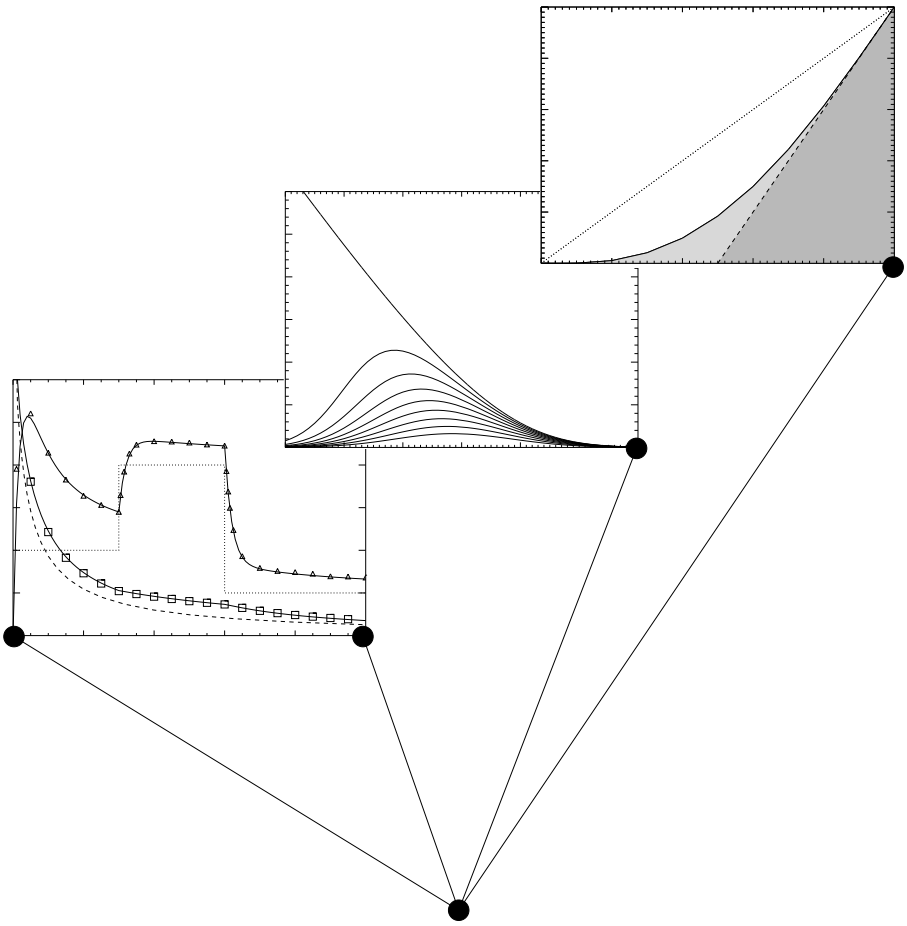
Dynamics of On-line Learning in Neural Networks



Peter Riegler

Institut für Theoretische Physik
Bayerische Julius–Maximilians–Universität Würzburg

1997

Dynamics of On-line Learning in Neural Networks

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Bayerischen Julius–Maximilians–Universität
Würzburg

vorgelegt von

Peter Riegler

geboren in Werneck

Institut für Theoretische Physik
Bayerische Julius–Maximilians–Universität Würzburg

1997

Eingereicht am 17.01.1997
bei der Fakultät für Physik und Astronomie

Beurteilung der Dissertation:

1. Gutachter: Prof. Dr. W. Kinzel
2. Gutachter: Priv. Doz. Dr. G. Reents

Mündliche Prüfung:

1. Prüfer: Prof. Dr. W. Kinzel
2. Prüfer: Prof. Dr. H. Langhoff

Tag der mündlichen Prüfung:

Abstract

One of the most important features of natural as well as artificial neural networks is their ability to adjust to their environment by "learning". This results in the network's ability to "generalize", i.e. to generate with high probability the appropriate response to an unknown input. The theoretical description of generalization in artificial neural networks by means of statistical physics is the subject of this thesis. The focus is on *on-line learning*, where the presentation of examples used in the learning process occurs in a sequential manner. Hence, the systems investigated are dynamical in nature. They typically consist of a large number of degrees of freedom, requiring a description in terms of order parameters.

In the first part of this work the most fundamental network, the perceptron, is investigated. Following a recent proposal by Kinouchi and Caticha it will be shown how one can derive a learning dynamics starting from first principles that results in an optimal generalization ability. Results will be presented for learning processes where the training examples are corrupted by different types of noise. The resulting generalization ability will be shown to be comparable to the noiseless case. Furthermore the results obtained reveal striking similarities to those obtained for batch learning.

The optimal algorithms derived will be shown to depend on the characteristics of the particular learning task including the type and strength of the corrupting noise. In general this requires an additional estimation of such characteristic quantities. For the strength of the noise this estimation leads to interesting dynamical phase transitions.

The second part deals with the dynamical properties of two-layer neural networks. This is of particular importance since these networks are known to represent universal approximators. Understanding the dynamical features will help to construct fast training algorithms that lead to best generalization.

Specifically, an exact analysis of learning a rule by on-line gradient descent (backpropagation of error) in a two-layered neural network will be presented. Hereby, the emphasis is on adjustable hidden-to-output weights which have been left out of the analysis in the literature so far. Results are compared with the training of networks having the same architecture but fixed weights in the second layer. It will be shown, that certain features of learning in a two-layered neural network are independent of the state of the second layer. Motivated by this result it will be argued that putting the dynamics of the hidden-to-output weights on a faster time scale will speed up the learning process.

For all systems investigated, simulations confirm the results.

Zusammenfassung

Eine der herausragendsten Eigenschaften natürlicher wie künstlicher neuronaler Netze ist ihre Fähigkeit zu „lernen“. Dies geschieht durch Anpassung ihrer Freiheitsgrade, den synaptischen Kopplungen. Außerdem können neuronale Netze „verallgemeinern“, d.h. sie können mit großer Wahrscheinlichkeit die richtige Ausgabe zu einer bisher unbekannten Eingabe produzieren. Der Gegenstand dieser Arbeit ist die theoretische Beschreibung der Verallgemeinerungsfähigkeit in künstlichen neuronalen Netzen mit Methoden der Statistischen Mechanik. Im Zentrum steht dabei *Einschritt–Lernen* (*on-line learning*). Darunter versteht man Verfahren, bei denen die Beispiele während des Lernprozesses sequentiell präsentiert werden. Nachdem die lernenden Netzwerke typischerweise eine sehr große Zahl von Freiheitsgraden besitzen, ist eine Beschreibung durch einige wenige Ordnungsparameter erstrebenswert. Dies motiviert einen Zugang mittels Statistischer Physik. Aufgrund der dynamischen Natur der lernenden Systeme werden insbesondere Methoden der Nichtgleichgewichtsthermodyamik für die Analyse benötigt.

Im ersten Teil wird das grundlegende Netzwerk, das Perceptron, untersucht. Neben der Beschreibung mit Methoden der Physik steht die Konstruktion geeigneter Trainingsalgorithmen im Vordergrund. Physikalisch gesehen legt der Algorithmus die Bewegungsgleichung der synaptischen Kopplungen fest. Hier wird gezeigt, wie man ausgehend von fundamentalen Prinzipien eine Lerndynamik herleiten, also einen Algorithmus *berechnen* kann, der zu optimaler Verallgemeinerungsfähigkeit führt. Dabei wird ein Verfahren von Kinouchi und Caticha weitergeführt und auf Lernprozesse angewandt, bei denen die Trainingsdaten verrauscht sind. Für zwei unterschiedliche Arten von Rauschprozessen werden optimale Algorithmen berechnet.

Die gewonnenen optimalen Algorithmen hängen von den Charakteristika des jeweiligen Lernproblems ab, hier von der Art und Stärke des Rauschprozesses. Im allgemeinen benötigt man daher ein zusätzliches Verfahren, das Schätzwerte für solche charakteristischen Größen liefert. Für die Stärke des Rauschprozesses führt eine solche Schätzung zu interessanten dynamischen Phasenübergängen.

Der zweite Teil dieser Arbeit behandelt die dynamischen Eigenschaften von Lernprozessen in Zweischichtnetzwerken. Dies ist von besonderem Interesse, weil solche Netzwerke universale Approximatoren darstellen, d.h. sie können stetige Funktionen mit beliebiger Genauigkeit approximieren. Ein Verständnis der dynamischen Eigenschaften solcher Netzwerke ist notwendig, um schnelle Trainingsalgorithmen konstruieren zu können, die zu einer optimalen Verallgemeinerungsfähigkeit führen.

Die Dynamik von Zweischichtnetzwerken, die durch Gradientenabstieg (Backpropagation–Algorithmus) trainiert werden, wird exakt berechnet. In der Analyse steht die Anpassung der Freiheitsgrade im Mittelpunkt, die die verborgene Schicht mit der Ausgabe verbinden. In den Zweischichtnetzwerken, die bisher in der Literatur behandelt wurden, wurde die Dynamik dieser

Freiheitsgrade vereinfachend nicht berücksichtigt. Die hier gewonnenen Ergebnisse werden mit den Ergebnissen für solche Netzwerke verglichen, bei denen die Freiheitsgrade in der zweiten Schicht (Ausgabeschicht) von vorneherein bekannt sind. Es wird gezeigt, daß beim Einschritt–Lernen in Zweischichtnetzwerken einige wesentliche Eigenschaften von Zustand und Dynamik der zweiten Schicht unabhängig sind. Dies motiviert die Idee, die Dynamik der zweiten Schicht adiabatisch an die der ersten Schicht zu koppeln. Die Dynamik der zweiten Schicht verläuft dann auf einer wesentlich schnelleren Zeitskala. Obwohl die Freiheitsgrade der zweiten Schicht dadurch ihre Selbstmittelungseigenschaften verlieren, führt dies insgesamt zu einer Beschleunigung des Lernprozesses.

Für alle untersuchten Systeme wurden zur Bestätigung der theoretischen Ergebnisse Simulationen durchgeführt.

Contents

Chapter 1

Introduction

The past decade has seen the formation of new research fields, crossing traditional boundaries between biology, computer science, physics, and others. Known variously as learning theory, theory of neural networks, and biological computation, the common theme is a focus on adaptation in natural and artificial systems. This research has the aim both to further our understanding of living and adaptive mechanisms in nature, and to construct artificial systems which show the same flexibility, robustness, and capacity for adaptation as is seen in living creatures.

Natural as well as artificial adaptive systems typically consist of a large number of degrees of freedom. For instance, on average each neuron in the human brain is connected to 10^4 other neurons, therefore representing a system of 10^4 degrees of freedom (synaptic couplings). Artificial networks often consist of hundreds of neurons connected to each other, where again each connection represents a single degree of freedom.

This large number of variables suggests a statistical mechanics approach in order to describe the properties of these systems. Statistical mechanics allows one to compress the laws governing the microscopic behaviour of each single degree of freedom into a macroscopic theory which only depends on a small number of variables, the *order parameters*.

Of course, a description in terms of a small number of order parameters facilitates the analysis of such systems. It should be emphasized, however, that the physicist's approach is not just an alternative to the approaches of other scientific communities. The tools provided by equilibrium and nonequilibrium statistical mechanics to describe large interacting systems enable us to obtain new nontrivial results.

Furthermore, the statistical mechanics of neural networks comprises far more than only a description of features of neural nets. As the equation of motion for a particle in a harmonic potential is applicable to more than only a mass attached to a spring, the results gained in this field provide a conceptual tool that allows an understanding of other phenomena in other branches of science. Related systems and application appear not only in biology but also in information theory, complex optimization, or even financial theory.

In this introductory chapter Section 1.1 presents the very basics of artificial neural networks. Section 1.2 introduces the concept of on-line learning. Both sections are intended to make this thesis selfcontained. They can be skipped by the reader familiar with the topic. In Section 1.3 basic concepts from physics are used in order to comprise the microscopic on-line dynamics with

its large number of degrees of freedom into a small number of macroscopic equations of motion. The last section gives an outline of this thesis.

1.1 Artificial Neural Networks

Many of the tasks that we routinely perform during our daily activities have been learnt through experience. Although these tasks may appear to be simple, it is often very difficult to find algorithms on the basis of which such tasks can be carried out reliably. This observation has prompted the search for systems that can learn from examples. Prominent among these are so-called *neural networks*.

Artificial neural networks resemble their biological counterparts in the essential features: Computation and information processing are done in a parallel manner. This is in contrast to computers (Turing machines) where computation and information processing are done sequentially.

Like in biological neural networks the basic unit (the neuron) takes the outputs of many other units as inputs, performs some sort of computation on them and returns the result as the output. For artificial neural networks this basic processing unit is called *perceptron*. Typically, a neural network consists of many of these basic units. As in the nervous system there is a plethora of ways how perceptrons can be combined to build larger structures. The way this is done depends on the desired functionality. The possible network architectures can roughly be grouped into two classes, attractor networks and feedforward networks; this applies for artificial as well as biological systems.

In *attractor networks* the connectivity is very large. In the extreme case, each unit is connected with every other unit in the system. These networks have been shown to realize an *associative memory*, e.g. the Hopfield model [Hop92].

In contrast, *feedforward networks* have a layered structure of units. This type of architecture resembles areas of the vertebrate brain that perform sensoric tasks or motor control. In the extreme case, there are no feedback connections between units in different layers and no lateral connections between units in the same layer. The information is processed from the input unit to the next layer, which in turn feeds its outputs to the next layer of units and so forth. Figure 1.1 shows a two-layer feedforward network consisting of an input layer, one layer of so-called hidden units, and a single output unit.[1] The notion *hidden units* stems from the fact that only inputs and outputs are connected to the outside (the environment). All other units are hidden to the environment.

Mathematically, feedforward neural networks represent functions $\mathbb{R}^N \to \mathbb{R}$, mapping the N-dimensional input to the (here) one-dimensional output. The perceptron, for instance, realizes a functional relation of the following form:

$$\sigma = g\left(\sum_{i=1}^{N} J_i \xi_i\right) = g(\boldsymbol{J} \cdot \boldsymbol{\xi}), \tag{1.1}$$

[1] The input layer is a kind of dummy layer since it does not contain any degrees of freedom. Hence, this network is called a *two*-layer network, consisting of two active layers of degrees of freedom.

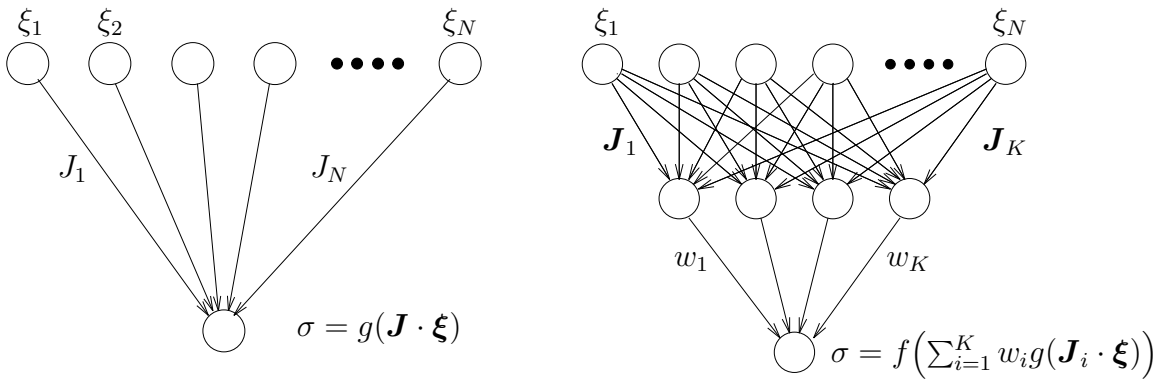


Figure 1.1: Perceptron (left) and two-layer feedforward network (right), showing the notation for units and weights.

where $\boldsymbol{\xi}$ denotes the inputs and σ the output of the perceptron (cf. Fig. 1.1). The components of the vector $\boldsymbol{J}$ are called *weights* or *couplings*.[2] They embody the fundamental degrees of freedom allowing the network to learn from examples. Obviously, different choices of $\boldsymbol{J}$ in Eq. (1.1) realize different functions.

Learning signifies finding weights $\boldsymbol{J}$ such that for a given *training set* of examples (input/output pairs) $\{(\boldsymbol{\xi}^\mu, \tau^\mu)\}_{\mu=1..p}$

$$\tau^\nu \equiv g(\boldsymbol{J} \cdot \boldsymbol{\xi}^\nu) \quad \forall \ (\boldsymbol{\xi}^\nu, \tau^\nu) \in \{(\boldsymbol{\xi}^\mu, \tau^\mu)\}_{\mu=1..p}. \tag{1.2}$$

In general, however, it might not be possible to satisfy (1.2) for all examples in the training set. Then one would search for that solution $\boldsymbol{J}$ which fulfills (1.2) for as many examples as possible. Formulated this way, learning in feedforward networks so far is nothing but a (nonlinear) parametric regression in N dimensions.

However, daily experience tells us that learning comprises more than just the ability to reliably reproduce already learnt training data. There is an additional quality to learning: *generalization*, i.e. the ability to infer information from hitherto unseen similar data. One can think of the training data being generated by a function $\tau(\boldsymbol{\xi})$. Good generalization then corresponds to couplings $\boldsymbol{J}$ such that $g(\boldsymbol{J} \cdot \boldsymbol{\xi})$ is a good approximation to $\tau(\boldsymbol{\xi})$.

It is advantageous to interpret the function $\tau(\boldsymbol{\xi})$ to be represented by another neural network. This network generates the examples used for training. More pictorially, it acts as a teacher, providing the data necessary for training. For this reason an imaginary network which realizes $\tau(\boldsymbol{\xi})$ is usually called the *teacher network*, while the network to be trained is called the *student network*.

Thus, the ultimate goal is to find a student network that well approximates the teacher network, i.e. some metric $\epsilon(\sigma, \tau)$ has to be minimized given the information contained in the training set. To this end the weights $\boldsymbol{J}$ have to be changed according to some algorithm. This training

[2]The weights $\boldsymbol{J}$ play an analogous role as the synaptic couplings in biological neurons.

