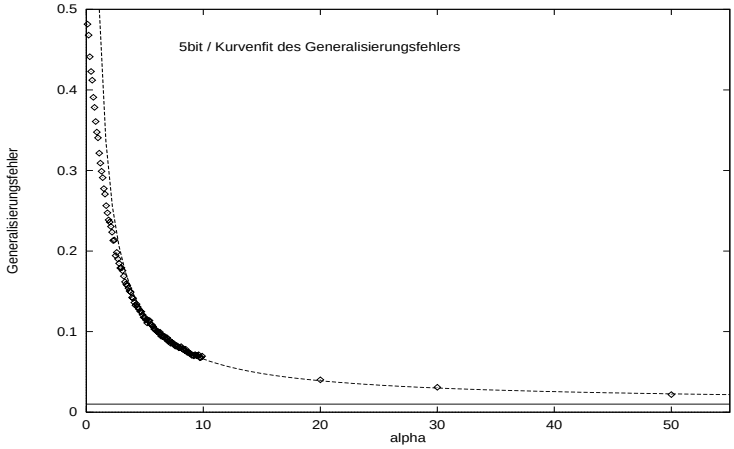

Figure 7.8: *Fit curve of the generalization error for 5bit, N=100; Fit with a power-function after including only the last 30 points.*

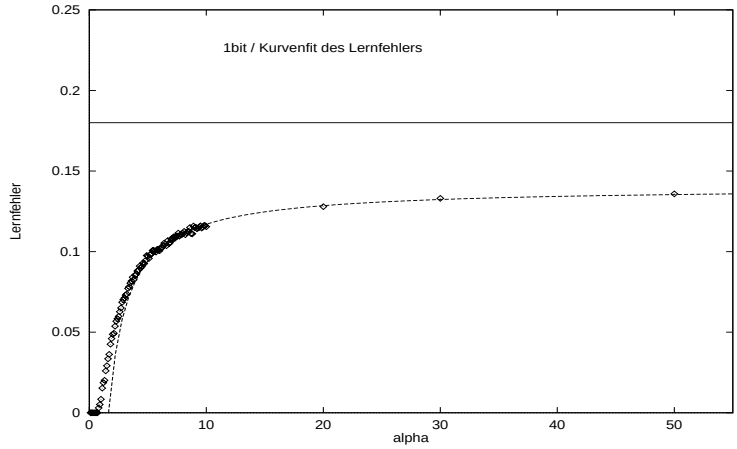

Figure 7.9: *Fit curve for the learning error for 1bit, N=100; Fit with power - Function.*

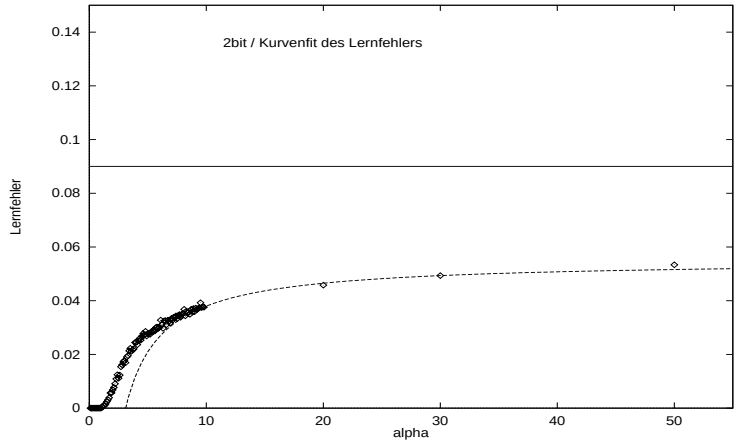

Figure 7.10: *Fit curve for the learning error for 2bit, N=100; Fit with power - Function.*

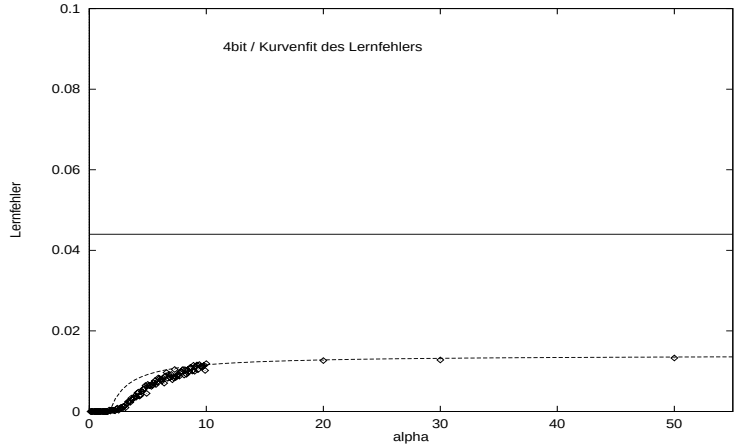

Figure 7.11: *Fit curve for the learning error for 4bit, N=100; Fit with power - Function.*

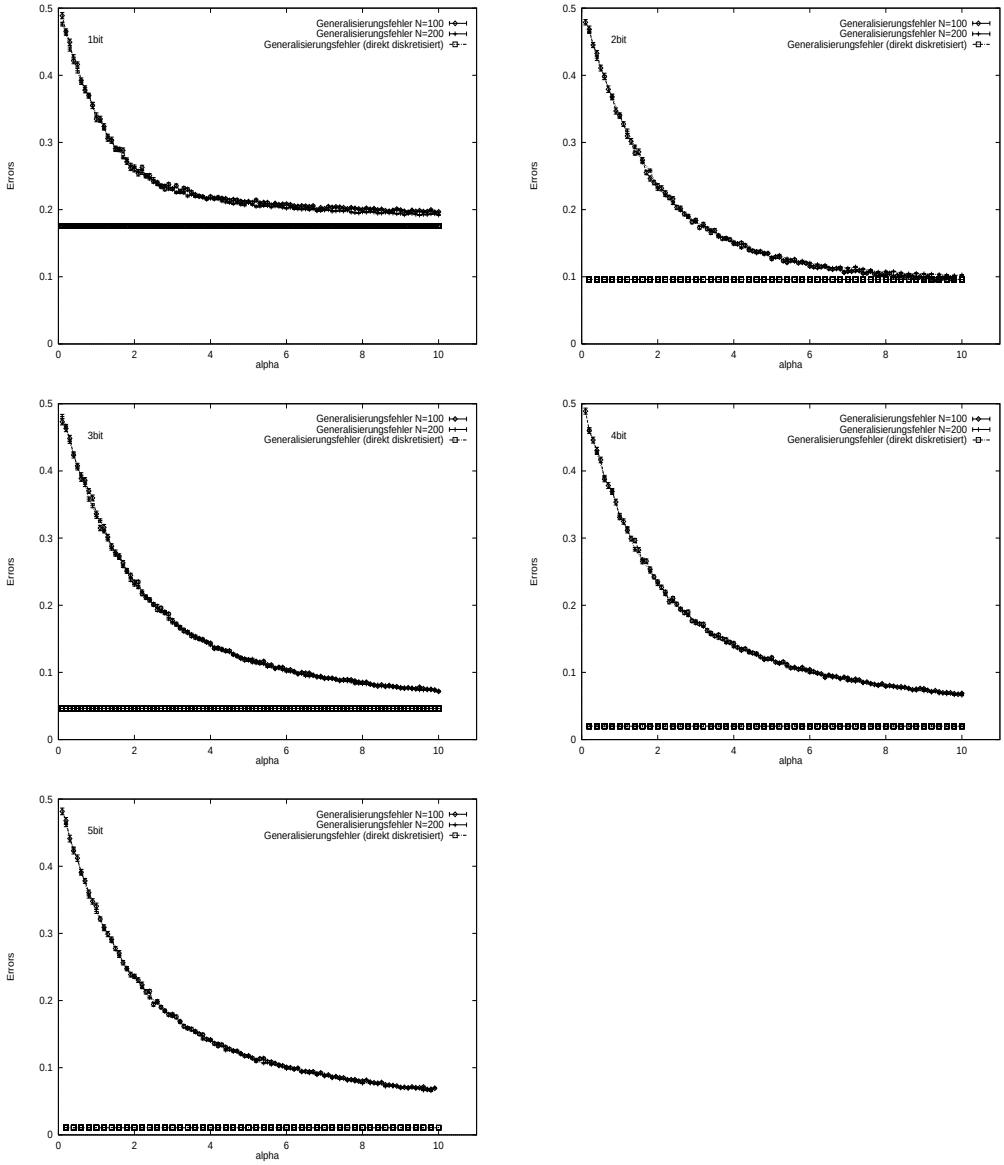

Figure 7.12: *comparison of the Generalization error for perceptrons with N=100 and N=200 input neurons*

