DISSERTATION
submitted to the
Combined Faculties of the Natural Sciences
and of Mathematics
of the Ruperto Carola University of Heidelberg, Germany
for the degree of
Doctor of Natural Sciences

presented by
Merab Kutsia MSc
born in Tbilisi, Georgia

Oral examination: July 23, 2007

Learning and Generalisation in Neural Networks with Local Preprocessing

Referees: Prof. Dr. Heinz Horner
Prof. Dr. Franz Wegner

Learning and Generalisation in Neural Networks with Local Preprocessing

We study learning and generalisation ability of a specific two-layer feed-forward neural network and compare its properties to that of a simple perceptron. The input patterns are mapped nonlinearly onto a hidden layer, much larger than the input layer, and this mapping is either fixed or may result from an unsupervised learning process. Such preprocessing of initially uncorrelated random patterns results in the correlated patterns in the hidden layer. The hidden-to-output mapping of the network is performed by a simple perceptron, trained using a supervised learning process. We investigate the effects of the correlations on the learning and generalisation properties as opposed to those of a simple perceptron with uncorrelated patterns. As it turns out, this architecture has some advantages over a simple perceptron.

Lernen und Generalisierung in neuronalen Netzen mit lokaler Vorverarbeitung

Wir untersuchen Lern- und Generalisierungsverhalten eines zweischichtigen feed-forward neuronalen Netzes und vergleichen sie mit den Eigenschaften eines einfachen Perzeptrons. Die Eingangsmuster werden nichtlinear in einer Zwischenschicht abgebildet, die viel größer als die Eingangsschicht ist. Diese Abbildung ist entweder vorgegeben oder wird durch einen unüberwachten Lernprozess bestimmt. Derartige Vorverarbeitung ursprünglich unkorrelierter zufälliger Muster erzeugt Korrelationen in der Zwischenschicht. Von der Zwischenschicht werden die Muster durch ein einfaches Perzeptron klassifizert, das mithilfe eines überwachten Lernprozesses trainiert wird. Wir untersuchen den Einfluss der Korrelationen auf das Lern- und Generalisierungsverhalten des Netzes und vergleichen die Ergebnisse mit dem Fall eines einfachen Perzeptrons mit unkorrelierten Mustern. Diese Architektur weist einige Vorteile gegenüber einem einfachen Perzeptron.

ვუძღვნი ჩემს ოჯახს

To my family

მეუფეო ზეცათაო, ნუგეშინისმცემელო, სულო ჭეშმარიტებისაო, რომელი ყოველგან ხარ და ყოველსავე აღავსებ მადლითა შენითა, საუნჯეო კეთილთაო, მომნიჭებელო ცხოვრებისაო, მოვედ და დაემკვიდრე ჩვენ შორის, და წმიდა მყვენ ჩვენ ყოვლისაგან ბიწისა და აცხოვნენ, სახიერო, სულნი ჩვენნი.

მაკურთხე უფალო და შემწე მეყავ მე, ცოდვილს, აღსრულებასა ზედა ამის საქმისასა სადიდებელად შენდა.

უფალო, იესუ ქრისტე, ძეო მხოლოდშობილო დაუსაბამოისა მამისაო, შენ სთქუ ყოვლადწმიდითა ბაგითა შენითა: ვითარმედ გარეშე ჩემსა არცა ერთი რაი ძალგიძსთ. უფალო ჩემო, უფალო, რწმენით მომიცავს გულსა და სულსა შინა ჩემსა შენ მიერ თქმული, შეუვრდები სახიერებასა შენსა: შემეწიე მე ცოდვილს, რათა საქმეი ესე, ჩემ მიერ დაწყებული განვასრულო შენ მიერ, სახელითა მამისათა და ძისათა და სულისა წმიდისათა, ლოცვითა ყოვლადწმიდისა დედისა შენისათა და ყოველთა წმიდათათა, ამინ.

Contents

Chapter 1

Introduction

The research in the field of *neural networks* was originated with the objective to model the neurons and neural networks in the brain. However, the possibility of constructing artificial neural architectures capable of performing complicated computational tasks has been attracting much more interest towards this area.

From the neurophysiological point of view the models of neural networks are rather simplified and abstract. The complexity of the internal dynamics of single neurons is neglected. The neurons are seen as threshold elements which *fire* if their post synaptic potential reaches or exceeds certain threshold. Still, these models give a good insight into the global collective behaviour of the network.

The post synaptic potential of a given neuron is formed as a weighted sum of signals received from all other neurons. The weighting takes place in synapses and a weight can be either positive or negative corresponding to an excitatory or inhibitory synapse respectively. For a given configuration of weights, thresholds and initial states the dynamics of a neural network is determined.

In Chapter 2 we give a brief introduction to the models of neural networks. Two different architectures can be distinguished: feed-forward nets and networks with feedback loops.

In feed-forward networks the information flow is directed. At their output they perform certain mapping of the patterns fed into their input layer. The mapping can be trained by *learning* with examples. The feed-forward networks can serve as neural classifiers.

Networks with feedback loops, by contrast, consist of neurons with connections in arbitrary directions. These networks are characterised by nontrivial dynamic attractors and are referred to as *associative memory.*

The main objective of the present work is to study learning and generalisation ability of a two-layer feed-forward neural network proposed by Bethge et al. [19] and compare its properties to that of a simple perceptron.

The input patterns are mapped nonlinearly onto a hidden layer, much larger than the input layer, and this mapping is either fixed or may result from an unsupervised learning process. Such preprocessing of initially uncorrelated random patterns results in the correlated patterns in the hidden layer. The hidden-to-output mapping of the network is performed by a simple perceptron, trained using a supervised learning process. Of main interest is to investigate the effects of the correlations on the learning and generalisation properties as opposed to those of a simple perceptron with uncorrelated patterns.

The next chapters contain discussions of the basic components of the network.

Chapter 3 is devoted to the properties of the simplest neural network called the simple perceptron [1]. The learning and generalisation, the ability to extract rules from examples, of the simple perceptron is well understood [1, 2, 3, 4]. Simple and fast algorithms have been designed and their convergence can be proved provided the rule is *learnable* i.e. realizable.

However, the classification ability of the simple perceptron is limited to the so-called *linearly separable* problems [6]. A simple example of tasks, which are unrealisable by a simple perceptron is the Boolean exclusive-OR or XOR function. This limitation can be overcome by combining several simple perceptrons to multilayer feed-forward networks serving as universal classifiers [1, 2, 3, 4]. Such networks are discussed in Chapter 4. Nevertheless, the corresponding fast learning algorithms and convergence proofs are still missing, and the solutions to the classification problems, if any, can be ambiguous. The learning in such systems using the so-called error back-propagation algorithm is usually slow and complex [7, 8].

An interesting detour to the problems associated with multilayer networks provide the so-called Support Vector Machines (SVMs) introduced by Vapnik et al. [10, 11, 12, 13, 14, 15, 16, 17, 18]. The general properties of the SVMs are presented in Chapter 5. The SVMs have several advantages over the common neural architectures. For these systems, it is possible to formulate learning algorithms which iteratively solve a convex optimisation problem having a single solution. On the one hand, the SVMs use the heuristics known from perceptrons and can handle quite complex problems despite the simplicity of their learning dynamics.

A specific learning scenario called *unsupervised learning* is introduced in Chapter 6. In unsupervised learning there is no teacher producing the training examples. Rather, a network is supposed to learn on its own from unsorted examples, provided there is some structure within the distribution of these examples. It extracts the features, regularities, correlations from the example set and codes them at the output. Thus, there is a degree of selforganisation present.

To achieve even better learning properties in multilayer networks the so-called hybrid learning schemes have been investigated [2]. A hybrid network is of SVM kind. The input-to-hidden layer couplings are adjusted using unsupervised learning and then fixed, while the hidden-to-output layer couplings represent a simple perceptron, which is trained using the common learning algorithms. Due to the preprocessing performed by unsupervised learning at the first stage, the input-to-hidden layer becomes adjusted to the input data structure, which in turn increases the overall learning rate of the network.

Bethge et al. proposed a two-layer hybrid network of SVM type [19]. The input-to-hidden layer couplings are fixed after a hypothetical unsupervised learning so that the input data are recoded by representing them locally in terms of mutually exclusive features. This requires a large hidden layer and divergent pathways. One of the advantages of this architecture lies in the comfortable scaling of the number of parameters with the size of the system. In Chapter 7 the evaluation of the capacity of this network performed in [19] is presented. While Chapter 8, the original part of the work, is devoted to the investigation of the generalisation ability of Bethge et al. architecture.

For the calculations performed in this work, the method of replicas, originally designed for the spin glass theory, was exploited. This method was first devised by Kac [20], modified for the analysis of rubber elasticity by Edwards [21] and used to study spin glasses by Edwards and Anderson [22] and Sherrington and Kirkpatrick [23]. The method was applied to neural networks by Amit et al. [24] and reapplied to analyse the space of couplings $\mathbf{J}$ of networks by Gardner [25, 26]. The replica method is beneficial for problems in which it is easier to calculate an average of the partition function Z than that of $\ln Z$.

In the final Chapter the results are analysed and an outlook given.

Chapter 2

Neural Networks

The human brain is composed of nerve cells called *neurons*. A typical neuron (Fig. 2.1) consists of the cell body – *soma*, where the cell nucleus is located; a tree-like structure known as *dendrites* which is connected to the soma; and the *axon* – a long nerve fiber branching in strands and substrands and reaching distant parts of the brain. With the *synaptic junctions*, or *synapses* the axons are connected to the other neurons. A typical neuron has a few thousand synaptic connections with other neurons. The whole brain contains 10^{10} to 10^{11} neurons which are interconnected with more than 10^{14} synaptic junctions, thus forming a *neural network* (Fig. 2.2).

The transmission of a signal from one neuron to another at a synapse involves a complex chemical process. Specific substances are transferred resulting in the rise or fall of the electrical potential inside the body of the receiving neuron. If this potential reaches or exceeds a threshold, the neuron *fires*, i.e. an electrochemical pulse of fixed shape is sent through the axon to the synapses connected with other neurons.

The first steps to the modelling of a neural network was made by Mc-

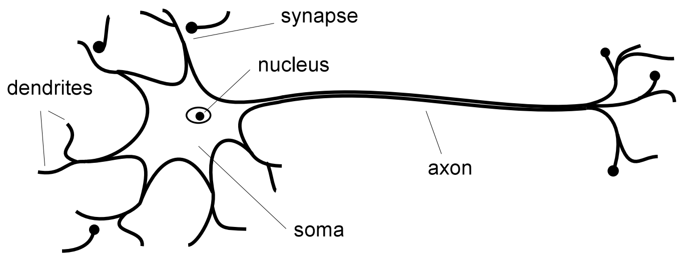

Figure 2.1: Schematic drawing of a neuron.

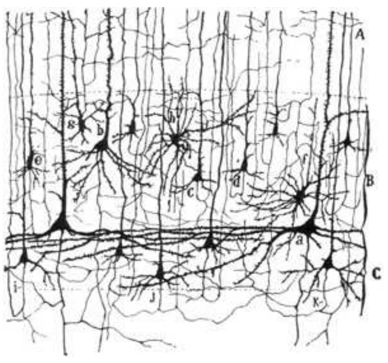

Figure 2.2: Neurons in the cerebral cortex (from Ramón y Cajal (1880)). The drawing is made using Golgi's method of chemical staining.

Culloch and Pitts [32]. They proposed a simple model of a neuron being a binary threshold unit. This unit evaluates the weighted sum of its inputs. At the output we have *one* if the sum is above a threshold and *zero* if it is below (Fig. 2.3):

$$\sigma_i(t+1) = \Theta\left(\sum_i J_{ij}\sigma_j(t) - \theta_i\right). \tag{2.1}$$

Time t is discrete, with one unit per process; $\Theta(x)$ is the unit step function, or Heaviside function:

$$\Theta(x) = \begin{cases} 1, & \text{if} \quad x \geq 0 \\ 0, & \text{if} \quad x < 0. \end{cases} \tag{2.2}$$

The weight or coupling J_{ij} indicates the strength of the synapse from neuron j to neuron i. It is either positive (*excitatory*) or negative (*inhibitory*); ν_i is the threshold of unit i.

Despite the simplicity and the lack of many complicated features of a real neuron, a McCulloch-Pitts unit is a powerful tool for both modelling and computations. Initially, the models of neural networks were used to get the insights of the brain activities. However, the possibility of constructing artificial networks capable of performing complicated computations raised crucially the interest in this field. Very often the word *artificial* is omitted and the terms *neuron*, *neural networks* are used to refer to the non-biological man-made networks.

Another advantage of the networks with McCulloch-Pitts neurons is that one can focus on the *collective* properties of a *large* network. Previous ex-

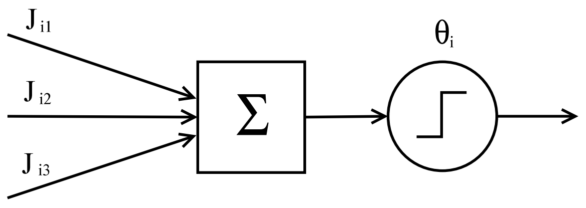

Figure 2.3: Schematic diagram of a McCulloch-Pitts neuron.

perience with complex physical systems such as magnets and liquid crystals has proved that the collective properties are often independent of the microscopic details. Thus, the simplified models for the building blocks are often advantageous for the macroscopic properties of the whole system.

There are several possibilities of building an artificial neural network using the McCulloch-Pitts neurons, characterised by the connectivity graph of the synaptic matrix J_{ij}. Fig. 2.4 shows some examples of networks.

A mathematical analysis is possible only for some architectures, which have extreme kinds of connectivity. Two main types are of special interest. In the first one all neurons are interconnected with each other (Fig. 2.4 a), there are *feed-back loops*. The dynamics is non-trivial and may become chaotic. The scenario becomes simpler for symmetric couplings $J_{ij} = J_{ji}$. In this case an *energy*

$$H(\mathbf{S}) = -\sum_{ij} J_{ij} S_i S_j, \tag{2.3}$$

where $\mathbf{S} = (S_1, ..., S_N)$ denote neuron activities, can be introduced. The dynamics can be chosen such that $H(\mathbf{S})$ never increases. This leads the system to an *attractor state*, which is a local minimum of $H(\mathbf{S})$. By appropriately choosing the couplings J_{ij}, these attractors can even be certain desired neuron configurations $\boldsymbol{\xi^\mu} = (\xi_1^\mu, ..., \xi_N^\mu)$, $\xi_i^\mu = \pm 1$. If the initial configuration $\mathbf{S}$ is close to one of the attractors $\boldsymbol{\xi^\nu}$, the dynamics will tend to this attractor and thus *restore* the pattern $\boldsymbol{\xi^\nu}$. That is why the attractor neural networks can serve as *associative memories* which restore a stored pattern if it is initially given as a noisy or incomplete version of this pattern. The important questions in the statistical mechanics of attractor neural networks include the maximal storage capacity and the properties of learning rules [2, 27, 28, 29].

The other type of the architecture is the *feed-forward* neural network (Fig. 2.4 c). The neurons can be arranged in layers, which are named the input, hidden and output layers. The neurons in each layer are only con-

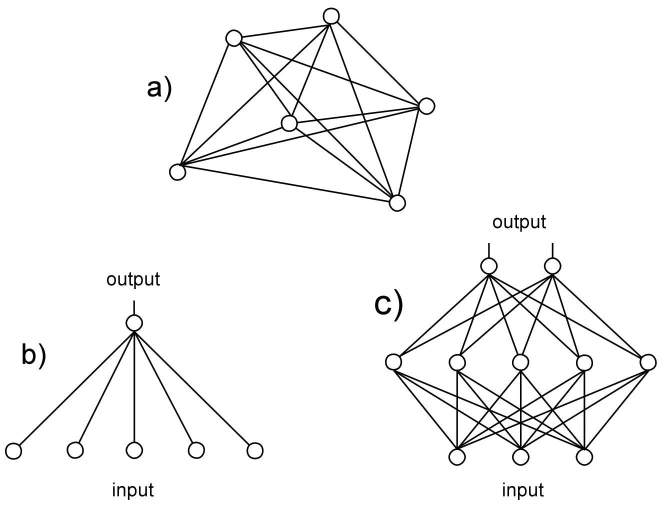

Figure 2.4: Different types of neural networks. a) attractor neural network, b) simple perceptron, c) feed-forward network with one hidden layer.

nected with the two neighbouring ones. In such networks the information flow is directed, which makes its dynamics very simple. The outputs produce a certain *map* or function of the input, i.e. the task is the *classification* of the input vectors into classes determined by the different configurations at the output. The feed-forward networks are very good for learning from examples. The simplest feed-forward network having no hidden layers called *simple perceptron* (Fig. 2.4 b) was introduced by Rosenblatt in 1962 [1]. Due to its simplicity, the perceptron can be analysed completely. In the statistical mechanics of learning the perceptron plays the role of *hydrogen atom*. Although the perceptron is a simple model for some features of the human brain, and it caused the euphoria about the artificial neural networks in the 1960s, there are some obvious limitations too, which forced the consideration of more complicated architectures. The techniques developed for the perceptron were also applied to the more effective *multilayer networks* with one or more hidden layers.

Chapter 3

Simple Perceptron

In this chapter we discuss the properties of the simplest neural network, the *simple perceptron.* The first section concerns the storage capacity of a simple perceptron, and the second sections deals with the generalisation. Several learning rules are discussed in the third section.