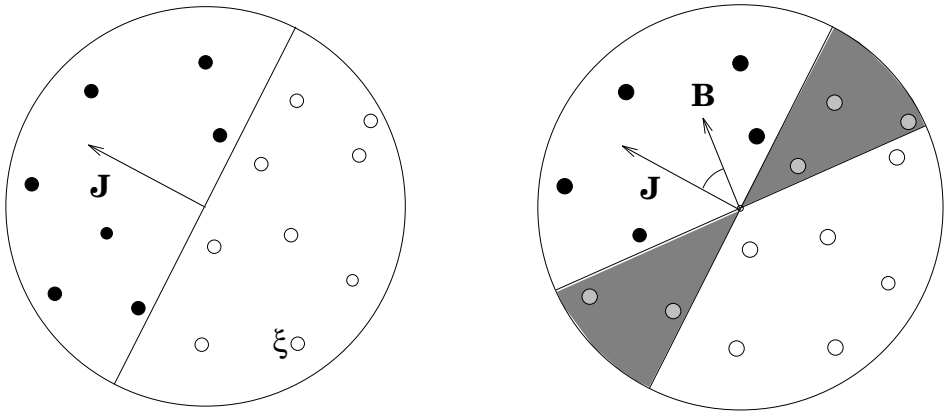


Figure 1.2: The perceptron with binary outputs (1.3) realizes a linearly separable classification. Left: All inputs $\boldsymbol{\xi}$ above (below) the hyperplane defined by $\boldsymbol{J} \cdot \boldsymbol{\xi} = 0$ are mapped to $+1$ (-1); $+1$ is represented by $\bullet$, -1 by $\circ$. Right: Perceptron with weights $\boldsymbol{J}$ learning a linearly separable map represented by a teacher network with weights $\boldsymbol{B}$. The examples that are misclassified by the student vector are situated in the shaded area. For an isotropic distribution of inputs $\boldsymbol{\xi}$ the probability that an example will be misclassified is proportional to the angle between $\boldsymbol{J}$ and $\boldsymbol{B}$.

algorithm is a mathematical prescription allowing to infer the appropriate weights given the training examples.

The choice of an appropriate training algorithm for a given task is one of the central topics in the theory of learning. The generalization ability of a neural network depends in a very essential way on the algorithm used for training. How the training algorithm determines the generalization ability is most easily illustrated by a simple example: Consider a perceptron (1.1) with binary outputs ± 1:

$$\sigma = \mathrm{sign}\,(\boldsymbol{J} \cdot \boldsymbol{\xi}). \tag{1.3}$$

Such a network acts as a *classifier*: The inputs $\boldsymbol{\xi}$ are grouped into (here) two classes which are labeled by $+1$ and -1, respectively. Which label is assigned depends on the sign of the scalar product $\boldsymbol{J} \cdot \boldsymbol{\xi}$ which is the projection of the input vector $\boldsymbol{\xi}$ on the weight vector $\boldsymbol{J}$. Consequently, in input space the regions with different labels are separated by the $(N-1)$-dimensional hyperplane perpendicular to the weight vector $\boldsymbol{J}$ (see Fig. 1.2). The map realized by a perceptron of the type (1.3) is called *linearly separable*.

Obviously, the perceptron (1.3) can implement any linearly separable map: If the training examples are generated by a teacher network of the same architecture, i.e. the training examples satisfy $\tau^\mu = \mathrm{sign}(\boldsymbol{B} \cdot \boldsymbol{\xi}^\mu)$, then a state $\boldsymbol{J} \propto \boldsymbol{B}$ represents a student network that has perfectly learnt the required task; its generalization ability is maximum.

A prominent algorithm for training a perceptron by examples is *Hebb's rule* [HKP91]. It computes the couplings $\boldsymbol{J}$ as the average correlation of input and the corresponding output label in the training set:

$$\boldsymbol{J} = \frac{1}{N} \sum_{\mu=1}^{p} \tau^\mu \boldsymbol{\xi}^\mu = \frac{1}{N} \sum_{\mu=1}^{p} \mathrm{sign}(\boldsymbol{B} \cdot \boldsymbol{\xi}^\mu) \boldsymbol{\xi}^\mu. \tag{1.4}$$

By the geometry of the perceptron (Fig. 1.2) it is easy to see that for an *isotropic* distribution of inputs $\boldsymbol{\xi}$ Hebb's rule leads to the convergence $\boldsymbol{J} \to \boldsymbol{B}$ as $p \to \infty$: First consider the examples that have been labeled by the teacher network with $+1$. Their mean value is $\boldsymbol{B}$ if the inputs are distributed isotropically. Analogously, the mean value of inputs labeled with -1 is $-\boldsymbol{B}$. Consequently the r.h.s. of (1.4) approximates $\boldsymbol{B}$. The approximation is the better the larger the sample size p is.

However, by the same geometry argument it can be seen that Hebb's algorithm in general does not lead to a convergence $\boldsymbol{J} \to \boldsymbol{B}$. This is the case if the distribution of inputs is not isotropic. Imagine, for example, a distribution of $\boldsymbol{\xi}$ that has a characteristic direction different from $\boldsymbol{B}$. Then according to Hebb's rule the couplings $\boldsymbol{J}$ will not converge to the desired teacher vector $\boldsymbol{B}$ but will be biased towards the characteristic direction of the distribution. For instance if the distribution consists of two overlapping clusters, $\boldsymbol{J}$ will converge to a linear combination of $\boldsymbol{B}$ and the vector that describes the cluster separation [RBSM96].

This exemplifies how important the choice of an appropriate training algorithm is. Obviously, the quality of an algorithm depends on many factors, including the distribution of inputs. Also it can make a difference whether or not the training set is split into parts for training. In the theory of learning one essentially distinguishes between two classes of training algorithms: batch and on-line algorithms. The focus of this work is on the latter type of algorithms which will be introduced in the following section.

1.2 On-line Learning

In learning theory much effort has been taken to analyze *batch* or *off-line learning*, in which the learner has free access to a fixed set of examples $\{(\boldsymbol{\xi}^\mu, \tau^\mu)\}_{\mu=1..p}$ stored in memory. This learning situation naturally leads to a description in terms of equilibrium statistical mechanics [HKP91, SST92, WRB93]. The approach is based on an energy function which is the learner's error on the training set.

For instance, for the perceptron discussed in the last section (Eq. (1.3)) an often used energy function is the *training error*

$$\epsilon_t(\boldsymbol{J}, \{\boldsymbol{\xi}\}) = \sum_{\mu=1}^{p} \Theta(-\operatorname{sign}(\boldsymbol{J} \cdot \boldsymbol{\xi}^\mu)\tau^\mu) \tag{1.5}$$

which measures the number of misclassified examples in the training set. By calculating an appropriate partition function one can study the equilibrium properties of this system in the thermodynamic limit, i.e. for a large number N of degrees of freedom $\boldsymbol{J}$.

Note that in general one is interested in typical results, i.e. results that are independent of the particular choice of the training set. In addition to the thermal average one therefore has to average over all possible sets of training examples $\boldsymbol{\xi}$. This average over the distribution of $\boldsymbol{\xi}$ is equivalent to the average over the disorder known from the physics of disordered systems [MPV87].

From a theoretical point of view the advantage of this equilibrium formulation is that the generalization ability as a function of the number of examples can be investigated without ad-

dressing the complex dynamical aspects of learning. The disadvantage is that one does not gain any information about the learner's performance as a function of training time.

Many learning phenomena, however, do not fit into the framework of batch learning. They are characterized by a sequential presentation of learning examples. Such learning procedures are fundamentally dynamical in nature and, hence, in general one cannot assume that the system is in equilibrium. In learning theory the notion *on-line learning* has been coined for this type of learning algorithms (e.g. [Ama67, HK93]). The examples are presented one at a time, i.e. on-line[3], and are discarded afterwards.

The weight change due to the actual example is of the following form:[4]

$$J_i^{p+1} = J_i^p + \frac{1}{N} F(\boldsymbol{J}^p, \boldsymbol{\xi}^p, \tau^p)\xi_i^p. \tag{1.6}$$

The current example is weighted by F and added to the present couplings $\boldsymbol{J}^p$. The scalar function F is known as *weight function*. It is only allowed to depend on quantities that are accessible to the student network. The scaling of the weight function with $1/N$ keeps the length of $\boldsymbol{J}$ of $\mathcal{O}(1)$. This will become clear in the following section. The index p denoting the number of examples so far presented can be viewed as a discrete time index. Due to the randomness of the presentation of the examples $\boldsymbol{\xi}$ the dynamics of the couplings $\boldsymbol{J}$ is stochastic. Moreover, the state at time $p+1$ exclusively depends on the state at the previous time step and the latest example. Hence, the on-line dynamics (1.6) of the couplings $\boldsymbol{J}$ is a first order Markov process [vK92] representing the microscopic description of the time evolution.

For on-line learning the weight function used for updating the weight vector $\boldsymbol{J}$ represents the *algorithm* used for training. The weight function for Hebb's algorithm considered in the last section is given by $F = \tau^\mu$. As pointed out earlier the performance of a network essentially depends on the choice of algorithm or weight function. An appropriate quantity to measure the performance of on-line learning is the generalization error. Given the present state of a network this is the average error that an arbitrary example will be misclassified. For the simple perceptron with binary output (1.3) the generalization error measures the correlation of the outputs of teacher and student network on random, hitherto unseen inputs:

$$\epsilon_g(\boldsymbol{J}^p) = \langle \Theta(-\operatorname{sign}(\boldsymbol{J}\cdot\boldsymbol{\xi})\tau(\boldsymbol{\xi}))\rangle_\xi. \tag{1.7}$$

The more the outputs are correlated the less the generalization error will be. For couplings $\boldsymbol{J} \propto \boldsymbol{B}$ the generalization error will be perfectly zero, i.e. the student network has become a perfect representation of the function $\tau(\boldsymbol{\xi}) = \operatorname{sign}(\boldsymbol{B}\cdot\boldsymbol{\xi})$.

[3]Unfortunately the usage of "on-line" is not unique in the literature. It differs somewhat from discipline to discipline. Here it is only used as defined in the text.

[4]Note that this is not the most general form. The update rule (1.6) depends on the symmetry of the input distribution and the *a priori* knowledge about the properties of the rule to be learnt. For instance, for correlated examples $\langle \xi_i \xi_j \rangle \neq \delta_{ij}$ the weight function F could be a tensor which allows one to cope with the structural peculiarities of the input space. However, throughout this work the examples $\boldsymbol{\xi}$ are exclusively assumed to be uncorrelated.

Furthermore, the r.h.s. of (1.6) could contain an additional term proportional to J_i^p. Since such a term only changes the length of the vector $\boldsymbol{J}$ but not its direction it is not relevant for the case of the perceptron (1.3) which is invariant under a transformation $\boldsymbol{J} \to \gamma\boldsymbol{J}$. Moreover, for the backpropagation algorithm in multi-layer networks considered in Chapter 3 such a term does not occur by definition of the algorithm. It should be emphasized, however, that the computational tools developed in this work would also apply if such a term had been taken into account.

On-line learning is well suited for a plethora of learning scenarios. Examples are training sets that are generated by some time series, or a rule that changes with time. Clearly, in the latter case the performance of on-line learning algorithms will be superior to that of batch learning. In contrast to batch learning, on-line learning does not require the storage of the whole training set since only the latest in a series of examples is taken into account for changing the couplings. Storing can be very expensive in applications.

1.3 From Microscopics to Macroscopics

Neural networks learn from examples by modifying the values of their internal degrees of freedom. For on-line learning this is done according to Eq. (1.6). The success of the physical description of neural networks is largely based on the fact that the large number of nonlinearly interacting degrees of freedom $\boldsymbol{J}$ can be comprised into a small number of order parameters and applying standard tools of statistical physics.

In the following, appropriate order parameters will be derived for perceptron learning. This will allow to transform the equation of motion (1.6) for the N many microscopic degrees of freedom $\boldsymbol{J}$ into a set of only a few macroscopic equations of motion for these order parameters.

Which quantities are appropriate order parameters essentially depends on the symmetries of the problem, in particular on the properties of the input distribution. Throughout this work the examples $\boldsymbol{\xi}$ will be considered to be independently drawn vectors with uncorrelated random components of zero mean and unit variance:

$$\langle \xi_i \rangle = 0, \quad \langle \xi_i \xi_j \rangle = \delta_{ij}. \tag{1.8}$$

Furthermore, it will be assumed that no component of the teacher vector $\boldsymbol{B}$ is distinguished.[5] Hence, all microscopic quantities are completely invariant under rotations in the N-dimensional space. Consequently, the order parameters have to be invariant under rotation as well. This immediately restricts the possible candidates to the following two overlaps (and any function of them):

$$R \equiv \boldsymbol{B} \cdot \boldsymbol{J}, \quad Q \equiv \boldsymbol{J} \cdot \boldsymbol{J}. \tag{1.9}$$

The dynamics of these order parameters is easily obtained from the on-line dynamics (1.6) by taking the scalar products with $\boldsymbol{B}$ and $\boldsymbol{J}^{p+1}$, respectively:

$$\begin{aligned} R^{p+1} &= R^p + \frac{1}{N} \langle F(H_J^p, \tau^p; Q^p) H_B^p \rangle_\xi \\ Q^{p+1} &= Q^p + \frac{1}{N} \Big\langle 2F(H_J^p, \tau^p; Q^p) H_J^p + F^2(H_J^p, \tau^p; Q^p) \Big\rangle_\xi . \end{aligned} \tag{1.10}$$

In order to obtain typical results the randomness in the presentation of the inputs $\boldsymbol{\xi}$ has been averaged out. This average is denoted by $\langle \ldots \rangle_\xi$. Note that due to the assumed symmetry the weight function can only depend on rotation-invariant quantities as well.

[5] In the language of Bayesian Inference this corresponds to an isotropic prior distribution for the components B_i.

The randomness of the inputs $\boldsymbol{\xi}$ enters the dynamics (1.10) exclusively through the *internal fields*

$$H_J \equiv \boldsymbol{J} \cdot \boldsymbol{\xi} \quad \text{and} \quad H_B \equiv \boldsymbol{B} \cdot \boldsymbol{\xi}, \tag{1.11}$$

which have zero mean and correlations

$$\left\langle H_J^2 \right\rangle = |\boldsymbol{J}|^2 = Q, \quad \left\langle H_B^2 \right\rangle = |\boldsymbol{B}|^2, \quad \text{and} \quad \langle H_J H_B \rangle = \boldsymbol{J} \cdot \boldsymbol{B} = R. \tag{1.12}$$

Therefore the average over $\boldsymbol{\xi}$ in (1.10) can be replaced by the average over the joint probability $P(H_J, H_B)$ of the internal fields. In the thermodynamic limit $N \to \infty$ this distribution is a Gaussian according to the Central Limit Theorem. Thus, $P(H_J, H_B)$ is completely determined by the first and second moments of the internal fields. Therefore it depends on the order parameters R and Q, according to (1.12). See Appendix A for the functional form of this distribution. It is important to point out that after averaging (1.10) over the internal fields one obtains a recursion relation for the order parameters R and Q in a closed form, i.e. the right hand sides depend on the order parameters only.

Defining $\alpha = p/N$ as the number of examples so far presented in the training process, scaled with the number of inputs N, one can interpret α as a continuous macroscopic time variable. Since the infinitesimal is $d\alpha = 1/N$ the recursion relation (1.10) can be cast into the form of first order differential equations for the order parameters R and Q:

$$\begin{aligned} \frac{dR}{d\alpha} &= \langle F(H_J, \tau; Q) H_B \rangle_{P(H_J, H_B)} \\ \frac{dQ}{d\alpha} &= \left\langle 2F(H_J, \tau; Q) H_J + F^2(H_J, \tau; Q) \right\rangle_{P(H_J, H_B)}. \end{aligned} \tag{1.13}$$

Consequently, the set of N microscopic equations of motion (1.6) for the couplings $\boldsymbol{J}$ has been replaced by a set of only two differential equations for the macroscopic order parameters. The essential feature of order parameters is that they are selfaveraging in the thermodynamic limit, i.e. their fluctuations vanish as $N \to \infty$. That this is indeed the case can be shown selfconsistently by means of a van Kampen system size expansion [vK92, WH94]. An alternative way to justify the assumed selfaveraging properties of R and Q will be given in Appendix D.

For the perceptron with binary outputs (1.3) there is an additional inherent symmetry: Since the output is given by the sign of the internal fields the length of the weight vectors does not matter. Hence, an appropriate order parameter for this situation is the *normalized* student-teacher overlap

$$\rho \equiv \frac{\boldsymbol{J} \cdot \hat{\boldsymbol{B}}}{\sqrt{Q}} = \frac{R}{\sqrt{Q}}. \tag{1.14}$$

Here, the teacher vector $\hat{\boldsymbol{B}}$ has been normalized without loss of generality. It is also convenient to normalize the internal fields:

$$h_J \equiv \frac{\boldsymbol{J} \cdot \boldsymbol{\xi}}{\sqrt{Q}}, \quad h_B \equiv \hat{\boldsymbol{B}} \cdot \boldsymbol{\xi}. \tag{1.15}$$

The resulting *normalized internal fields* are of zero mean and unit variance. Their correlation is $\langle h_J h_B \rangle = \hat{\boldsymbol{J}} \cdot \hat{\boldsymbol{B}} = \rho$ and therefore their joint distribution is given by the Gaussian

$$P(h_J, h_B) = \frac{1}{2\pi\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2}\frac{h_J^2 - 2\rho h_J h_B + h_B^2}{1-\rho^2}\right) \tag{1.16}$$

in the thermodynamic limit (cf. Appendix A).

The corresponding equations of motion can be easily obtained from (1.13):

$$\begin{aligned} \frac{d\rho}{d\alpha} &= \left\langle \frac{1}{\sqrt{Q}} F(h_J, \tau; Q)(h_B - \rho h_J) - \frac{\rho}{2Q} F^2(h_J, \tau; Q) \right\rangle_{P(h_J, h_B)} \\ \frac{dQ}{d\alpha} &= \left\langle 2\sqrt{Q} F(h_J, \tau; Q) h_J + F^2(h_J, \tau; Q) \right\rangle_{P(h_J, h_B)} . \end{aligned} \tag{1.17}$$

Note that although the length of the teacher vector can be normalized this cannot be done for the student vector. The on-line dynamics (1.6) does not preserve the norm of the student vector, in general.

Using the distribution (1.16) it is an easy task to compute the generalization error (1.7) of a perceptron with binary outputs (1.3) learning a realizable task:

$$\epsilon_g(\alpha) = \frac{1}{\pi} \arccos \rho(\alpha). \tag{1.18}$$

Thus, for an isotropic distribution of inputs the generalization error corresponds to the angle between teacher and student vectors (apart from a factor of π; cf. Fig. 1.2).

1.4 Outline of this Thesis

The research of this thesis aims towards two goals: The derivation of meaningful and nontrivial results for on-line learning situations and the development of mathematical tools in order to obtain these results. This will be done for two types of networks: perceptrons and two-layer networks. Being the fundamental neural network, developing and testing new nontrivial concepts is most easily done for the perceptron.

The previous section has shown how the dynamics of perceptron learning can be sufficiently described by a set of two first order differential equations. Obviously the only task that remains is to think about a clever algorithm. One then plugs the corresponding weight function into the right hand sides of these differential equations and solves for the solution. Even better, one could parametrize the weight function and by playing around with the parameters one could search for nice weight functions with nice properties, e.g. a fast increase of the generalization error.

Obviously such a "plug in and play" procedure is not very satisfactory. Instead of an heuristic approach a way to obtain algorithms with a desired functionality starting from first principles is certainly preferable. That this can be easily done for perceptron learning will be shown in Chapter 2. Specifically, algorithms will be *derived* that have two properties: First, they are robust with respect to noise and second, they result in a generalization error which is optimal at each time

α. This means there is no algorithm that typically gives rise to a lower generalization error at a given α.