Synaptic depth	Fit-function	Asymptotic limit
1 bit	$0.14 - \frac{0.23}{\alpha}$	0.14 (0.185)
2 bit	$0.055 - \frac{0.17}{\alpha}$	0.055 (0.071)
3 bit	$0.027 - \frac{0.1}{\alpha}$	0.027 (0.034)
4 bit	$0.014 - \frac{0.024}{\alpha}$	0.014 (0.022)

Table 7.2: *Functional dependency of the learning error on α for different synaptic depth; Fit with a power-function after including only the the last 30 points; The asymptotic limit of the generalization error is presented between brackets.*

Carlo simulation accepts only weight changes which reduces the cost function, energy function, so it is in this case not possible to avoid local minima of the system. Thus we would expect that the difference of the two algorithms, Monte-Carlo simulation at zero temperature and simulated annealing, must be larger than the difference observed in Figs. (7.13),(7.14), (7.15). Because of the very small difference we could conclude that the simple perceptron has only few number of local minima which do not affect the generalization error calculated by averaging over many pupil perceptrons. The replica theory predicted for a pupil with continuous weights a generalization error of $\epsilon = 0.07$ for a training set of 10α. Monte-Carlo simulation at zero temperature and simulated annealing lead to pupil perceptron with the same generalization error for the discrete perceptron in the case of synaptic depth of 5bit. Simulated annealing requires a higher computing time than Monte-Carlo simulation at $T = 0$ because Monte-Carlo simulation does not need a cooling procedure. Thus it is advantageous to utilize Monte-Carlo simulation in training a simple perceptron with discrete weights, if we have a large enough training set.
We expect that Multilayer perceptrons with discrete weights have much more local minima than the simple perceptron, so it is better to use simulated annealing for its training because training with finite temperature allows us to avoid local minima. In this case training at $T = 0$ with Monte-Carlo simulation is not recommendable.

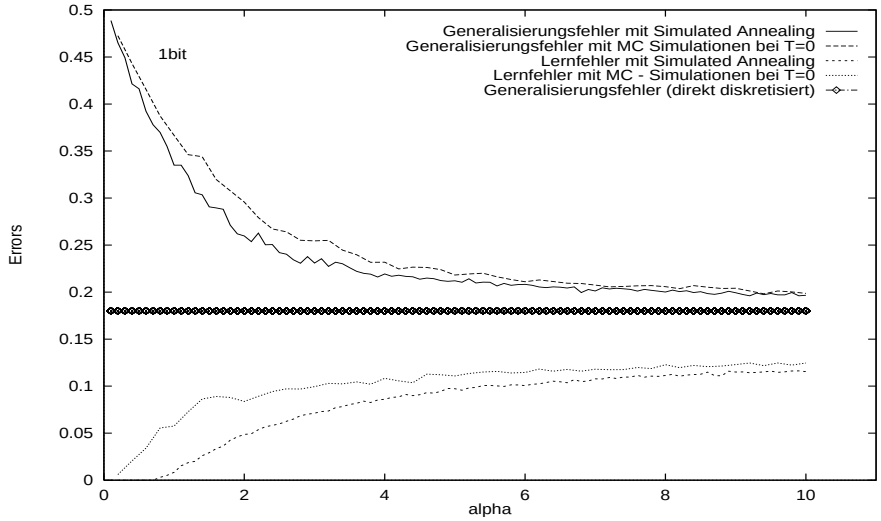

Figure 7.13: *Learning and generalization errors vs. α for a pupil trained with simulated annealing and Monte-Carlo simulations; 1bit synaptic depth, N=100, averaged over 50 pupils. Monte-Carlo simulations at T=0 with 700 MCS .*

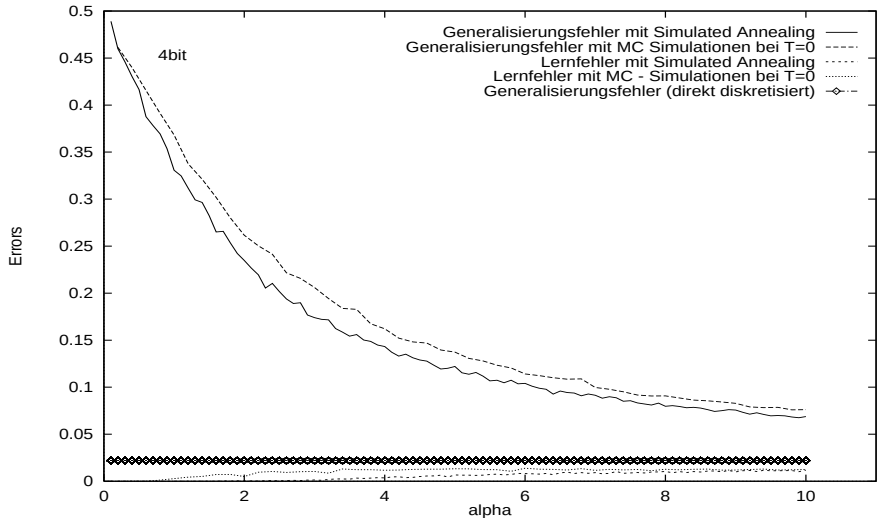

Figure 7.14: *Learning and generalization errors vs. α for a pupil trained with simulated annealing and Monte-Carlo simulations; 4bit synaptic depth, N=100, averaged over 50 pupils. Monte-Carlo simulations at T=0 with 700 MCS .*

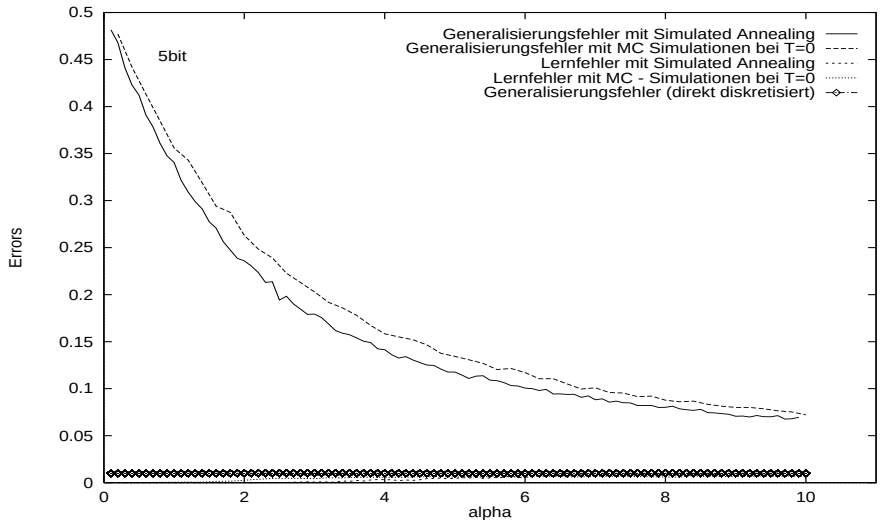

Figure 7.15: *Learning error vs. α for a pupil trained with simulated annealing and Monte-Carlo simulations; 5bit synaptic depth, N=100, averaged over 50 pupils. Monte-Carlo simulations at T=0 with 700 MCS.*