A simple perceptron is an element which sums a single N-dimensional layer of given inputs $\boldsymbol{\xi} = (\xi_1, ..., \xi_N)$ with synaptic weights $\mathbf{J} = (J_1, ..., J_N)$ and passes the result through a transfer function (Fig. 3.1):

$$\sigma^\mu = \operatorname{sign}\left(\sum_{i=1}^{N} J_i \xi_i^\mu - \theta\right), \tag{3.1}$$

where θ is a threshold and $\mu \in \{1, 2, ..., p\}$ is the index of the input-output pattern. Note that the Θ function (as in (2.1)) is replaced by sign. This is plausible since the mean activity of random patterns is 0.5 which justifies the ± 1 representation [30]. During the learning process the perceptron is required to adjust its weights $\mathbf{J}$ so that the desired input-output pattern $(\boldsymbol{\xi}^\mu, \sigma_0^\mu)$ can be realized, i.e. $\sigma^\mu = \sigma_0^\mu$. Assuming $\sigma_0^\mu, \xi_i^\mu = \pm 1$, the task of the perceptron is to classify correctly p input patterns into two classes, with the output $\sigma_0^\mu = 1$ and $\sigma_0^\mu = -1$.

For the illustration we choose the case $N = 2$ (Fig. 3.2). The vector $\mathbf{J}$ is perpendicular to the line $\sum_{i=1}^{N} J_i \xi_i^\mu - \theta = 0$. In this arrangement $\sigma^1 = 1$ is the output of the input $\boldsymbol{\xi}^1 = (1, 1)$ (full circle) and $\sigma^\mu = -1$ for the three other inputs (open circles). Thus the line separates the full and the open circles. In case of two full circles at $(-1, -1)$ and $(1, 1)$ (Boolean exclusive-OR or XOR function) no line could separate the circles. Thus a simple perceptron can only realize classifications corresponding to a bisection of $\boldsymbol{\xi}^\mu$-space by a hypersurface, i.e the task should be *linearly separable.* Despite the constraint of the linear separability, a simple perceptron is very important for the theory of neural networks. It is a kind of *hydrogen atom* in this area.

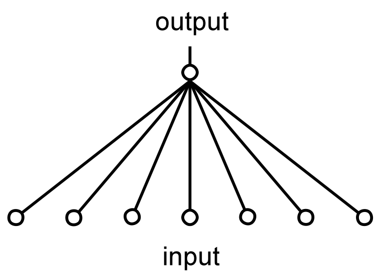

Figure 3.1: Simple perceptron.

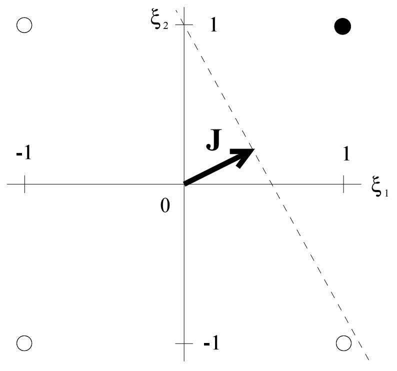

Figure 3.2: Linearly separable task.

3.1 Storage Capacity

Let us find out how many random input-output pairs p can be realized by a simple perceptron. The output generated by the perceptron is given by

$$\sigma^\mu = \text{sign}\left(\frac{1}{\sqrt{N}}\sum_{j=1}^{N} J_j \xi_j^\mu\right). \tag{3.2}$$

Here the threshold θ from (3.2) is set to zero for symmetry, and the factor $1/\sqrt{N}$ ensures correct normalisation for large N. For the couplings $\mathbf{J}$ we consider the spherical constraints

$$\sum_j J_j^2 = N. \tag{3.3}$$

With the presentation of the examples to the simple perceptron, the region in the coupling space correctly reproducing these examples ($\sigma^\mu = \sigma_0^\mu$) shrinks. Beyond a certain number of examples p_{max}, the volume of this region vanishes. The critical value p_{max} is called the *storage capacity* of the perceptron and is known to be $2N$ [33, 34]. This result will be generalised using the tools of spin glass theory (Gardner [25, 26]).

The condition that the μth example is correctly reproduced is given by

$$\Delta_\mu \equiv \frac{\sigma_0^\mu}{\sqrt{N}}\sum_j J_j \xi_j^\mu \geq \kappa > 0, \tag{3.4}$$

where κ is the *stability constant* used to reduce the influence of noise. The volume of the subspace of the $\mathbf{J}$-space satisfying (3.4), the *Gardner volume* is the following expression:

$$V = \frac{1}{V_0}\int \prod_j dJ_j \delta\Big(\sum_j J_j^2 - N\Big)\prod_\mu \Theta(\Delta_\mu - \kappa), \tag{3.5}$$

where

$$V_0 = \int \prod_j dJ_j \delta\Big(\sum_j J_j^2 - N\Big), \tag{3.6}$$

and $\Theta(x)$ is a step function.

The Gardner volume is a random quantity due to the randomness of the input-output patterns. Its typical value can be obtained from the specific *quenched entropy* as in the spin glass theory:

$$s = \frac{\langle \ln V\rangle}{N}, \tag{3.7}$$

where $\langle\cdot\rangle$ means the average over ξ_i^μ and σ_0^μ. In order to calculate the averages over the examples we apply *the replica method*

$$\langle \ln V \rangle = \lim_{n\to 0} \frac{\langle V^n \rangle - 1}{n}, \tag{3.8}$$

which can be reformulated as

$$\langle \ln V \rangle = \lim_{n\to 0} \frac{\ln\langle V^n \rangle}{n}. \tag{3.9}$$

We introduce the replicated couplings J_j^α, and a spin glass *order parameter* in the space of couplings

$$q_{\alpha\beta} = \frac{1}{N}\mathbf{J}^\alpha \cdot \mathbf{J}^\beta, \tag{3.10}$$

where α is the replica index. For V^n we write

$$V^n = V_0^{-n} \int \prod_{j\alpha} dJ_j^\alpha \delta\Big(\sum_j J_j^{\alpha 2} - N\Big) \prod_{\alpha\mu} \Theta\Big(\frac{\sigma_0^\mu}{\sqrt{N}} \sum_j J_j^\alpha \xi_j^\mu - \kappa\Big). \tag{3.11}$$

The averaging $\langle\cdot\rangle$ involves only the part with Θ functions

$$\begin{aligned} \langle V^n \rangle &= V_0^{-n} \int \prod_{j\alpha} dJ_j^\alpha \delta\Big(\sum_j J_j^{\alpha 2} - N\Big) \\ &\times \Big\langle \prod_{\alpha\mu} \Theta\Big(\frac{\sigma_0^\mu}{\sqrt{N}} \sum_j J_j^\alpha \xi_j^\mu - \kappa\Big)\Big\rangle. \end{aligned} \tag{3.12}$$

Using the order parameter $q_{\alpha\beta}$ the previous expression can be rewritten as

$$\begin{aligned} \langle V^n \rangle &= V_0^{-n} \int \prod_{\alpha<\beta} dq_{\alpha\beta} \delta\Big(\sum_j J_j^\alpha J_j^\beta - N q_{\alpha\beta}\Big) \int \prod_{j\alpha} dJ_j^\alpha \delta\Big(\sum_j J_j^{\alpha 2} - N\Big) \\ &\times \Big\langle \prod_{\alpha\mu} \Theta\Big(\frac{\sigma_0^\mu}{\sqrt{N}} \sum_j J_j^\alpha \xi_j^\mu - \kappa\Big)\Big\rangle. \end{aligned} \tag{3.13}$$

In order to evaluate (3.13) conveniently, we split the right-hand side of the equation into two parts. The first part, which includes the δ functions, does not depend on the inputs $\boldsymbol{\xi}^\mu$ and will be denoted as I_1. The second part containing the brackets $\langle\cdot\rangle$ depends on the inputs and will be known as I_2. Thus, we get

$$\langle V^n \rangle = I_1 \cdot I_2, \tag{3.14}$$

where

$$\begin{aligned} I_1 &= V_0^{-n} \int \prod_{\alpha<\beta} dq_{\alpha\beta} \delta\Big(\sum_j J_j^\alpha J_j^\beta - N q_{\alpha\beta} \Big) \\ &\quad \times \int \prod_{j\alpha} dJ_j^\alpha \delta\Big(\sum_j J_j^{\alpha 2} - N \Big) \end{aligned} \tag{3.15}$$

and

$$I_2 = \left\langle \prod_{\alpha\mu} \Theta\Big(\frac{\sigma_0^\mu}{\sqrt{N}} \sum_j J_j^\alpha \xi_j^\mu - \kappa \Big) \right\rangle . \tag{3.16}$$

The specific entropy, using the replica identity, I_1 and I_2 can be expressed as

$$s = \lim_{n\to 0} \frac{\ln\langle V^n \rangle}{Nn} = g_1 + g_2, \tag{3.17}$$

where

$$g_1 = \lim_{n\to 0} \frac{\ln I_1}{Nn} \quad \text{and} \quad g_2 = \lim_{n\to 0} \frac{\ln I_2}{Nn}. \tag{3.18}$$

To calculate I_1, we use the Fourier representation of the δ function

$$\delta(x - x_0) = \int \frac{d\hat{x}}{2\pi} \exp(i\hat{x}(x - x_0)) \tag{3.19}$$

by introducing two conjugate variables $\hat{q}_{\alpha\beta}$, $\hat{N}_\alpha$, and omit the constant factors:

$$\begin{aligned} I_1 &= \int \prod_{\alpha<\beta} dq_{\alpha\beta} d\hat{q}_{\alpha\beta} \int \prod_\alpha d\hat{N}_\alpha \int \prod_{\alpha j} dJ_j^\alpha \\ &\quad \times \exp\left\{ i \left(\sum_{\alpha j} \hat{N}_\alpha J_j^{\alpha 2} + \sum_{\alpha<\beta, j} \hat{q}_{\alpha\beta} J_j^\alpha J_j^\beta \right) \right\} \\ &\quad \times \exp\left\{ -iN \left(\sum_{\alpha<\beta} \hat{q}_{\alpha\beta} q_{\alpha\beta} + \sum_\alpha \hat{N}_\alpha \right) \right\} \end{aligned} \tag{3.20}$$

In order to evaluate I_2, we use the integral representation of the Θ function:

$$\Theta(y - \kappa) = \int_\kappa^\infty \frac{d\lambda}{2\pi} \int_{-\infty}^\infty dx \exp(ix(\lambda - y)), \tag{3.21}$$

perform the average $\langle \cdot \rangle$ over $\boldsymbol{\xi}^\mu$ and apply $\ln \cos x \approx -x^2/2$ for small x:

$$\begin{aligned}(I_2)^{\frac{1}{\alpha N}} &= \int \prod_\alpha dx^\alpha \int_\kappa^\infty \prod_\alpha d\lambda^\alpha \\ &\times \exp\left(i\sum_\alpha x^\alpha \lambda^\alpha - \frac{1}{2}\sum_\alpha x^{\alpha 2} - \sum_{\alpha<\beta} q_{\alpha\beta} x^\alpha x^\beta \right), \quad (3.22)\end{aligned}$$

where $\alpha = p/N$.

We omit the integrals over the parameters $q_{\alpha\beta}$, $\hat{q}_{\alpha\beta}$, $\hat{N}_\alpha$ in I_1 in the anticipation of the use of the steepest descent method. Further, we assume the replica symmetry (RS)

$$q_{\alpha\beta} = q, \quad \hat{q}_{\alpha\beta} = \hat{q}, \quad \hat{N}_\alpha = \hat{N}, \tag{3.23}$$

and transform the Gaussian integrals. This yields for g_1

$$g_1 = -\frac{1}{2}\ln\left(\hat{N} - \frac{\hat{q}}{2}\right) - \frac{\hat{q}}{4\hat{N} - 2\hat{q}} - i\hat{N} + \frac{i\hat{q}q}{2}. \tag{3.24}$$

The evaluation of I_2 under the assumption of RS gives the following expression for g_2

$$g_2 = \alpha \int Dy \ln H(u), \tag{3.25}$$

where

$$Dt = \frac{dt}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) \tag{3.26}$$

is the Gaussian measure;

$$H(x) = \int_x^\infty Dt \tag{3.27}$$

and

$$u = \frac{\kappa + y\sqrt{q}}{\sqrt{1-q}}. \tag{3.28}$$

The method of steepest descent can now be applied. We extremise the specific entropy $s = g_1 + g_2$ with respect to $\hat{N}$, $\hat{q}$, q and get the following saddle-point equations

$$\begin{aligned} i &= -\frac{1}{2\hat{N} - \hat{q}} + \frac{\hat{q}}{(2\hat{N} - \hat{q})^2} && (3.29) \\ \frac{iq}{2} &= -\frac{1}{4\hat{N} - 2\hat{q}} + \frac{\hat{N}}{(2\hat{N} - \hat{q})^2} && (3.30) \\ \frac{i\hat{q}}{2} &= -\frac{\alpha}{2}\int Dy \frac{\exp(-u^2/2)}{\sqrt{2\pi}H(u)}\left(\frac{u}{1-q} + \frac{y}{\sqrt{q(1-q)}}\right) && (3.31)\end{aligned}$$

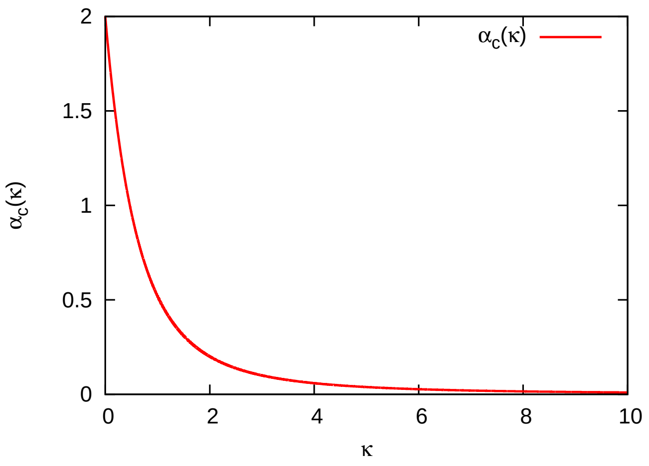

Figure 3.3: Capacity of a simple perceptron.

The two conjugate variables $\hat{N}$ and $\hat{q}$ can be eliminated algebraically after expressing them in terms of q

$$\hat{N} = \frac{i(1-2q)}{2(1-q)^2}, \quad \hat{q} = -\frac{iq}{(1-q)^2}. \tag{3.32}$$

In the remaining equation we put $q \to 1$. This is plausible since the Gardner volume shrinks with the increasing numbers of patterns. This equation has the form

$$\frac{1}{2(1-q)^2} = \frac{\alpha}{2(1-q)^2} \int_{-\kappa}^{\infty} Dy(\kappa + y)^2. \tag{3.33}$$

The final result for the capacity of the perceptron α_c is

$$\alpha_c(\kappa) = \frac{1}{\int_{-\kappa}^{\infty} Dy(\kappa + y)^2}. \tag{3.34}$$

In the limit $\kappa \to 0$, $\alpha_c(0) = 2$, which is the aforementioned capacity $2N$. The function $\alpha_c(\kappa)$ is monotonically decreasing as shown in Fig. 3.3.

3.2 Generalisation

The previous section dealt with the storage capacity of a simple perceptron. The input-output patterns were randomly chosen. In case of generalisation the problem is different. Assume the input-output patterns are generated

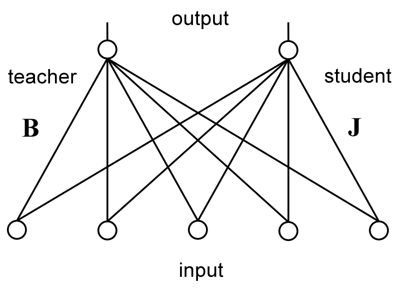

Figure 3.4: Teacher and student perceptrons.

according to some rule. A network can be trained using some of those patterns. The question is now, if the network can produce the correct output for an input pattern which has not been presented before, and how this ability to generalise depends on the number of examples used for training. In particular, the rule can be presented by the output of a *teacher* perceptron with fixed couplings. The perceptron called *student* is supposed to imitate the teacher. This is achieved by *learning.* The teacher perceptron generates input-output examples which are presented to the student. The learning process should achieve that the couplings of the student are adjusted gradually so that it classifies like the teacher. In this way the probability of the correct answers increases. The crucial task of the theory of learning is to evaluate the relation between the number of patterns and the expected error rate for a given learning algorithm.

Fig. 3.4 shows two perceptrons: A teacher perceptron with the fixed couplings $\mathbf{B} = (B_1, ..., B_N)$ and a student with $\mathbf{J} = (J_1, ..., J_N)$. The two perceptrons have the same input patterns $\boldsymbol{\xi}^\mu = (\xi_1^\mu, ..., \xi_N^\mu)$. The outputs σ_0^μ, σ^μ are different in general. For each input $\boldsymbol{\xi}^\mu$ the student, not knowing the couplings of the teacher, compares its output σ^μ to that of the teacher σ_0^μ. If they are different $(\sigma^\mu \neq \sigma_0^\mu)$, the couplings of the student are updated using a certain learning rule. This learning process is called *supervised learning.*

The input patterns $\boldsymbol{\xi}^\mu$, where $\mu \in \{1, 2, ..., p\}$, are chosen randomly according to some distribution $d\mu(\boldsymbol{\xi}^\mu)$ in the input space. We discuss the case where the components of the input vector are independent stochastic variables with zero mean and unit variance. The statistical mechanical approach to the problems in neural networks was originated by Gardner [25].