Chapter 3 focuses on multi-layer networks. Although being ubiquitous in applications not very much about their dynamical properties is understood so far. Even less is known about how to describe them by theoretical methods. The two-layer networks which have been investigated so far in the physics literature have fixed hidden-to-output weights. This restriction essentially makes these networks be an ensemble of perceptrons.

Here it will be shown, how adaptive hidden-to-output weights can be incorporated into the analysis. This is done for the case of a finite number of hidden-to-output weights. Although this is not the most general two-layer network the obtained results will justify the investigation of this limit as a first step. More important than that, however, is that the emerging concepts will turn out to be of value not only for learning in multi-layer networks but also for other on-line learning processes.

Chapter 2

Learning from Noisy Data

Besides their ability to learn a rule from examples, one of the striking features of artificial as well as natural neural networks is their tolerance to noise. They can extract information from the training set even if the latter is noisy or if the state of the network itself is corrupted by noise. Learning from a noisy data set bears a lot of interesting questions:

- If the task under consideration is learnable in the noiseless case, is it perfectly learnable in the presence of noise as well, or does a certain finite error remain?
- How do the results depend on the amount of noise corrupting the learning process?
- Which features must a learning algorithm have in order to successfully cope with the corrupting noise?

These questions will be answered in this chapter for the case of a linearly separable rule. The main focus will be on the third of the above questions. Following a streamlined and powerful scheme introduced by Kinouchi and Caticha [KC92a, KC92b] algorithms will be derived which automatically reveal the features necessary for an optimal learning of the given task.

It will turn out that the gained algorithms are *parameterfree*. The algorithms only depend on quantities describing the physics of the problem. However, they do not depend on parameters introduced in an *ad hoc* fashion, like for example a learning rate.

This chapter is organized in the following way: Section 2.1 introduces the specific learning task investigated in this chapter. Next, the concept of optimal feature algorithms will be introduced and in Section 2.3 this will be applied to the case where the outputs of the training examples are noisy (output noise). Section 2.4 proceeds along the same line for the case that the weights of the teacher network are corrupted by noise (weight noise). In Section 2.5 it will be shown how one can estimate the seemingly inaccessible quantities the optimal algorithm will turn out to make use of. Section 2.6 discusses the optimal algorithms for both weight and output noise for a more sophisticated learning scenario where the learning network can choose its training examples to a certain extent. Finally, in Section 2.7 a thorough discussion of the advantages of the optimal feature algorithms will be given. Parts of the results of this chapter have already been published in [BRS95]. Some of them have also been obtained parallelly by [CKC96].

2.1 The Framework

The specific network architecture considered in this chapter is the thresholded perceptron. As discussed in Section 1.1 such a simple feed-forward neural network realizes a linearly separable input/output relation of the form

$$\sigma(\boldsymbol{\xi}) = \mathrm{sign}\,(\boldsymbol{J}\cdot\boldsymbol{\xi}) \tag{2.1}$$

where $\boldsymbol{\xi}$ represents a N-dimensional input vector. $\boldsymbol{J}\in I\!R^N$ is the vector of perceptron weights, which are to be adapted in the course of the training process.

The rule to be learnt is defined through a *teacher* vector $\boldsymbol{B}\in I\!R^N, \boldsymbol{B}^2=1$:

$$\tau(\boldsymbol{\xi}) = \mathrm{sign}(h_B) \quad \text{with} \quad h_B = \boldsymbol{B}\cdot\boldsymbol{\xi}. \tag{2.2}$$

Since the student and the teacher network are of the same architecture, the concept is in principle learnable for the perceptron student (2.1). However, assume that a sequence of examples $\{\boldsymbol{\xi}^\mu, \tau_n^\mu\}$ is provided for training, where the example labels $\tau_n^\mu = \pm 1$ can differ from τ^μ due to some stochastic process which will be specified later on. Then it is not *a priori* clear whether the student network can realize the true concept (2.2) despite the noise.

The generic form of on-line perceptron training considered here is (cf. Eq. (1.6))

$$\boldsymbol{J}^{\mu+1} = \boldsymbol{J}^\mu + \frac{1}{N}F(h_J^\mu, \tau_n^\mu, Q^\mu)\boldsymbol{\xi}^\mu \tag{2.3}$$

where the weight function F defines the actual algorithm. This function depends on quantities available to the student, such as the (noisy) example label τ_n^μ, the student norm $Q^\mu = \boldsymbol{J}^\mu\cdot\boldsymbol{J}^\mu$, and the normalized local field $h_J^\mu = \boldsymbol{J}^\mu\cdot\boldsymbol{\xi}^\mu/\sqrt{Q^\mu}$.

The input vectors are taken to be N-dimensional vectors of independent and identically distributed random components ξ_j^μ with zero mean and unit variance. At each "time step" μ a new, uncorrelated example is drawn and used for an update of $\boldsymbol{J}^\mu$ according to Eq. (2.3). Proceeding as in Chapter 1 it is straightforward to obtain recursion equations for the quantities Q^μ and $\rho^\mu = \boldsymbol{J}^\mu\cdot\boldsymbol{B}/\sqrt{Q^\mu}$. In the limit $N\to\infty$ these overlaps are selfaveraging with respect to the randomness in the training data. The evolution of these order parameters in "continuous time" $\alpha = \mu/N$ is given in terms of first order differential equations (cf. Eq. (1.17)):

$$\begin{aligned}\frac{d\rho}{d\alpha} &= \left\langle \frac{F}{\sqrt{Q}}(h_B - \rho h_J) - \frac{\rho}{2}\frac{F^2}{Q}\right\rangle_{\xi,\tau_n}\\ \frac{dQ}{d\alpha} &= \left\langle 2\sqrt{Q}Fh_J + F^2\right\rangle_{\xi,\tau_n}\end{aligned} \tag{2.4}$$

where $\langle\ldots\rangle_{\xi,\tau_n}$ denotes the averages over the input-distribution and also over the randomness in the evaluation of τ_n. It is convenient to write the weight function F in the form

$$F(h_J^\mu, \tau_n^\mu, Q^\mu) = \sqrt{Q^\mu}f(h_J^\mu, \tau_n^\mu, Q^\mu). \tag{2.5}$$

In terms of the *scaled weight function* f the equations of motion read

$$\begin{aligned} \frac{d\rho}{d\alpha} &= \left\langle f(h_B - \rho h_J) - \frac{\rho}{2} f^2 \right\rangle_{\xi,\tau_n} \\ \frac{dQ}{d\alpha} &= Q \left\langle 2 f h_J + f^2 \right\rangle_{\xi,\tau_n} \end{aligned} \tag{2.6}$$

In the following the term weight function will be used invariably for both F and f and a distinction will be made only where necessary.

Given the weight function f and a specific type of noise, one can work out the corresponding differential equations analytically. They can be integrated at least numerically and hence one obtains the evolution of the order parameters with α, the scaled number of examples used for training. In the following the initial conditions considered will be $\rho(0) = Q(0) = 0$ exclusively. The initial condition for ρ corresponds to the most likely one for any choice of $Q(0)$ as $N \to \infty$, whereas the initial condition for Q represents a start from *tabula rasa*, i.e. all weights are set to zero initially.

It is useful to distinguish two performance measures, the *generalization error*

$$\epsilon_g = \langle \Theta(-h_J \tau) \rangle_\xi = \frac{1}{\pi} \arccos \rho \tag{2.7}$$

and the *prediction error*

$$\epsilon_p = \langle \Theta(-h_J \tau_n) \rangle_{\xi,\tau_n} . \tag{2.8}$$

The quantity ϵ_g is the probability for disagreement between the student and the genuine rule output and the average is over the input distribution only. In contrast, ϵ_p compares the student output with the noisy τ_n for an arbitrary input. For a student with normalized overlap ρ with the teacher, the generalization error is given by $\epsilon_g = (1/\pi) \arccos \rho$ on average over the uniform input distribution [WRB93]; cf. (1.18). Obviously, ϵ_g is zero for perfect alignment $\boldsymbol{J} \propto \boldsymbol{B}$. The minimal ϵ_p, however, will remain non-zero in general; of course it is impossible to predict the randomized τ_n without errors. The relation between generalization and prediction error will depend on the specific noise considered.

A simple choice for the weight function is $F = \tau_n$, which corresponds to Hebbian learning [Val89]. For this case it is easy to work out and solve the differential equations for the two different types of training noise considered in Sections 2.3 and 2.4. Various other learning schemes have also been considered in the literature, for both noiseless and noisy training (e.g. [KR90, BS93, BR94, BSS95, Ste95]). Instead of testing this plethora of available algorithms it is certainly more appealing if one could determine an algorithm that performs best for the task under consideration. This will be achieved in the next section.

2.2 The Optimal Feature Algorithm

The most desirable training algorithm would certainly be the one that converges to the best solution as fast as possible. The best solution is characterized by the lowest generalization error

ϵ_g achievable. Consequently, the optimal algorithm is the one which approaches the minimum of ϵ_g at the fastest rate, i.e. with a maximum temporal change of the generalization error $(-d\epsilon_g/d\alpha)$. For the noiseless case these considerations led Kinouchi and Caticha [KC92a, KC92b] to the construction of the optimal training algorithm.

Since $\epsilon_g = (1/\pi)\arccos\rho$ is a monotonic function of the student-teacher overlap ρ the minimization of $d\epsilon_g/d\alpha$ is equivalent to the maximization of the change of ρ per training time α. Hence, the optimal algorithm is defined by that weight function f_{opt} that maximizes

$$\begin{aligned}\frac{d\rho}{d\alpha} &= \Big\langle f(h_J,\tau_n,Q)(h_B-\rho h_J) - \frac{\rho}{2}f^2(h_J,\tau_n,Q)\Big\rangle_{P(h_J,h_B,\tau_n)} \\ &= \int dh_J d\tau_n P(h_J,\tau_n)\Big[f(h_J,\tau_n,Q)\Big(\langle h_B\rangle_{P(h_B|h_J,\tau_n)} - \rho h_J\Big) \\ &\qquad - \frac{\rho}{2}f^2(h_J,\tau_n,Q)\Big], \end{aligned} \tag{2.9}$$

where $P(\tau_n)$ formally denotes the statistics of the noise that gives rise to the corrupted output τ_n. Here, the average over the randomness of $\boldsymbol{\xi}$ in (2.6) has been replaced by the equivalent average over the fields h_B and h_J.

Varying the last equation with respect to the weight function f results in the *optimal weight function*[1]

$$f_{opt}(h_J,\tau_n,Q) = \frac{1}{\rho}\langle h_B\rangle_{P(h_B|h_J,\tau_n)} - h_J. \tag{2.10}$$

Inserting f_{opt} back into (2.9) one obtains the equation of motion for a student network trained by this on-line algorithm. It will be shown in Section 2.5.1 that this equation of motion can be cast into the form

$$\frac{d\rho}{d\alpha} = \frac{\rho}{2}\Big\langle f_{opt}^2(h_J,\tau_n,Q)\Big\rangle = \frac{\rho}{2}\left\langle\left(\frac{1}{\rho}\langle h_B\rangle_{P(h_B|h_J,\tau_n)} - h_J\right)^2\right\rangle. \tag{2.11}$$

From this it is not difficult to see that $d\rho/d\alpha$ vanishes at $\rho = 1$, since at this point the internal fields h_J and h_B are completely correlated ($\rho \equiv \langle h_J h_B\rangle$). Therefore, by training a perceptron with the weight function f_{opt} the generalization error assumes its minimum value at $\epsilon_g = 0$ even if the training set is corrupted by noise! Moreover, by construction the approach to the minimum $\epsilon_g = 0$ is on average faster than for any other weight function.

In addition to leading to the fastest typical convergence for a sequence of αN examples the functional form of f_{opt} contains the features indispensable for this. In other words, the variational principle can be used to obtain the features an algorithm must exploit in order to successfully cope with the given learning task. Hence such algorithms are optimal feature algorithms. Their features will be investigated in the consecutive sections.

Note that f_{opt} in (2.10) explicitly depends on the statistics of the noise process that corrupts the output τ_n through the probability $P(h_B|h_J,\tau_n)$. To begin with, it will be assumed that the type

[1] The second variation of (2.9) with respect to f yields as the necessary condition for a maximum the reasonable condition $\rho > 0$.

and strength of the noise process are known to the student. This information can therefore be used and determines the specific form of f_{opt} according to (2.10). Inserting f_{opt} into (2.6) and solving the resulting differential equations will then yield the optimal on-line learning curve $\epsilon_g(\alpha)$ that can be achieved by an algorithm of the form (2.3). Later, in Section 2.5, this assumption will be abandoned.

2.3 Output Noise

In this first scenario the rule outputs are inverted independently with a probability $\lambda \leq 1/2$:

$$\tau_n = \begin{cases} \tau & \text{with probability } 1-\lambda \\ -\tau & \text{with probability } \lambda \end{cases}, \tag{2.12}$$

where τ is the output of the teacher perceptron (2.2) in the noiseless case. Thus, only a fraction $(1-\lambda)$ of the examples provides correct information about the rule. This type of noise was considered in [OH91b, OH91a] in the framework of batch learning, and recently in [Hes94, BSS95] for on-line learning with a perceptron.

Here, the random inputs enter only through the fields h_B and h_J. Their joint distribution is a two-dimensional Gaussian with zero means, unit variances and correlation $\langle h_J h_B \rangle_\xi = \rho$ (see Section 1.3) In addition one has to average explicitly over the randomness in τ_n according to (2.12). This can be done following the general scheme

$$\begin{aligned} \langle g(h_J, h_B, \tau_n) \rangle_{P(h_J,h_B,\tau_n)} &= \Big\langle \lambda g(h_J, h_B, -\operatorname{sign}(h_B)) \\ &\quad + (1-\lambda) g(h_J, h_B, \operatorname{sign}(h_B)) \Big\rangle_{P(h_J,h_B)} \end{aligned} \tag{2.13}$$

since the distribution of the noise that gives rise to τ_n and the distribution of the internal fields h_J, h_B are independent from each other in the case considered here.

Applying (2.13) to the definition (2.8) it is easy to verify that the prediction error obeys the relation

$$\epsilon_p = \lambda + (1-2\lambda)\epsilon_g \geq \lambda. \tag{2.14}$$

As explained in Section 2.1 ϵ_p is bounded from below by the inversion rate λ due to the randomness of τ_n.

Note that according to (2.12) output noise is isotropic in input space. The output of any input example $\boldsymbol{\xi}$ can be affected by the noise irrespective of the position of $\boldsymbol{\xi}$ relative to the system's characteristic directions $\boldsymbol{B}$ and $\boldsymbol{J}$.

2.3.1 Hebbian Learning

For Hebbian learning ($F = \tau_n$) the differential equations (2.4) are most easily solved in terms of the unnormalized student-teacher overlap $R = \boldsymbol{J} \cdot \boldsymbol{B} = \rho\sqrt{Q}$:

$$\frac{dR}{d\alpha} = \sqrt{\frac{2}{\pi}}(1-2\lambda)$$

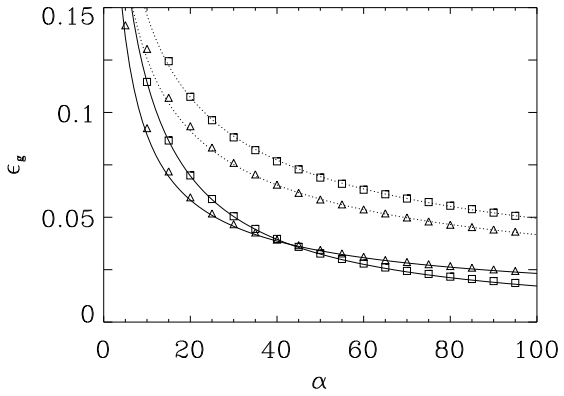


Figure 2.1: Learning curves $\epsilon_g(\alpha)$ in the presence of output noise ($\Box$) with $\lambda = 0.1$ and weight noise ($\triangle$) with $\omega = \cos(0.1\pi)$. The dotted lines are for simple Hebbian learning, the solid lines correspond to the respective optimal weight functions (2.17,2.32). The symbols represent the results of simulations for input dimension $N = 1000$, averaged over 100 independent runs. Error bars would be smaller than the symbol size.

$$\frac{dQ}{d\alpha} = 2\sqrt{\frac{2}{\pi}}(1-2\lambda)R + 1. \tag{2.15}$$

This set of equations is analytically solvable for all α. One obtains for the generalization error (2.7)

$$\epsilon_g(\alpha) = \frac{1}{\pi}\arccos\left[\left(1 + \frac{\pi}{2(1-2\lambda)^2\alpha}\right)^{-1/2}\right]. \tag{2.16}$$

The rule is learnt perfectly for $\lambda < 1/2$ as $\alpha \to \infty$ despite the noise corrupting the teacher output. Moreover, the asymptotic generalization error decays like $\alpha^{-1/2}$ as for noise free learning [Val89] but with a modified prefactor which gives rise to a slower decay of the generalization error the larger the noise is.

Note that this prefactor diverges for $\lambda \to 1/2$, which corresponds to training labels completely uncorrelated with the rule. In this case $\epsilon_g \equiv 1/2$, reflecting the fact that the output does not convey any information about the underlying rule. Hence, the correct output can only be guessed randomly.

Figure 2.1 shows the learning curve $\epsilon_g(\alpha)$ for $\lambda = 0.1$ together with the results of numerical simulations.

2.3.2 Optimal Weight Function

For the optimized weight function one obtains after carrying out the conditional average in (2.10)

$$f_{opt} = \frac{1-2\lambda}{\sqrt{2\pi}\rho}\sqrt{1-\rho^2}\frac{\exp\left(-\frac{1}{2}\frac{\rho^2}{1-\rho^2}h_J^2\right)}{(1-2\lambda)\,\Phi\left(\frac{\rho}{\sqrt{1-\rho^2}}h_J\tau_n\right) + \lambda}\tau_n \tag{2.17}$$

where $\Phi(x) = (1 + \mathrm{erf}(x/\sqrt{2}))/2$ denotes the cumulative Gaussian distribution. Note that in the noiseless case ($\lambda = 0$) the optimal weight function reduces to the result previously obtained in [KC92b]. The properties of this weight function will be analyzed in the next section in further detail.

Inserting f_{opt} into the Eqs. (2.6) and performing the averages over h_B, h_J, and τ_n one obtains decoupled differential equations for ρ and Q. The equation of motion for the order parameter ρ reads

$$\frac{d\rho}{d\alpha} = \frac{(1-2\lambda)^2}{2\pi} \frac{(1-\rho^2)^{3/2}}{\rho^2} \int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi}} \frac{\exp\left(-\frac{1}{2}\frac{1+\rho^2}{\rho^2}x^2\right)}{(1-2\lambda)\Phi(x)+\lambda} \tag{2.18}$$

which can be integrated only numerically. The solution for Q will turn out to be $Q(\alpha) \equiv \rho^2(\alpha)$. See Section 2.5 for the proof.

The resulting generalization error is shown in Figure 2.1 for $\lambda = 0.1$. From the numerical solution depicted in the figure it can be seen that the generalization error decays considerably faster than for Hebbian learning. This is of course not surprising since by construction of f_{opt} the corresponding learning curve is optimal.