8 Conclusion

The optical implementation of neural networks can be realized by storing the weights in holograms with a limited number of gray values. Motivated by this fact, we focused our investigation in this thesis on analyzing the dependence of the generalization and training errors of a simple perceptron with discrete weights, on the training set size, and on the number of allowed discrete values (there are $L_w = 2^L$ allowed discrete values for a bit precision of L).
Our starting point is the teacher pupil paradigm which is a natural approach to study the properties of neural networks and especially their generalization ability, if we want to get general information about their behavior independent of a special training set. The teacher is a single-layer perceptron. The components of the teacher's weight vector are continuous valued numbers from the Interval [-1,+1]. The pupil is also a single-layer perceptron whose weight vector is only allowed to have discrete equidistant values spanning the interval [-1,+1]. The pupil was trained to learn the rule produced by the teacher by exposing it to a stream of αN input output examples $(\xi^1, \zeta_0^1), (\xi^2, \zeta_0^2), ..., (\xi^{\mathcal{P}}, \zeta_0^{\mathcal{P}})$, where the inputs $\xi^\mu = (\xi_1^\mu, \xi_2^\mu, ..., \xi_N^\mu)$ are chosen to be random, and the outputs are generated by the teacher. This problem has been investigated analytically and by computer simulations. For the analytical investigation we used sophisticated techniques of statistical mechanics, which have been originally developed to study complex spin systems. Statistical mechanics is a useful tool to analyze learning in ANNs for different reasons. One is that many algorithms are stochastic and correspond exactly to Langevin or Glauber dynamics on a noisy energy landscape. Another reason for using statistical mechanics is that learning is essentially a problem of statistical inference and fits naturally into the same mathematical framework.
By utilizing the replica trick, the replica symmetric ansatz and the saddle point method we obtained the saddle point equations, which describe the behavior of the learning system in the thermodynamical limit.
By determining the learning curves, we have shown that restricting the weights of the pupil to discrete values makes the rule produced by the continuous teacher unlearnable. This means that there is no pupil with discrete weights which can learn the rule perfectly. This is in contrast to the case of pupil with continuous weights which would learn the teacher's rule with a suitable training algorithm, after a training with a large enough training set. For the case of binary pupil learning a rule produced by a binary teacher György(1990) and Seung *et al* (1991) found that there is a discontinuous tran-

sition to perfect generalization at $\alpha = 1.245$. This is not the case here. There is no discontinuous transition to perfect learning. By solving the saddle point equations for a fixed temperature, we have found that for any finite T, the generalization error is a monotonically decreasing function of the training set size α. Analogous behavior was found for the dependence of the generalization error as a function of the temperature for a fixed α. The generalization error has been found to go as $\frac{1}{\sqrt{\alpha}}$ and $\sqrt{T}$ towards the optimal values $\epsilon_g^{opt}(T, L_w)$ and $\epsilon_g^{opt}(\alpha, L_w)$ as $\alpha \to \infty$ or $T \to 0$, respectively. We found that $\epsilon_g^{opt}(\alpha, L_w)$ behaves as an inverse power function of α. This has been also predicted for the case of a continuous pupil learning a continuous teacher. The difference between the two cases is that $\epsilon_g^{opt}(\alpha, L_w)$ has a non zero asymptotic limit $\epsilon_g^{opt}(L_w)$ as $\alpha \to \infty$. Whereas, the optimal generalization error curve of the continuous pupil goes towards 0 as $\alpha \to \infty$. $\epsilon_g^{opt}(L_w)$ is found to behave as an inverse power function of the number of allowed weights L_w. This is the most important result of this thesis for the implementation of neural networks. Since the constraints imposed by the optics for the implementation of neural networks allow a maximal precision of 6 bit, we can see that 5 bit are enough to get a generalization error of about only 1%. If we used 8 bit to represent the weight matrix by the optical implementation of the vector matrix multiplier the generalization error is only about 8.10^{-3} smaller than generalization error in the case of 5 bit.
The optimal training error curve has also been studied for the case of 5 bit. We found that there is a critical value α_c which marks the loading capacity, i.e., the point up to which the examples are memorized perfectly. α_c has been found to be $2.7, 2.3, 1.6, 1.1, 0.7$ for $5, 4, 3, 2, 1$ bit synaptic depth, respectively. Above α_c the training error increases, and approaches the same limit as that of the generalization error, i.e. ϵ_t approaches $\epsilon_g^{opt}(L_w)$, as expected, from below as $\alpha \to \infty$. Although this critical α_c does not mark a transition to optimal generalization, it gives a reasonable scale for the decrease of ϵ_g with respect to α. This α_c is greater than the capacity calculated for the discrete perceptron for random pattern which has a maximum of $2N$ in the case of the continuous perceptron.
The influence of the stability parameter κ on the performance has been found constructive as expected. Tuning of κ allows us to find an optimal κ so as to minimize the generalization error for fixed α and T. For $\kappa > \kappa^{opt}$ the generalization error begins to increase. κ^{opt} is found to be a decreasing function of the training set size α, when T is fixed. The stability parameter κ biases the training algorithm to favor networks less sensitive to the effects of noise in the training data. This helps to absorb the performance fluctuations of the optical elements representing the patterns and the weights in the optical implementation of neural networks.
By solving the saddle point equations, we found that for a fixed α and $T > T_{GD}(\alpha)$ the equations possess solutions with $q_1 < q_0$. For a fixed α, $q_1 \to q_0$ as $T \to T_{GD}(\alpha)$. At a temperature $T_n(\alpha) > T_{GD}(\alpha)$, $q_1 - q_0$ becomes so small that terms in the saddle point equations which contain $q_1 - q_0$ in the denominator very large. For $T < T_n(\alpha)$, numerical solutions with available double precision arithmetic are then no longer feasible.

At $T_{GD}(\alpha)$, $q_1 - q_0$ becomes zero, and the saddle point equations become singular. The fact that $q_0 - q_1 = 0$ at $T_{GD}(\alpha)$ can be understood as follows: As $q_1 \to q_0$ the subspace of solutions shrinks until it contains only one solution, so that the overlap between two different replicas becomes the same as the length of the weight vector of the optimal solution. At this point the saddle point equations become singular. In contrast to the case where the pupil perceptron can take continuous weights, the replica symmetric solution has been found to be unstable with respect to small local variations of the order parameters and their conjugate variables in some regions in the α, T phase space. The so called replicon eigenvalue λ_R determines the stability boundary of the RS solution with respect to RSB-fluctuations. The limit of stability is given by $\lambda_R = 0$. This condition defines a function in the parameter space $T_{AT}(\alpha)$, which is the De Almeida Thouless line. Above this line the replica symmetric solution is stable, so the RS assumption is exact in the high temperature phase. Below the AT-line the replica symmetric solutions are unstable with respect to fluctuations towards replica symmetry breaking. This fact indicates that the assumption of independence of the order parameters and their conjugates on their replica index is too simple. The onset of the RSB signals the occurrence of a spin glass phase. Formally, the spin glass phase is characterized by a non trivial dependence of the order parameters on the replica indices. So to find stable and exact solutions in the region where the RS is broken, we have to make another assumption about the dependence of the order parameters and their conjugate variables on the replica index. This has not been investigated in this work, because the saddle point equations become so complicated that no numerical solution could be found even for the one step RSB. The occurrence of the spin glass phase indicates that many degenerate ground states of the energy exists which are well separated in the configuration space. Furthermore, these degenerate ground states occupy disconnected regions in the configuration space that are separated by energy barriers that diverges with N. This leads to anomalous slow learning dynamics. The $T_c(\alpha)$ and the $T_{GD}(\alpha)$ for the case of 1 bit synaptic depth are below the $T_{AT}(\alpha)$ line which imply that the solutions at $T_c(\alpha)$ and $T_{GD}(\alpha)$ are unstable with respect to RSB. This means the system never escapes from the spin glass phase even as $\alpha \to \infty$. In contrast, for a synaptic depth of 5 bit the $T_{GD}(\alpha)$ intersects the $T_{AT}(\alpha)$ at the point $(T = 0.68, \alpha = 24.5)$. This implies that for $(T > 0.68, \alpha > 24.5)$ all the points on the $T_c(\alpha)$ the RS solution become stable. This suggests that the effects of RSB become less severe, as α become larger. This suggests that the fluctuations in the training energy do not shrink as α increases.

For all studied synaptic precisions the entropy is a positive quantity. The entropy is also positive at the $T_c(\alpha)$ line. At the $T_{GD}(\alpha)$ the entropy becomes negative, because the volume of solutions shrinks to zero, so the zero entropy line must be in the area between $T_c(\alpha)$ and $T_{GD}(\alpha)$.

Because the RS picture is not quantitively accurate in the whole α, T space, we tried to confirm our results with computer simulations. The results for $\kappa = 0$ have been confirmed semiquantitatively with computer simulations.

Because training a neural network with discrete weights belongs to the class of hard op-

timization problems referred to as NP-complete problems, we used simulated annealing as a training algorithm, which is now a well established statistical technique for tackling such hard optimization problems. In the simulations we have investigated the case of pupil perceptrons with $N = 100$, $N = 200$, and synaptic depth of $L = 1, 2, 3, 4, 5, 8$. We found also here that the generalization error is a decreasing function of α and L_w. The $\frac{1}{\alpha}$ dependence of the generalization error, which is found in the theoretical investigation, is confirmed here by the simulations. For all the investigated synaptic depths, we found a good $\frac{1}{\alpha}$ fit curve for the generalization error. If we compare the curves of different synaptic depth we observe that we need a smaller training set to obtain a certain generalization error, if we used a higher precision. The difference between the theoretical $\alpha \to \infty$ limits for the generalization error and the simulation limits are in the 0.15% range. The limits obtained by directly clipping the weight vector of the teacher to the nearest allowed discrete weight vector are for the synaptic depths of $2, 3, 4$ as expected higher than those obtained theoretically or by simulated annealing.