We introduce the *training energy*:

$$E(\mathbf{J}) = \sum_{\mu=1}^{p} \epsilon(\mathbf{J}, \boldsymbol{\xi}^\mu), \tag{3.35}$$

where

$$\epsilon(\mathbf{J}, \boldsymbol{\xi}^\mu) = \Theta(-\sigma^\mu \sigma_0^\mu) \tag{3.36}$$

is the *error function*, the measure of the deviation of the perceptron's output $\sigma^\mu(\mathbf{J}, \boldsymbol{\xi}^\mu)$ from the target output $\sigma_0^\mu(\boldsymbol{\xi}^\mu)$. $\Theta(x)$ is the step function. The training energy characterises the performance of a perceptron on a set of training examples.

In order to measure the performance also on the whole input space, we introduce the *generalisation function.* It is defined as the average error function over the whole input space:

$$\epsilon(\mathbf{J}) = \int d\mu(\boldsymbol{\xi}) \epsilon(\mathbf{J}, \boldsymbol{\xi}) \tag{3.37}$$

It can be shown that for the present arrangement the generalisation function

$$\epsilon(\mathbf{J}) = \frac{1}{\pi} \arccos r = \frac{\varphi}{\pi}, \tag{3.38}$$

where

$$r = \frac{1}{N} \mathbf{J} \cdot \mathbf{B} \tag{3.39}$$

is the overlap between the student and teacher couplings (Fig. 3.5). During the learning $\mathbf{J}$ comes close to $\mathbf{B}$ and r tends to unity correspondingly.

The training can be described by a stochastic dynamics which tends to decrease the training energy, although the energy can also increase when considering the thermal noise. Such description yields a Gibbs distribution in the weight space

$$P(\mathbf{J}) = \frac{1}{Z} \exp(-\beta E(\mathbf{J})), \tag{3.40}$$

with the canonical partition function

$$Z = \int d\mu(\mathbf{J}) \exp(-\beta E(\mathbf{J})), \tag{3.41}$$

where $\beta = 1/T$. T is *temperature*, a parameter characterising noise. $d\mu(\mathbf{J})$ is the constraint in the couplings space. We consider the normalised spherical distribution:

$$d\mu(\mathbf{J}) = \frac{1}{(\sqrt{2\pi e})^N} d\mathbf{J} \delta \left(\sum_{i=1}^{N} J_i^2 - N \right). \tag{3.42}$$

Using the formalism of equilibrium statistical mechanics one calculates the thermal averages with respect to $P(\mathbf{J})$ which we denote by $\langle \cdot \rangle_T$. In the thermodynamic limit ($N \to \infty$) such averages provide information about the

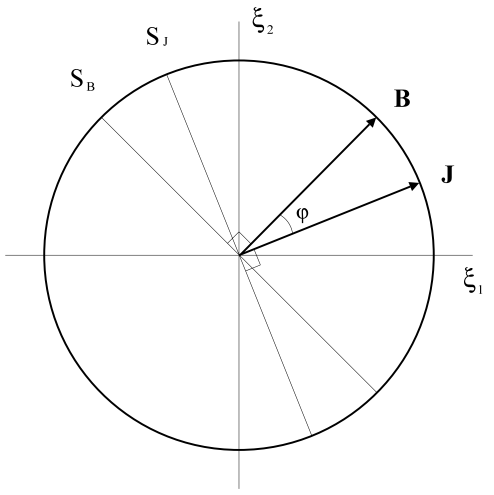

Figure 3.5: The generalisation error is the ratio of the subspace spanned by S_J and S_B to the whole space i.e. $2\varphi/2\pi$, where $\varphi = \arccos r$ is the angle between **J** and **B**.

typical performance of the perceptron independent of the initial conditions of the learning dynamics.

The examples present the *quenched disorder* since they are chosen randomly and then fixed during the learning. Thus, we have to perform also a second *quenched average* over the distribution of example sets denoted by $\langle \cdot \rangle$.

The important parameter, *generalisation error*, is then defined in the following way

$$\epsilon_g = \langle \langle \epsilon(\mathbf{J}) \rangle_T \rangle. \tag{3.43}$$

The generalisation error $\epsilon_g(\beta, p)$ as a function of number of examples p is referred to as the *learning curve*.

The central quantity of this canonical approach is the specific *free energy* of the system:

$$f = -\frac{T\langle \ln Z \rangle}{N}. \tag{3.44}$$

It turns out that it is not necessary to perform the averaging over the quenched variables since each training set would typically yield the same result. This feature is called *self-averaging*. However, performing the averaging is easier than considering specific realizations. The same situation occurs in the analysis of disordered condensed matter systems such as spin glasses. Also, a proper thermodynamic limit requires that the size of the training set scales as $p = \alpha N$ with fixed α.

The average $\langle \ln Z \rangle$ can be evaluated using the replica method

$$\langle \ln Z \rangle = \lim_{n \to 0} \frac{\ln \langle Z^n \rangle}{n} \tag{3.45}$$

In order to perform the replica calculation we introduce the replicated couplings J_j^α the *order parameters*

$$r_\alpha = \frac{1}{N} \mathbf{J}^\alpha \cdot \mathbf{B}, \quad q_{\alpha\beta} = \frac{1}{N} \mathbf{J}^\alpha \cdot \mathbf{J}^\beta. \tag{3.46}$$

r_α measures the overlap of the αth replica perceptron with the teacher, while $q_{\alpha\beta}$ is the overlap of the αth and βth replicas. We also need the parameters

$$X_\mu^\alpha = \frac{1}{\sqrt{N}} \mathbf{J}^\alpha \cdot \boldsymbol{\xi}_\mu, \quad Y_\mu = \frac{1}{\sqrt{N}} \mathbf{B} \cdot \boldsymbol{\xi}_\mu. \tag{3.47}$$

The partition function acquires the form (we omit the constant factors):

$$Z = \int \prod_j dJ_j \delta\Big(\sum_j J_j^2 - N \Big) \prod_\mu \exp\Big(-\beta \Theta(-X_\mu Y_\mu) \Big). \tag{3.48}$$

After the replication

$$Z^n = \int \prod_{j\alpha} dJ_j^\alpha \delta\Big(\sum_j J_j^{\alpha 2} - N \Big) \prod_{\alpha\mu} \exp\Big(-\beta \Theta(-X_\mu^\alpha Y_\mu) \Big). \tag{3.49}$$

The averaging over the inputs $\langle \cdot \rangle$ affects only the part of Z^n which contains the exponential functions:

$$\langle Z^n \rangle = \int \prod_{j\alpha} dJ_j^\alpha \delta\Big(\sum_j J_j^{\alpha 2} - N \Big) \left\langle \prod_{\alpha\mu} \exp\Big(-\beta \Theta(-X_\mu^\alpha Y_\mu) \Big) \right\rangle. \tag{3.50}$$

Using the order parameters r_α and $q_{\alpha\beta}$ the previous expression can be rewritten as

$$\begin{aligned} \langle Z^n \rangle \quad = \quad & \int \prod_\alpha dr_\alpha \delta\Big(\sum_j B_j J_j^\alpha - N r_\alpha \Big) \int \prod_{\alpha<\beta} dq_{\alpha\beta} \delta\Big(\sum_j J_j^\alpha J_j^\beta - N q_{\alpha\beta} \Big) \\ & \times \int \prod_{j\alpha} dJ_j^\alpha \delta\Big(\sum_j J_j^{\alpha 2} - N \Big) \\ & \times \left\langle \prod_{\alpha\mu} \exp\Big(-\beta \Theta(-X_\mu^\alpha Y_\mu) \Big) \right\rangle. \end{aligned} \tag{3.51}$$

In order to calculate (3.51) comfortably, we split the right-hand side of the equation into two parts. The first part with the δ functions is independent of the inputs $\boldsymbol{\xi}^\mu$ and will be known as I_1. The second part in the brackets $\langle\cdot\rangle$ depends on the input and will be denoted I_2. Thus, we get

$$\langle Z^n\rangle = I_1 \cdot I_2, \tag{3.52}$$

where

$$\begin{aligned} I_1 &= \int \prod_\alpha dr_\alpha \delta\Big(\sum_j B_j J_j^\alpha - N r_\alpha\Big) \int \prod_{\alpha<\beta} dq_{\alpha\beta}\delta\Big(\sum_j J_j^\alpha J_j^\beta - N q_{\alpha\beta}\Big) \\ &\quad \times \int \prod_{j\alpha} dJ_j^\alpha \delta\Big(\sum_j J_j^{\alpha 2} - N\Big) \end{aligned} \tag{3.53}$$

and

$$I_2 = \left\langle \prod_{\alpha\mu} \exp\Big(- \beta\Theta(-X_\mu^\alpha Y_\mu)\Big)\right\rangle. \tag{3.54}$$

The specific free energy, using the replica identity, I_1 and I_2 obtains the form

$$-\beta f = \lim_{n\to 0} \frac{\ln\langle Z^n\rangle}{Nn} = G_1 + G_2, \tag{3.55}$$

where

$$G_1 = \lim_{n\to 0}\frac{\ln I_1}{nN} \quad \text{and} \quad G_2 = \lim_{n\to 0}\frac{\ln I_2}{nN}. \tag{3.56}$$

First we calculate I_1. We use the integral representation of the δ functions by introducing three conjugate variables $\hat r_\alpha$, $\hat q_{\alpha\beta}$, $\hat N_\alpha$, and omit the constant factors:

$$\begin{aligned} I_1 &= \int \prod_\alpha dr_\alpha d\hat r_\alpha \int \prod_{\alpha<\beta} dq_{\alpha\beta} d\hat q_{\alpha\beta} \int \prod_\alpha d\hat N_\alpha \int \prod_{\alpha j} dJ_j^\alpha \\ &\quad \times \exp\left\{ i \left(\sum_{\alpha j} \hat N_\alpha J_j^{\alpha 2} + \sum_{\alpha<\beta, j} \hat q_{\alpha\beta} J_j^\alpha J_j^\beta + \sum_{\alpha j} \hat r_\alpha B_j J_j^\alpha \right)\right\} \\ &\quad \times \exp\left\{ -iN \left(\sum_\alpha \hat r_\alpha r_\alpha + \sum_{\alpha<\beta} \hat q_{\alpha\beta} q_{\alpha\beta} + \sum_\alpha \hat N_\alpha \right)\right\} \end{aligned} \tag{3.57}$$

We omit the integrals over the parameters $r_\alpha, q_{\alpha\beta}, \hat r_\alpha, \hat q_{\alpha\beta}, \hat N_\alpha$ in I_1 in the anticipation of the use of the steepest descent method; use the RS ansatz:

$$r_\alpha = r, \quad q_{\alpha\beta} = q, \tag{3.58}$$

$$\hat{r}_\alpha = \hat{r}, \quad \hat{q}_{\alpha\beta} = \hat{q}, \quad \hat{N}_\alpha = \hat{N}; \tag{3.59}$$

and transform the Gaussian integrals. For the G_1 we get

$$G_1 = -\frac{1}{2}\ln\left(\hat{N} - \frac{\hat{q}}{2}\right) - \frac{\hat{q} + i\hat{r}^2}{4\hat{N} - 2\hat{q}} - i\hat{r}r + \frac{i\hat{q}q}{2} - i\hat{N}. \tag{3.60}$$

The evaluation of I_2 yields for G_2

$$\begin{aligned} G_2 &= 2\alpha \int Dt \int_{-rt/\sqrt{q-r^2}}^{\infty} Dz_0 \\ &\times \ln\left\{\exp(-\beta) + (1 - \exp(-\beta)) \int_{-t\sqrt{q/(1-q)}}^{\infty} Dz\right\}. \end{aligned} \tag{3.61}$$

From now on we consider the noise-free case ($T = 0$). The calculations for finite temperatures has been performed in [31]. The G_2 simplifies by setting $\beta \to \infty$:

$$G_2 = 2\alpha \int Dt H\left(-\frac{tr}{\sqrt{q - r^2}}\right) \ln H\left(-t\sqrt{\frac{q}{1-q}}\right). \tag{3.62}$$

The method of steepest descent can now be applied. By extremising the specific free energy $-\beta f = G_1 + G_2$ with respect to the order parameters $\hat{r}, \hat{q}, \hat{N}, r, q$ and after some partial integration we can write the following saddle-point equations:

$$r = -\frac{2\hat{r}}{4\hat{N} - 2\hat{q}} \tag{3.63}$$

$$\frac{iq}{2} = -\frac{1}{4\hat{N} - 2\hat{q}} + \frac{4\hat{N}}{(4\hat{N} - 2\hat{q})^2} + \frac{2i\hat{r}^2}{(4\hat{N} - 2\hat{q})^2} \tag{3.64}$$

$$i = -\frac{1}{2\hat{N} - \hat{q}} + \frac{4\hat{q}}{(4\hat{N} - 2\hat{q})^2} + \frac{4i\hat{r}^2}{(4\hat{N} - 2\hat{q})^2} \tag{3.65}$$

$$i\hat{r} = \frac{\alpha\sqrt{q}}{\pi\sqrt{(q - r^2)(1-q)}} \int Dt \frac{\exp\left(-\frac{t^2r^2}{2(q-r^2)}\right)\exp\left(-\frac{t^2q}{2(1-q)}\right)}{H\left(-t\sqrt{\frac{q}{1-q}}\right)} \tag{3.66}$$

$$\frac{i\hat{q}}{2} = \frac{\alpha}{2\pi(1-q)} \int Dt \frac{H\left(-\frac{tr}{\sqrt{q-r^2}}\right)\exp\left(-\frac{t^2q}{1-q}\right)}{H^2\left(-t\sqrt{\frac{q}{1-q}}\right)} \tag{3.67}$$

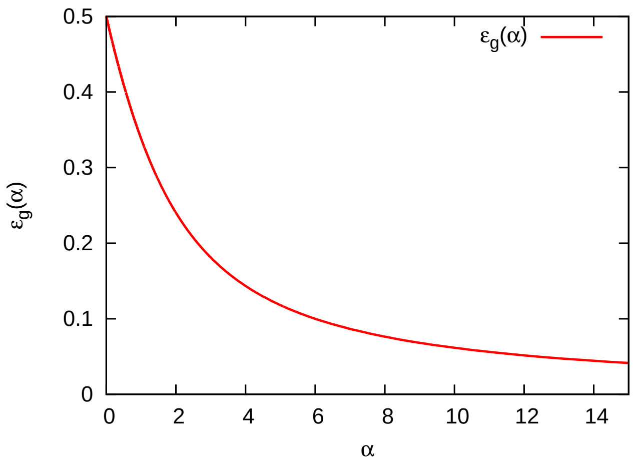

Figure 3.6: Learning curve of a simple perceptron.

After eliminating the three conjugate variables $\hat{r}, \hat{q}, \hat{N}$ algebraically only two equations remain:

$$\frac{r\sqrt{q-r^2}}{\sqrt{q(1-q)}} = \frac{\alpha}{\pi}\int Dt \frac{\exp\left(-\frac{t^2 r^2}{2(q-r^2)}\right)\exp\left(-\frac{t^2 q}{2(1-q)}\right)}{H\left(-t\sqrt{\frac{q}{1-q}}\right)} \tag{3.68}$$

$$\frac{q-r^2}{1-q} = \frac{\alpha}{\pi}\int Dt \frac{H\left(-\frac{tr}{\sqrt{q-r^2}}\right)\exp\left(-\frac{t^2 q}{1-q}\right)}{H^2\left(-t\sqrt{\frac{q}{1-q}}\right)} \tag{3.69}$$

The solution of these equations satisfies $r = q$. For q we get:

$$q = \frac{\alpha}{\pi}\int Dt \frac{\exp\left(-\frac{t^2 q}{1-q}\right)}{H\left(-t\sqrt{\frac{q}{1-q}}\right)}. \tag{3.70}$$

Fig. 3.6 shows the corresponding learning curve $\epsilon_g = \epsilon_g(\alpha)$.

The asymptotic ($\alpha \to \infty$) learning curve is thus given by

$$\epsilon_g(\alpha) \approx \frac{0.625}{\alpha}. \tag{3.71}$$

3.3 Learning Rules

In the previous section we discussed the Gibbs learning rule. This rule is very well suited for the theoretical analysis of the neural networks, however, it is not easy to use for practical problems. There is a whole variety of other direct learning algorithms, the most important of which we are going to mention now.