As $\rho \to 1$, i.e. for small generalization errors, the last equation can be approximated by

$$\begin{aligned} \frac{dz}{d\alpha} &= -\frac{(1-2\lambda^2)}{2\pi} \tan^2 z \int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi}} \frac{\exp\left(-\frac{1}{2}\frac{1+\cos^2 z}{\cos^2 z}x^2\right)}{(1-2\lambda)\Phi(x)+\lambda} \\ &= -\frac{z^2}{\pi C_o(\lambda)} + \mathcal{O}(z^4) \end{aligned} \tag{2.19}$$

where $z = \pi\epsilon_g = \arccos\rho$ and

$$C_o(\lambda) = \frac{2}{(1-2\lambda)^2} \left[\int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi}} \frac{e^{-x^2}}{(1-2\lambda)\,\Phi(x)+\lambda} \right]^{-1}. \tag{2.20}$$

This differential equation can be integrated analytically and one obtains the asymptotic solution for the generalization error

$$\epsilon_g(\alpha \to \infty) = \frac{C_o(\lambda)}{\alpha}. \tag{2.21}$$

As in the noiseless case the optimal decay of the generalization error is inversely proportional to the number of presented examples. This is in contrast to the results of [BSS95] for the standard perceptron algorithm [HKP91], where the decay is only proportional to $\alpha^{-1/2}$ for any nonzero λ as in the case of Hebbian learning discussed above.[2]

The asymptotics of the learning curve (2.21) depends on $C_o(\lambda)$ which is plotted as a function of the inversion rate λ in Figure 2.2. $C_o(\lambda)$ is monotonic in λ. Therefore, the larger the inversion

[2]More precisely the *standard* on-line perceptron algorithm does *not* converge to the desired solution in this case. In order to achieve convergence one has to introduce an additional learning rate which furthermore has to be α-dependent in a specific way; for details see [BSS95]. Comparison to this algorithm shows one of the main advantages of the optimal weight function. A detailed discussion of this will be presented in Section 2.7.

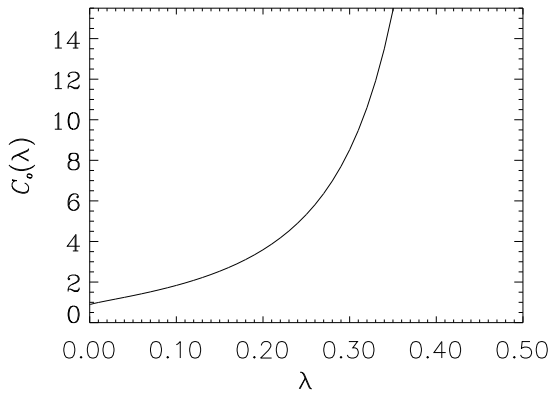


Figure 2.2: For output noise, application of the optimal weight function f_{opt} leads to an asymptotic decrease of the generalization error like $\epsilon_g = C_o(\lambda)/\alpha$. Here the numerator C_o is shown as a function of the noise level λ. Since $C_o(\lambda)$ is monotonic the generalization error increases with λ for fixed α. Note that $C_o(0) = 0.884$ consistent with the result previously obtained for the noiseless case [KC92b].

rate the larger the generalization error . This does not only hold in the asymptotic regime, but for all values of α since the monotonicity of the weight function (2.17) in λ inhibits a crossing of the learning curves for different values of λ.

In the noisefree case $\lambda = 0$ one obtains $\epsilon_g \approx 0.88/\alpha$ for large α in accordance to the results of [KC92b]. For $\lambda \to 1/2$ the prefactor $C_o(\lambda)$ diverges reflecting the fact that the rule cannot be learnt in this limit.

Note that the asymptotic decay of the generalization error (2.21) differs from the replica-symmetric result for Bayes optimal batch learning [OH91b, OH91a] by exactly a factor of 2 (and thus by a factor $\sqrt{2}$ from the off-line Gibbs procedure considered in [OK96, OH91b, OH91a]). This interesting relation was also found in [KC92b] for noisefree training where the replica-symmetric result is believed to be exact. Apparently it persists for the learning from corrupted data. An independent derivation of this result has been given recently by Opper [Opp].

2.3.3 Optimal Features

In order to understand the features of the optimal weight function (2.17) which give rise to the optimal decay of the generalization error (2.21) it is useful to analyze the functional form (2.17). To this end, it is advantageous to transform the scaled form (2.17) back to the original form (2.5)

$$F_{opt} = \frac{1-2\lambda}{\sqrt{2\pi}} \frac{\sqrt{Q}}{\rho} \sqrt{1-\rho^2} \frac{\exp\left(-\frac{1}{2}\frac{\rho^2}{1-\rho^2} h_J^2\right)}{(1-2\lambda)\,\Phi\left(\frac{\rho}{\sqrt{1-\rho^2}} h_J \tau_n\right) + \lambda} \tau_n. \tag{2.22}$$

Note that the second fraction cancels because of $\rho \equiv \sqrt{Q}$ (cf. Section 2.5). It should be stressed that the optimal weight function realizes a Hebbian type of update rule since $F_{opt} \propto \tau_n$.

According to (2.22) the optimal weight function essentially depends on three quantities:

- the *aligned field* of the actual example: $\kappa \equiv h_J \tau_n$,
- the inversion rate λ, and
- the actual overlap with the teacher $\rho(\alpha)$.

It is instructive to recall the noiseless case ($\lambda = 0$, $\tau_n \equiv \tau$) first:

$$F_{opt} = \frac{1}{\sqrt{2\pi}} \frac{\sqrt{Q}}{\rho} \sqrt{1-\rho^2} \frac{\exp\left(-\frac{1}{2}\frac{\rho^2}{1-\rho^2} h_J^2\right)}{\Phi\left(\frac{\rho}{\sqrt{1-\rho^2}} h_J \tau\right)} \tau. \quad \text{(noiseless case)} \tag{2.23}$$

Here, the aligned field contains two pieces of information. First, it allows to distinguish whether the actual example has been correctly classified ($\kappa = h_J \tau > 0$) or not ($\kappa < 0$). Second, it measures how stable[3] the classification of the actual example is before the learning step is made. The absolute value $|\kappa|$ measures the Euclidean distance of the actual input to the decision plane of the student network. The larger this quantity the larger a change of the couplings $\boldsymbol{J}$ is required to alter the currently implemented classification of the actual example.

Figure 2.3 shows the dependence of the optimal weight function on the aligned field κ. Note, that the optimal weight function for the noiseless case $\lambda = 0$ essentially distinguishes between two regimes: correct and wrong classification. If the actual example has been misclassified the resulting weight change is the larger the larger $|\kappa|$ is. This is a very reasonable property since for decreasing $\kappa < 0$ the input is farther away from the decision boundary of the student network. Hence, the decision boundary needs to be rotated by a larger angle in order to achieve correct classification.[4] On the other hand virtually no learning step is made if the actual example has already been classified correctly.

In the case of noisy training examples the interpretation of the aligned field is slightly different. Here, κ only provides the information whether or not the classification of the actual example has been *predicted* correctly. For instance, an example could have been deceptively misclassified because the output provided by the teacher network has been corrupted by noise. This uncertainty is profoundly reflected in the dependence of the optimal weight function on the aligned field. For large negative values of κ almost no weight is assigned to the corresponding examples (cf. Figure 2.3). For κ not too small, however, examples are weighted virtually as in the noiseless case.

Again, this is a very appealing feature. As discussed above, an example with a large negative aligned field would require a large weight change in order to achieve correct classification. A large change of $\boldsymbol{J}$, however, would have a devastating effect, if the actual example has been deceptively misclassified due to the noisy teacher output; this situation is exemplified in Figure 2.4. For this reason f_{opt} does not assign a considerable weight to such examples in the presence of output noise.

[3] For this reason the aligned field is also known as *stability*.

[4] It is easy to show that in the noiseless case f_{opt} has the property that the aligned field of a misclassified example is always positive after the learning step. See also Figure 2.4.

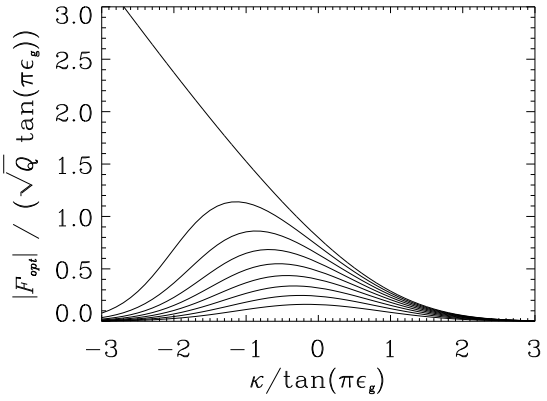


Figure 2.3: Dependence of the optimal weight function for learning subject to output noise as a function of the aligned field κ for various values of the inversion rate λ ($\lambda = 0 \ldots 0.4$ top to bottom, step size $\Delta\lambda = 0.05$). In order to scale out the ρ-dependence $F_{opt}\tau_n/(\sqrt{Q}\tan(\pi\epsilon_g))$ is plotted versus $\kappa/\tan(\pi\epsilon_g)$, where $\tan(\pi\epsilon_g) = \sqrt{(1-\rho^2)/\rho^2}$.

Copelli and Caticha [CC95, Cop95] have coined the notion *confidence* for this behaviour. The student vector does not assign a considerable weight to an actual example, although it is seemingly misclassified. Instead the student is "blaming" the output noise for this event and is confident in the own classification. According to Figure 2.4 a lack of confidence would result in a nonzero final generalization error. This statement will also be manifested analytically in Section 2.5.

Effectively, the optimal weight function distinguishes among three domains depending on the value of the aligned field. For examples with $\kappa \gg 1$ the weights $\boldsymbol{J}$ essentially remain unchanged. Only examples with aligned field close to zero lead to a considerable weight change. For decreasing $\kappa < 0$ there is an effective cutoff value such that for κ less than this cutoff virtually no weight change occurs. This cutoff value is λ-dependent (see Figure 2.3). Obviously, the larger the inversion rate λ the larger is the probability that an example with negative aligned field causes an undesirable weight change. Therefore it is advantageous that the cutoff increases with increasing λ. A good estimate for this λ-dependent cutoff will be given below.

The dependence of the optimal weight function on the overlap ρ is shown in Figure 2.5 for different values of λ. In the limit $\rho \to 0$, i.e. at the beginning of the learning process the optimal weight function reduces to $\sqrt{2/\pi}(1-2\lambda)\tau_n$. This is the weight function of Hebbian learning.[5]

As ρ increases the weight function assumes more and more a bell shape giving more weight to probably misclassified examples ($\kappa < 0$) than to probably correctly classified examples ($\kappa > 0$). Examples with $\kappa \ll -1$ are given less and less weight as discussed above. For fixed λ the absolute value of the cutoff, roughly determined by the width of the Gaussian in (2.22), decreases like

$$\tan(\pi\epsilon_g) = \frac{\rho}{\sqrt{1-\rho^2}}, \tag{2.24}$$

[5]The dynamics of Hebbian learning is invariant under the transformation $F \to \eta F$, where η is an arbitrary positive constant [Rie94].

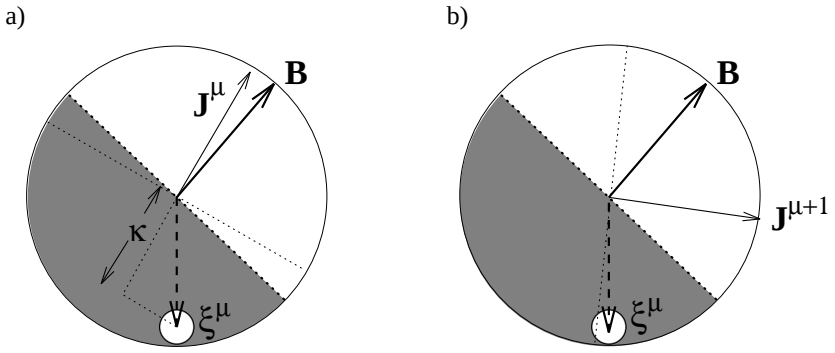


Figure 2.4: Geometrical illustration of the behaviour of the optimal weigh function F_{opt} (2.22) for large negative values of the aligned field $\kappa = h_J\tau_n$. If F_{opt} were noise-independent as in (2.23) the deceptively misclassified example $\boldsymbol{\xi}^\mu$ would result in a new weight vector $\boldsymbol{J}^{\mu+1}$ with considerably less overlap to the desired rule $\boldsymbol{B}$; a) before, b) after the update.

i.e. like $1/\alpha$ as $\alpha \to \infty$. In the limit $\rho \to 1$ the weight function approaches $2(1-2\lambda)\sqrt{1-\rho^2}$ $\delta(h_J)\tau_n$, thus only taking into account examples close to the decision boundary and weighting them by the decreasing factor $\sqrt{1-\rho^2} \approx \epsilon_g$.

The prefactor $\sqrt{1-\rho^2}$ in f_{opt} can be viewed as a *learning rate*. Such a parameter is often used in *ad hoc* algorithms in order to make them convergent. Usually the learning rate has to be "annealed", i.e. decreased at a certain rate. Obviously, the optimal weight function incorporates this annealing automatically and more than that it does so at the optimal rate.

As pointed out above the region where the optimal weight function differs from the one for the noiseless case is for misclassified examples ($h_J\tau_n < 0$). Due to noise, however, the misclassification can be deceptive. This is the case for $h_J\tau_n < 0$ but $h_J\tau > 0$, i.e. the current example is actually correctly classified but appears to be misclassified because the teacher's output has been flipped due to noise. Such an event occurs with conditional probability $P(h_J h_B > 0 \mid h_J\tau_n < 0)$, where

$$P(h_J h_B > 0 \mid h_J\tau_n) = \frac{(1-\lambda)\Phi\left(\frac{h_J\tau_n}{\tan(\pi\epsilon_g)}\right)\Theta(h_J\tau_n) + \lambda\Phi\left(-\frac{h_J\tau_n}{\tan(\pi\epsilon_g)}\right)\Theta(-h_J\tau_n)}{(1-\lambda)\Phi\left(\frac{h_J\tau_n}{\tan(\pi\epsilon_g)}\right) + \lambda\Phi\left(-\frac{h_J\tau_n}{\tan(\pi\epsilon_g)}\right)}. \tag{2.25}$$

This probability is shown in Figure 2.6 as a function of the aligned field $h_J\tau_n$.

For those values of $h_J\tau_n$ where $P(h_J h_B > 0 \mid h_J\tau_n < 0) > 1/2$ the corresponding example is most likely noisy and has therefore been deceptively misclassified. Vice versa, for $P(h_J h_B > 0 \mid h_J\tau_n < 0) < 1/2$ the example is more likely to be truly misclassified. As can be seen in Figure 2.6 these examples are situated closely to the student's decision boundary $h_J\tau_n = 0$. The value κ_{co} defined via $P(h_J h_B > 0 \mid \kappa_{co}) = 1/2$ can be viewed as a good measure for the cutoff value of f_{opt} (2.17). From (2.25) one obtains

$$\kappa_{co} = \tan(\pi\epsilon_g)\Phi^{(-1)}(\lambda) \tag{2.26}$$

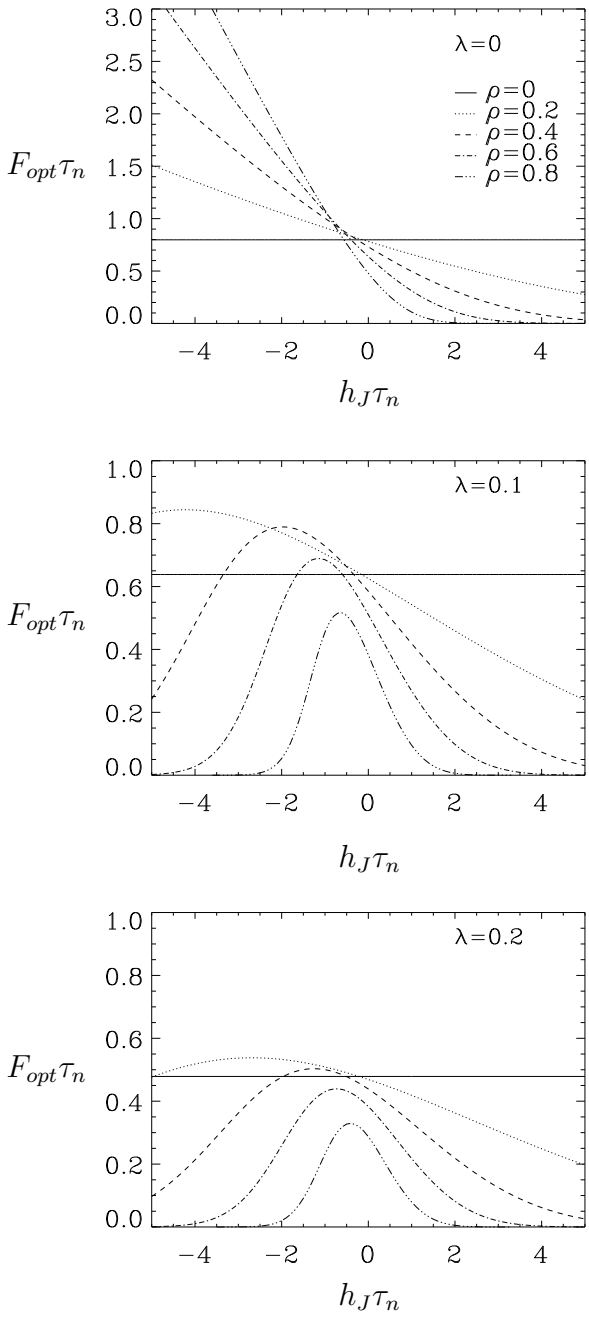


Figure 2.5: Optimal weight function F_{opt} for learning subject to output noise for various values of the order parameter ρ. For $\rho = 0$ and given inversion rate λ all examples are weighted equally irrespective of the value of the aligned field $h_J\tau_n$. This corresponds to Hebbian learning. In the noiseless case ($\lambda = 0$) examples with a negative aligned field receive more weight as ρ increases while those with positive aligned field gain less weight. For $\lambda > 0$ also examples with large negative aligned fields are not taken into account for updating. These examples are most likely affected by the noise and therefore are deceptively misclassified. The optimal weight function possesses a cutoff value at negative values of $h_J\tau_n$. This cutoff decreases in absolute value with increasing λ and ρ.