The effect of the network size on the generalization curves has been found to be very small. No significant difference has been found between the generalization curve for the case of 100 input neuron and that for 200 input neurons. We have not investigated networks with more than than 200 input neurons, because a very high computing time is required to investigate larger networks.
The Monte-Carlo simulations at a constant temperature $T = 0$ have shown that the generalization error in this case is only a little bit higher than that obtained by simulated annealing. Because of the very small difference we could conclude that the simple perceptrons have only few number of local minima which do not affect the average generalization. Simulated annealing requires a higher computing time than Monte-Carlo simulation at $T = 0$, because $T = 0$ Monte-Carlo simulation does not need a cooling procedure. Thus it is advantageous to utilize such Monte-Carlo simulation in training a simple perceptron with discrete weights, if we have a large enough training set.

There are also many open question for which there was not enough time to be studied. What is the influence of the Neural network architecture on the network performance if the synaptic depth is limited? How robust are optical neural networks against accuracy fluctuation which can happen by storing its weights in holograms? How do neural networks with discrete weights operate on real data?

9 Zusammenfassung

Aufgrund der Geschwindigkeit und der wechselwirkungsfreien Überlagerung von Photonen wird der optischen Implementierung neuronaler Netze immer mehr Aufmerksamkeit gewidmet. Eine der möglichen Implementierungsformen kann realisiert werden indem man die Gewichte in Hologrammen mit limitierter Anzahl von Graustufen speichert. Motiviert durch diese Tatsache, haben wir uns in dieser Arbeit auf das Analysieren der Abhängigkeit des Generalisierungs- und Trainingsfehlers eines simplen Perceptrons von der Anzahl der Trainingsmuster und der Anzahl der erlaubten diskreten Gewichte (es gibt $L_w = 2^L$ erlaubte Gewichtswerte für eine synaptische Tiefe von L) konzentriert. Unser Ausgangspunkt ist das Lehrer-Schüler Paradigma, welches ein natürliches Werkzeug darstellt, um Eigenschaften neuronaler Netze zu studieren, insbesondere ihre Generalisierungsfähigkeit, wenn wir unabhäbgig von einem speziellen Trainingsset allgemein gültige Informationen über ihr Verhalten erhalten möchten.
Der Lehrer ist ein simples Perceptron dessen Gewichtsvektorkomponenten reelle Zahlen aus dem Intervall $[-1, +1]$ sind. Der Schüler ist auch ein simples Perceptron, dessen Gewichte nur equidistante diskrete Werte aus dem Intervall $[-1, +1]$ annehmen dürfen. Der Schüler lernt die von dem Lehrer hergestellte Regel, indem man ihm eine Trainingsmenge von Input-Output Mustern zur Verfügung stellt. Dieses Problem wurde analytisch untersucht und mit Computersimulationen bestätigt. Die analytische Untersuchung wurde mit Hilfe von Techniken aus der statistischen Mechanik, die ursprünglich für die Untersuchung von komplexen Spin Systemen entwickelt worden sind, durchgeführt. Statistische Mechanik ist aus verschiedenen Gründen eine hilfreiche Technik, um das Lernen neuronaler Netze zu untersuchen. Einer dieser Gründe ist, daß viele der Algorithmen einer Langevin oder Glauber Dynamik über einer Energielandschaft entsprechen. Ein anderer Grund für die Anwendung statistischer Mechanik ist, daß das Lernen bei neuronalen Netzen ursprünglich ein Problem statistischer Folgerung ist, was in demselben mathematischem Rahmen passt.
Bei der Anwendung des Replica Tricks, des Replica symmetrischen Ansatzes, und der Sattelpunktmethode haben wir die Sattelpunktgleichungen, die das Verhalten des Schülers im thermodynamischen Limes beschreiben, abgeleitet.
Die Einschränkung der Gewichte des Schülers auf diskrete Werte macht die von dem Lehrer produzierte Regel nicht perfekt erlernbar. Dies bedeutet, daß es keinen Schüler mit diskreten Werten gibt, der diese Regeln perfekt erlernen kann. Im Gegensatz dazu kann ein Schüler mit kontinuierlichen Gewichten, wenn er mit einem entsprechend

gutem Trainingsalgorithmus trainiert wurde, nachdem man ihm eine ensprechend große Anzahl von Trainingsmustern präsentiert hat, die vom Lehrer produzierte Regel perfect lernen. Bei der Untersuchung der Learningskurven (Generalisierung- und Trainingskurve) haben wir festgestellt, daß kein diskontinuierlicher Übergang zum Perfekten Lernen stattfinden kann, wie es der Fall ist wenn sowohl Lehrer als auch Schüler nur diskrete Gewichte annehmen dürfen. Nach der Lösung der Sattelpunktgleichungen haben wir festgestellt, daß für eine konstante Temperatur der Generalisierungsfehler eine monoton fallende Funktion der Anzahl der Trainingsmuster ist. Der Generalisierungsfehler geht in diesem Fall wie $\frac{1}{\sqrt{\alpha}}$ asymptotisch gegen ein von der Temperatur und der Bittiefe abhängiges Optimum $\epsilon_g^{opt}(T, L_w)$. Ein analoges Verhalten haben wir für die Abhängigkeit des Generalisierungsfehlers von T bei konstanten Muster Anzahl beobachtet. In diesem Fall geht der Generalisierungsfehler wie $\sqrt{T}$ asymptotisch gegen $\epsilon_g^{opt}(\alpha, L_w)$. $\epsilon_g^{opt}(\alpha, L_w)$ hat, im Gegensatz zum Fall eines Kontinuierlichen Schülers, einen von Null verschieden limes $\epsilon_g^{opt}(L_w)$, wenn $\alpha \to \infty$. $\epsilon_g^{opt}(L_w)$ ist auch eine monoton fallende Funktion von L_w. $\epsilon_g^{opt}(L_w)$ gehet wie $0.3033\frac{1}{L_w}$ gegen Null, wenn $L_w \to \infty$. Das ist das wichtigste Resultat dieser Arbeit im Hinblick auf der optischen Imlementierung neuronaler Netze. Dieses Ergebniss zeigt, daß aufgrund der bei höherer Bittiefe großen Anzahl der zum Training des Netzes zur Verfügung stehenden diskreten Gewichten generalisiert das neuronale Netz umso besser, je höher die Präzision gewählt wird. Jedoch werden die Verbesserungen beim Übergang von einer Bit Tiefe zur nächst höheren immer geringfügiger. Der Vergleich zwischen dem Fall von 5 und 8 Bittiefen läßt fast keinen Unterschied erkennen. Demzufolge sollte die durch die optischen Randbedingungen auf 6bit eingeschränkte Präzision das Generalisierungsverhalten des Netzes bei genügend großem Trainingssatz nur wenig einschränken.
Die Wirkung des Stabilitätsparameters κ auf dem Generalisierungsfehler wurde auch untersucht. Die Einführung eines von Null verschiedenen κ kann hilfreich sein, um mögliches Rauschen sowohl an den Gewichten als auch an den Trainingsmustern zu absorbieren. Wir haben, wie erwartet gesehen, daß ein von Null verschiedenes κ constructive auf dem Generalisierungsfehler wirkt. Bei der richtigen Einstellung von κ können wir einen optimalen Generalisierungsfehler für feste α und T bei einer κ^{opt} erreichen. Für $\kappa > \kappa^{opt}$ der fängt Generalisierungsfehler an zu steigen. Für eine feste Bittiefe ist κ^{opt} eine monoton fallende Function des Trainingssatzes α.
Es hat sich gezeigt, daß die Sattelpunkt Gleichungen im bestimmten Bereichen des α, T Raumes singulär werden. Die singularitätsgerenze wird als T_{GD} (Gardner Derrida) Linie bezeichnet. Für $T_n > T_{GD}$ könnten wir mit der benuzten double precision Arithmetik keine numerischen Lösungen mehr finden.
Die Untersuchung der Stabilität der Lösungen der Sattelpunktgleichungen bezüglich kleine Fluktuationen der Ordnungsparameter hat gezeigt, daß Lösungen in der hoch Temperatur Phase stabil sind für alle Bittiefen. Aber sobald die Temeratur einen Wert $T_{AT}(\alpha, L_w)$ erreicht, werden die Lösungen der Sattelpunktgleichungen instabil im Hinblick auf Replica Symmetrie Brechung. Dies deutet auf dem Eintritt einer Spin Gals Phase, die auf einer nicht trivialen Abhängigkeit der Ordnungsparameter von dem Replica Index hindeutet. Im Falle einer 1 Bit synaptischen Tiefe bleiben alle Lösungen