The oldest learning rule was introduced by Hebb [35]. It goes back to Pavlov's coincidence training. The basic idea is to strengthen the connection of neurons which fire together. As a result the firing of one of them will stimulate the other neuron to fire as well. The Hebb rule for the simple perceptron has the following form. Let again $\mathbf{J}$ and $\mathbf{B}$ be the couplings of a student and a teacher respectively, and $(\boldsymbol{\xi}^\mu, \sigma_0^\mu)$ a training pattern. If the required output

$$\sigma_0^\mu = \operatorname{sign}(\mathbf{B} \cdot \boldsymbol{\xi}^\mu) \tag{3.72}$$

has the same sign as ξ_i^μ, there is a coincidence and the corresponding coupling J_i is strengthened

$$J_i' = J_i + 1. \tag{3.73}$$

If the signs are different in the original Hebb rule there is no change. However, for symmetry the coupling can be weakened

$$J_i' = J_i - 1. \tag{3.74}$$

In vector notation the Hebb rule takes on the form

$$\mathbf{J}' = \mathbf{J} + \boldsymbol{\xi}^\mu \sigma_0^\mu. \tag{3.75}$$

After presenting a set of p examples to the perceptron the normalised student coupling becomes

$$\mathbf{J}^H = \frac{1}{\sqrt{N}} \sum_\mu \boldsymbol{\xi}^\mu \sigma_0^\mu. \tag{3.76}$$

It has been shown (see e.g. [4]) that the generalisation error is given by

$$\epsilon_g = \frac{1}{\pi} \arccos \sqrt{\frac{2\alpha}{2\alpha + \pi}}. \tag{3.77}$$

and considering the asymptotic behaviour

$$\epsilon_g \sim \frac{1}{2} - \frac{\sqrt{2\alpha}}{\pi^{3/2}} \quad \text{for} \quad \alpha \to 0, \tag{3.78}$$

$$\epsilon_g \sim \frac{0.40}{\sqrt{\alpha}} \quad \text{for} \quad \alpha \to \infty, \tag{3.79}$$

with $\alpha = p/N$. As we see, the generalisation error decreases slower than for the Gibbs rule for large α. However, for small values of α the Hebb algorithm yields even better generalisation than the Gibbs rule.

Another important learning algorithm, the so-called *perceptron learning rule* was designed, together with the perceptron itself, by Rosenblatt [1]. This rule is a modification of the Hebb rule. For each training example $\boldsymbol{\xi}^\mu$ which is correctly classified, the coupling $\mathbf{J}$ remains unchanged. Otherwise, the correction is like the one in the Hebb rule:

$$\mathbf{J}' = \begin{cases} \mathbf{J} + \frac{1}{\sqrt{N}}\boldsymbol{\xi}^\mu \sigma_0^\mu, & \text{if} \quad \mathbf{J}\cdot\boldsymbol{\xi}^\mu\sigma_0^\mu < 0 \\ \mathbf{J} & \text{otherwise.} \end{cases} \tag{3.80}$$

By introducing the *embedding strengths* ν^μ the perceptron rule for the whole set of examples can be written as

$$\mathbf{J} = \frac{1}{\sqrt{N}} \sum_\mu \nu^\mu \boldsymbol{\xi}^\mu \sigma_0^\mu, \tag{3.81}$$

where ν^μ is zero for the correctly classified patterns, and unity otherwise.

Chapter 4

Multilayer Networks

In the previous chapter we investigated the properties of the simplest feed-forward architecture, a simple perceptron. As we saw, this network has some limitations, which can be overcome if we add some more hidden layers. Multilayer networks built up of simple perceptrons, each representing linearly separable functions, has been studied by Minsky and Papert [6]. For such networks, there is no general learning algorithm known. However, the situation changes and simplifies if the elementary perceptrons of the network have a smooth, differentiable input-output relation. The so-called *back-propagation* algorithm plays the central role for multilayer networks. It was designed independently several times since the late 1960s (see [2]). The algorithm is based on gradient descent in an error-energy landscape.

We consider a two-layer network (Fig. 4.1). The inputs are denoted by ξ_k, the hidden-layer units by σ_j and the outputs by ν_i. The J_{jk} are the input-to-hidden and W_{ij} hidden-to-output layer couplings. Let $(\boldsymbol{\xi}^\mu, \zeta_i^\mu)$, $\mu = 1, ..., p$ be a set of input-output pairs.

While presenting pattern μ, hidden unit j gets as an input

$$h_j^\mu = \sum_k J_{jk} \xi_k^\mu \tag{4.1}$$

and gives output

$$\sigma_j^\mu = \Phi(h_j^\mu) = \Phi\Big(\sum_k J_{jk} \xi_k^\mu\Big), \tag{4.2}$$

i.e. output unit i receives

$$h_i^\mu = \sum_j W_{ij} \sigma_j^\mu = \sum_j W_{ij} \Phi\Big(\sum_k J_{jk} \xi_k^\mu\Big). \tag{4.3}$$

It produces the final output

$$\nu_i^\mu = \Phi(h_i^\mu) = \Phi\Big(\sum_j W_{ij} \sigma_j^\mu\Big) = \Phi\Big(\sum_j W_{ij} \Phi\Big(\sum_k J_{jk} \xi_k^\mu\Big)\Big). \tag{4.4}$$

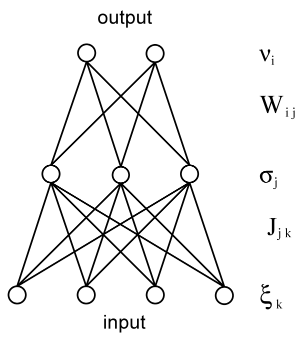

Figure 4.1: A two-layer feed-forward network.

The network error measure or cost function over the set of patterns is defined as

$$E(J,W) = \frac{1}{2}\sum_{\mu i}(\zeta_i^\mu - \nu_i^\mu)^2 \tag{4.5}$$

it has the form

$$E(J,W) = \frac{1}{2}\sum_{\mu i}\left(\zeta_i^\mu - \Phi\Big(\sum_j W_{ij}\Phi\Big(\sum_k J_{jk}\xi_k^\mu\Big)\Big)\right)^2 \tag{4.6}$$

This function is continuous differentiable and it depends on the couplings. So one can use a gradient descent algorithm to learn the right couplings.

We apply the gradient descent rule to the hidden-to-output connections:

$$\Delta W_{ij} = -\eta\frac{\partial E}{\partial W_{ij}} = \eta\sum_\mu(\zeta_i^\mu - \nu_i^\mu)\Phi'(h_i^\mu)\sigma_j^\mu = \eta\sum_\mu \delta_i^\mu\sigma_j^\mu, \tag{4.7}$$

where

$$\delta_i^\mu = \Phi'(h_i^\mu)(\zeta_i^\mu - \nu_i^\mu), \tag{4.8}$$

and η is a learning rate.

For the input-to-hidden connections the chain rule can be used:

$$\Delta J_{jk} = -\eta\frac{\partial E}{\partial J_{jk}} = -\eta\sum_\mu\frac{\partial E}{\partial\sigma_j^\mu}\frac{\partial\sigma_j^\mu}{\partial J_{jk}} \tag{4.9}$$

So we get

$$\begin{aligned}\Delta J_{jk} &= -\eta\sum_{\mu i}(\zeta_i^\mu - \nu_i^\mu)\Phi'(h_i^\mu)W_{ij}\Phi'(h_j^\mu)\xi_k^\mu \\ &= \eta\sum_{\mu i}\delta_i^\mu W_{ij}\Phi'(h_j^\mu)\xi_k^\mu \\ &= \eta\sum_{\mu}\delta_j^\mu\xi_k^\mu, \end{aligned} \tag{4.10}$$

where

$$\delta_j^\mu = \Phi'(h_j^\mu)\sum_i W_{ij}\delta_i^\mu. \tag{4.11}$$

Generalising this result for networks with more layers, the update rule is always of the form

$$\Delta J_{ij} = \eta\sum_\mu \delta_{output}\sigma_{input} \tag{4.12}$$

where *output* is the i end and *input* the j end of the coupling in question; the error δ for a given hidden unit σ_j is determined in terms of the error δs of the next unit ν_i. The coefficients are W_{ij}, which are propagating errors backwards. This feature gives the algorithm its name *error back-propagation* or simply *back-propagation.*

This kind of updating of couplings is biologically implausible. Moreover, the back-propagation algorithm looks for the nearest local minimum in the error-energy landscape and remains there. In general, this algorithm is very slow and requires vast computational resources. Therefore its practical use is restricted to small networks having only a few layers.

An interesting detour to the problems with multilayer networks will be discussed in the next chapter.

Chapter 5

Support Vector Machines

In this chapter we consider the properties of the so-called *Support Vector Machines* (SVMs). Introduced by Vapnik et al. [10, 11], they proved to have advantages over the common multilayer neural networks. For SVMs it is possible to formulate learning algorithms which iteratively solve a convex optimisation problem having a single solution. The SVMs use the heuristics known from perceptrons and can handle quite complex problems despite the simplicity of their learning dynamics.

To explain the idea behind SVMs we can compare it to the simple perceptron. Consider a simple perceptron performing the following classification

$$y = \operatorname{sign}\left(\sum_{i=1}^{N} J_i x_i\right). \tag{5.1}$$

N dimensional vector $\mathbf{x}$ of input features x_i is assigned to either -1 or 1 at the output. The couplings J_i are adjusted so that the perceptron classifies well the patterns of the training set.

As already mentioned, the abilities of a simple perceptron as a classifier are limited to the linearly separable tasks. For such tasks the perceptron learning algorithm discussed in Section 3.3 can always find a coupling vector $\mathbf{J}$, such that an $N-1$ dimensional hyperplane perpendicular to $\mathbf{J}$ separates well the negative and positive examples from each other.

In order to overcome the limitations of perceptrons we could replace the linear input features x_i by some general features $\Psi_\mu(\mathbf{x})$ - fixed nonlinear functions of $\mathbf{x}$. Such modified neural network would give as an output:

$$y = \operatorname{sign}\left(\sum_{\mu=1}^{M} J_\mu \Psi_\mu(\mathbf{x})\right). \tag{5.2}$$

A simple example of the non-linear Ψ_μ is the set of linear x_i and quadratic $x_i x_j$ monomials. Such a network would be able to separate the two classes of the examples by a quadratic hypersurface, which is more flexible than a linear separation done by a perceptron. Still the machine could be trained using the perceptron learning algorithm. However, the dimension of feature space M would increase ($M \sim N^2$) and so would the number of couplings J_μ to be adjusted. It was shown [16, 17], that a good generalisation is achieved, when the number of training example is of the order M, i.e. N^2 which is high.

This situation changes if we look at the perceptron learning rule. Every time an input-output example $(\mathbf{x}^k, y_0^k)$ is presented to the network, the change of coupling J_μ is proportional to $y_0^k \Psi_\mu(\mathbf{x}^k)$, if the student does not classify correctly. Thus, after presenting a set of m examples the coupling vector components will have the form (starting value of $\mathbf{J}$ is 0)

$$J_\mu = \sum_{k=1}^{m} \nu^k y_0^k \Psi_\mu(\mathbf{x}^k). \tag{5.3}$$

The embedding strength ν^k is equal to the number of times the network classified wrongly the kth example (i.e. number of times coupling updating was performed). $\nu^k = 0$ for examples which are correctly classified.

The input vectors $\mathbf{x}^k$ which contribute to the updating of J_μ and thus correspond to the nonzero ν^ks are called the *support-vectors*. The parameters ν^k are more advantageous to consider for learning than the huge number of couplings J_μ

A practical question is how to construct the mappings $\Psi_\mu(\mathbf{x})$ suitable for a particular problem. It turns out that the explicit form of $\Psi_\mu(\mathbf{x})$ is not needed [10]. It is sufficient to know the scalar product of the feature vectors $\Psi_\mu(\mathbf{x})$ and $\Psi_\mu(\mathbf{x}')$ which is given by the so-called *kernel*:

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\Psi}_\mu(\mathbf{x}) \cdot \boldsymbol{\Psi}_\mu(\mathbf{x}') = \sum_{\mu=1}^{M} \Psi_\mu(\mathbf{x}) \Psi_\mu(\mathbf{x}') \tag{5.4}$$

The reason for this situation is the following. During the learning process for the determination of the couplings $\mathbf{J}$ one needs the correlation matrix

$$C_{kl} = \boldsymbol{\Psi}(\mathbf{x}^k) \cdot \boldsymbol{\Psi}(\mathbf{x}^l) y_0^k y_0^l = K(\mathbf{x}^k, \mathbf{x}^l) y_0^k y_0^l. \tag{5.5}$$

Also, in order to determine the classification of a new example $\mathbf{z}$ after the training one has to evaluate

$$y = \operatorname{sign}\left(\sum_{\mu=1}^{M} J_\mu \Psi_\mu(\mathbf{z}) \right) = \operatorname{sign}\left(\sum_{k=1}^{m} \nu^k K(\mathbf{x}^k, \mathbf{z}) y_0^k \right) \tag{5.6}$$

which is determined by the kernel K.

Hence, it is not necessary to deal with the complications associated with mappings $\boldsymbol{\Psi}$. Instead we choose an appropriate kernel K representing e.g. a polynomial decision surface in the input space. It is clear that not every function K can be written as a scalar product, but the mathematical conditions ensuring this decomposition are well known (Mercer theorem). In this manner the feature space can even be infinite dimensional.

Chapter 6

Unsupervised Competitive Learning

In the previous chapters, every time we considered any kind of learning we assumed the existence of a teacher. That is why, we call these learning scenarios *supervised learning.* However, one can admit that learning from examples does not necessarily require the presence of a teacher. Learning from unsorted examples is possible if there is some structure within the distribution of the examples. The main problem of *unsupervised learning* is to extract the features, regularities, correlations from the example set and code them at the output. Thus, the network has to show a degree of *selforganisation.* This problem is encountered in many pattern recognition and data compression tasks.

In the case of *competitive learning* only one output unit is on at a time The output units compete to fire, and are therefore called *winner-take-all* units.

The task of unsupervised competitive learning is to cluster or categorise the input data. Similar inputs should be categorised similarly and fire the same output unit. The network is required to find the classes from the correlations in the input data.

This sort of classification has many practical uses. E.g. it can be used for data encoding and compression through *vector quantisation.* In this task an input data vector is replaced by the index of the output unit, which is fired by this input.

6.1 Simple Competitive Learning

We consider a simple competitive learning network with a single layer of output units σ_i and inputs ξ_j (Fig. 6.1). The input-output connections are excitatory $J_{ij} \geq 0$. From the output units only one unit, the *winner*, can be active at a time. The winner is defined as the unit with the largest net

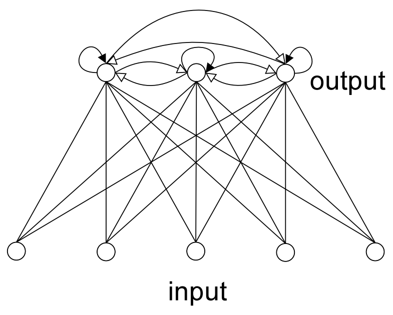

Figure 6.1: A simple competitive learning network. The connections with open arrows are inhibitory.

input

$$h_i = \sum_j J_{ij}. \tag{6.1}$$

Thus

$$\sigma_{i'} = 1, \qquad \text{if} \quad \mathbf{J}_{i'} \cdot \boldsymbol{\xi} \geq \mathbf{J}_i \cdot \boldsymbol{\xi} \quad \text{for all} \quad i, \tag{6.2}$$

the output unit with the label i' is the winner. If the couplings are normalised

$$|\mathbf{J}_i| = 1 \tag{6.3}$$

then the previous inequality is equivalent to

$$|\mathbf{J}_{i'} - \boldsymbol{\xi}| \leq |\mathbf{J}_i - \boldsymbol{\xi}| \quad \text{for all} \quad i. \tag{6.4}$$

Consequently the winner is the unit, the coupling vector $\mathbf{J}_i$ of which is closest to the input vector $\boldsymbol{\xi}$.

One of the possibilities of the implementation of the winner-take-all character is shown in Fig. 6.1. The output units have *lateral inhibition*, i.e. each unit inhibits the others. A self-excitatory connection is also required.

The problem is how to find clusters in the input data and choose the couplings $\mathbf{J}_i$ using these networks. One start with small random values for the couplings. A set of input patterns is presented to the network. For each input the winner output unit i' is found and the corresponding couplings $J_{i'j}$ are updated to make the $\mathbf{J}_{i'}$ closer to the input $\boldsymbol{\xi}^\mu$. This change makes it more likely to win on that input in future. The *standard competitive learning rule* has the form

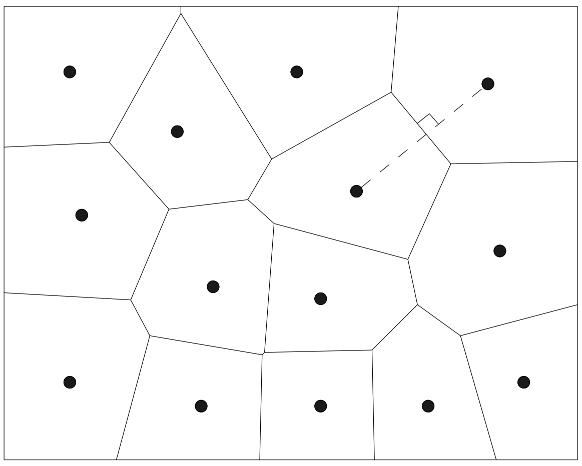

Figure 6.2: Voronoi tessellation.

$$\Delta J_{i'j} = \eta(\xi_j^\mu - J_{i'j}) \tag{6.5}$$

for pre-normalised inputs.

6.2 Vector Quantisation

Vector quantisation for data compression is a very important example of competitive learning. Vector quantisation is used for storage and transmission of speech and image data [2].

The idea behind vector quantisation is simple. The continuous-valued input vectors ξ^μ are to be categorised into M classes defined by a set of *M prototype vectors*. Only the index of the corresponding class can be transmitted or stored, after a set of classes, a *codebook* is defined. The class of a given input can be determined by finding the *nearest* prototype vector using the Euclidian metric. This arrangement divides the input space into a *Voronoi tessellation* (or *Dirichlet tessellation*) (Fig.6.2).