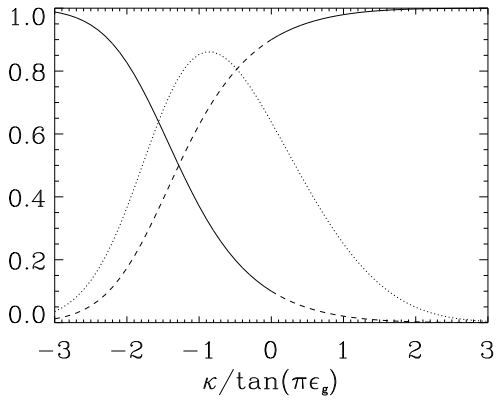


Figure 2.6: Probabilities $P(h_J h_B > 0 \mid \kappa)$ (solid) and $P(h_J h_B < 0 \mid \kappa)$ (dashed) as a function of the aligned field $\kappa = h_J \tau_n$ for an inversion rate $\lambda = 0.1$. For negative values of the aligned field, $P(h_J h_B > 0 \mid \kappa)$ denotes the probability that the corresponding example has been deceptively misclassified, i.e. the teacher output has been corrupted by noise. $P(h_J h_B < 0 \mid \kappa < 0)$ is the probability that the example has been truly misclassified. Therefore examples with large negative values of $\kappa = h_J \tau_n$ are with high probability deceptively misclassified and consequently should not be taken into account for learning. For comparison the corresponding optimal weight function is also shown (dotted).

where $\Phi^{(-1)}$ denotes the inverse function of the cumulative Gaussian distribution.

The scaling of κ_{co} with the generalization error ϵ_g is consistent with the result obtained in (2.24) from the width of f_{opt}. Furthermore, κ_{co} includes the noise-dependence of the cutoff; κ_{co} approaches 0 for increasing noise level λ. For $\lambda \to 0$, κ_{co} diverges which is also consistent with the properties of the optimal weight function in the noiseless case.

Although the investigation of the probability (2.25) leads to quantitatively the same results, it cannot be overemphasized that the derivation of f_{opt} from first principles is preferable and more rigorous. Eq. (2.25) only gives an additional means for an understanding of the noise-dependent properties of the optimal weight function.

In a qualitative way the optimal features which the weight function f_{opt} makes use of can be summarized as follows: First and most importantly the weight function has to incorporate the knowledge (or suspicion) that the provided examples are corrupted by output noise. Next, the weight assigned to a particular example depends on its actual aligned field. The weight assigned must not increase as $\kappa \to -\infty$. Examples with small positive aligned field receive a small but nonzero weight. As the performance of the student network increases only those examples give rise to a considerable weight change whose aligned fields lie in a decreasingly small interval around zero. This is in contrast to the noiseless case.

Primarily the advantage of the optimal weight function is not that it gives rise to an optimal decay of the generalization error, but that it allows to identify the necessary features of a successful algorithm. This is done by deriving f_{opt} by means of a simple variation method in an *a priori* fashion. This is what makes this approach so powerful.

Once f_{opt} has been derived and the optimal features have been identified it is an easy task to construct an algorithm which exploits these optimal features but with a weight function which is

easier to implement or of simpler algebraic form, leading to a comparable decay of the generalization error as f_{opt}.

2.4 Weight Noise

In the second scenario, the actual weights used to evaluate the training output τ_n^μ are subject to random fluctuations. This can be modeled by

$$\tau_n^\mu = \mathrm{sign}(\widetilde{\boldsymbol{B}}^\mu \cdot \boldsymbol{\xi}^\mu) \tag{2.27}$$

where $\widetilde{\boldsymbol{B}}^\mu$ is a normalized random vector with $\widetilde{\boldsymbol{B}}^\mu \cdot \boldsymbol{B} = \omega \leq 1$; i.e. at each time step the true teacher weights $\boldsymbol{B}$ are corrupted by noise such that the overlap of the resulting vector $\widetilde{\boldsymbol{B}}^\mu$ equals ω on average. The noise is assumed to be isotropic, i.e. independent and identically distributed in all components B_j. So far, weight noise has been investigated only in the context of off-line learning (e.g. [OK96, GT90]).

Note that this scenario is identical to introducing an equivalent noise in the inputs $\boldsymbol{\xi}$ used for the evaluation of τ_n, but with $\boldsymbol{B}$ unchanged. Furthermore, the effect will be the same if this noise is imposed on the student input vectors in (2.3), with τ_n being the true rule output for the true $\boldsymbol{\xi}$. This is to say that noisy inputs in either the student or the teacher network are equivalent to weight noise.[6]

The average over this type of noise can be performed by introducing the additional Gaussian variable $\widetilde{h_B} = \widetilde{\boldsymbol{B}} \cdot \boldsymbol{\xi}$ with zero mean, unit variance and correlations

$$\left\langle \widetilde{h_B} h_J \right\rangle_\xi = \omega\rho, \;\; \left\langle \widetilde{h_B} h_B \right\rangle_\xi = \omega. \tag{2.28}$$

All averages in the equations of motion (2.6) reduce to integrals over the corresponding three-dimensional density $P(h_J, h_B, \widetilde{h_B})$ with $\tau_n = \mathrm{sign}(\widetilde{h_B})$. See Appendix A for the complete functional form of the distribution P.

Note that the relation between generalization and prediction error (2.8) is now

$$\epsilon_p = \frac{1}{\pi} \arccos\left(\omega \cos(\pi\epsilon_g)\right) \tag{2.29}$$

implying that ϵ_p is bounded from below by $\lambda_\omega \equiv \epsilon_p^\infty = (1/\pi)\arccos\omega$. Hence, a fraction λ_ω of the teacher outputs is flipped due to noise. The usage of the quantity λ_ω is convenient for two reasons. First, measuring the inversion rate, it allows for a direct comparison of the results with those of output noise. Second, λ_ω provides a more favorable measure of the strength of the noise since its value increases as the effect of weight noise increases, while ω decreases in this case.

In contrast to output noise, the distribution of those inputs $\boldsymbol{\xi}$ whose output labels have been corrupted by noise is not isotropic in input space. Only those examples are affected by weight noise that are situated between the decision boundaries of $\boldsymbol{B}$ and $\widetilde{\boldsymbol{B}}$. These hyperplanes span

[6] Note that noise in the student weights $\boldsymbol{J}$ [Ste95] is not equivalent to weight noise. This case corresponds to a drifting rule [BS93, KC93].

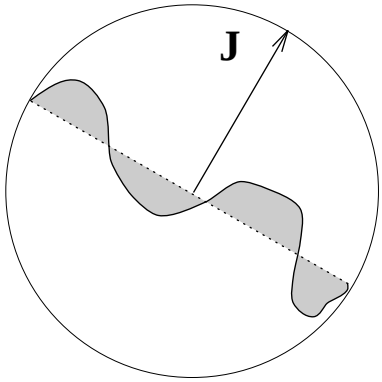


Figure 2.7: Weight noise might serve as a simple model for unlearnable rules. Imagine the student being a perceptron capable of realizing a *linearly* separable rule. If the true rule is of the type shown here (the solid line represents the teacher's decision boundary) the distribution of systematically mismatched examples (shaded) is similar to the one for weight noise, i.e. centered at the teacher's decision boundary.

an angle $\arccos\omega$ on average. Therefore, noisy examples are more likely to have aligned fields $|h_J\tau_n| < \arccos\omega$.

Due to the characteristic distribution of examples which are corrupted by noise, weight noise might serve as a simple model for unlearnable rules. Consider a separable rule which is "almost" linearly separable, see Figure 2.7. If such a rule is to be learnt by a neural network that realizes a linearly separable perceptron, the examples that are systematically misclassified because of the architecture mismatch will be located closely the teacher's decision boundary. Hence, the situation is similar to the weight noise case discussed here. The difference, of course, is that for weight noise the misclassification is probabilistic in nature while it is deterministic for an unlearnable rule. Whether weight noise actually is a suitable model for a certain class of unlearnable rules has still to be investigated, however.

2.4.1 Hebbian Learning

For Hebbian learning ($F = \tau_n$) the equations of motion (2.4) read in terms of the unnormalized student-teacher overlap $R = \boldsymbol{J} \cdot \boldsymbol{B} = \rho\sqrt{Q}$

$$\begin{aligned} \frac{dR}{d\alpha} &= \sqrt{\frac{2}{\pi}}\omega \\ \frac{dQ}{d\alpha} &= 2\sqrt{\frac{2}{\pi}}\omega R + 1. \end{aligned} \tag{2.30}$$

These equations of motion coincide with those for output noise (2.15) except that the inversion rate λ is replaced by $(1-\omega)/2$. Hence, by comparison with (2.16) one obtains for the generalization error

$$\epsilon_g(\alpha) = \frac{1}{\pi}\arccos\left[\left(1 + \frac{\pi}{2\omega^2\alpha}\right)^{-1/2}\right] \tag{2.31}$$

which asymptotically decreases as $\alpha^{-1/2}$. The corresponding $\epsilon_g(\alpha)$ in Figure 2.1 can therefore be interpreted as the learning curve for Hebbian learning with weight noise parameter $\omega = 1 - 2\lambda$. Note, however, that the respective prediction errors are not identical.

2.4.2 Optimal Weight Function

As in the case of output noise the optimal weight function is simply obtained by calculating the conditional average in (2.10):

$$f_{opt} = \frac{1}{\sqrt{2\pi}} \frac{\omega(1-\rho^2)}{\rho\sqrt{1-\omega^2\rho^2}} \frac{\exp\left(-\frac{1}{2}\frac{\omega^2\rho^2}{1-\omega^2\rho^2}h_J^2\right)}{\Phi\left(\frac{\omega\rho}{\sqrt{1-\omega^2\rho^2}}h_J\tau_n\right)}\tau_n. \tag{2.32}$$

For the noiseless case $\omega = 1$ this reduces to the result of [KC92b], Eq. (2.23). Note that again the optimal weight function requires knowledge about the existence of the noise and the characteristics of its distribution (here parametrized by ω).

By solving the corresponding differential equation for the order parameter ρ (2.6)

$$\frac{d\rho}{d\alpha} = \frac{1}{2\pi} \frac{\omega(1-\rho^2)^2}{\rho^2\sqrt{1-\omega^2\rho^2}} \int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi}} \frac{\exp\left(-\frac{1}{2}\frac{1+\omega^2\rho^2}{\omega^2\rho^2}x^2\right)}{\Phi\left(x\right)} \tag{2.33}$$

numerically, one obtains the generalization error $\epsilon_g(\alpha)$. Figure 2.1 shows the numerical solution for ϵ_g for a particular choice of ω. The differential equation for ρ is completely decoupled from the equation of motion for the norm Q. The latter can therefore be omitted for the purpose of determining the solution for ϵ_g. An asymptotic solution of (2.33) for small ϵ_g yields for $\alpha \to \infty$ and $\omega < 1$ the analytic result

$$\epsilon_g(\alpha \to \infty) = C_w(\omega)\alpha^{-1/2}. \tag{2.34}$$

The asymptotics of the learning curve depends on the amount of corrupting noise measured by ω through

$$C_w(\omega) = \frac{(1-\omega^2)^{1/4}}{(\omega\pi)^{1/2}} \left[\int_{-\infty}^{\infty} \frac{dx}{\sqrt{2\pi}} \frac{\exp\left(-\frac{1}{2}\frac{1+\omega^2}{\omega^2}x^2\right)}{\Phi(x)}\right]^{-1/2} \tag{2.35}$$

which is shown in Figure 2.8. For increasing noise (decreasing ω) the prefactor $C_w(\omega)$ is increasing resulting in a larger generalization error for given α. Furthermore, the constant $C_w(\omega)$ diverges as $\omega \to 0$. The latter case corresponds to noisy teacher weights $\widetilde{\boldsymbol{B}}$ perpendicular to the true teacher vector $\boldsymbol{B}$ ($\omega = \langle \boldsymbol{B}\widetilde{\boldsymbol{B}} \rangle = 0$), i.e. the training label τ_n and the output of the true teacher $\mathrm{sign}(h_B)$ agree with only 50% probability. Therefore the rule $\boldsymbol{B}$ cannot be learnt in this limit.

It is instructive to compare the optimal on-line decay of the generalization error with the result one can obtain by using batch training. Recall that in the case of output noise the optimal on-line result differs only by a factor of 2 from the optimal batch result (Bayes optimum), cf. Section 2.3.2.

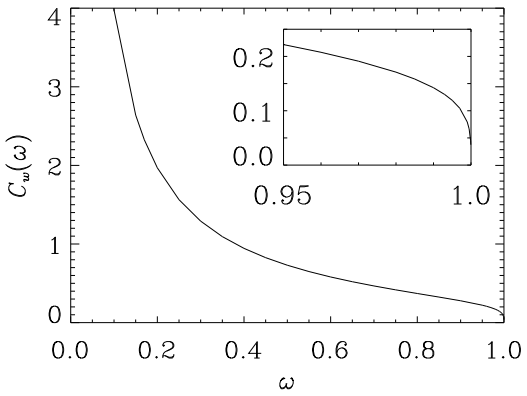


Figure 2.8: For weight noise, application of the optimal weight function f_{opt} leads to an asymptotic decrease of the generalization error like $\epsilon_g = C_w(\omega)/\sqrt{\alpha}$. Here, the numerator C_w is shown as a function of the noise level ω. Smaller values of ω correspond to more noise corrupting the training data; $\omega = 1$ represents the noiseless case. Since $C_w(\omega)$ is monotonic the generalization error decreases with increasing ω for fixed α. The inset shows $C_w(\omega)$ in the weak noise limit $\omega \to 1$.

For weight noise, it can be shown [Opp] that on-line optimum and Bayes optimum lead to the same generalization error asymptotically, i.e. in the presence of weight noise on-line learning is as effective as batch training. Stated differently, in this case there is no advantage in using batch learning.[7]

This result has been explained recently by Opper [Opp96]: The essential quantity is the distribution of the noise. If this function is differentiable then optimal on-line learning will be as effective as the Bayes optimum. If this is not the case then the generalization error for optimal on-line is worse by an exact factor of 2.

2.4.3 Optimal Features

For the case of output noise (Section 2.3.3) the analysis revealed that the optimal weight function depends on three quantities: the aligned field $\kappa = h_J\tau_n$, the strength of the noise, and the actual performance, i.e. the overlap $\rho(\alpha)$ of the student vector with the true teacher vector. These are also the essential quantities the optimal weight function (2.32) makes use of in the case of weight noise:

$$F_{opt} = \sqrt{\frac{Q}{2\pi}} \frac{\omega(1-\rho^2)}{\rho\sqrt{1-\omega^2\rho^2}} \frac{\exp\left(-\frac{1}{2}\frac{\omega^2\rho^2}{1-\omega^2\rho^2}h_J^2\right)}{\Phi\left(\frac{\omega\rho}{\sqrt{1-\omega^2\rho^2}}h_J\tau_n\right)}\tau_n. \tag{2.36}$$

Note that the dependence on ω is only through the actual value of the prediction error $\epsilon_p(\alpha)$

[7] Taking into account the generic drawbacks of batch learning, on-line training can even be considered superior. The drawbacks of batch learning include for instance the necessary storage of training examples and the inability to cope with a time changing rule.

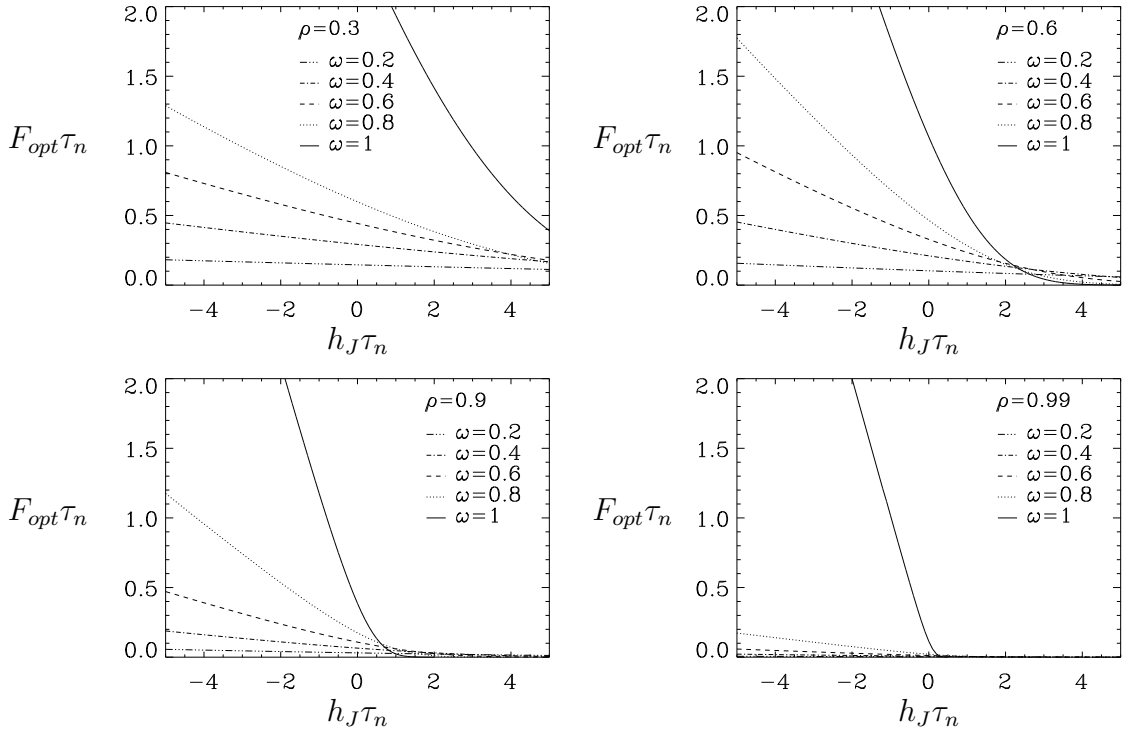


Figure 2.9: Optimal weight function F_{opt} for learning subject to weight noise as a function of the aligned field $h_J\tau_n$ for various values of ω and ρ. As in the noiseless case ($\omega = 1$, cf. Figure 2.5), F_{opt} assigns more weight the smaller $h_J\tau_n$. However, the weight function decreases with increasing teacher-student overlap ρ. Moreover, for increasing strength of the weight noise (decreasing ω) F_{opt} decreases in amplitude.

(2.29) since (2.36) can be written as

$$F_{opt} = \sqrt{\frac{Q}{2\pi}} \frac{\tan^2(\pi\epsilon_g)}{\tan(\pi\epsilon_p)} \frac{\exp\left(-\frac{1}{2}\frac{(h_J\tau_n)^2}{\tan^2(\pi\epsilon_p)}\right)}{\Phi\left(\frac{h_J\tau_n}{\tan(\pi\epsilon_p)}\right)} \tau_n. \tag{2.37}$$

The dependence on the aligned field exclusively enters through the last factor. This is of the same form as in the optimal weight function for the noiseless case (2.23). Hence, F_{opt} typically assigns more weight to examples with a negative value of the aligned field. For decreasing values of the aligned field the weight function becomes roughly proportional to $-h_J\tau_n$ while it approaches zero as the aligned field $h_J\tau_n$ increases (cf. Fig. 2.9).

In order to understand the essential features of the optimal weight function it is useful to consider f_{opt} for small values of the prediction error ϵ_p. In this limit f_{opt} can be well approximated by

$$-\sqrt{Q}\frac{\tan^2(\pi\epsilon_g)}{\tan^2(\pi\epsilon_p)}\Theta(-h_J\tau_n)h_J\tau_n. \tag{2.38}$$

For the noiseless case ($\epsilon_g \equiv \epsilon_p$) this expression is known as AdaTron weight function or relaxation rule. It leads to a slightly worse performance than the optimal weight function in the

noiseless case [BR94].