auf der T_{GD}, und T_n instabil auch für grosse α. Im Gegensatz dazu werden Lösungen auf den Linien T_{GD}, und T_n für eine synaptische Tiefe grösser als 1 Bit, stabil für eine Trainingsmenge, die grosser als $\alpha(T, L_w)$ ist. Wir haben festgestellt, daß die Entropie der Lösungen an den Linien T_{AT} und T_n positive ist. Jedoch auf der T_{GD} muss sie negative sein, da an dieser Linie der Lösungsraum auf 0 schrupmft. So die Zero Entropie Linie muss zwischen T_n und T_{GD} sein.
Diese Resultate wurden semiqualitativ mit Simulationen bestätigt. Die Simulationen wurden durchgeführt mit Simulated Annealing als Trainingsalgorithmus. Die funktionale Abhängigkeit des Generalisierungsfehlers von der Trainingssatzgroße wird beim Trainieren mit Simulated Annealing und diskreten Gewichten auch, wie in der Theorie vorausgesagt wurde , durch eine Funktion $\sim 1/\alpha$ approximiert. Dies stimmt mit der für das kontinuierliche Zwei-Perzeptronen Szenario von der Theorie für einige Lernverfahren vorhergesagten $1/\alpha$ Proportionalität für große α überein. Trotz Einschränkung auf diskrete Kopplungen ändert sich also das asymptotische Verhalten des Generalisierungsfehlers nicht. In den Simulationen haben wir Schüler Perceptronen mit $N = 100$, und $N = 200$. Die Erhöhung der Anzahl der Eingangsneuronen führte zu keinen signifikanten Aussagen bzgl. der Skalierung von Lern- und Generalisierungsfehler mit der Netzgröße. Der Untersuchung noch grösserer Netze sind zeitliche und rechentechnische Grenzen gesetzt.
Da der Simulated Annealing Algorithmus sehr rechenaufwendig ist, wurden Monte - Carlo - Simulationen bei T=0 durchgeführt, um abzuschätzen, inwieweit damit ähnliche Ergebnisse erzielt werden können. Bei gleich großem Trainingssatz liegen die Fehler beim Lernen und Generalisieren geringfügig über den mit Simulated Annealing erreichten. In den Monte - Carlo - Simulationen muß aber kein Abkühlungsschema eingehalten werden, so daß sie wesentlich schneller eine diskrete Lösung finden. Kann das Problem also mit einem Simple Perzeptron gelöst werden und ist eine ausreichende Anzahl an Trainingsmstern vorhanden, so wären sie das Mittel der Wahl. Da in den Monte - Carlo - Simulationen nur die Kosten verringernde Schritte akzeptiert werden kann der Algorithmus bei mehrschichtigen Netzen leicht in einem lokalen Minima veharren. Hier erreicht man mit Simulated Annealing sicher bessere Resultate.

10 Appendix

Appendix A: The Replicon Eigenvalue $\lambda_{\mathcal{R}}$ [7]

The onset of RSB is signaled by a change of sign in at least one of the eigenvalues of of the stability matrix. Specifically, the spectrum of the stability matrix, the second derivative matrix of the free energy, should be evaluated at the replica symmetric values of the order parameter and their conjugate parameters. To check the local stability of the RS-solution against RSB-fluctuation, it will be shown that it is enough to check the sign of the so called replicon eigenvalue ($\lambda_{\mathcal{R}}$).
The stability matrix can be represented schematically, in view of in block from as follow

$$\overline{\mathbf{H}} = \begin{bmatrix} \overline{\mathbf{A}} & \overline{\mathbf{X}} \\ \overline{\mathbf{X}}^T & \overline{\mathbf{B}} \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{A}}^{22} & \overline{\mathbf{A}}^{21} & 1\!\!1_d & \overline{\mathbf{0}} \\ \overline{\mathbf{A}}^{21T} & \overline{\mathbf{A}}^{11} & \overline{\mathbf{0}} & \overline{\mathbf{X}}^{11} \\ 1\!\!1_d & \overline{\mathbf{0}} & \overline{\mathbf{B}}^{22} & \overline{\mathbf{B}}^{21} \\ \overline{\mathbf{0}} & \overline{\mathbf{X}}^{11T} & \overline{\mathbf{B}}^{21T} & \overline{\mathbf{B}}^{11} \end{bmatrix}, \qquad d = \frac{1}{2}n(n-1). \tag{10.1}$$

with

$$A^{22}_{ab,cd} \equiv \frac{\partial^2 G}{\partial q_{ab}\partial q_{cd}} \qquad (a<b, c<d = 1,\ldots,n), \tag{10.2}$$

$$B^{22}_{ab,cd} \equiv \frac{\partial^2 G}{\partial \hat{q}_{ab}\partial \hat{q}_{cd}} \qquad (a<b, c<d = 1,\ldots,n), \tag{10.3}$$

$$A^{11}_{a:i,b:j} \equiv \frac{\partial^2 G}{\partial f^i_a \partial f^j_b} \qquad (a,b = 1,\ldots,n; i,j = 1,2), \tag{10.4}$$

$$B^{11}_{a:i,b:j} \equiv \frac{\partial^2 G}{\partial \hat{f}^i_a \partial \hat{f}^j_b} \qquad (a,b = 1,\ldots,n; i,j = 1,2), \tag{10.5}$$

$$X^{11}_{a:i,b:j} \equiv \frac{\partial^2 G}{\partial f^i_a \partial \hat{f}^j_b} \qquad (a,b = 1,\ldots,n; i = 1,2; j = 1,2), \tag{10.6}$$

[7] This appendix follows calculations done by J. van Mourik

$$A^{21}_{ab,c:j} \equiv \frac{\partial^2 G}{\partial q_{ab} \partial f^j_b} \qquad (a < b, c = 1, \ldots, n; i = 1, 2), \tag{10.7}$$

$$B^{21}_{ab,c:i} \equiv \frac{\partial^2 G}{\partial \hat{q}_{ab} \partial \hat{f}^i_c} \qquad (a < b, c = 1, \ldots, n; i = 1, 2), \tag{10.8}$$

where G is given in Eq.(5.34), and f^i_a and $\hat{f}^i_a$ are the order parameters involved in the saddle point integration, with only one replica index (R_a, q_{aa} and $\hat{R}_a$, $\hat{q}_{aa}$).
Here a runs over the replicas, while i labels the *type* of order parameters that we consider (e.g. $\hat{f}^1_a \equiv \hat{R}_a$ and $\hat{f}^2_a \equiv \hat{q}_{aa}$).
In the RS-saddle point all the replicas are totally equivalent. Hence the elements of $\overline{\mathbf{A}}^{22}$ and $\overline{\mathbf{B}}^{22}$ take only three values

$$A^{22}_{ab,cd} = \begin{cases} P & | \quad a = c \quad \text{and} \quad b = d, \\ Q & | \quad b = c, and a \neq d \\ R & | \quad a \neq c, d \quad \text{and} \quad b \neq c, d, \end{cases} \tag{10.9}$$

$$B^{22}_{ab,cd} = \begin{cases} P' & | \quad a = c \quad \text{and} \quad b = d, \\ Q' & | \quad b = c, and a \neq d \\ R' & | \quad a \neq c, d \quad \text{and} \quad b \neq c, d, \end{cases} \tag{10.10}$$

Furthermore we get

$$A^{11}_{a:i,b:j} = \begin{cases} K_{ij} & | \quad a = b, \\ L_{ij} & | \quad a \neq b, \end{cases} \tag{10.11}$$

$$B^{11}_{a:i,b:j} = \begin{cases} K'_{ij} & | \quad a = b, \\ L'_{ij} & | \quad a \neq b, \end{cases} \tag{10.12}$$

$$X^{11}_{a:i,b:j} = \begin{cases} V_{ij} & | \quad a = b, \\ W_{ij} & | \quad a \neq b, \end{cases} \tag{10.13}$$

$$A^{21}_{ab,c:i} = \begin{cases} C_i & | \quad c = a \quad \text{or} \quad c = b, \\ D_i & | \quad c \neq a \quad \text{and} \quad c \neq b, \end{cases} \tag{10.14}$$

$$B^{21}_{ab,c:i} = \begin{cases} C'_i & | \quad c = a \quad \text{or} \quad c = b, \\ D'_i & | \quad c \neq a \quad \text{and} \quad c \neq b, \end{cases} \tag{10.15}$$

We now determine the eigensystems of $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ separately. Following [43], we consider three *classes* of eigenvectors and their corresponding eigenvalues

- eigenvectors invariant under permutations of all replicas,
- eigenvectors invariant under permutations of all replicas but 1,
- eigenvectors invariant under permutations of all replicas but 2.