In terms of competitive learning, when an input $\boldsymbol{\xi}^\mu$ is applied, the winning output determines the corresponding class. The couplings J_i are the prototype vectors, and the winner is found using

$$|\mathbf{J}_{i'} - \boldsymbol{\xi}| \leq |\mathbf{J}_i - \boldsymbol{\xi}| \qquad \text{for all} \quad i. \tag{6.6}$$

This is not equivalent to maximising $\mathbf{J}_i \cdot \boldsymbol{\xi}$ unless the couplings are normalised.

An appropriate set of prototype vectors can be found using the standard competitive learning algorithm (6.5). After presenting sample data the couplings change and divide up the input space into Voronoi polyhedra, which contain almost equal numbers of sample points. They represent a discretised map of the input distribution. After training the couplings can be fixed and thus determine a static codebook.

Chapter 7

Bethge Architecture - Storage Capacity

An interesting neural network architecture, a kind of SVM, was introduced by Bethge et al. [19]. In this chapter we discuss the storage capacity of this network. One of the advantages of this architecture lies in the comfortable scaling of the number of parameters with the size of the system. It also fits well the problems where the classification of input patterns is based on local features of the patterns.

7.1 The coding

The input patterns are random. Each of them consists of N' bits $\{\pm 1\}$. The input neurons are divided in m disjoint modules, each containing k input bits ($N' = m \cdot k$) to represent $d = 2^k$ states which have the same probability (Fig. 7.1). These states are further mapped onto the $N = m \cdot d$ neurons in the hidden layer in the following manner. All d states of each module are represented by a pool of d $\{0, 1\}$ neurons in the hidden layer of which only one is active ($= 1$) at a time, thus unambiguously representing the k-bit state of the corresponding module in the input layer. The $(n + 1)$th neuron in a module of the hidden layer is active, when the original module in the input layer shows the binary representation of the number n. Fig. 7.1 shows the $d = 4$ possible states of a module with $k = 2$. The input states encoded in this way are further classified by a common perceptron having the hidden layer of the system as an input. The connections between the input and hidden layer are thus preset, whereas the connections between the hidden layer and the output change during the learning process. The modules arranged in this manner can be regarded as *receptive fields* or *filters* extracting mutually exclusive features. This can be realized using vector quantisation units working in parallel.

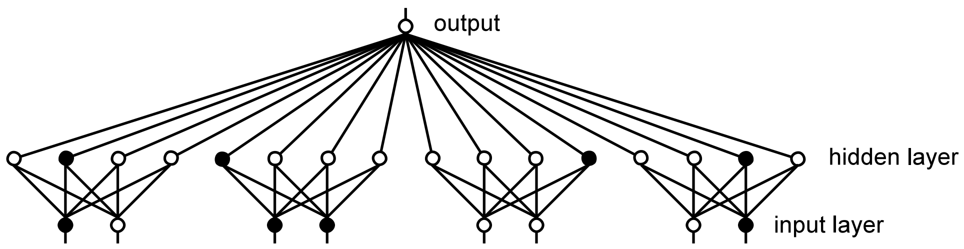

Figure 7.1: The neural network with fixed preprocessing.

7.2 The correlations

The coding presented above leads to the low activity of patterns. Globally and locally within each module in the hidden layer

$$\langle \xi_i^\mu \rangle = \frac{1}{d} = \frac{1}{2^k}, \tag{7.1}$$

where $\xi_i^\mu \in \{0, 1\}$ are the pattern bits represented in the hidden layer, $i \in \{1, 2, ..., N\}$ enumerates the bits in the hidden layer, and μ the patterns of bits. The angular brackets $\langle \cdot \rangle$ denote an average over the distribution of these patterns. The different patterns are correlated,

$$\langle \xi_i^\mu \xi_j^\nu \rangle = \frac{1}{d^2} \qquad \text{for } \mu \neq \nu. \tag{7.2}$$

There is also a spatial correlation within each pattern:

$$\langle \xi_i^\mu \xi_j^\mu \rangle = C_{ij}. \tag{7.3}$$

The C_{ij} are the elements of the correlation matrix C, which, given our mapping, has the following structure:

$$C = \begin{pmatrix} D & A & \ldots & \ldots & A \\ A & D & A & \ldots & \vdots \\ \vdots & A & \ddots & A & \vdots \\ \vdots & \ldots & A & \ddots & A \\ A & \ldots & \ldots & A & D \end{pmatrix} \tag{7.4}$$

with

$$D = \frac{1}{d} I_d \qquad A = \frac{1}{d^2} 1_d, \tag{7.5}$$

where I_d is the d-dimensional unit matrix and 1_d the d-dimensional matrix with 1 everywhere. The matrix C has three different eigenvalues. These are (with degeneracies):

$$\begin{aligned}
\lambda_1 &= \frac{1}{d}\left(1 + N\frac{1}{d} - d\frac{1}{d}\right) = \frac{N}{d^2} && \text{single} \\
\lambda_2 &= \frac{1}{d}\left(1 - d\frac{1}{d}\right) = 0 && \left(\frac{N}{d} - 1\right)\text{-fold} \\
\lambda_3 &= \frac{1}{d} && \left(N - \frac{N}{d}\right)\text{-fold}
\end{aligned} \tag{7.6}$$

7.3 Storage capacity

The goal is to calculate the maximum number of correlated patterns which can be stored in the perceptron. Patterns are regarded as stored, if

$$X_\mu \equiv \zeta^\mu \frac{1}{\sqrt{N}} \sum_j J_j \xi_j^\mu \geq \kappa > 0, \tag{7.7}$$

where κ is the desired stability of the solution and $\zeta^\mu \in \{\pm 1\}$ the output value. The input-output pairs $(\boldsymbol{\xi}^\mu, \zeta^\mu)$ are drawn randomly (there is no teacher). The number of patterns is p, thus $\mu \in \{1, 2, ..., p\}$. These arrangements yield $\langle X_\mu \rangle = 0$. In order to calculate the critical storage capacity the Gardner approach is used [25, 26]. For the couplings $\mathbf{J}$ we consider the spherical constraints $\sum_j J_j^2 = N$. The volume spanned by the couplings which solve (7.7) for a given number of patterns, the Gardner volume, is given by

$$V = \frac{1}{V_0} \int \prod_j dJ_j \delta(\sum_j J_j^2 - N) \prod_\mu \Theta(X_\mu - \kappa), \tag{7.8}$$

where

$$V_0 = \int \prod_j dJ_j \delta(\sum_j J_j^2 - N). \tag{7.9}$$

The specific *quenched entropy* in the sense of the spin glass theory is given by:

$$s = \frac{\langle \ln V \rangle}{N}, \tag{7.10}$$

where the brackets $\langle \cdot \rangle$ denote the averaging over the patterns.

Since $\ln V$ is extensive and thus self-averaging, it is independent of the choice of patterns in the limit $N \to \infty$. The average $\langle \ln V \rangle$ over the patterns can be carried out using the replica identity:

$$\langle \ln V \rangle = \lim_{n \to 0} \frac{\ln \langle V^n \rangle}{n}. \tag{7.11}$$

We introduce the replicated couplings J_i^σ and stabilities X_μ^σ, where σ is the replica index. The averaging over the correlated patterns $\boldsymbol{\xi}^\mu$ can be replaced by the averaging over the X_μ^σ, since V depends on the $\boldsymbol{\xi}^\mu$ only through the X_μ^σ. The moments of their distribution are

$$\langle X_\mu^\sigma \rangle = \langle \zeta^\mu \rangle \frac{1}{\sqrt{N}} \sum_j J_j^\sigma \langle \xi_j^\mu \rangle = 0, \quad \text{since} \quad \langle \zeta^\mu \rangle = 0, \tag{7.12}$$

$$\langle X_\mu^\sigma X_\nu^{\sigma'} \rangle = \delta_{\mu\nu} \frac{1}{N} \sum_{ij} C_{ij} J_i^\sigma J_j^{\sigma'}. \tag{7.13}$$

After diagonalising C

$$\langle X_\mu^\sigma X_\nu^{\sigma'} \rangle = \delta_{\mu\nu} \frac{1}{N} \sum_\gamma \lambda_\gamma J_\gamma^\sigma J_\gamma^{\sigma'} = \delta_{\mu\nu} q_{\sigma\sigma'}, \tag{7.14}$$

where

$$q_{\sigma\sigma'} = \frac{1}{N} \sum_\gamma \lambda_\gamma J_\gamma^\sigma J_\gamma^{\sigma'} \tag{7.15}$$

is the spin glass order parameter. According to the central limit theorem the X_μ^σ are independent Gaussian variables:

$$P(\{\boldsymbol{\xi}^\mu\}_\mu) = \prod_\mu P(\{X_\mu^\sigma\}_\sigma) = \prod_\mu \frac{1}{(\sqrt{2\pi})^n \sqrt{\det \mathbf{q}}} \exp\left(-\frac{1}{2} \mathbf{X} \mathbf{q}^{-1} \mathbf{X} \right). \tag{7.16}$$

Hence, the averaging means

$$\langle \cdot \rangle = \int \cdot \prod_\mu P(\{X_\mu^\sigma\}_\sigma) d\mathbf{X}_\mu. \tag{7.17}$$

For V^n we get

$$V^n = V_0^{-n} \int \prod_{\gamma\sigma} dJ_\gamma^\sigma \delta\Big(\sum_\gamma J_\gamma^{\sigma 2} - N \Big) \prod_{\mu\sigma} \Theta(X_\mu^\sigma - \kappa). \tag{7.18}$$

The averaging $\langle \cdot \rangle$ involves only the part with Θ functions

$$\langle V^n \rangle = V_0^{-n} \int \prod_{\gamma\sigma} dJ_\gamma^\sigma \delta\Big(\sum_\gamma J_\gamma^{\sigma 2} - N \Big) \left\langle \prod_{\mu\sigma} \Theta(X_\mu^\sigma - \kappa) \right\rangle. \tag{7.19}$$

Using the order parameter $q_{\sigma\sigma'}$ we rewrite the previous expression as

$$\begin{aligned}\langle V^n\rangle &= V_0^{-n}\int\prod_{\sigma\le\sigma'} dq_{\sigma\sigma'}\delta\Big(\sum_\gamma \lambda_\gamma J_\gamma^\sigma J_\gamma^{\sigma'} - Nq_{\sigma\sigma'}\Big)\\ &\quad\times\int\prod_{\gamma\sigma} dJ_\gamma^\sigma \delta\Big(\sum_\gamma J_\gamma^{\sigma 2} - N\Big)\left\langle\prod_{\mu\sigma}\Theta(X_\mu^\sigma-\kappa)\right\rangle \qquad (7.20)\end{aligned}$$

To make the evaluation of (7.20) more convenient, its right-hand side is split into two parts. The part I_1, which includes the δ functions, does not depend on the inputs. The second part I_2 containing the brackets $\langle\cdot\rangle$, depends on the inputs. Thus, we have

$$\langle V^n\rangle = I_1\cdot I_2, \qquad (7.21)$$

where

$$\begin{aligned}I_1 &= V_0^{-n}\int\prod_{\sigma\le\sigma'} dq_{\sigma\sigma'}\delta\Big(\sum_\gamma \lambda_\gamma J_\gamma^\sigma J_\gamma^{\sigma'} - Nq_{\sigma\sigma'}\Big)\\ &\quad\times\int\prod_{\gamma\sigma} dJ_\gamma^\sigma \delta\Big(\sum_\gamma J_\gamma^{\sigma 2} - N\Big) \qquad (7.22)\end{aligned}$$

and

$$I_2 = \left\langle\prod_{\mu\sigma}\Theta(X_\mu^\sigma-\kappa)\right\rangle. \qquad (7.23)$$

The specific entropy, using the replica identity, I_1 and I_2 is given by

$$s = \lim_{n\to 0}\frac{\ln\langle V^n\rangle}{Nn} = g_1 + g_2, \qquad (7.24)$$

where

$$g_1 = \lim_{n\to 0}\frac{\ln I_1}{Nn} \quad\text{and}\quad g_2 = \lim_{n\to 0}\frac{\ln I_2}{Nn}. \qquad (7.25)$$

In order to evaluate I_1, we use the integral representation of the δ function by introducing two conjugate variables $\hat{q}_{\sigma\sigma'}$, $\hat{N}_\sigma$, and omit the constant factors

$$\begin{aligned}\langle V^n\rangle &= \int\prod_{\sigma\le\sigma'} dq_{\sigma\sigma'}d\hat{q}_{\sigma\sigma'}\int\prod_\sigma d\hat{N}_\sigma\int\prod_{\gamma\sigma} dJ_\gamma^\sigma\\ &\quad\times\exp\left\{-i\left(\sum_{\sigma\gamma}\hat{N}_\sigma J_\gamma^{\sigma 2} + \sum_{\sigma\le\sigma',\gamma}\hat{q}_{\sigma\sigma'}\lambda_\gamma J_\gamma^\sigma J_\gamma^{\sigma'}\right)\right\}\\ &\quad\times\exp\left\{iN\left(\sum_{\sigma\le\sigma'}\hat{q}_{\sigma\sigma'}q_{\sigma\sigma'} + \sum_\sigma \hat{N}_\sigma\right)\right\} \qquad (7.26)\end{aligned}$$

To calculate I_2, we use the definition of the Θ function

$$I_2 = \left(\int_{\kappa}^{\infty} \prod_{\sigma} dX^{\sigma} P(X^{\sigma}) \right)^{\alpha N}, \tag{7.27}$$

where $\alpha = p/N$.

In the thermodynamic limit ($N \to \infty$) the integral in I_1 over the order parameters is dominated by the saddle point. The method of steeplest descent can be applied. Thus, we omit the integrals over $q_{\sigma\sigma'}, \hat{q}_{\sigma\sigma'}, \hat{N}_{\sigma}$ in I_1. Also, we assume the replica symmetry:

$$q_{\sigma\sigma} = q_0, \quad q_{\sigma\sigma'} = q \quad (\sigma \neq \sigma'), \tag{7.28}$$

$$\hat{q}_{\sigma\sigma} = \hat{q}_0, \quad \hat{q}_{\sigma\sigma'} = \hat{q} \quad (\sigma \neq \sigma'), \quad \hat{N}_{\sigma} = \hat{N}, \tag{7.29}$$

and use the following convention for any function of the eigenvalues in the limit $N \to \infty$:

$$\frac{1}{N} \sum_{\gamma} f(\lambda_{\gamma}) \to \{f(\lambda)\}_{\lambda}. \tag{7.30}$$

For g_1 we get

$$\begin{aligned} g_1 &= iq_0\hat{q}_0 - \frac{iq\hat{q}}{2} + i\hat{N} \\ &\quad - \left\{ \frac{\ln(i[\hat{N} + \lambda(\hat{q}_0 - \frac{\hat{q}}{2})])}{2} \right\}_{\lambda} - \left\{ \frac{\hat{q}\lambda}{4[\hat{N} + \lambda(\hat{q}_0 - \frac{\hat{q}}{2})]} \right\}_{\lambda}. \end{aligned} \tag{7.31}$$

The evaluation of I_2 under the assumption of RS gives the following equation for g_2

$$g_2 = \alpha \int Dz \ln H \left(\frac{\kappa + \sqrt{q} z}{\sqrt{q_0 - q}} \right), \tag{7.32}$$

where $H(x) = \int_x^{\infty} Dt$ with Dt being the Gaussian measure.