For negative aligned fields the approximate proportionality to $-h_J\tau_n$ is governed by the factor $\tan^2(\pi\epsilon_g)/\tan^2(\pi\epsilon_p)$ which reflects the dependence of the optimal weight function on the performance ρ. This factor essentially acts as a learning rate which is α-dependent through the dependence on ϵ_g or ρ, respectively. Hence, the weight assigned to a particular example depends on the actual state of the network measured by the performance ρ, see Fig. 2.9. The weight assigned to any example decreases roughly with $\epsilon_g^2 \propto 1/\alpha$.

It is interesting to compare the $(1/\alpha)$-decay of the effective learning rate in f_{opt} to learning rates required to achieve a convergence $\epsilon_g \to 0$ of the AdaTron weight function in the presence of weight noise. In [Ste95] it has been shown that this requires a learning rate decreasing like $1/\alpha$ with a resulting asymptotic generalization error $\epsilon_g \propto 1/\sqrt{\alpha}$. Here, this result has been found in a straightforward way simply be analyzing the optimal weight function.

In addition to the dependence on the actual performance ρ and the aligned field $h_J\tau_n$, the optimal weight function (2.36) depends on the strength of the corrupting weight noise. This dependence is essentially given by the factor $1/\tan^2(\pi\epsilon_p)$ in (2.38) where the strength of the weight noise is measured through the prediction error $\epsilon_p = (1/\pi)\arccos(\omega\rho)$. As can be seen in Figure 2.9 this means that the weight assigned to a particular example is the smaller the larger the strength of the corrupting noise, i.e. the smaller ω.

In a qualitative way the noise dependence of the optimal weight function has already been found in the case of output noise. Also the performance dependence of f_{opt} is already familiar from this case. Note however, that for weight noise the dependence on the aligned field is profoundly different. While for output noise the optimal weight function requires an effective cutoff, this is not the case here. This is even more surprising if one considers the same kind of argument that has been used in Section 2.3 in order to understand the necessity of a cutoff if output noise is present. For weight noise the probability that an example with negative aligned field has been deceptively misclassified is given by

$$P(h_J h_B > 0 \mid h_J\tau_n) = \frac{A_+(\omega,\rho,h_J\tau_n)\Theta(-h_J\tau_n) + A_-(\omega,\rho,h_J\tau_n)\Theta(h_J\tau_n)}{A_+(\omega,\rho,h_J\tau_n) + A_-(\omega,\rho,h_J\tau_n)}, \tag{2.39}$$

where

$$A_\pm(\omega,\rho,h_J\tau_n) = \int_{-\infty}^{\mp h_J\tau_n\rho/\sqrt{1-\rho^2}} Dy\,\Phi\left[\pm\frac{\omega}{\sqrt{1-\omega^2}}\left(\sqrt{1-\rho^2}y \pm \rho h_J\tau_n\right)\right]. \tag{2.40}$$

An analysis of this expression shows that as long as $\omega < 1$ examples with sufficiently small values of the aligned field $h_J\tau_n$ are always more likely to be deceptively misclassified. Hence, one would suspect that the optimal weight function does not assign a weight to such examples. Effectively the optimal weight function should possess a cutoff value, similar to the case of output noise. Obviously this is not the case. The intuitive probability argument does not predict the dependence of f_{opt} on the aligned field correctly.

It is not quite clear why the probability argument helps do understand the $(h_J\tau_n)$-dependence of the optimal weight function for output noise while it does not for weight noise. It should be

emphasized, however, that this is not very crucial. The optimal weight function has been derived from first principles by means of a variational principle. This is, of course, more satisfying and powerful than a possible intuitive argument that would help to understand the particular functional form of f_{opt}.

2.4.4 Comparison with Output Noise

The analysis of on-line learning using the optimal weight function has shown that the decay of the generalization error occurs on different time scales depending on the nature of the corrupting noise. For output noise the generalization error decays on the same $(1/\alpha)$-time scale as in the noiseless case whereas it is considerably slower if weight noise is present: $\epsilon_g \propto \alpha^{-1/2}$. Since the on-line algorithms used to obtain these results are optimal, weight noise is obviously a much harder type of noise than output noise. This conclusion is confirmed by the results for optimal batch learning [OH91a] mentioned above.

In order to understand why this is the case it is instructive to compare the models for the same asymptotic value of the prediction error ϵ_p, i.e. for $\lambda = (1/\pi) \arccos \omega$. Then, the expected number of corrupted outputs τ_n is the same for both types of noise. Yet, weight noise will produce a label $\tau_n = -\tau$ with a probability which depends on the value of h_B. Mainly examples with $\boldsymbol{\xi}$ close to the decision boundary $h_B = 0$ will be affected, whereas the output noise was defined to be independent of h_B. In other words, as $\rho \to 1$ the noisy examples are more likely situated closely to the student's decision boundary. Therefore one would expect that asymptotically output noise is easier to cope with.

Figure 2.1 depicts the numerically obtained solution of the generalization error for $\omega = \cos(0.1\pi)$. This value of ω leads to an asymptotic prediction error $\epsilon_p^\infty = 0.1$ and therefore allows for a direct comparison with the results obtained for output noise with $\lambda = \epsilon_p^\infty = 0.1$. As can be seen in Figure 2.1, for small α the generalization error decreases considerably faster in the case of weight noise than for output noise, and only for large α the decay is slower according to the different power law.

This behavior can be understood as follows: For weight noise only those input vectors are corrupted that are close to the decision boundary of the noiseless rule vector $\boldsymbol{B}$. Therefore the effect of the noise is not very pronounced for small α, where the student's decision boundary is still far away from the one of the teacher. In contrast, output noise inverts all inputs with the same probability, regardless of their overlap with student or teacher. Hence, learning in the case of output noise leads to a larger ϵ_g in the beginning compared to weight noise.

2.5 Realization of the Optimal Generalization

The optimal weight functions (2.17,2.32) depend both on the (normalized) student-teacher overlap $\rho = \cos(\pi\epsilon_g)$ and the noise parameters λ and ω, respectively, which will not be accessible to the student network in general. Nevertheless, by construction of f_{opt}, there is no algorithm of the form (2.3) that could yield a smaller value of the generalization error for a given α.

In the following it will be shown how to make the weight function independent of the inaccessible quantities without changing the asymptotic behaviour of the generalization error.

2.5.1 Dependence on the Teacher-Student Overlap

In order to circumvent the dependence of f_{opt} on ρ it is instructive to investigate the dynamics of Q using the weight function f_{opt}. As a first step the equation of motion for the order parameter Q (2.6) is rewritten in terms of the length of the student vector $\sqrt{Q}$:

$$\frac{d\sqrt{Q}}{d\alpha} = \sqrt{Q}\Big\langle f_{opt}(h_J,\zeta;\rho)h_J + \frac{1}{2}f_{opt}(h_J,\zeta;\rho)^2\Big\rangle_{P(h_J,h_B,\zeta)} \tag{2.41}$$

Here, ζ represents any quantities to be averaged over in addition to the internal fields h_B, h_J. For instance, in the case of output noise $\zeta = \tau_n$ or $\zeta = \widetilde{h_B}$ for weight noise.

The first term on the r.h.s. of (2.41) vanishes since

$$\begin{aligned}
\Big\langle \langle h_B\rangle_{P(h_B|h_J,\zeta)} h_J\Big\rangle_{P(h_J,h_B,\zeta)} &= \int dh_J dh_B d\zeta P(h_J,h_B,\zeta)\langle h_B\rangle_{P(h_B|h_J,\zeta)} h_J \\
&= \int dh_J d\zeta P(h_J,\zeta) h_J \underbrace{\int dh_B P(h_B|h_J,\zeta)}_{=1} \times \\
&\quad \times \int dh'_B h'_B P(h'_B|h_J,\zeta) \\
&= \int dh_J dh'_B d\zeta P(h_J,h'_B,\zeta) h_J h'_B \\
&= \langle h_J h_B\rangle_{P(h_J,h_B,\zeta)} = \rho
\end{aligned} \tag{2.42}$$

and therefore by definition of f_{opt} (2.10) and because of $\langle h_J^2\rangle = 1$

$$\langle h_J f_{opt}(h_J,\zeta;\rho)\rangle = \left\langle \frac{1}{\rho}\langle h_B\rangle_{P(h_B|h_J,\zeta)} h_J - h_J^2 \right\rangle = 0. \tag{2.43}$$

Hence the equation of motion for $\sqrt{Q}$ can be written as

$$\frac{d\sqrt{Q}}{d\alpha} = \frac{\sqrt{Q}}{2}\Big\langle f_{opt}^2(h_J,\zeta;\rho)\Big\rangle_{P(h_J,h_B,\zeta)}. \tag{2.44}$$

Proceeding in a similar fashion as in (2.42) it is straightforward to proof that

$$\Big\langle \langle h_B\rangle_{P(h_B|h_J,\zeta)} h_B\Big\rangle_{P(h_J,h_B,\zeta)} = \Big\langle \big(\langle h_B\rangle_{P(h_B|h_J,\zeta)}\big)^2\Big\rangle_{P(h_J,h_B,\zeta)}. \tag{2.45}$$

This result can be used to simplify the equation of motion (2.6) for the normalized teacher-student overlap ρ:

$$\frac{d\rho}{d\alpha} = \left\langle \rho f_{opt}(h_J,\zeta;\rho)\left(\frac{h_B}{\rho} - h_J\right) - \frac{\rho}{2} f_{opt}^2(h_J,\zeta;\rho)\right\rangle_{P(h_J,h_B,\zeta)}$$

$$= \left\langle \rho f_{opt}(h_J,\zeta;\rho) \underbrace{\left(\frac{1}{\rho}\langle h_B\rangle_{P(h_B|h_J,\zeta)} - h_J\right)}_{=f_{opt}} - \frac{\rho}{2} f_{opt}^2(h_J,\zeta;\rho)\right\rangle_{P(h_J,h_B,\zeta)}$$

$$= \frac{\rho}{2}\left\langle f_{opt}^2(h_J,\zeta;\rho)\right\rangle_{P(h_J,h_B,\zeta)}. \tag{2.46}$$

Comparing (2.44) and (2.46) one arrives at the remarkable fact that the equations of motion for the order parameters ρ and Q are related by

$$\frac{1}{\sqrt{Q}}\frac{d\sqrt{Q}}{d\alpha} = \frac{1}{\rho}\frac{d\rho}{d\alpha}. \tag{2.47}$$

By choosing the initial condition $\sqrt{Q(0)} = \rho(0)$ one therefore guarantees $\sqrt{Q(\alpha)} \equiv \rho(\alpha)$, as already observed in [CC95]. Hence, the solution for $\epsilon_g(\alpha)$ does not change if ρ is replaced by $\sqrt{Q}$ in $f_{opt}(h_J,\zeta;\rho)$. However, Q is the norm of the weight vector $\boldsymbol{J}$ and of course is available to the student network.

Note that by deriving (2.47) one has only made use of Bayes formula, but has not made any assumption about the nature of the joint probability distribution $P(h_J,h_B,\zeta)$. Therefore it has been shown that the optimal weight function $f_{opt}(h_J,\zeta,\sqrt{Q})$ satisfies $\sqrt{Q(\alpha)} \equiv \rho(\alpha)$ for any distribution of the inputs and the noise. Since all transformations in the above proof are equivalence transformation the reverse result holds true also: The only weight function that satisfies $\sqrt{Q(\alpha)} \equiv \rho(\alpha)$ is f_{opt}, provided of course that initially $\sqrt{Q(0)} = \rho(0)$.

2.5.2 Dependence on the Noise Parameter

The other peculiarity of the optimal weight function is its explicit dependence on the value of the quantities that characterize the corrupting noise, i.e. λ and ω, respectively. It has been argued in 2.3.3 that the knowledge about the presence of the corrupting noise is essential for the learning process. In general, however, one cannot expect that the characteristic quantities of the noise are known *quantitatively*, i.e. there is no *a priori* knowledge about the value λ or ω, respectively.

How can one circumvent this difficulty? This question will first be answered for the case of output noise. The results will be easily transfered to weight noise.

Output Noise

One could try to replace the unknown noise level λ with a constant estimate Λ in the expression for f_{opt} (2.17) in order to find out, how sensitively the optimal weight function depends on the knowledge of λ. Inserting the resulting weight function

$$f_\Lambda = f_{opt}\Big|_{\lambda\to\Lambda} = \frac{1-2\Lambda}{\sqrt{2\pi}\rho}\sqrt{1-\rho^2}\frac{\exp\left(-\frac{1}{2}\frac{\rho^2}{1-\rho^2}h_J^2\right)}{(1-2\Lambda)\,\Phi\left(\frac{\rho}{\sqrt{1-\rho^2}}h_J\tau_n\right)+\Lambda}\tau_n \tag{2.48}$$

into the equation of motion for ρ (2.6) leads to the asymptotic solution

$$\epsilon_g(\alpha \to \infty) \begin{cases} = & C_o(\lambda, \Lambda)/\alpha & \text{for } \Lambda_c(\lambda) < \Lambda < \frac{1}{2} \\ > & 0 & \text{for } 0 \leq \Lambda \leq \Lambda_c(\lambda) \end{cases} . \tag{2.49}$$

The inverse of the constant $C_o(\lambda, \Lambda)$ is given by

$$\begin{aligned} C_o^{-1}(\lambda, \Lambda) &= \frac{1}{2}(1-2\Lambda) \int_{-\infty}^{\infty} \frac{dz}{\sqrt{2\pi}} \frac{e^{-z^2}}{(1-2\Lambda)\Phi(z)+\Lambda} \times \\ &\quad \times \left[2(1-2\lambda) - (1-2\Lambda) \frac{(1-2\lambda)\Phi(z)+\lambda}{(1-2\Lambda)\Phi(z)+\Lambda} \right] \end{aligned} \tag{2.50}$$

and $\Lambda_c(\lambda)$ is defined via

$$\frac{1}{C_o(\lambda, \Lambda_c)} = 0. \tag{2.51}$$

Hence, there exists a critical value $\Lambda_c(\lambda)$ such that $\epsilon_g \propto 1/\alpha$ for $\Lambda > \Lambda_c$, but ϵ_g remains finite for $\Lambda < \Lambda_c < \lambda$. Consequently the dynamics defined by (2.6) with (2.48) as a weight function exhibits a far off equilibrium phase transition: The attractor of the dynamics depends on the value of Λ and a critical value Λ_c separates between phases with a zero ($\Lambda > \Lambda_c$) and a nonzero ($\Lambda < \Lambda_c$) value of the final generalization error.

Obviously a wrong estimate $\Lambda \neq \lambda$ in (2.48) does not destroy the fast $1/\alpha$-decay of the generalization error if $\Lambda > \Lambda_c(\lambda)$. However, it is not very advantageous to choose a large value for Λ. This is because $C_o(\lambda, \Lambda)$ increases as Λ increases. Hence, the number of examples α needed in order to reach a fixed value of ϵ_g increases with Λ. The prefactor $C_o(\lambda, \Lambda)$ is shown in Figure 2.10 as a function of the estimate Λ. It diverges as $\Lambda \to \infty$ and $\Lambda \to \Lambda_c$. The critical value Λ_c as a function of the noise level λ is also shown in Figure 2.10.

Quite surprising is the fact that the critical estimate Λ_c is less than the actual value of the inversion rate λ, i.e. one can even underestimate the inversion rate λ level as long as $\Lambda > \Lambda_c$. But again, $C_o(\lambda, \Lambda)$ will increase the more the estimate Λ deviates from the true noise level λ. Of course, only for the choice $\Lambda = \lambda$ the generalization error decreases optimally as in (2.21).

Copelli [Cop95, CEK$^+$97] pointed out that in the region $\Lambda < \Lambda_c$ there is another phase transition between values of Λ that lead to generalization errors $\epsilon_g(\infty) = 1/2$ and $0 < \epsilon_g(\infty) < 1/2$, respectively. This transition line is also shown in Figure 2.10.

Note that $\Lambda_c = 0$ for $\lambda = 0$ only. Hence, if output noise is present, the student network has to exploit a nonzero estimate of λ. Otherwise a zero final generalization error is not possible. This illustrates again the necessity of a λ-dependent weight function with its characteristic cutoff as discussed in Section 2.3.

The fact that there exist estimates $\Lambda \neq \lambda$ resulting in a still fast decay of ϵ_g suggests to estimate the true value of λ by means of an on-line adaptation of the parameter Λ. To this end, one has to define a simple dynamics for Λ such that it tends to λ asymptotically as desired. For instance, one can change Λ by $(1-\Lambda)/2N$ every time the student disagrees with the noisy output of the teacher. In the case of agreement Λ is changed by $-\Lambda/2N$:

$$\Lambda^{\mu+1} = \Lambda^{\mu} + \frac{1}{2N} \left[(1-\Lambda^{\mu})\, \Theta(-h_J^{\mu}\tau_n^{\mu}) - \Lambda^{\mu} \Theta(h_J^{\mu}\tau_n^{\mu}) \right]. \tag{2.52}$$

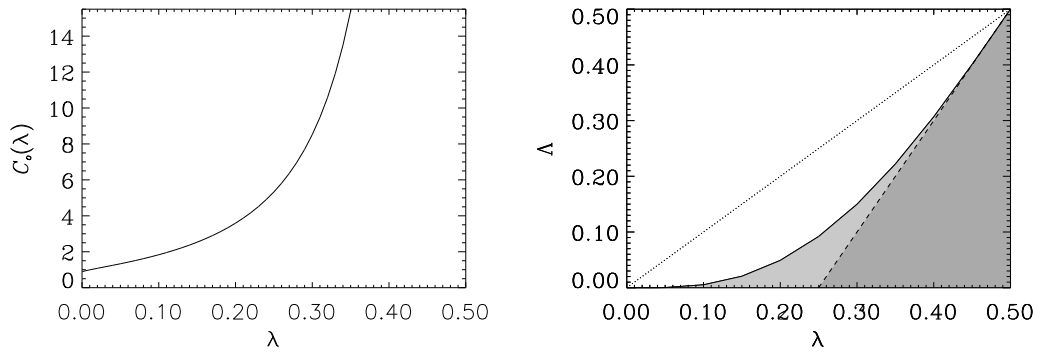


Figure 2.10: The optimal weight function for output noise requires knowledge of the noise level λ. If one replaces the noise level by an estimate Λ the asymptotic power law behaviour of ϵ_g remains unchanged: $\epsilon_g = C_o(\lambda, \Lambda)/\alpha$ for $\Lambda_c < \Lambda < 1/2$. Left: The constant $C_o(\lambda, \Lambda)$ for a fixed inversion rate $\lambda = 0.2$. The decay of ϵ_g is fastest for $\Lambda = \lambda$ and becomes slower as $\Lambda \to 1/2$ or $\Lambda \to \Lambda_c$. Below the critical value Λ_c (dashed line) the generalization error does not converge to zero. Right: Phase diagram $\Lambda(\lambda)$. Above the solid line (Λ_c) the generalization error converges to zero as $\alpha \to \infty$. On the dotted line ($\Lambda = \lambda$) the convergence is optimal. Below the solid line ϵ_g converges to a finite nonzero value, below the dashed line $\epsilon_g = 1/2$ for all values of α.