The order of $\overline{\mathbf{A}}$ is $\frac{1}{2}n(n-1)+2n$. Since the matrix is real symmetric, this is the number of linearly independent eigenvectors to be found.
The eigenvectors $\vec{x}$ of $\overline{\mathbf{A}}$ have the form

$$\vec{x} = \begin{pmatrix} \{\eta_{ab}\} \\ \{\varepsilon_a^1\} \\ \{\varepsilon_a^2\} \end{pmatrix}, \qquad (a < b = 1, \ldots, n), \tag{10.16}$$

where $\{\eta_{ab}\}$ and $\{\varepsilon_a^i\}$ are column vectors with $\frac{1}{2}n(n-1)$ and n elements respectively.

- First we consider normalized vectors $\vec{x}_0^l$ with elements given by:

$$\eta_{ab} = c^0 \qquad \text{and} \qquad \varepsilon_a^i = c^i, \qquad (\forall a < b = 1, \ldots, n). \tag{10.17}$$

These vectors span a (3)-dimensional invariant subspace and therefore yield (3) eigenvectors. For such a vector the eigenvalue equation can be written as

$$c^0(P - \lambda + 2(n-2)Q + \frac{1}{2}(n-2)(n-3)R) + \sum_{i=1}^{2} c^i(2C_i + (n-2)D_i) = 0,$$

$$c^0((n-1)C_j + \frac{1}{2}(n-2)(n-1)D_j) + c^j(K_{jj} - \lambda + (n-1)L_{jj}) \tag{10.18}$$

$$+ \sum_{i \neq j}^{2} c^i(K_{ij} + (n-1)L_{ij}) = 0, \qquad (j = 1, 2).$$

This gives (3) eigenvalues and corresponding eigenvectors

$$\lambda_0^l \to \vec{x_0}^{lT} \equiv (\vec{\eta}_0^{lT}, \vec{\varepsilon}_0^{lT}), \qquad (l = 1, 2, 3). \tag{10.19}$$

With $\vec{x}^T$ we mean now a vector, i.e. the transposed of the column vector $\vec{x}$. Note that since these vectors are invariant under any permutation of the replica indices, they describe fluctuations within RS and not fluctuations towards RSB.

- Next we consider normalized vectors $\vec{x}_1^l$ of the form

$$\begin{aligned} \eta_{ab} &= c^0, \qquad (a \quad \text{or} \quad b = \theta) \quad \text{and} \quad \eta_{ab} = d^0, \qquad (a, b \neq \theta), \\ \varepsilon_a^i &= c^i, \qquad\qquad (a = \theta) \quad \text{and} \quad \varepsilon_a^i = d^i, \qquad (a \neq \theta). \end{aligned} \tag{10.20}$$

these vectors span a $n(3)$-dimensional invariant subspace and therefore yield $n(3)$ eigenvectors, including those already obtained. To ensure orthogonality to the eigenvectors $\vec{x}_0^l$, we take $c^0 = (1 - \frac{1}{2}n)d^0, c^i = (1 - n)d^i$, and the eigenvalue equation can then be rewritten as

$$\begin{aligned} &c^0(P - \lambda + (n-4)Q - (n-3)R) + 2\sum_{i=1}^{2} c^i(C_i - D_i) = 0, \\ &c^0(n-2)(C_j - D_j) + c^j(K_{jj} - \lambda - L_{jj}) \\ &+ \sum_{i \neq j}^{2} c^i(K_{ij} - L_{ij}) = 0, \qquad (j = 1, \ldots, 2). \end{aligned} \tag{10.21}$$

This gives (3) eigenvalues with degeneracy $(n-1)$ and corresponding eigenvectors

$$\lambda_1^l \to \vec{x}_1^{lT}(r) \equiv (\vec{\eta}_1^{lT}(r), \vec{\varepsilon}_1^{lT}(r)), \qquad (l = 1, \ldots, 3), (r = 1, \ldots, n-1). \tag{10.22}$$

- Finally, we take normalized vectors $\vec{x}_2$ of the form

$$\begin{aligned} \eta_{\theta\nu} = c^0, \quad \eta_{\theta a} = \eta_{\nu a} = d^0, (a \neq \theta, \nu), \quad \eta_{ab} = e^0, (a, b \neq \theta, \nu) \\ \varepsilon_a^i = d^i, (a = \theta \quad \text{or} \quad \nu) \qquad \text{and} \qquad \varepsilon_a^i = e^i, (a \neq \theta, \nu). \end{aligned} \tag{10.23}$$

Orthogonality to the eigenvectors already found, imposes the conditions
$d^i = e^i = 0, (i = 1, \ldots, 2)$ and $c^0 = 2 - n)d^0 = \frac{1}{2}(2-n)(3-n)e^0$. The eigenvalue equation can then be rewritten as

$$\lambda_2 = P - 2Q + R, \tag{10.24}$$

with degeneracy $\frac{1}{2}n(n-3)$ and corresponding eigenvectors

$$\lambda_2 \to \vec{x}_2^T(t) \equiv (\vec{\eta}_2^T(t), \vec{0}^T), \qquad (t = 1, \ldots, \frac{1}{2}n(n-3)). \tag{10.25}$$

Combined the eigensystems of $\overline{\mathbf{A}}$ reads

$$\begin{cases} \lambda_0^l \to \vec{x}_0^l & \equiv & \binom{\vec{\eta}_0^l}{\vec{\varepsilon}_0^l}, & (l = 1, \ldots, 3), \\ \lambda_1^l \to \vec{x}_1^l(r) & \equiv & \binom{\vec{\eta}_1^l(r)}{\vec{\varepsilon}_1^l(r)}, & (l = 1, \ldots, 3), (r = 1, \ldots, (n-1)), \\ \lambda_2 \to \vec{x}_2(t) & \equiv & \binom{\vec{\eta}_2(t)}{\vec{0}}, & (t = 1, \ldots, \frac{1}{2}n(n-3)). \end{cases} \tag{10.26}$$

Similarly the order of $\overline{\mathbf{B}}$ is $\frac{1}{2}n(n-1)+2n$ and $\overline{\mathbf{B}}$ can be diagonalized with eigensystem

$$\begin{cases} \rho_0^l \to \vec{y}_0^k & \equiv & \binom{\vec{\chi}_0^k}{\vec{\delta}_0^k}, & (k = 1, \ldots, 3), \\ \rho_1^k \to \vec{y}_1^k(r) & \equiv & \binom{\vec{\chi}_1^k(r)}{\vec{\delta}_1^k(r)}, & (k = 1, \ldots, 3), \quad (r = 1, \ldots, (n-1)), \\ \rho_2 \to \vec{y}_2(t) & \equiv & \binom{\vec{\chi}_2(t)}{\vec{0}}, & (t = 1, \ldots, \frac{1}{2}n(n-3)). \end{cases} \tag{10.27}$$

- Now we note the important fact that in the limit $n \to 0$, the eigenvectors $\vec{x}_1^l(r)(\vec{y}_1^k(r))$, the corresponding eigenvalue equation and thus the $\lambda_1^l(\rho_1^k)$, reduce to $\vec{x}_0^l(\vec{y}_0^k)$ and $\lambda_0^l(\rho_0^k)$. Since these describe fluctuations within RS, the only eigenvalues involved in the stability against RSB are λ_2 and ρ_2.

- A second important fact is that the whole procedure is dependent on the structure of $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ only, and not on the specific values of their elements. So we have

$$\begin{cases} \vec{\eta}_2(t) & \equiv & \vec{\chi}_2(t) & \\ & & & \left(\begin{array}{lcll} t & = & 1, \ldots, \frac{1}{2}n(n-3) & \\ r & = & 1, \ldots, (n-1) & \\ l & = & 1, \ldots, 3, & k = 1, \ldots, 3 \end{array} \right) \\ \vec{\eta}_2(t) & \perp & \vec{\eta}_0^l, \vec{\chi}_0^k, & \\ & & & \\ \vec{\eta}_2(t) & \perp & \vec{\eta}_1^l(r), \vec{\chi}_1^k(r). & \end{cases} \tag{10.28}$$

So far, we have examined $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ separately. Now we will show that in analyzing the full stability matrix $\overline{\mathbf{H}}$, in the limit $n \to 0$ no mixing between the eigenvectors corresponding to fluctuations within RS and those describing RSB-fluctuations, occurs.