The specific entropy $s = g_1 + g_2$ obtains the form:

$$\begin{aligned} s(q_0, q, \hat{q}_0, \hat{q}, \hat{N}) &= \alpha \int Dz \ln H \left(\frac{\kappa + \sqrt{q} z}{\sqrt{q_0 - q}} \right) + iq_0\hat{q}_0 - \frac{iq\hat{q}}{2} + i\hat{N} \\ &\quad - \left\{ \frac{\ln(i[\hat{N} + \lambda(\hat{q}_0 - \frac{\hat{q}}{2})])}{2} \right\}_{\lambda} \\ &\quad - \left\{ \frac{\hat{q}\lambda}{4[\hat{N} + \lambda(\hat{q}_0 - \frac{\hat{q}}{2})]} \right\}_{\lambda}, \end{aligned} \tag{7.33}$$

By extremising the specific entropy with respect to $\hat{q}_0, \hat{q}, \hat{N}, q_0, q$, we get the following saddle-point equations:

$$\begin{aligned}
2i(q_0 - q) &= \left\{ \frac{\lambda}{[\hat{N} + \lambda(\hat{q}_0 - \frac{\hat{q}}{2})]} \right\}_\lambda && (7.34)\\
q &= \left\{ \frac{i\hat{q}\lambda^2}{4[\hat{N} + \lambda(\hat{q}_0 - \frac{\hat{q}}{2})]^2} \right\}_\lambda && (7.35)\\
i &= \left\{ \frac{1}{2[\hat{N} + \lambda(\hat{q}_0 - \frac{\hat{q}}{2})]} \right\}_\lambda - \left\{ \frac{\hat{q}\lambda}{4[\hat{N} + \lambda(\hat{q}_0 - \frac{\hat{q}}{2})]^2} \right\}_\lambda && (7.36)\\
-i\hat{q}_0 &= \frac{\alpha}{2} \int Dz \frac{\exp(-u^2/2)}{\sqrt{2\pi}H(u)} \frac{u}{q_0 - q} && (7.37)\\
-\frac{i\hat{q}}{2} &= \frac{\alpha}{2} \int Dz \frac{\exp(-u^2/2)}{\sqrt{2\pi}H(u)} \left(\frac{u}{q_0 - q} + \frac{z}{\sqrt{q(q_0 - q)}} \right) && (7.38)
\end{aligned}$$

where

$$u = \frac{\kappa + \sqrt{q}z}{\sqrt{q_0 - q}}. \tag{7.39}$$

The three conjugate variables $\hat{q}_0, \hat{q}, \hat{N}$ can be eliminated algebraically. In the remaining two equations we put $q_0 = q = q_c$, where q_c is the so-called Derrida-Gardner limit. The Gardner volume shrinks with the increasing number of patterns, which leads to $q_0 = q$. The two remaining equations with the variables q_c and the critical capacity α_c have the form:

$$\alpha_c(\kappa, \{\lambda\}) = \alpha_p \left(\frac{\kappa}{\sqrt{q_c}} \right) \left\{ \frac{\lambda q_c}{[\alpha_p(\frac{\kappa}{\sqrt{q_c}})H(-\frac{\kappa}{\sqrt{q_c}})(\lambda - q_c) + q_c]^2} \right\}_\lambda \tag{7.40}$$

$$\alpha_c(\kappa, \{\lambda\}) = \alpha_p \left(\frac{\kappa}{\sqrt{q_c}} \right) \left\{ \frac{\lambda^2}{[\alpha_p(\frac{\kappa}{\sqrt{q_c}})H(-\frac{\kappa}{\sqrt{q_c}})(\lambda - q_c) + q_c]^2} \right\}_\lambda \tag{7.41}$$

where $\alpha_p(x)$ is the storage capacity of a simple perceptron storing uncorrelated patterns with stability x discussed in Section 3.1:

$$\alpha_p(x) = \frac{1}{\int_{-x}^{\infty} Dz(x + z)^2}. \tag{7.42}$$

After the evaluation we obtain the final results for q_c and the critical storage capacity α_c:

$$q_c = \lambda_3 = \frac{1}{d}, \tag{7.43}$$

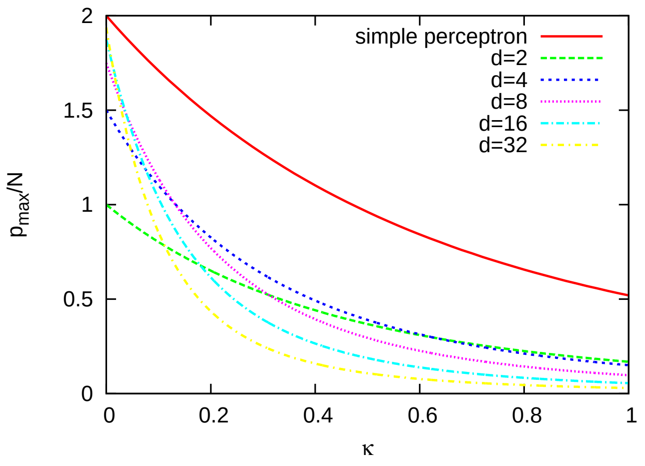

Figure 7.2: Capacity of Bethge architecture as a function of stability κ for $d = 2, 4, 8, 16, 32$. For comparison, the dotted curve shows the capacity of a simple perceptron.

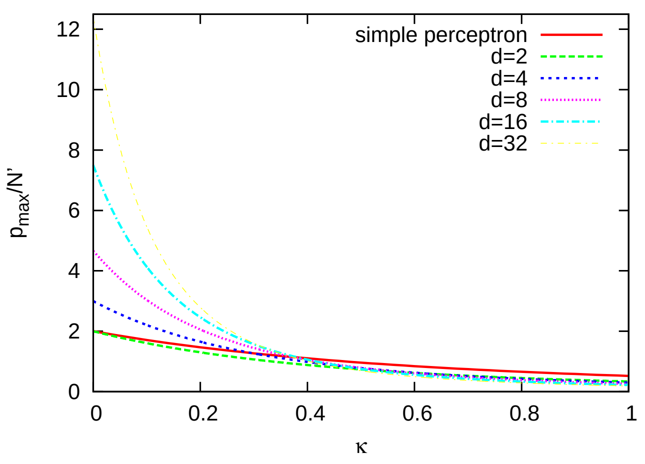

Figure 7.3: Capacity of Bethge architecture as a function of stability κ, measured relative to the size N' of the input layer, for $d = 2, 4, 8, 16, 32$. For comparison, the dotted curve shows the capacity of a simple perceptron.

$$\alpha_c(\kappa) \equiv \frac{p_{max}}{N} = \alpha_p(\kappa\sqrt{d})\left(1 - \frac{1}{d}\right). \tag{7.44}$$

The interpretation of this result is twofold.

On the one hand, it gives the storage capacity of a simple perceptron for correlated patterns. One observes a reduction compared to the capacity $\alpha_p(\kappa)$ of a simple perceptron storing uncorrelated patterns (Fig. 7.2).

On the other hand, (7.44) yields also the number of patterns that can be stored in the present two-layer network. The number of patterns normalised with respect to the size of the input layer N' is

$$\alpha'_c(\kappa) \equiv \frac{p_{max}}{N'} = \alpha_p(\kappa\sqrt{d})\frac{d-1}{\log_2 d}. \tag{7.45}$$

This capacity may become much larger than $\alpha_p(\kappa)$ (Fig. 7.3).

However, for any $\kappa > 0$ the capacity $\alpha'_c(\kappa)$ is lower than $\alpha_p(\kappa)$, if d is sufficiently large, since

$$\frac{\alpha_p(\kappa)}{\alpha'_c(\kappa)} \sim \alpha_p(\kappa)\frac{\kappa^2 d}{d-1}\log_2 d > 1, \quad \text{when} \quad d \to \infty. \tag{7.46}$$

Noting that $N' = m \cdot k$, where m is the number of modules and k the number of units per module in the input layer, one can show that the number of storable patterns at $\kappa = 0$ scales exponentially with N'

$$p_{max} = 2m(2^{\frac{N'}{m}} - 1). \tag{7.47}$$

Chapter 8

Bethge Architecture - Generalisation

In the previous chapter we discussed the architecture of a two-layer neural network (Fig. 8.1) introduced in [19] and showed how its storage capacity was evaluated. As it turned out, this network has advantages over the simple perceptron. In the present chapter we study the generalisation properties of the network. The techniques are again borrowed from the spin glass theory.

8.1 The Problem

The state of the N-dimensional hidden layer is characterised by the vector

$$\boldsymbol{\xi}^{\mu} = (\xi_1^{\mu}, ..., \xi_N^{\mu}), \tag{8.1}$$

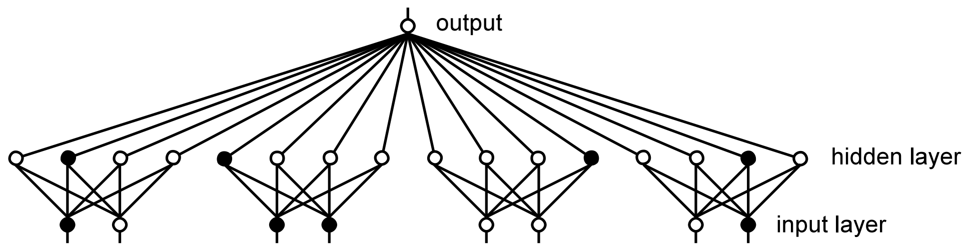

Figure 8.1: The neural network with fixed preprocessing.

where $\mu \in \{1, 2, ..., p\}$ enumerates the input-output patterns (training examples) and the pattern bits $\xi_i^\mu \in \{0, 1\}$. The output signal is given by

$$s = \text{sign}\left(\frac{1}{\sqrt{N}}\sum_{i=1}^{N} J_i \xi_i\right), \tag{8.2}$$

thus the student and the teacher yield the following outputs respectively

$$s^\mu = \text{sign}\left(\frac{1}{\sqrt{N}}\sum_{i=1}^{N} J_i \xi_i^\mu\right), \tag{8.3}$$

$$s_0^\mu = \text{sign}\left(\frac{1}{\sqrt{N}}\sum_{i=1}^{N} B_i \xi_i^\mu\right), \tag{8.4}$$

where $\mathbf{J} = (J_1, ..., J_N)$ denotes the couplings of the student and $\mathbf{B} = (B_1, ..., B_N)$ are those of the teacher.

To investigate the learning and generalisation properties of the system an approach from the statistical physics is applied. We consider an ensemble of systems to extract a *typical* behaviour of their learning abilities. Such ensembles are defined by a *Gibbs distribution*:

$$P(\mathbf{J} \mid \{\sigma_\mu\}) = \frac{1}{Z} P(\mathbf{J}) \exp\left(-\beta E(\mathbf{J})\right), \tag{8.5}$$

where $P(\mathbf{J})$ denotes the constraints on the couplings $\mathbf{J}$, Z is the canonical partition function:

$$Z = \int d\mu(\mathbf{J}) \exp\left(-\beta E(\mathbf{J})\right) \qquad \text{with} \quad d\mu(\mathbf{J}) = P(\mathbf{J}) d\mathbf{J}, \tag{8.6}$$

$E(\mathbf{J})$ is *the training energy*:

$$E(\mathbf{J}) = \sum_{\mu=1}^{p} \epsilon(\mathbf{J}, \boldsymbol{\xi}^\mu). \tag{8.7}$$

The error function is defined as

$$\epsilon(\mathbf{J}, \boldsymbol{\xi}^\mu) = \Theta(-(\mathbf{J} \cdot \boldsymbol{\xi}^\mu)(\mathbf{B} \cdot \boldsymbol{\xi}^\mu)), \tag{8.8}$$

where Θ is the step function. We also define *the generalisation function*:

$$\epsilon(\mathbf{J}) = \int d\mu(\boldsymbol{\xi}) \epsilon(\mathbf{J}, \boldsymbol{\xi}). \tag{8.9}$$

By introducing new variables in analogy to (7.7)

$$X_\mu = \frac{1}{\sqrt{N}} \eta_\mu \mathbf{J} \cdot \boldsymbol{\xi}^\mu, \quad Y_\mu = \frac{1}{\sqrt{N}} \eta_\mu \mathbf{B} \cdot \boldsymbol{\xi}^\mu \tag{8.10}$$

the error function gets the form

$$\epsilon(\mathbf{J}, \boldsymbol{\xi}^\mu) = \Theta(-X_\mu Y_\mu). \tag{8.11}$$

Here, $\eta_\mu = \pm 1$. The actual choice of η_μ is not relevant since it does not affect the error function (8.8) or the generalisation error. Later we are going to average over η_μ.

We consider the spherical constraints for the couplings $\mathbf{J}$: the a priori measure $d\mu(\mathbf{J})$ is uniform on the sphere $\sum_{i=1}^N J_i^2 = N$ of radius $\sqrt{N}$, i.e.

$$d\mu(\mathbf{J}) = \prod_{i=1}^N \left(\frac{dJ_i}{\sqrt{2\pi e}} \right) \delta \left(\sum_{i=1}^N J_i^2 - N \right), \tag{8.12}$$

hence it is normalised

$$\int d\mu(\mathbf{J}) = 1. \tag{8.13}$$

The partition function Z gets the following form:

$$Z = \int \prod_{i=1}^N \left(\frac{dJ_i}{\sqrt{2\pi e}} \right) \delta \left(\sum_{i=1}^N J_i^2 - N \right) \exp \left(-\beta \sum_{\mu=1}^p \Theta(-X_\mu Y_\mu) \right). \tag{8.14}$$

We intend to calculate the following quantities:

$$f(T,p) = -\frac{T \langle \ln Z \rangle}{N} \quad \textit{the specific free energy;} \tag{8.15}$$

$$\epsilon_g = \langle \langle \epsilon(\mathbf{J}) \rangle_T \rangle \quad \textit{the generalisation error,} \tag{8.16}$$

where

$$\langle \cdot \rangle_T = \int \cdot \frac{1}{Z} d\mu(\mathbf{J}) \exp\left(-\beta E(\mathbf{J})\right) \tag{8.17}$$

is *the thermal average*, and

$$\langle \cdot \rangle = \int \cdot \prod_{\mu=1}^p d\mu(\boldsymbol{\xi}^\mu) = \int \cdot \prod_{\mu=1}^p \left(d\boldsymbol{\xi}^\mu \right) P(\{\boldsymbol{\xi}^\mu\}_\mu) \tag{8.18}$$

is *the quenched average* over the distribution of the patterns. The generalisation error is linked to the so-called *teacher-student overlap P*:

$$\epsilon_g = \frac{1}{\pi} \arccos \frac{P}{1-\varepsilon}, \quad \text{where} \quad P = \frac{\mathbf{B} \cdot \mathbf{J}}{|\mathbf{B}||\mathbf{J}|} = \frac{1}{N} \sum_{\gamma=1}^N B_\gamma J_\gamma, \tag{8.19}$$

where $\varepsilon = 1/d$. The factor $(1-\varepsilon)^{-1}$ results from the fact that only those components of $\mathbf{J}$ are changed during the learning process which correspond to the nonzero eigenvalues λ_γ of the correlation matrix.

8.2 Replica Calculations

We use again the replica method:

$$\langle \ln Z \rangle = \lim_{n\to 0} \frac{\ln\langle Z^n \rangle}{n}. \tag{8.20}$$

In physical sense the expression Z^n is a partition function of n identical systems, *replicas*, labeled by $\sigma = 1, \dots, n$, which do not interact.

Thus, in order to determine the specific free energy (8.15), we have to calculate $\langle Z^n \rangle$ first, since

$$-\beta f = \lim_{n\to 0} \frac{\ln\langle Z^n \rangle}{Nn}. \tag{8.21}$$

The partition function acquires the form (we omit the constant factors):

$$Z = \int \prod_i dJ_i \delta\Big(\sum_i J_i^2 - N\Big) \prod_\mu \exp\left(-\beta\Theta(-X_\mu Y_\mu)\right). \tag{8.22}$$

After the replication

$$Z^n = \int \prod_{i\sigma} dJ_i^\sigma \delta\Big(\sum_i J_i^{\sigma 2} - N\Big) \prod_{\sigma\mu} \exp\left(-\beta\Theta(-X_\mu^\sigma Y_\mu)\right). \tag{8.23}$$

The averaging $\langle\cdot\rangle$ affects only the part containing the exponential functions

$$\langle Z^n \rangle = \int \prod_{i\sigma} dJ_i^\sigma \delta\Big(\sum_i J_i^{\sigma 2} - N\Big) \left\langle \prod_{\sigma\mu} \exp\left(-\beta\Theta(-X_\mu^\sigma Y_\mu)\right) \right\rangle. \tag{8.24}$$

Since the training examples $\boldsymbol{\xi}^\mu, \mu \in \{1, \dots, p\}$ are correlated, it is complicated to perform the averaging over these variables. We choose the variables X_σ^μ, Y^μ instead and average over the η_μ as well. These variables are uncorrelated and Gaussian distributed according to the central limit theorem. The moments of the distribution are

$$\langle X_\mu^\sigma \rangle = 0, \qquad \langle Y_\mu \rangle = 0, \tag{8.25}$$

$$\langle X_\mu^\sigma Y_\nu \rangle = \delta_{\mu\nu} r_\sigma, \tag{8.26}$$

$$\langle X_\mu^\sigma X_\nu^\rho \rangle = \delta_{\mu\nu} q_{\sigma\rho}, \tag{8.27}$$

where

$$r_\sigma = \frac{1}{N} \sum_\gamma \lambda_\gamma J_\gamma^\sigma B_\gamma, \tag{8.28}$$

$$q_{\sigma\rho} = \frac{1}{N} \sum_{\gamma} \lambda_\gamma J_\gamma^\sigma J_\gamma^\rho \tag{8.29}$$

The parameters r_σ and $q_{\sigma\rho}$ are the spin glass order parameters.