In the limit of large N this leads to the differential equation

$$\frac{d\Lambda}{d\alpha} = \frac{1}{2}(\epsilon_p - \Lambda). \tag{2.53}$$

Therefore Λ approaches ϵ_p and as α increases, ϵ_p becomes λ:

$$\lim_{\alpha\to\infty} \Lambda(\alpha) = \lim_{\alpha\to\infty} \epsilon_p(\alpha) = \lambda. \tag{2.54}$$

Now the system is described by a set of three[8] coupled differential equations for the dynamical variables ρ, Q, and Λ. By construction Λ approaches λ and the resulting generalization error asymptotically becomes identical to the optimal solution. Therefore the weight function (2.17) with the replacements $\rho \to \sqrt{Q}$ and $\lambda \to \Lambda$ from equation (2.52) provides an algorithm that realizes the optimal $(1/\alpha)$-decrease of the generalization error without requiring knowledge of ϵ_g or the noise parameter λ.

Note that one could easily speed up the estimation of ϵ_p by increasing the inverse time constant $1/2$ in (2.52). Furthermore, one could use $\sqrt{Q}$ as an estimate for the overlap ρ and access a direct estimate for the inversion rate λ via relation (2.14), rather than estimating ϵ_p as in (2.52).

Finally, the ability of this on-line algorithm to adapt to a changing noise level λ is illustrated, see Figure 2.11. The student network rapidly adapts to the noise and the generalization error decays proportional to $1/\alpha$. The dynamics of Λ can be further improved by replacing (2.52) with a schedule that approaches the asymptotic value λ even faster, as mentioned above.

[8]The identity $\rho \equiv \sqrt{Q}$ holds true only if the true noise level λ is used in f_{opt}.

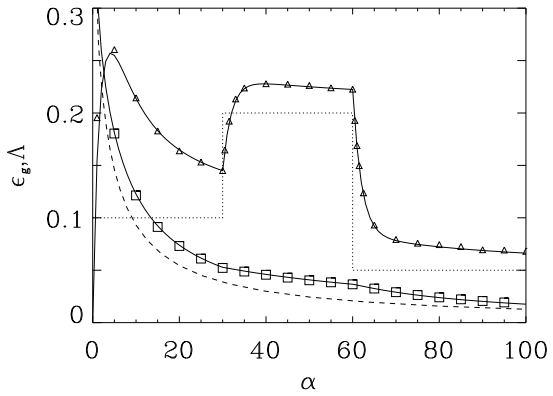


Figure 2.11: Learning with variable output noise parameter λ (dotted). Shown is the learning curve ϵ_g ($\Box$) and the evolution of Λ ($\triangle$) according to (2.52) with $\Lambda(0) = 0$. The dashed line depicts the optimal ϵ_g as obtained for a constant noise level $\lambda = 0.05$ for comparison. Simulations as in Figure 2.1.

Weight Noise

Replacing the parameter ω which measures the strength of weight noise by a constant estimate Ω results in the weight function

$$f_\Omega = f_{opt}\Big|_{\omega\to\Omega} = \frac{1}{\sqrt{2\pi}}\frac{\Omega(1-\rho^2)}{\rho\sqrt{1-\Omega^2\rho^2}}\frac{\exp\left(-\frac{1}{2}\frac{\Omega^2\rho^2}{1-\Omega^2\rho^2}h_J^2\right)}{\Phi\left(\frac{\Omega\rho}{\sqrt{1-\Omega^2\rho^2}}h_J\tau_n\right)}\tau_n. \tag{2.55}$$

The corresponding generalization error decreases asymptotically as

$$\epsilon_g(\alpha\to\infty)\begin{cases} = & C_w(\omega,\Omega)/\sqrt{\alpha} \quad \text{for } 0<\Omega<\Omega_c(\omega) \\ > & 0 \qquad\qquad\qquad\quad \text{for } 1\geq\Omega\geq\Omega_c(\omega) \end{cases}. \tag{2.56}$$

Analogously to the previous section $\Omega_c(\omega)$ is defined as the solution of the equation $C_w^{-1}(\omega,\Omega) = 0$, where

$$C_w(\omega,\Omega) = \frac{(1-\Omega^2)^{1/4}}{(\pi\Omega)^{1/2}}\left[\int_{-\infty}^{\infty}\frac{dz}{\sqrt{2\pi}}\left(2\frac{\omega}{\Omega}\sqrt{\frac{1-\Omega^2}{1-\omega^2}}\frac{\exp\left(-\frac{z^2}{2}\frac{1-\omega^2\Omega^2}{\Omega^2(1-\omega^2)}\right)}{\Phi(z)}\right.\right.$$
$$\left.\left. - \exp\left(-\frac{z^2}{2}\frac{1+\Omega^2}{\Omega^2}\right)\frac{\Phi\left(\frac{\omega}{\Omega}\sqrt{\frac{1-\Omega^2}{1-\omega^2}}z\right)}{\Phi^2(z)}\right)\right]^{-1/2}. \tag{2.57}$$

The result (2.56) states that the optimal on-line dynamics is to a certain extent robust against a misestimation of the noise parameter ω. Only if Ω has been chosen to be larger than a critical value $\Omega_c(\omega)$ the generalization error converges to a nonzero value. For all other estimates the optimal $(1/\sqrt{\alpha})$-decay for learning subject to weight noise is preserved. However, in the interval $0 < \Omega < \Omega_c(\omega)$ a mismatched value $\Omega \neq \omega$ will lead to $C_w(\omega,\Omega) > C_w(\omega,\omega) = C_w(\omega)$, Eq.

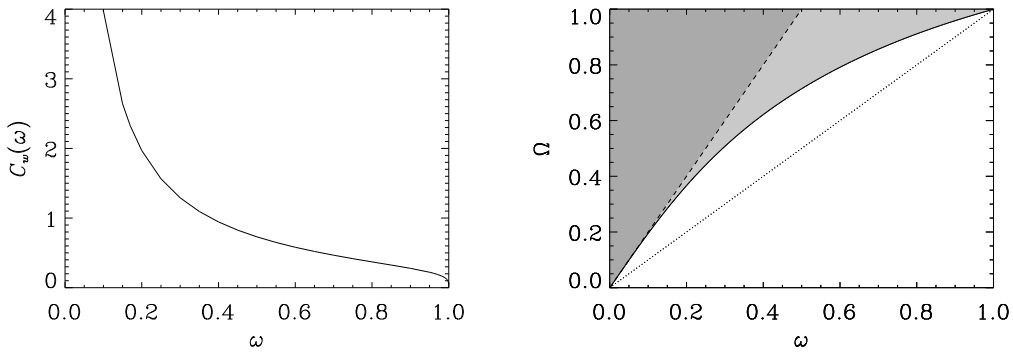


Figure 2.12: The optimal weight function makes use of the characteristics of the corrupting noise, here the fact that it is weight noise and that its strength is measured by ω which is unknown in general. If one uses in f_{opt} a constant estimate Ω instead of ω the power law behaviour of the asymptotic generalization error remains unchanged: $\epsilon_g = C_w(\omega, \Omega)/\sqrt{\alpha}$. However, this holds only true if the estimate is smaller than some ω-dependent critical value $\Omega_c(\omega)$. Left: The constant $C_w(\omega, \Omega)$ for $\omega = 0.8$. It is minimum for $\Omega = \omega$ and increases as Ω approaches 0 or Ω_c (dashed line). Right: Phase diagram $\Omega(\omega)$. Below the solid line (Ω_c) the generalization error converges to zero as $\alpha \to \infty$, above the line to a nonzero value. The convergence is optimal on the dotted line ($\Omega = \omega$).

(2.35). Hence, the asymptotics of ϵ_g will be slightly worse than in the optimal case. As for output noise the algorithm (2.55) is also robust if the estimated noise level is somewhat smaller than the actual one ($\Omega_c > \Omega > \omega$). Figure 2.12 shows the prefactor $C_w(\omega, \Omega)$ and the critical value Ω_c as a function of ω.

Qualitatively, the replacement of ω by Ω in the optimal weight function leads to the same results as in the case of output noise discussed in the previous section. The dynamics of the generalization error shows a far off equilibrium phase transition. Depending on the values of the noise parameter ω and its estimate Ω one can distinguish the following phases: A phase with zero final generalization error and one with nonzero generalization error (see Fig. 2.12). The latter one can again be separated into a phase with $0 < \epsilon_g^\infty < 1/2$ and one where $\epsilon_g = 1/2$ for all α [CEK$^+$97].

The important result is the dominance of the ($\epsilon_g^\infty = 0$)-phase in the phase diagram (Fig. 2.12). This allows for an on-line estimation of Ω, such that it approaches ω and consequently the dynamics of the generalization error approaches the optimal decay (2.34) in the process of learning. To this end one would need to define a dynamics for the estimate Ω similar to the one used to estimate the inversion rate in the last paragraph (Eq. (2.52)).

2.6 Queries

So far, the optimal algorithms discussed have been obtained by varying the equation of motion for the generalization error (2.7) with respect to the weight function. However, there is another

quantity in (2.6) which is free of choice for the student network: the probability distribution $P(h_J)$ of the student's local field h_J. By appropriately *choosing* the new examples, the student network can influence the distribution $P(h_J)$. Note that this does not change the conditional distribution of the teacher's internal field h_B given h_J. The distribution $P(h_B|h_J,\zeta)$ is still a Gaussian with mean ρh_J and variance $1-\rho^2$, if the examples are uncorrelated.

Kinzel and Ruján [KR90] argued for the noiseless case that by choosing inputs with a distribution $P(h_J) = \delta(h_J)$ the student network exclusively makes use of the examples with the highest information content. This is because these examples lie on the actual decision boundary of the student network. Pictorially speaking, the student network picks those examples it is most uncertain about and asks the teacher network for the corresponding output, hence the name *learning by queries*. In [KR90] it was shown that this learning procedure reduces the generalization error for Hebbian learning by a factor of 2 asymptotically.

For sake of simplicity, $P(h_J)$ will be restricted to Gaussian distributions of mean h_J^0 and variance Σ in the following. The optimal strategy for choosing queries is then obtained by varying the change of the generalization error per example with respect to h_J^0 and Σ.

2.6.1 Output Noise

Inserting the optimal weight function (2.17) for the case of output noise into the equations of motion (2.6) and performing the average over h_B and τ_n one obtains for the equation of motion for ρ

$$\frac{d\rho}{d\alpha} = \frac{(1-2\lambda)^2}{4\pi}\frac{1-\rho^2}{\rho}\int dh_J P(h_J) g_o(h_J), \tag{2.58}$$

where

$$\begin{aligned} g_o(h_J) &= \exp\left(-\frac{\rho^2}{1-\rho^2}h_J^2\right)\left[\left((1-2\lambda)\Phi\left(\frac{\rho}{\sqrt{1-\rho^2}}h_J\right)+\lambda\right)^{-1}\right. \\ &\qquad \left. + \left((1-2\lambda)\Phi\left(-\frac{\rho}{\sqrt{1-\rho^2}}h_J\right)+\lambda\right)^{-1}\right]. \end{aligned} \tag{2.59}$$

As stated above, $P(h_J)$ is chosen to be a Gaussian

$$P(h_J) = \frac{1}{\sqrt{2\pi}\Sigma}\exp\left(-\frac{(h_J-h_J^0)^2}{2\Sigma^2}\right). \tag{2.60}$$

Writing the integral in (2.58) as $\int Dh_J g_o(\Sigma(h_J+h_J^0))$ and varying with respect to the width Σ, it is straightforward to show that $d\rho/d\alpha$ is maximized for $\Sigma = 0$ since $g_o'(0) = 0$ and $g_o''(0) < 0$. Further variation with respect to h_J^0 yields the result that the best distribution the student can choose for queries is given by $P_{opt}(h_J) = \delta(h_J)$ as already postulated in [KR90].

Inserting $P_{opt}(h_J)$ into (2.58) yields

$$\frac{d\rho}{d\alpha} = \frac{(1-2\lambda)^2}{\pi}\frac{1-\rho^2}{\rho}, \tag{2.61}$$

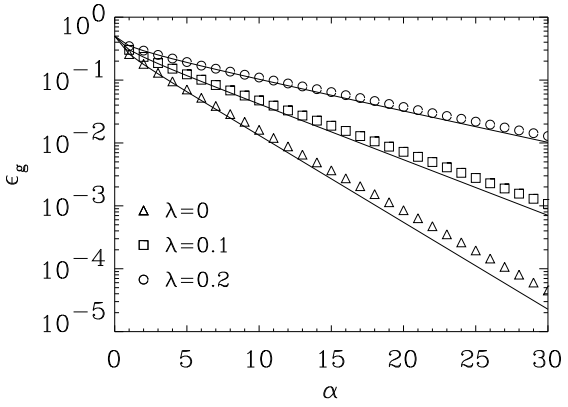


Figure 2.13: Generalization error $\epsilon_g(\alpha)$ for perceptron learning with an optimized queries strategy. The training examples are corrupted by output noise with an inversion rate λ. The analytic result (2.62) (curves) is compared to simulations (symbols). Simulations are for an input dimension $N = 100$, averaged over 50 independent runs. Error bars would be smaller than the symbol size.

which for the initial condition $\rho(0) = 0$ has the solution

$$\rho(\alpha) = \sqrt{1 - \exp\left(-\frac{2}{\pi}(1-2\lambda)^2\alpha\right)}. \tag{2.62}$$

Therefore, as $\alpha \to \infty$ the generalization error decays exponentially

$$\epsilon_g(\alpha \to \infty) = \frac{1}{\pi}\exp\left(-(1-2\lambda)^2\alpha/\pi\right). \tag{2.63}$$

For the noiseless case this result has already been obtained in [KC92b]. Figure 2.13 shows the learning curve $\epsilon_g(\alpha)$ for several values of the inversion rate λ together with simulations.

Note that since $P_{opt}(h_J) = \delta(h_J)$, the optimal weight function is effectively given by

$$f_{opt}^{(queries)} = \sqrt{\frac{2}{\pi}}(1-2\lambda)\frac{\sqrt{1-\rho^2}}{\rho}\delta(h_J)\tau_n. \tag{2.64}$$

It is interesting to not that this is essentially the same weight function obtained in Section 2.3 for the optimal weight function without queries in the limit $\rho \to 1$.

2.6.2 Weight Noise

For weight noise the equation of motion reads

$$\frac{d\rho}{d\alpha} = \frac{1}{4\pi}\frac{\omega^2(1-\rho^2)^2}{\rho(1-\omega^2\rho^2)}\int dh_J P(h_J) g_w(h_J), \tag{2.65}$$

where

$$g_w(h_J) = \exp\left(-\frac{\omega^2\rho^2}{1-\omega^2\rho^2}h_J^2\right)\left[\Phi^{-1}\left(\frac{\omega\rho}{\sqrt{1-\omega^2\rho^2}}h_J\right) + \Phi^{-1}\left(-\frac{\omega\rho}{\sqrt{1-\omega^2\rho^2}}h_J\right)\right]. \tag{2.66}$$

Note that g_w belongs to the same functional class as g_o (2.59); it can be obtained by the replacements $\lambda \to 0$ and $\rho \to \omega\rho$. Following the arguments of the previous section one finds that again the best distribution of examples the student can choose for queries is $P_{opt}(h_J) = \delta(h_J)$.

Inserting $P_{opt}(h_J)$ into the equation of motion (2.65) one obtains for the asymptotic solution

$$\epsilon_g(\alpha \to \infty) = \sqrt{\frac{1-\omega^2}{2\pi\omega^2}}\alpha^{-1/2}. \tag{2.67}$$

Hence, queries do not lead to a considerable improvement of the generalization ability for learning subject to weight noise. The generalization error decays with the same power law whether or not a query strategy is used. Note, however, that queries at least give rise to a small improvement since the generalization error (2.67) is less than $C_w(\omega)\alpha^{-1/2}$, Eq. (2.34), for all $\omega < 1$.

2.7 Interpretation and Discussion of Results

In this chapter it has been shown that for on-line learning from noisy training data the optimal learning curves can be calculated exactly. Learning has been investigated for two different types of noise in the training set: output noise and weight noise. In each case the learning algorithm converges asymptotically to the desired solution uniquely defined by the noiseless teacher network, i.e. zero generalization error can be achieved even if the training examples are noisy. On-line algorithms which achieve this have not been known in the literature before.

The optimal weight functions have been derived by a simple variational principle in a similar way as in the well-known brachistochrone problem (see e.g. [Gol80]). This general procedure is very appealing from two points of view. First, it represents a powerful approach to a general theory of machine learning. It is independent of *ad hoc* assumptions and provides a way to design algorithms starting from first principles. This will be discussed further below. Second, as already mentioned above, this procedure allows one to obtain the features necessary for an optimal decay of the generalization error . This is more reasonable and convincing than the usual approach where the features are put in by hand based on some sort of intuition or human insight into the problem. Moreover, due to the nature of the variational approach the obtained weight functions are completely parameterfree, again an inherent property of a first principles' theory.

By construction the discussed algorithms are optimal, i.e. at each time step α in the on-line learning process the generalization error is as low as possible, on average over the randomness of the examples. However, the most central result of this chapter is not that the presented algorithms are optimal. More than that, the optimal weight function teaches which kind of features an algorithm has to exploit in order to give rise to fast convergence.

Obviously, the two types of corrupted data require rather different learning strategies. Nevertheless the optimal features the respective weight functions make use of for either type of noise are connected to the following: The dependence of the weight function on the aligned field, the dependence on the characteristics of the noise process, and the dependence on the performance $\rho(\alpha)$. These dependences have been determined easily by applying the variational principle.

In the case of output noise the most important feature of the algorithm has been shown to be the cutoff of aligned fields which are negative and large in absolute value. Due to this noise level-

and performance-dependent cutoff, examples that have most likely been corrupted by output noise are not taken into account for learning. For weight noise, the dependence of the weight function on the aligned field is such that the weight assigned decreases with an increasing value of the aligned field and increasing performance ρ.

The general dependence of the optimal weight functions on the performance $\rho(\alpha)$ is an essential and quite appealing feature. One can view the performance-dependent prefactors or amplitudes in the optimal weight functions (2.17,2.32) to be the equivalent of what is usually called a *learning rate*. Such learning rates are often introduced phenomenologically into a training algorithm in order to make it convergent. In general, it is necessary that the learning rate decreases with α [HK91]. This result has been found here quite naturally by application of the variational principle: The prefactors of the optimal weight functions decay with α.

However, the important difference is that the effective learning rate of the optimal weight functions truly depends on ρ and not simply on α. The weight assigned to a particular example for a given α is the less the smaller the generalization error, and not the larger α. Imagine the rule to be learnt changes with time (on a time scale large compared to the sample rate). In this case the ρ-dependent equivalent of the learning rate implemented in the optimal weight function is relatively high in value. This allows to "overwrite" the information provided by the old rule which is not valid any more. In contrast, a simple learning rate decreasing with α would assign less weight to the more relevant, new examples than it did for the previous, now irrelevant examples.