The transformation matrices that diagonalize $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ are orthogonal matrices with the eigenvectors as columns

$$\overline{\mathbf{U}}_A \equiv \left[\begin{array}{c|c} \overline{\beta}_2 & \overline{\beta}_{01} \\ \hline \overline{\mathbf{0}} & \overline{\varepsilon}_{01} \end{array}\right] \qquad \overline{\mathbf{U}}_B \equiv \left[\begin{array}{c|c} \overline{\beta}_2 & \overline{\chi}_{01} \\ \hline \overline{\mathbf{0}} & \overline{\delta}_{01} \end{array}\right]. \tag{10.29}$$

We now apply the following transformation matrix on $\overline{\mathbf{H}}$

$$\overline{\mathbf{U}} = \begin{bmatrix} \overline{\mathbf{U}}_A & \overline{\mathbf{0}} \\ \overline{\mathbf{0}} & \overline{\mathbf{U}}_B \end{bmatrix}. \tag{10.30}$$

Using (*B.29*), this gives

$$\overline{\mathbf{U}}^T \overline{\mathbf{H}}\, \overline{\mathbf{U}} = \begin{bmatrix} \lambda_2 \mathbb{1}_c & \overline{\mathbf{0}} & \mathbb{1}_c & \overline{0} \\ \overline{\mathbf{0}} & \overline{\mathbf{D}}_A & \overline{0} & \overline{\mathbf{M}}_{RS} \\ \mathbb{1}_c & \overline{0} & \rho_2 \mathbb{1}_c & \overline{\mathbf{0}} \\ \overline{0} & \overline{\mathbf{M}}_{RS}^T & \overline{\mathbf{0}} & \overline{\mathbf{D}}_B \end{bmatrix}, \qquad (c = \frac{1}{2}n(n-3)), \tag{10.31}$$

where $\overline{\mathbf{D}}_A$ and $\overline{\mathbf{D}}_B$ are diagonal matrices with the eigenvalues λ_0^l, λ_1^l and ρ_0^k, ρ_1^k respectively on the diagonal and $\overline{\mathbf{M}}_{RS}$ is a $n(3) \times n(3)$ dimensional matrix which couples the fluctuations within the RS-subspace of $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$

$$\overline{\mathbf{M}}_{RS} = \overline{\beta}_{01}^T \overline{\chi}_{01} + \overline{\varepsilon}_{01}^T \overline{\mathbf{X}}^{11} \overline{\delta}_{01}. \tag{10.32}$$

We see that the RSB-eigenvalue equation simplifies to

$$(\lambda_2 - \lambda)(\rho_2 - \lambda) - 1 = 0. \tag{10.33}$$

This gives us two $\frac{1}{2}n(n-3)$-fold degenerate eigenvalues

$$\lambda_\pm = \frac{1}{2}(\lambda_2 + \rho_2 \pm \sqrt{(\lambda_2 - \rho_2)^2 + 4}\,). \tag{10.34}$$

Since $\lambda_+ > \lambda_-$, the sign of the product of these two eigenvalues determines the stability against RSB. This is the so-called replicon eigenvalue

$$\lambda_R \equiv \lambda_- \lambda_+ = \lambda_2 \rho_2 - 1. \tag{10.35}$$

remark:
In fact the result for the decoupling of the RS- and RSB-fluctuations is more general. Take

$$\overline{\mathbf{H}} = \begin{bmatrix} \overline{\mathbf{A}} & \overline{\mathbf{X}} \\ \overline{\mathbf{X}}^T & \overline{\mathbf{B}} \end{bmatrix} = \begin{bmatrix} \overline{\mathbf{A}}^{22} & \overline{\mathbf{A}}^{21} & \overline{\mathbf{X}}^{22} & \overline{\mathbf{X}}^{21} \\ \overline{\mathbf{A}}^{21T} & \overline{\mathbf{A}}^{11} & \overline{\mathbf{X}}^{12} & \overline{\mathbf{X}}^{11} \\ \overline{\mathbf{X}}^{22T} & \overline{\mathbf{X}}^{12T} & \overline{\mathbf{B}}^{22} & \overline{\mathbf{B}}^{21} \\ \overline{\mathbf{X}}^{21T} & \overline{\mathbf{X}}^{11T} & \overline{\mathbf{B}}^{21T} & \overline{\mathbf{B}}^{11} \end{bmatrix}, \tag{10.36}$$

with $\overline{\mathbf{A}}^{22}, \overline{\mathbf{A}}^{21}, \overline{\mathbf{A}}^{11}, \overline{B}^{22}, \overline{\mathbf{B}}^{21}, \overline{\mathbf{B}}^{11}$ and $\overline{\mathbf{X}}^{11}$ as in (*B.2-B.8*) and

$$X^{22}_{ab,cd} \equiv \frac{\partial^2 g}{\partial q_{ab} \partial \hat{q}_{cd}} \qquad (a < b, c < d = 1, \ldots, n), \tag{10.37}$$

$$X^{21}_{ab,c:j} \equiv \frac{\partial^2 g}{\partial q_{ab} \partial \hat{f}^i_c} \qquad (a < b, c = 1, \ldots, n; i = 1, \ldots, 2), \tag{10.38}$$

$$X^{12}_{a:i,cd} \equiv \frac{\partial^2 g}{\partial f^i_a \partial \hat{q}_{cd}} \qquad (a, c < d = 1, \ldots, n; i = 1, \ldots, 2). \tag{10.39}$$

Since we have from (*B..29*), that $\overline{\beta}_2^T \overline{\mathbf{X}}^{21} = \overline{0}$ and $\overline{\mathbf{X}}^{12} \overline{\beta}_2 = \overline{0}$, we get

$$\overline{\mathbf{U}}^T \overline{\mathbf{H}}\, \overline{\mathbf{U}} = \begin{bmatrix} \lambda_2 \mathbb{1}_c & \overline{\mathbf{0}} & \overline{\mathbf{M}}_{RSB} & \overline{\mathbf{0}} \\ \overline{\mathbf{0}} & \overline{\mathbf{D}}_A & \overline{0} & \overline{\mathbf{M}}_{RS} \\ \overline{\mathbf{M}}^T_{RSB} & \overline{\mathbf{0}} & \rho_2 \mathbb{1}_c & \overline{\mathbf{0}} \\ \overline{\mathbf{0}} & \overline{\mathbf{M}}^T_{RS} & \overline{\mathbf{0}} & \overline{\mathbf{D}}_B \end{bmatrix}, \qquad (c = \frac{1}{2} n(n-3)), \tag{10.40}$$

with

$$\begin{aligned} \overline{\mathbf{M}}_{RSB} &= \overline{\beta}_2^T \overline{\mathbf{X}}^{22} \overline{\beta}_2, \\ \overline{\mathbf{M}}_{RS} &= \overline{\beta}_{01}^T \overline{\mathbf{X}}^{22} \overline{\chi}_{01} + \overline{\beta}_{01}^T \overline{\mathbf{X}}^{21} \overline{\delta}_{01} + \overline{\varepsilon}_{01}^T \overline{\mathbf{X}}^{12} \overline{\chi}_{01} + \overline{\varepsilon}_{01}^T \overline{\mathbf{X}}^{11} \overline{\delta}_{01}. \end{aligned} \tag{10.41}$$

So the eigenvalues determining the stability against RSB-fluctuations can be obtained from the simplified eigenvalue equation

$$\begin{bmatrix} \lambda_2 \mathbb{1}_c & \overline{\mathbf{M}}_{RSB} \\ \overline{\mathbf{M}}_{RSB} & \rho_2 \mathbb{1}_c \end{bmatrix} - \lambda \mathbb{1}_{2c} = 0, \qquad (c = \frac{1}{2} n(n-3)). \tag{10.42}$$

It is clear that the decoupling of the eigenvalue equations concerning RS- and RSB-fluctuations can easily be generalized to any model with n_2 order parameters with two replica indices and n_1 order parameters with one replica index. Technically the essential points are

- **1)** The eigenvectors (and corresponding eigenvalue) invariant under permutations of all replicas but one, reduce to the fluctuations within RS for $n \to 0$.
- **2)** The columns of matrices of the form

$$X_{ab,c} = \frac{\partial^2 G}{\partial u_{ab} \partial v_c}, \qquad (a < b, c = 1, \ldots, n), \tag{10.43}$$

  are vectors of length $\frac{1}{2}n(n-1)$ and are invariant under permutations of all replicas but one (the column index). Consequently they are orthogonal to the $\vec{\eta}_2(t), (t = 1 \ldots, \frac{1}{2}n(n-3))$ for all n.