For the averaging we get the following operation

$$\langle \cdot \rangle = \int \cdot \prod_{\mu=1}^{p} [d\mathbf{Z}^\mu] \, P\left(\{\mathbf{Z}^\mu\}_\mu\right) = \int \cdot \prod_{\mu=1}^{p} [d\mathbf{Z}^\mu P(\mathbf{Z}^\mu)] \, . \tag{8.30}$$

Using the order parameters r_σ, $q_{\sigma\rho}$ and P_σ the $\langle Z^n \rangle$ can be rewritten as

$$\begin{aligned} \langle Z^n \rangle \;\; = \;\; & \int \prod_{\sigma} dr_\sigma \delta\Big(\sum_\gamma \lambda_\gamma B_\gamma J_\gamma^\sigma - N r_\sigma \Big) \int \prod_{\sigma \le \rho} dq_{\sigma\rho} \delta\Big(\sum_\gamma \lambda_\gamma J_\gamma^\sigma J_\gamma^\rho - N q_{\sigma\rho} \Big) \\ & \times \int \prod_\sigma dP_\sigma \delta\Big(\sum_\gamma B_\gamma J_\gamma^\sigma - N P_\sigma \Big) \int \prod_{i\sigma} dJ_i^\sigma \delta\Big(\sum_i J_i^{\sigma 2} - N \Big) \\ & \times \left\langle \prod_{\sigma\mu} \exp\Big(-\beta \Theta(-X_\mu^\sigma Y_\mu) \Big) \right\rangle . \end{aligned} \tag{8.31}$$

For the sake of convenience we split the right-hand side of (8.31) into two parts. The first part I_1 contains the δ functions. The second part I_2 contains the averaging $\langle \cdot \rangle$. Thus, we have

$$\langle Z^n \rangle = I_1 \cdot I_2, \tag{8.32}$$

where

$$\begin{aligned} I_1 \;\; = \;\; & \int \prod_{\sigma} dr_\sigma \delta\Big(\sum_\gamma \lambda_\gamma B_\gamma J_\gamma^\sigma - N r_\sigma \Big) \int \prod_{\sigma \le \rho} dq_{\sigma\rho} \delta\Big(\sum_\gamma \lambda_\gamma J_\gamma^\sigma J_\gamma^\rho - N q_{\sigma\rho} \Big) \\ & \times \int \prod_\sigma dP_\sigma \delta\Big(\sum_\gamma B_\gamma J_\gamma^\sigma - N P_\sigma \Big) \int \prod_{i\sigma} dJ_i^\sigma \delta\Big(\sum_i J_i^{\sigma 2} - N \Big) \end{aligned} \tag{8.33}$$

and

$$I_2 = \left\langle \prod_{\sigma\mu} \exp\Big(-\beta \Theta(-X_\mu^\sigma Y_\mu) \Big) \right\rangle . \tag{8.34}$$

The specific free energy, using the replica identity, I_1 and I_2 obtains the form

$$-\beta f = \lim_{n \to 0} \frac{\ln \langle Z^n \rangle}{Nn} = G_1 + G_2, \tag{8.35}$$

where

$$G_1 = \lim_{n\to 0} \frac{\ln I_1}{nN} \quad \text{and} \quad G_2 = \lim_{n\to 0} \frac{\ln I_2}{nN}. \tag{8.36}$$

In order to calculate I_1, we use integral representation of the δ functions by introducing conjugate variables $\hat{r}_\sigma$, $\hat{q}_{\sigma\rho}$, $\hat{N}_\sigma$, $\hat{P}_\sigma$, and omit the constant factors:

$$\begin{aligned} I_1 &= \int \prod_\sigma dr_\sigma d\hat{r}_\sigma \int \prod_{\sigma\le\rho} dq_{\sigma\rho} d\hat{q}_{\sigma\rho} \int \prod_\sigma dP_\sigma d\hat{P}_\sigma \int \prod_\sigma d\hat{N}_\sigma \int \prod_{\sigma i} dJ_i^\sigma \\ &\times \exp\left\{ i \left(\sum_{\sigma i} \hat{N}_\sigma J_i^{\sigma 2} + \sum_{\sigma\gamma} \hat{r}_\sigma \lambda_\gamma B_\gamma J_\gamma^\sigma + \sum_{\sigma\le\rho,\gamma} \hat{q}_{\sigma\rho} \lambda_\gamma J_\gamma^\sigma J_\gamma^\rho + \sum_{\sigma\gamma} \hat{P}_\sigma B_\gamma J_\gamma^\sigma \right) \right\} \\ &\times \exp\left\{ -iN \left(\sum_\sigma \hat{r}_\sigma r_\sigma + \sum_{\sigma\le\rho} \hat{q}_{\sigma\rho} q_{\sigma\rho} + \sum_\sigma \hat{P}_\sigma P_\sigma + \sum_\sigma \hat{N}_\sigma \right) \right\}. \end{aligned} \tag{8.37}$$

In the thermodynamic limit the integral in I_1 over the order parameters is dominated by the saddle point in r_σ, $q_{\sigma\rho}$ and P_σ. The specific free energy is obtained by analytically continuing the saddle point to $n = 0$. Thus, we omit the integrals over the parameters $r_\sigma, q_{\sigma\rho}, P_\sigma, \hat{r}_\sigma, \hat{q}_{\sigma\rho}, \hat{P}_\sigma, \hat{N}_\sigma$ in I_1 in the anticipation of the use of the steepest descent method.

8.3 The Replica Symmetric (RS) Solution

We use the following replica symmetric (RS) ansatz

$$q_{\sigma\rho} = \delta_{\sigma\rho} q_0 + (1-\delta_{\sigma\rho}) q \tag{8.38}$$
$$r_\sigma = r \tag{8.39}$$
$$P_\sigma = P \tag{8.40}$$
$$\hat{q}_{\sigma\rho} = \delta_{\sigma\rho} \hat{q}_0 + (1-\delta_{\sigma\rho}) \hat{q} \tag{8.41}$$
$$\hat{r}_\sigma = \hat{r} \tag{8.42}$$
$$\hat{P}_\sigma = \hat{P}. \tag{8.43}$$

For the G_1 we get

$$\begin{aligned} G_1 &= -i\hat{N} - i\hat{q}_0 q_0 + \frac{1}{2} i\hat{q}q - i\hat{r}r - i\hat{P}P - \frac{1}{N}\sum_\gamma \frac{1}{2}\ln(\hat{N} + (\hat{q}_0 - \frac{1}{2}\hat{q})\lambda_\gamma) \\ &- \frac{1}{N}\sum_\gamma \frac{\lambda_\gamma \hat{q}}{4(\hat{N} + (\hat{q}_0 - \frac{1}{2}\hat{q})\lambda_\gamma)} - \frac{1}{N}\sum_\gamma \frac{i\lambda_\gamma^2 \hat{r}^2}{4(\hat{N} + (\hat{q}_0 - \frac{1}{2}\hat{q})\lambda_\gamma)} \\ &- \frac{1}{N}\sum_\gamma \frac{\lambda_\gamma i\hat{r}\hat{P}}{2(\hat{N} + (\hat{q}_0 - \frac{1}{2}\hat{q})\lambda_\gamma)} - \frac{1}{N}\sum_\gamma \frac{i\hat{P}^2}{4(\hat{N} + (\hat{q}_0 - \frac{1}{2}\hat{q})\lambda_\gamma)} \end{aligned} \tag{8.44}$$

The evaluation of I_2 yields the G_2

$$\begin{aligned} G_2 &= 2\alpha \int DtH\left(-\frac{rt}{\sqrt{qs-r^2}}\right) \\ &\quad \times \ln\left[\exp(-\beta) + (1-\exp(-\beta))H\left(-\sqrt{\frac{qt^2}{q_0-q}}\right)\right], \end{aligned} \tag{8.45}$$

where

$$s = \frac{1}{N}\sum_{\gamma} \lambda_\gamma B_\gamma B_\gamma. \tag{8.46}$$

In the $T = 0$ limit G_2 gets the form

$$G_2 = 2\alpha \int DtH\left(-\frac{rt}{\sqrt{qs-r^2}}\right) \ln H\left(-\sqrt{\frac{qt^2}{q_0-q}}\right). \tag{8.47}$$

8.4 The Saddle Point Equations

To calculate the parameters $r, q_0, q, P, \hat{r}, \hat{q}_0, \hat{q}, \hat{P}, \hat{N}$ the following system of *saddle point equations* should be solved

$$\frac{\partial f}{\partial L_i} = 0, \qquad L_i \in \{r, q_0, q, P, \hat{r}, \hat{q}_0, \hat{q}, \hat{P}, \hat{N}\}. \tag{8.48}$$

After inserting the corresponding values and performing the following transformations for convenience:

$$\hat{q_0} = i\tilde{q_0}, \quad \hat{q} = i2\tilde{q}, \quad \hat{r} = i\tilde{r}, \quad \hat{N} = i\tilde{N}, \quad \tilde{v} = \tilde{N}/(\tilde{q_0} - \tilde{q}) \tag{8.49}$$

we get the equations:

$$\frac{1}{N}\sum_\gamma \frac{\lambda_\gamma^2}{(\tilde{v}+\lambda_\gamma)^2} = -\frac{2q(\tilde{q}_0-\tilde{q})^2}{\tilde{q}} + \frac{\tilde{r}^2}{2\tilde{q}N}\sum_\gamma \frac{\lambda_\gamma^3}{(\tilde{v}+\lambda_\gamma)^2} \tag{8.50}$$

$$\frac{1}{N}\sum_\gamma \frac{\lambda_\gamma}{\tilde{v}+\lambda_\gamma} = 2(q_0-q)(\tilde{q}_0-\tilde{q}) \tag{8.51}$$

$$\frac{1}{N}\sum_\gamma \frac{\lambda_\gamma^2}{\tilde{v}+\lambda_\gamma} = -\frac{2r(\tilde{q}_0-\tilde{q})}{\tilde{r}} \tag{8.52}$$

$$\frac{1}{N}\sum_\gamma \frac{\lambda_\gamma}{(\tilde{v}+\lambda_\gamma)^2} = -\frac{2(\tilde{q}_0-\tilde{q})^2}{\tilde{q}} + \frac{\tilde{q}_0-\tilde{q}}{\tilde{q}N}\sum_\gamma \frac{1}{\tilde{v}+\lambda_\gamma} + \frac{\tilde{r}^2}{2\tilde{q}N}\sum_\gamma \frac{\lambda_\gamma^2}{(\tilde{v}+\lambda_\gamma)^2} \tag{8.53}$$

$$r\tilde{r} = 2(q\tilde{q} - q_0\tilde{q}_0) \tag{8.54}$$

$$\frac{1}{N}\sum_\gamma \frac{\lambda_\gamma}{\tilde{v}+\lambda_\gamma} = -\frac{2P(\tilde{q}_0-\tilde{q})}{\tilde{r}} \tag{8.55}$$

$$\hat{P} = 0 \tag{8.56}$$

$$\tilde{q} = -\frac{\alpha}{2\pi(q_0-q)}\int Dt \frac{\exp\left(-\frac{qt^2}{q_0-q}\right) H\left(-\frac{rt}{\sqrt{sq-r^2}}\right)}{H^2\left(-\sqrt{\frac{qt^2}{q_0-q}}\right)} \tag{8.57}$$

$$\tilde{r} = -\frac{\alpha\sqrt{q}}{\pi\sqrt{(sq-r^2)(q_0-q)}} \tag{8.58}$$

$$\times \int Dt \frac{\exp\left(-\frac{r^2t^2}{2(sq-r^2)}\right)\exp\left(-\frac{qt^2}{2(q_0-q)}\right)}{H\left(-\sqrt{\frac{qt^2}{q_0-q}}\right)} \tag{8.59}$$

The eigenvalues (7.6) can now be inserted in the equations. From now on it will be assumed that $N \gg d \gg 1$. For convenience, the following scaled variables will be used

$$Q_0 = q_0 d, \quad Q = qd, \quad R = rd, \tag{8.60}$$

$$\tilde{Q}_0 = \tilde{q}_0 d^{-1}, \quad \tilde{Q} = \tilde{q} d^{-1}, \quad \tilde{R} = \tilde{r} d^{-1}, \quad \tilde{z} = \tilde{v} d. \tag{8.61}$$

The system of the saddle point equations gets the form:

$$\frac{1-\varepsilon}{(\tilde{z}+1)^2} = -\frac{2Q(\widetilde{Q}_0-\widetilde{Q})^2}{\widetilde{Q}} + \frac{\tilde{R}^2\varepsilon}{2\widetilde{Q}} + \frac{\tilde{R}^2(1-\varepsilon)}{2\widetilde{Q}(\tilde{z}+1)^2} \tag{8.62}$$

$$\frac{1-\varepsilon}{\tilde{z}+1} = 2(Q_0-Q)(\widetilde{Q}_0-\widetilde{Q}) \tag{8.63}$$

$$\frac{1-\varepsilon}{\tilde{z}+1} = -\varepsilon - \frac{2R(\widetilde{Q}_0-\widetilde{Q})}{\tilde{R}} \tag{8.64}$$

$$\frac{1-\varepsilon}{(\tilde{z}+1)^2} = -\frac{2(\widetilde{Q}_0-\widetilde{Q})^2}{\widetilde{Q}} + \frac{\tilde{R}^2(1-\varepsilon)}{2\widetilde{Q}(\tilde{z}+1)^2} + \frac{(\widetilde{Q}_0-\widetilde{Q})(1-\varepsilon)}{\widetilde{Q}(\tilde{z}+1)} + \frac{(\widetilde{Q}_0-\widetilde{Q})\varepsilon}{\widetilde{Q}\tilde{z}} \tag{8.65}$$

$$R\tilde{R} = 2(Q\widetilde{Q} - Q_0\widetilde{Q}_0) \tag{8.66}$$

$$\frac{1-\varepsilon}{\tilde{z}+1} = -\frac{2P(\widetilde{Q}_0-\widetilde{Q})}{\tilde{R}} \tag{8.67}$$

$$\tilde{Q} = -\frac{\alpha u^2}{2\pi Q}\int Dt \frac{\exp\left(-u^2t^2\right)H\left(-vt\right)}{H^2\left(-ut\right)} \tag{8.68}$$

$$\tilde{R} = -\frac{\alpha uv}{\pi R}\int Dt \frac{\exp\left(-\frac{v^2t^2}{2}\right)\exp\left(-\frac{u^2t^2}{2}\right)}{H\left(-ut\right)}, \tag{8.69}$$

where

$$\varepsilon = \frac{1}{d}, \quad u = \sqrt{\frac{Q}{Q_0-Q}}, \quad v = \frac{R}{\sqrt{Q-R^2}} \tag{8.70}$$

8.5 The Linear Approximation

The solution of the equations can be written using the following linear ansatz

$$Q_0 = Q_0^{(0)} + \varepsilon Q_0^{(1)}, \quad Q = Q^{(0)} + \varepsilon Q^{(1)}, \quad R = R^{(0)} + \varepsilon R^{(1)}, \tag{8.71}$$

$$\widetilde{Q}_0 = \widetilde{Q}_0^{(0)} + \varepsilon \widetilde{Q}_0^{(1)}, \quad \widetilde{Q} = \widetilde{Q}^{(0)} + \varepsilon \widetilde{Q}^{(1)}, \quad \tilde{R} = \tilde{R}^{(0)} + \varepsilon \tilde{R}^{(1)}, \tag{8.72}$$

$$\tilde{z} = \tilde{z}^{(0)} + \varepsilon \tilde{z}^{(1)}, \tag{8.73}$$

where the indices (0) and (1) denote the leading term and the first correction respectively. The leading terms represent the solution of the equations for the case, when all eigenvalues are equal

$$Q^{(0)} = q\lambda^{-1} = \frac{\alpha}{\pi}\int Dt \frac{\exp\left(-\frac{Q^{(0)}}{1-Q^{(0)}}t^2\right)}{H\left(-\sqrt{\frac{Q^{(0)}}{1-Q^{(0)}}}t\right)}, \tag{8.74}$$

$$Q_0^{(0)} = 1, \quad R^{(0)} = Q^{(0)}, \tag{8.75}$$

$$\tilde{Q}_0^{(0)} = 0, \quad \tilde{Q}^{(0)} = -\frac{Q^{(0)}}{2(1-Q^{(0)})}, \quad \tilde{R}^{(0)} = -\frac{Q^{(0)}}{1-Q^{(0)}}, \tag{8.76}$$

$$\tilde{z}^{(0)} = \frac{1-Q^{(0)}}{Q^{(0)}}. \tag{8.77}$$

After inserting this ansatz and considering the terms up to the linear order in ε, a system of linear equations is obtained, where the linear corrections are the variables. The first five equations can be expressed in the following form:

$$A * x = B, \tag{8.78}$$

where A is the following matrix (here $Q \equiv Q^{(0)}$):

$$\left|\begin{array}{c|c|c|c|c|c|c} 0 & 0 & 1 & 2Q(1-Q) & 4(1-Q) & -2(1-Q^2) & 2Q^2 \\ 0 & -Q & Q & 0 & -2(1-Q)^2 & 2(1-Q)^2 & -Q^2(1-Q) \\ 0 & 0 & 0 & -2Q^2(1-Q) & -2(1-Q^2) & 2(1+Q^2)(1-Q) & -Q^2(1+Q) \\ -1 & 0 & 0 & -(1-Q) & -2(1-Q) & 2(1-Q) & -Q^2 \\ -Q & 0 & Q & Q(1-Q) & 2(1-Q) & -2Q(1-Q) & 0 \end{array}\right|$$

$$x = (R^{(1)}, Q_0^{(1)}, Q^{(1)}, \tilde{R}^{(1)}, \tilde{Q}_0^{(1)}, \tilde{Q}^{(1)}, \tilde{z}^{(1)}), \tag{8.79}$$

$$B = (1-Q, Q(1-Q), 0, -(1-Q), 0). \tag{8.80}$$

Using these five equations the variables $Q_0^{(1)}, Q^{(1)}, R^{(1)}, \tilde{Q}_0^{(1)}, \tilde{z}^{(1)}$ can be expressed in terms of $\tilde{Q}^{(1)}$ and $\tilde{R}^{(1)}$

$$Q_0^{(1)} = 0 \tag{8.81}$$

$$R^{(1)} = \frac{2Q^{(0)}(1-Q^{(0)})^2}{1+Q^{(0)}}\tilde{Q}^{(1)} + \frac{(1-Q^{(0)})(2{Q^{(0)}}^2 - 1 - Q^{(0)})}{1+Q^{(0)}}\tilde{R}^{(1)} \tag{8.82}$$

$$Q^{(1)} = -\frac{2(1-Q^{(0)})({Q^{(0)}}^2 + 1 - 2Q^{(0)})}{1+Q^{(0)}}\tilde{Q}^{(1)} - \frac{2Q^{(0)}(1-Q^{(0)})^2}{1+Q^{(0)}}\tilde{R}^{(1)} \tag{8.83}$$

$$\tilde{Q}_0^{(1)} = \frac{2Q^{(0)}}{1+Q^{(0)}}\tilde{Q}^{(1)} - \frac{Q^{(0)}}{1+Q^{(0)}}\tilde{R}^{(1)} \tag{8.84}$$

$$\tilde{z}^{(1)} = \frac{2(1-Q^{(0)})^3}{{Q^{(0)}}^2(1+Q^{(0)})}\tilde{Q}^{(1)} + \frac{2(1-Q^{(0)})^2}{Q^{(0)}(1+Q^{(0)})}\tilde{R}^{(1)} \tag{8.85}$$

For the variables $\tilde{Q}^{(1)}$ and $\tilde{R}^{(1)}$ we have the following equations:

$$\begin{aligned}\tilde{R}^{(1)} = {} & \frac{\alpha B}{\sqrt{8}\pi^{3/2}\sqrt{Q^{(0)}}(1-Q^{(0)})^{5/2}}Q^{(1)} \\ & + \left(\frac{1}{1-Q^{(0)}} + \frac{\alpha C}{\pi(1-Q^{(0)})^3} - \frac{\alpha A}{\pi Q^{(0)}(1-Q^{(0)})^2}\right) R^{(1)}\end{aligned} \tag{8.86}$$

$$\begin{aligned}\tilde{Q}^{(1)} = {} & -\frac{\alpha B}{\sqrt{8}\pi^{3/2}\sqrt{Q^{(0)}}(1-Q^{(0)})^{5/2}}R^{(1)} \\ & +\frac{1}{2}\left(\frac{1}{1-Q^{(0)}} + \frac{\alpha C}{\pi(1-Q^{(0)})^3} - \frac{\alpha A}{\pi Q^{(0)}(1-Q^{(0)})^2} + \frac{\alpha 3B}{\sqrt{8}\pi^{3/2}\sqrt{Q^{(0)}}(1-Q^{(0)})^{5/2}}\right) Q^{(1)},\end{aligned}$$

where

$$A = \int Dt \frac{\exp\left(-\frac{Q^{(0)}}{1-Q^{(0)}}t^2\right)}{H\left(-\sqrt{\frac{Q^{(0)}}{1-Q^{(0)}}}t\right)} \tag{8.87}$$

$$B = \int Dt \frac{t\exp\left(-\frac{3}{2}\frac{Q^{(0)}}{1-Q^{(0)}}t^2\right)}{H^2\left(-\sqrt{\frac{Q^{(0)}}{1-Q^{(0)}}}t\right)} \tag{8.88}$$

$$C = \int Dt \frac{t^2\exp\left(-\frac{Q^{(0)}}{1-Q^{(0)}}t^2\right)}{H\left(-\sqrt{\frac{Q^{(0)}}{1-Q^{(0)}}}t\right)} \tag{8.89}$$

8.6 The Solution and the Learning Curves

It can be easily shown that the solution of the saddle point equations satisfies the following relations:

$$R = Q, \quad \tilde{R} = 2\tilde{Q}, \quad Q_0 = 1, \quad \tilde{Q}_0 = 0, \tag{8.90}$$

the other variables get the form:

$$\tilde{Q} = -\frac{Q-\varepsilon}{2(1-Q)}, \quad \tilde{z} = \frac{1-Q}{Q-\varepsilon}, \quad P = Q-\varepsilon, \tag{8.91}$$

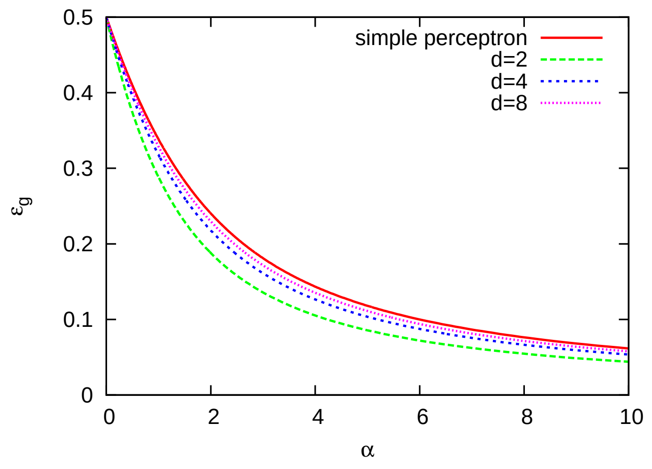

Figure 8.2: The learning curves $\epsilon_g = \epsilon_g(\alpha)$ for $d = 2, 4, 8$ compared to that of a simple perceptron.

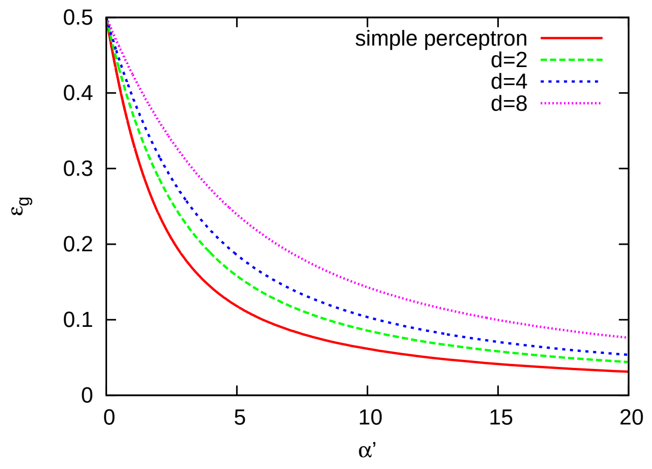

Figure 8.3: The learning curves $\epsilon_g = \epsilon_g(\alpha')$ for $d = 2, 4, 8$ compared to that of a simple perceptron.

$$Q = \epsilon + \frac{\alpha}{\pi} \int Dt \frac{\exp\left(-\frac{Q}{1-Q}t^2\right)}{H\left(-\sqrt{\frac{Q}{1-Q}}t\right)}. \tag{8.92}$$

The generalisation error obtains the form

$$\epsilon_g(Q, \varepsilon) = \frac{1}{\pi} \arccos \frac{Q-\varepsilon}{1-\varepsilon}. \tag{8.93}$$

Using (8.92) we get:

$$\alpha(Q, \varepsilon) = \pi(Q-\varepsilon) \left(\int Dt \frac{\exp\left(-\frac{Q}{1-Q}t^2\right)}{H\left(-\sqrt{\frac{Q}{1-Q}}t\right)} \right)^{-1}. \tag{8.94}$$

The generalisation error as a function of α is the sought learning curve

$$\epsilon_g = \epsilon_g(\alpha). \tag{8.95}$$

Fig. 8.2 shows learning curves corresponding to $d = 2, 4, 8$ as compared to the learning curve of a simple perceptron. It can be observed that the generalisation error decreases faster than that of a simple perceptron, depending on d. With the increasing d the curve of the Bethge network tends to approach the simple perceptron curve.

If we normalise the number of patterns with respect to the size of the input layer N'

$$\alpha' \equiv \frac{p}{N'} = \frac{d}{\log_2 d}\alpha \tag{8.96}$$

we obtain the learning curves $\epsilon_g = \epsilon_g(\alpha')$ shown in Fig. 8.3. As we see, in this normalisation of α the generalisation becomes worse compared to the simple perceptron.

We introduce a new parameter

$$\tau = \frac{\alpha}{\alpha_c(0)}, \tag{8.97}$$

where

$$\alpha_c(0) = 2\left(1 - \frac{1}{d}\right) \tag{8.98}$$

is the storage capacity for $\kappa = 0$, and consider $\epsilon_g = \epsilon_g(\tau)$ (Fig. 8.4). The learning curves lie over that of the simple perceptron.

This result may be interpreted in the following way. The preprocessing of patterns leads to the correlations in the hidden layer. The structure of these correlations is given by the eigenvalues of the C matrix. Due to

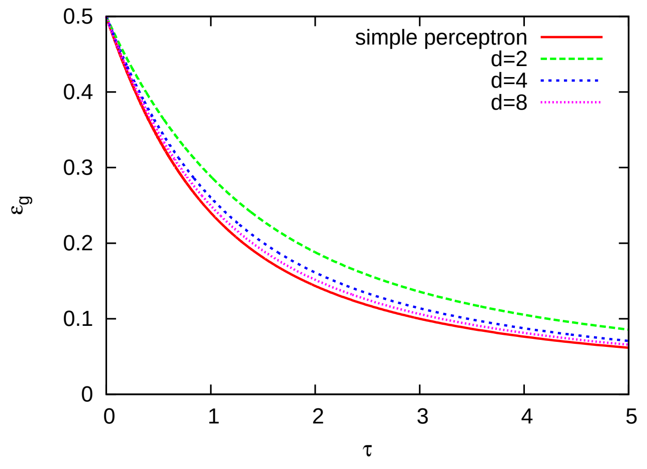

Figure 8.4: The learning curves $\epsilon_g = \epsilon_g(\tau)$ for $d = 2, 4, 8$ compared to that of a simple perceptron.

these correlations the generalisation error as a function of α decreases faster than in the case of the simple perceptron. However, as a function of α' the learning curves show slower decay compared to the perceptron. If the number of patterns is normalised by the maximal number of storable random patterns $\tau = p/p_{max}$ the above two effects seem to compensate each other, still making the generalisation fall a bit slower than for the perceptron.

Chapter 9

Conclusions and Outlook

We have considered a way to improve the learning and generalisation properties of the simple perceptron. For this purpose the Bethge et al. two-layer neural network has been investigated. It was shown that the specific preprocessing scheme in the input-to-output mapping induces correlations in the hidden layer. These correlations are given by the eigenvalues λ_γ. The correlations result in the reduction of the storage capacity and the generalisation error. Depending on the size of the modules d in the hidden layer, the learning curves lie under the simple perceptron learning curve, approaching it rapidly with increasing d. If the size of the input layer N' is considered as a normalising factor for the number of patterns, the storage capacity increases and the generalisation error decreases slower than for the simple perceptron. In order to find a measure for the performance, which is in some sense independent of details of the architecture, we introduce the ratio τ of the number of presented patterns p to the maximal number of random patterns p_{max}, which can be stored in the same network. $\tau = p/p_{max} = \alpha/\alpha_c$. This measure can even be used for networks of unknown architecture. Considering the generalisation error now as a function of τ, the two effects are combined resulting in the learning curves which decay slightly slower compared to the simple perceptron case. The reduction in performance is, however, rather small.

As for outlook, we would like to mention some further points which could be considered in future. It would be interesting to investigate the learning and generalisation of the Bethge network in the presence of noise (nonzero temperature) and using unlearnable rules. A rule is unlearnable if the structures of a student and teacher are different from each other.

Another step would be the application of the methods used in this work in the consideration of some other two-layer SVM architectures with different feature extractions in the input-to-hidden mapping. Performing numerical simulations would enrich our understanding of the problem.

Further, different unsupervised learning scenarios, e.g. the vector quantisation algorithm, can be applied to model the formation of preprocessing in the input-to-hidden layer. In this way, an optimal coding can be determined and fixed.

Using some realistic data set, the considered networks can be tested in their performance.

Bibliography

[1] F. Rosenblatt, *Principles of Neurodynamics*, (Spartan, New York. 1962)

[2] J. Hertz, A. Krogh, and R.G. Palmer: *Introduction to the Theory of Neural Computation*, (Addison-Wesley, Redwood City 1991).

[3] T.L.H. Watkin, A. Rau., and M. Biehl, *The Statistical Mechanics of Learning a Rule*, Rev. Mod. Phys. **65** 499 (1993).

[4] A. Engel, and C. Van den Broeck, *Statistical Mechanics of Learning*, (Cambridge University Press, 2001).

[5] H. Nishimori, *Statistical Physics of Spin Glasses and Information Processing*, (Oxford University Press, 2001).

[6] M. Minsky and S. Papert, *Perceptrons*, MIT Press: (Cambridge, Mass.1969); Partially reprented in 1988.

[7] G.J. Mitchison and R.M. Durbin, Biol. Cybern. **60**, 345 (1989)

[8] E. Barkai, D. Hansel, and H. Sompolinsky, Phys. Rev. A**45**, 4146 (1992); A. Engel, H.M. Köhler, F. Tschepke, H. Vollmayr, and A. Zippelius, Phys. Rev. A**45**, 7590 (1992)

[9] H. Horner, *Dynamics of Learning for the Binary Perceptron*, Z. Phys. B **86** 291 (1992)

[10] B. E. Boser, I. M. Guyon, and V. M. Vapnik: *A training algorithm for optimal margin classifiers* in: *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144-152, Pittsburgh, PA, (ACM Press, 1992)

[11] V. Vapnik *The Nature of Statistical Learning Theory* (Springer Verlag, 1995).

[12] B. Schölkopf, C. J. C. Burges, and A. J. Smola, *Advances in Kernel Methods Support Vector Learning*: (The MIT Press, 1999)

[13] A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmanns, (eds.): *Advances in Large Margin Classifiers* (The MIT Press, 2000)

[14] N. Cristianini and J. Shawe-Taylor: *Support Vector Machines*, (Cambridge University Press, 2000).

[15] M. Opper, *The Complexity of Learning with Supportvector Machines* in: *Perspectives on Adaptivity and Learning*, R. Kühn, R. Menzel, W. Menzel, U. Ratsch, M.M. Richter, und I.O. Stamatescu (Hrsg.), (Springer Verlag, Heidelberg, 2002) im Druck

[16] R. Dietrich, M. Opper and H. Sompolinsky: *Statistical Mechanics of Support Vector Networks*, Phys. Rev. Lett. **82**, 2975 (1999).

[17] S. Risau-Gusman and M. Gordon. Phys. Rev. E **62** 7092–7099, 2000.

[18] M. Opper and R. Urbanczik: *Universal learning curves of support vector machines*, Phys. Rev. Lett. **86** 4410-4413 (2001).

[19] A. Bethge, R. Kühn, and H. Horner, *Storage Capacity of a Two–Layer Perceptron with Fixed Preprocessing in the First Layer*, J. Phys. A **27** 1929 (1994).

[20] M. Kac, in *Trondheim Theoretical Physics Seminar*, Nordita Publ. No. 286 (1968)

[21] S.F. Edwards, in *4th International Conference on Amorphous Materials*, edited by R.W. Douglas, B. Ellis (Wiley, New York 1970)

[22] S.F. Edwards, P.W. Anderson, J. Phys. F **5** 965 (1975)

[23] D. Sherrington, S. Kirkpatrick, Phys. Rev. Lett. **35** 1792 (1975)

[24] D.J. Amit, H. Gutfreund, H. Sompolinsky, Phys. Rev. Lett. **55** 1530 (1985)

[25] E. Gardner, *Maximum storage capacity of neural networks*, Europhys. Lett. **4** 481 (1987).

[26] E. Gardner, *The space of interactions in neural network models*, J. Phys. A: Math. Gen. **21** 257 (1988).

[27] D.J. Amit, *Modelling Brain Function*, (Cambridge University Press, 1989)

[28] B. Müller, J. Reinhard, *Neural Networks – An Introduction*, (Springer, Berlin, 1991)

[29] P. Peretto, *An Introduction to the Modeling of Neural Networks*, (Cambridge University Press, 1992)

[30] H. Horner, Z. Phys. B **75** 133 (1989)

[31] G. Györgyi, N.Tishby: *Statistical theory of learning a rule* in: *Neural networks and spin glasses* (Ed. K.Thuemann, R. Köberle), pages 3-36, (World Scientific, Singapore 1990)

[32] W.S. McCulloch, W.Pitts *A logical calculus of ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics **5** 115-133 (1943). Reprinted in Anderson and Rosenberg (1988).

[33] T.M. Cover: *Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition*, IEEE Transactions on Electronic Computers **14** 326-334 (1965).

[34] G.J. Mitchison, R.M. Durbin: *Bounds on the learning capacity of some multi-layer networks*, Biological Cybernetics **60** 345-356 (1989).

[35] D.O. Hebb, *The Organization of Behavior*, (Wiley, New York, 1949).

Acknowledgements

I would like to express my profound gratitude to everyone who contributed to this project. I feel especially indebted to:

Prof.Dr. Heinz Horner for the excellent supervision and guidance.

Prof.Dr. Reimer Kühn for suggesting the interesting topic and for intensive feedback.

Prof.Dr. Franz Wegner for being a critical but fair co-supervisor.

Prof.Dr. Sigrid Böge and Prof.Dr. Siegfried Hunklinger for being co-referees.

Yvette Harbers and Dr. Eduard Thommes for solving so many problems for me.

Cristina Picus, Heiner Kohler, Talgat Daniyarov, Augusto Minatta for the competent technical assistance.

The whole collective of the institute of theoretical physics for the friendly atmosphere, particularly, Melanie Steiert, Cornelia Merkel and Sonja Bartsch.

All the students of the Wohnheim Seidel for creating lively studentlike ambience and still not being too loud.

And last but not least, my dear family for constant moral support, understanding and patience.

ალვსება ყოველთა კეთილთა შენ ხარ, ქრისტე, ღმერთო ჩემო, აღავსე სიხარულითა და მხიარულებითა გული ჩემი და მაცხოვნე მე, რამეთუ შენ ხარ ღმერთი მრავალმოწყალე.

Erklärung

Ich versichere, dass ich diese Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Heidelberg, den .... Unterschrift ....

(Merab Kutsia)