Following [44] we can determine the eigensystems of $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ separately for any order of RSB, by considering the same three classes of eigenvectors as before. Unlike in the RS-saddle point, the structure of each of the eigenvectors will be dependent on the explicit values of the elements of $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ respectively and not only on their structure. This will cause a complicated coupling between the fluctuations in both blocks, and an analytic evaluation becomes very soon an infeasible task. However, due to point **2)**, in the calculation of the eigenvalues of the combined system, the fluctuations of the third class still decouple from the fluctuations of the first and the second class. For the models with spherical couplings and no dilution, we are left with only one set of order parameters (q_{ab}), so in principle following [44] one can calculate the stability for any order of RSB.

Bibliography

[1] A.Engel. Modern Physics Letters B, 8(27):1683–1708, 1994.

[2] N.Metropolis A.W.Rosenbluth M.N.Rosenbluth A.H.Teller and E.Teller. Journal Of Chemical Physics, 21(6):1087–1092, 1953.

[3] Christoph Dietrich. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 1995.

[4] Erhard / Frey. Parallele digitale optische Recheneinheiten B.G.Teubner Stuttgart, 1994.

[5] Samarin Goossens, Mittelbach. The LATEX Companion, Addison-Wesley, 1994.

[6] Olaf Hartwig. C Referenz - Handbuch, Sybex, 1988.

[7] H. Horner. Zeitschrift für Physik B - Condensed Matter, 87:371–376, 1992.

[8] H. Horner. Zeitschrift für Physik B - Condensed Matter, 86:291-308, 1992.

[9] H. Horner. Physica A, 200:552-562, 1993.

[10] N.Tishby H.S.Seung, H.Sompolinsky. Physical Review, 45:6056–6091, 1992.

[11] Russell L.Pimmel Hyeonjoong Yoo. Neurocomputing, 6:541–549, 1994.

[12] Richard G. Palmer John Hertz, Anders Krogh. Introduction to the theory of neural computation, Addison-Wesley Publishing Company, 1991.

[13] W. Kinnebrock. Neuronale Netze - Grundlagen, Anwendungen, Beispiele., Oldenbourg, 1994.

[14] Guido Krüger. Programmieren in C: Grundlagen, Konzepte und Übungen. Addison-Wesley, 1995.

[15] Rupert Lange. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 1995.

[16] Lar Kaufmann Matt Welsh. Running LINUX, O'Reilly and Associates INC., 1995.

[17] S.Kirkpatrick C.D.Gelatt,Jr M.P.Vecchi. SCIENCE, 220(4598):671–679, 1983.

[18] Fiesler, Proceedings of the International Congress on Optical Science and Engineering, editors. volume SPIE-1281, 1990.

[19] H.-K. Patel. Zeitschrift für Physik B - Condensed Matter, 91:257–266, 1993.

[20] Michael Biehl Timothy L.H. Watkin, Albrecht Rau. Review Of Modern Physics, 65:499, 1993.

[21] Vallet F., . Europhys. Lett. 8, 747 (1989).

[22] W.Dzwinel. Pattern Recognition, 27(7):949–959, 1994.

[23] W.Dzwinel. In EUFIT '95, 1995.

[24] Andreas Zell. Simulation Neuronaler Netze. Addison-Wesley, 1994.

[25] D.Psaltis and N. Farhat. Optics letters 10(2), Feb 1985.

[26] J.W.Goodman, A.R.Dias, L.M. Woody Optics Letters, 2(1), 1-3,1978.

[27] D.Psaltis, D.Brady, K.Wagner Appl. Opt., 27(9),1752-1759, 1988.

[28] Györgi G. Physical rev. Lett., 64(24):2957-2960.

[29] Györgi G. Tishby N., Statistical theory of learning a rule Preprint.

[30] Patarnello S., Carnevali P. Europhys. Lett., 4(4):503-508, 1987.

[31] Mezard M., Parisi G., and Virasoro, Spin Glass Theory and beyond, World scientific, Singapore.

[32] Opper M., Kinzel W., Kleinz J., and Nehl R. Journal of physics A, 23:L581-L586, 1991.

[33] Solla S. A., Levin E., and Fleisher M. *complex systems 2*, 625.

[34] Hansel D., and Sompolinsky. Europhys. Lett. 11, 687.

[35] Edwars S.F., and Aderson P.W. . Journal of physics F: Metal Phys. 5,965.

[36] Gardner E., and Derrida B. . Journal of physics A: Math. Gen. 22,1983.

[37] Gardner E., . Europhys. Lett. 4, 481, 1987

[38] van Hemmen J.L., and Palmer R.G., . J. Phys. A12, 563(1979).

[39] Binder K. and D. W. Heerman Monte Carlo Simulation in statistical Mechanics, Springer-verlag, Berlin, 1988

[40] Agmon S., . Can. J. Math. 6, 382, (1954)

[41] Mays C. H., . IEEE Trans. Electron Comput. EC-13,465 (1964)

[42] Anlauf J.K., and Biehl M. . Europhys. Lett. 10, 687 (1989)

[43] De Almeida J.R.L. and Thouless D.J., . J. Phys. A11, 983 (1978)

[44] De Domenicis C. and Kondor I., . Phys. Rev. B27, 606 (1983)

[45] Hopfield, J. , Proceedings of the National Academy of sciences. USA, 79.

[46] Kolmogorove, A., Doklady Akademii Nauk USSR (russisch), 114(5):953-956

[47] Hecht-Nielson, R. IEEE International conference on neural networks, Volume 3, pages 11-14.

[48] Fontanari J.F., Phys. Rev. A volume 45 number 12 (1992).

[49] R. Kühn, H. Steffan Z. Phys. B95 249-260 (1994).

[50] J. Shamir, H.J. Caulfield, and R.B. Johnson. Appl. Opt., 28(2), 311-324, 1989.

[51] J. Van Mourik, Phd thesis, Katholieke Universiteit Leuven, (1996).

[52] S. Schwember, Diplomarbeit an der Karl Ruprecht Universität in Heidelberg (1997).

[53] White H.H., N.B. Aldridge, I. Lindsay Optical Eng., 27(1), 1988

[54] White H.H., W.A. Wright Appl. Opt. , 27(1), 331-338, 1988

[55] D.G. Feileson Optical Computing, The MIT Press Cambridge, Massachusetts, London England.

[56] J. Hertz, A Krogh, and R. Palmer Introduction to the theory of Neural Computation. Addison Wesley, Redwood City, CA.

[57] Rosenblatt F., Psych. Rev. 65:386-408. .

Ein Dankeschön

Herrn Prof. Dr. R. Männer möchte ich für die Möglichkeit zur Promotion am Lehrstuhl für Technische Informatik, für seine vielseitige Unterstützung, die weit über die Betreuung der Arbeit hinausging, und für die mir eingeräumte Freiheit, eigene Ideen verfolgen zu können danken. Herrn Priv. Doz. Dr. R. Kühn danke ich für die umfangreiche Betreuung. Sein riesiger Erfahrungsschatz und seine Diskussionsbereitschaft haben wesentlich zum Gelingen dieser Arbeit beigetragen. Herrn Dr. Steffen Noehte möchte ich für die sehr freundliche und vielfältige Unterstützung während der gesamten Arbeit, und für viele hilfreiche Dikussionen danken. Herrn Prof. Dr. H. Horner möchte ich für die vielen fruchtbaren Dikussionen danken. Allen Mitgliedern des Lehrstuhls für Technische Informatik in Mannheim und der theoretischen Physik in Heidelberg möchte ich für die stete Hilfsbereitschaft danken. Aufgrund der freundlichen und hilfsbereiten Atmosphäre an beiden Instituten habe ich mich immer sehr wohl gefühlt. Meinen Eltern möchte ich ganz besonders herzlich danken für ihre unendliche Liebe, und dafür, daß sie keine Opfer gescheut haben für ihre Kinder alle möglichen Tore im Leben zu öffnen. Ganz besonders bedanke ich mich auch bei meinen Geschwistern, die immer in guten wie in schlechten Zeiten die Sonnenstrahlen meiner Tage waren. Der Energy research group der Universität Damaskus gilt mein Dank für die finanzielle Unterstützung der Arbeit. Nicht zu vergessen meine besten Freunde Nahla Hayder, Jumana Younes, Muhammad Iskef, Tamim Asfour, und Ammar Kassis deren wunderbare Freundschaft sehr wertvoll für mich ist. Zu guter letzt möchte ich mich bei allen netten menschen, denen ich in Deutschland begegnet bin, ganz herzlich bedanken.