Consequently, *ad hoc* introduced learning rates (as in [KS96] for perceptron learning) have to be truly performance-dependent and not simply α-dependent. This has been known before (e.g. [BSS95]). However, while the phenomenological approach requires a try and error approach combined with human insight, the variational approach reveals the relevant features automatically. In the case of a rule change, it does so without even assuming the possibility of such a change [KC93].

By construction, the optimal feature algorithms require knowledge about certain quantities that are not readily accessible, such as the performance $\rho(\alpha)$ or the amount of noise corrupting the training data. Although it can be shown that the inaccessibility of the performance is not a real problem (since $\rho \equiv \sqrt{Q}$ and Q is an accessible quantity), the underlying message is quite different: *If an optimal weight function requires knowledge about some not directly accessible quantities one has to estimate them.* This can be easily done as has been shown in [KC93] for the estimation of the performance ρ and in this chapter for the estimation of the noise characteristics λ and ω, respectively.

The estimation of these noise parameters leads to an interesting dynamical phase transition. Depending on the value of the estimated noise level there are essentially two distinct phases, one that basically preserves the dynamics of the optimal weight function and one where the rule is not learnt perfectly. The dominance of the first phase makes the generalization properties of the optimal algorithms quite robust and allows for an on-line estimation of the noise parameter.

In consecutive work [CEK$^+$97] it has been shown that the encountered phase transitions are quite general. For simple two-layer networks the phase diagram obtained in Section 2.5 for the case of output noise is universal, i.e. it is the same for a wide class of architectures. Although the

corresponding phase diagram is not universal for weight noise, it does not change considerably for the two-layer networks considered in [CEK$^+$97].

Strictly speaking, the optimal weight functions do not only require knowledge about the characteristic parameters λ and ω, respectively, but also exploit the fact which *type* of noise has corrupted the training data. Analogous to the estimation of λ and ω, it is not too difficult a task to "estimate" the type of noise. For example, this can be done by assuming that the training data has been corrupted by both types of noise. The optimal weight function will then depend on both λ and ω which again can be replaced by estimates Λ and Ω. If, for instance, the noise is exclusively due to output noise then an on-line estimation of the noise strength will lead to $\Lambda = \lambda$ and $\Omega = 0$ [Grä97], hence predicting that the training data have been affected exclusively by output noise with an inversion rate λ.

Finally, it should be emphasized that the performance of on-line learning does not differ considerably from that of batch-learning which is often considered to be superior. For the two cases considered here, the asymptotic generalization error for optimal on-line learning differs from that of optimal batch learning (Bayes optimal) by at most a factor of 2. For weight noise the asymptotics is even the same for both types of training algorithms. Hence, optimal on-line algorithms can be suspected to give results comparable to batch-training for other training tasks as well.

Chapter 3

On-line Backpropagation in Two-layer Neural Networks

In the physics literature, so far most of the analysis has been restricted to very simple networks like the single layer perceptron [HKP91] or networks with one layer of hidden units and a fixed hidden-to-output relation, e.g. the so-called committee machine [WRB93]. In the following the recent investigation of learning by on-line gradient descent [BS95, SS95a] will be extended to two-layer networks with adjustable weights connecting the hidden units and the output. This topic is of crucial importance as systems with variable hidden-to-output relations can realize more complex classification schemes and are commonly used in practical applications of neural networks [HKP91, CR95].

In contrast to the previous chapters, the networks investigated here have continuous outputs. More specifically, their transfer functions are differentiable and therefore the outputs are differentiable functions of the inputs. This allows the application of a learning prescription which corresponds to an on-line version of the so-called *backpropagation of error* [HKP91], a method widely used in practice [CR95]. Nevertheless, the results obtained here for continuous transfer functions will indicate some properties of two-layer networks with threshold units.

This chapter is organized in the following way: Section 3.1 introduces the on-line version of the backpropagation algorithm. The equations of motion for the order parameters will be derived. In Section 3.2, the so-called Y-network, which is the basic unit of the two-layer networks considered in this chapter, will be discussed. The solution of its equations of motion will lead to the remarkable result that dynamical properties are independent of the second layer to a large extent. This result will be generalized to larger two-layer networks in Sections 3.3 (tree-like architecture) and 3.4 (fully connected architecture). The scaling properties of the characteristic quantities in the asymptotic regime will motivate that it is advantageous to put the dynamics of the second layer on a much faster time scale compared to the first layer. The corresponding equations of motion will be solved by adiabatically eliminating the second layer degrees of freedom. This will be discussed in Section 3.5. Section 3.6 will close with a summary. Results of this chapter have also partially been published in [RB95, BRW96].

3.1 The Framework

3.1.1 The Backpropagation Algorithm

Consider a student network with a single continuous output $\sigma(\boldsymbol{W}, \boldsymbol{\xi})$ where $\boldsymbol{W}$ denotes the set of all weights in the network. As before, $\boldsymbol{\xi}$ denotes an example input vector. A sequence of examples $\{\boldsymbol{\xi}^\mu, \tau^\mu\}$ for an unknown rule is provided on-line by the environment in order to train the network. Here, $\tau^\mu = \tau(\boldsymbol{\xi}^\mu) \in I\!R$ is the corresponding correct rule output.

The frequently used quadratic error measure

$$\epsilon(\boldsymbol{W}, \boldsymbol{\xi}) = \frac{1}{2}[\sigma(\boldsymbol{W}, \boldsymbol{\xi}) - \tau(\boldsymbol{\xi})]^2 \tag{3.1}$$

quantifies the degree of disagreement between the student and the rule output for a particular input. Throughout this chapter the inputs $\boldsymbol{\xi}$ will be considered to be independently drawn vectors with uncorrelated random components of zero mean and unit variance. Denoting the average over this input distribution by $\langle \ldots \rangle_\xi$ the generalization error is defined by

$$\epsilon_g(\boldsymbol{W}) = \langle \epsilon(\boldsymbol{W}, \boldsymbol{\xi}) \rangle_\xi. \tag{3.2}$$

This quantity measures the validity of the hypothesis for the rule. The hypothesis is defined through the student architecture and its weights $\boldsymbol{W}$.

In the on-line scheme a new, uncorrelated example is presented at each learning step μ and the set of student weights is updated instantaneously. In the following, the update of the weights $\boldsymbol{W}$ will be taken according to the gradient of $\epsilon(\boldsymbol{W}, \boldsymbol{\xi}^\mu)$ with respect to the weights:

$$\begin{aligned} \boldsymbol{W}^{\mu+1} &= \boldsymbol{W}^\mu - \tilde{\eta}\, \nabla_{\boldsymbol{W}} \epsilon(\boldsymbol{W}, \boldsymbol{\xi})|_{\boldsymbol{W}^\mu} \\ &= \boldsymbol{W}^\mu - \tilde{\eta}[\sigma(\boldsymbol{W}^\mu, \boldsymbol{\xi}^\mu) - \tau^\mu]\, \nabla_{\boldsymbol{W}} \sigma(\boldsymbol{W}, \boldsymbol{\xi}^\mu)|_{\boldsymbol{W}^\mu}\,. \end{aligned} \tag{3.3}$$

The learning rate $\tilde{\eta}$ controls the size of the steps made in the direction of steepest descent.

The update rule (3.3) is also known as backpropagation of error. The origin of the notion backpropagation will become clear in the next section, where this algorithm will be applied to two-layer networks.

3.1.2 Two-layer Networks

The type of student network considered in this chapter will be two-layer networks. Such feed-forward networks consist of a layer of N inputs, a layer of K hidden units, and a single output; see Figure 1.1. The particular input-output relation is given by

$$\sigma(\boldsymbol{\xi}) = f\left(\sum_{i=1}^{K} w_i g(\boldsymbol{J}_i \cdot \boldsymbol{\xi})\right). \tag{3.4}$$

$\boldsymbol{J}_i$ denotes the N-dimensional weight vector of the i-th input branch and w_i the weight connecting the i-th hidden unit with the output.

In the following, the choice of the hidden-to-output transfer function $f(\cdot)$ will be almost exclusively a linear function. This does not restrict the capability of a student network of type (3.4) to learn a complex rule. Two-layer networks with a linear output are known to be universal approximators as has been shown by Cybenko [Cyb89]:

Theorem 1 (Cybenko) *Let g be any continuous sigmoidal function and $\boldsymbol{J}_j \in I\!R^N$, $w_j, \theta_j \in I\!R$. Then finite sums of the form*

$$\sigma(\boldsymbol{\xi}) = \sum_{j=1}^{K} w_j g(\boldsymbol{J}_j \cdot \boldsymbol{\xi} + \theta_j) \tag{3.5}$$

are dense in the space of continuous functions $C(I_N)$. In other words, given any $\tau \in C(I_N)$ and $\epsilon > 0$, there is a sum, $\sigma(\boldsymbol{\xi})$, of the above form, for which

$$|\tau(\boldsymbol{\xi}) - \sigma(\boldsymbol{\xi})| < \sqrt{2\epsilon} \quad \forall \boldsymbol{\xi} \in I_N.$$

There are several remarks in place here. First, the restriction $\xi_i \in I = [0, 1]$ can be easily removed by rescaling the real valued weights $\boldsymbol{J}_j$. Second, the two-layer network (3.4) does not make use of the so-called *thresholds* θ_j. These are omitted here to simplify the analysis. The dynamics of the thresholds will be analyzed later in Section 3.5. Of course, due to this restriction the network (3.4) will not be a universal approximator in general. This is not important for the following, however, where the main concern will be the dynamics of two-layer networks and the way how to describe it by physics.

Finally, it is worth noting that Cybenko's Theorem does not put a limit on the number of hidden units K. This number can be very large in general. In this work, however, K is chosen to be finite, i.e. K does not scale with the dimension N of the weight vectors $\boldsymbol{J}_j$.

As a specific example, consider the learning of a rule which can be parametrized in terms of a teacher neural network with one hidden layer of M continuous nodes with sigmoidal activation function and a single linear output unit. Two different types of network models will be distinguished in the following:

In a *fully connected architecture* all M hidden units receive information from the same N input nodes. The total output of such a teacher network is given by

$$\tau(\boldsymbol{\xi}) = \sum_{n=1}^{M} v_n g(y_n) \quad \text{with} \quad y_n = \boldsymbol{B}_n \cdot \boldsymbol{\xi} \tag{3.6}$$

with weights $\boldsymbol{B}_n \in I\!R^N$, connecting the n-th hidden unit with the input $\boldsymbol{\xi} \in I\!R^N$, and the set $\{v_n\}_{n=1,\ldots,M}$ of hidden-to-output connections. A network of this type is depicted in Figure 1.1. Such machines are also said to have an *overlapping architecture*, because the receptive fields overlap.

The second type of network is the so-called *tree-like architecture*. Here, the hidden units are connected to nonoverlapping receptive fields, each of which is taken to be N-dimensional; see Fig. 3.1. Thus, the teacher output is also of the form (3.6) but with $y_n = \boldsymbol{B}_n \cdot \boldsymbol{\xi}_n$ where $\boldsymbol{\xi}_n$ is the n-th subset of the $(M \cdot N)$-dimensional input $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, \boldsymbol{\xi}_2, \ldots, \boldsymbol{\xi}_M)$. Since the receptive fields do not overlap the tree-like architecture is also referred to as *nonoverlapping architecture*. Note

that Cybenko's Theorem does not apply to the tree-like architecture. Although the corresponding neural networks are not necessarily universal generalizers, their complexity is still considerable.

The student is taken to be a network of the corresponding architecture with K hidden units, input-to-hidden weights $\boldsymbol{J}_i \in I\!R^N$, and adjustable hidden-to-output weights w_i, $i = 1, \ldots, K$. Its output is given by

$$\sigma(\boldsymbol{\xi}) = \sum_{i=1}^{K} w_i g(x_i). \tag{3.7}$$

The quantities x_i are defined as $x_i = \boldsymbol{J}_i \cdot \boldsymbol{\xi}$ in the overlapping architecture with an N-dimensional input and $x_i = \boldsymbol{J}_i \cdot \boldsymbol{\xi}_i$ in the case of K nonoverlapping receptive fields. The quantities $\{x_i, y_n\}$ correspond to the internal fields for perceptron learning (cf. Section 1.3) which absorb the randomness of the inputs and which will allow the transition from the microscopic equations of motion to macroscopic ones.

Given the functional form of student and teacher networks the equations of motion for the student network are readily obtained now. In both cases, the learning rate in equation (3.3) will be scaled with the number of inputs to a single hidden unit: $\tilde{\eta} = \eta/N$. For the microscopic equations of motion one therefore obtains

$$\begin{aligned} \boldsymbol{J}_i^{\mu+1} &= \boldsymbol{J}_i^{\mu} - \frac{\eta}{N} \Gamma_i^{\mu} \boldsymbol{\xi}^{\mu} \\ w_i^{\mu+1} &= w_i^{\mu} - \frac{\eta}{N} \Delta^{\mu} g(x_i^{\mu}) \end{aligned} \tag{3.8}$$

for the fully connected machine and

$$\begin{aligned} \boldsymbol{J}_i^{\mu+1} &= \boldsymbol{J}_i^{\mu} - \frac{\eta}{N} \Gamma_i^{\mu} \boldsymbol{\xi}_i^{\mu} \\ w_i^{\mu+1} &= w_i^{\mu} - \frac{\eta}{N} \Delta^{\mu} g(x_i^{\mu}) \end{aligned} \tag{3.9}$$

for the tree-like architecture, where

$$\Delta = \sigma(\boldsymbol{\xi}) - \tau(\boldsymbol{\xi}) = \left[\sum_{i=1}^{K} w_i g(x_i) - \sum_{n=1}^{M} v_n g(y_n) \right] \quad \text{and} \quad \Gamma_i = w_i \, g'(x_i) \, \Delta. \tag{3.10}$$

In both cases the update of the weights is of the generic form (1.6). In particular note that the change of the second layer weights w_i is proportional to their inputs $g(x_i)$. Furthermore, the change in w_i is weighted by the linear error Δ for the present example. The weight factor Γ_i in the update of the first layer couplings $\boldsymbol{J}_i$ can be viewed as the error Δ being "propagated back" through the nonlinearity $g(x_i)$ [HKP91].

As for the perceptron the next step will be to exploit the thermodynamic limit $N \to \infty$, while $M, K \propto \mathcal{O}(1)$, in order to transform the microscopic equations of motion for the $K \cdot (N+1)$ degrees of freedom into a finite number of macroscopic equations.

3.1.3 Equations of Motion

In order to obtain results for large systems, i.e. in the thermodynamic limit $N \to \infty$, $M, K \propto \mathcal{O}(1)$, it is necessary to identify quantities which are selfaveraging in this limit and which allow

to write macroscopic equations of motion in a closed form. In the following it will be argued that the $\{w_i\}$ and the overlaps

$$R_{im} = \boldsymbol{J}_i \cdot \boldsymbol{B}_m, \qquad Q_{ij} = \boldsymbol{J}_i \cdot \boldsymbol{J}_j, \qquad m = 1, \ldots, M \quad \text{and} \quad i, j = 1, \ldots, K. \tag{3.11}$$

are appropriate macroscopic degrees of freedom.

It might be surprising that the second layer weights $\{w_i\}$ by themselves are appropriate for a macroscopic description while the first layer weights $\{\boldsymbol{J}_i\}$ are not. The reason for this is twofold. First, the number K of w_i is $\mathcal{O}(1)$, i.e. it does not scale with the system size N. Therefore, there is no need to compress a large number of microscopic degrees of freedom into a small number of macroscopic order parameters as for the $\boldsymbol{J}_i$. Second, the change in w_i scales with $1/N$. As one can easily see this causes the variance $\sigma_i^2 = \langle (w_i)^2 \rangle - \langle w_i \rangle^2$ to obey the recursion relation

$$\begin{aligned} (\sigma_i^{\mu+1})^2 - (\sigma_i^{\mu})^2 &= -2\frac{\eta}{N}[\langle \Delta^\mu g(x_i^\mu) w_i^\mu \rangle - \langle \Delta^\mu g(x_i^\mu) \rangle \langle w_i^\mu \rangle] \\ &\quad + \frac{\eta^2}{N^2}\Big[\Big\langle (\Delta^\mu g(x_i^\mu))^2 \Big\rangle - \langle \Delta^\mu g(x_i^\mu) \rangle^2\Big], \end{aligned} \tag{3.12}$$

where $\langle \ldots \rangle$ denotes the average over the randomness of the inputs $\boldsymbol{\xi}$. Under reasonable assumptions it can be shown that the first term on the r.h.s. is also $\mathcal{O}(1/N^2)$; see Appendix D for details. Consequently the variance of the w_i is $\mathcal{O}(1/N)$, i.e. it vanishes in the thermodynamic limit $N \to \infty$ for fixed initial condition w_i^0 ($(\sigma_i^0)^2 = 0$). In other words, the randomness in the presentation of $\boldsymbol{\xi}$ does not result in a finite width of the distribution of w_i. Hence, the second layer weights w_i are selfaveraging.[1]

At this point it should be noted that the origin of the selfaveraging properties of the second layer weights w are the same as for the noise estimator Λ discussed in Section 2.5. There, this degree of freedom was selfaveraging in the thermodynamic limit for the same reason, namely the $(1/N)$-scaling of the microscopic dynamics.

The overlaps R_{im} and Q_{ij} are selfaveraging quantities as well, for the same reasons as for the perceptron order parameter R and Q discussed in Chapter 1. This is not too surprising since for instance R_{im} is the overlap of the perceptron representing the i-th student branch with the perceptron representing the m-th teacher branch.

It is straightforward to derive from the microscopic backpropagation dynamics (3.8,3.9) recursion relations for the mean values of all these quantities by performing the average over the latest example input [BS95, SS95a]. Since in the thermodynamic limit the overlap parameters $\{R_{im}, Q_{ij}\}$ as well as the adjustable weights $\{w_i\}$ become selfaveraging quantities, the description in terms of their mean values is sufficient. In the same limit one can interpret $\alpha = \mu/N$ as a "continuous time" and obtains ordinary differential equations for the evolution of the learning

[1] It should be emphasized that the selfaveraging property is with respect to the average over an ensemble of different realizations of sequences $\left\{\boldsymbol{\xi}^1, \ldots, \boldsymbol{\xi}^{\alpha N}\right\}$. Of course the selfaveraging is not with respect to an ensemble of student networks that start with different initial values of $\{w_i\}$. The same holds true for the perceptron order parameter ρ, for instance. If one prepared an ensemble of perceptrons such that the initial values of ρ had finite variance, then the variance of ρ would remain finite for intermediate values of α.

network:

$$\frac{dR_{im}}{d\alpha} = -\eta\langle\Gamma_i y_m\rangle, \qquad \frac{dQ_{ij}}{d\alpha} = -\eta\langle\Gamma_i x_j + \Gamma_j x_i\rangle + \eta^2\langle\Gamma_i\Gamma_j\rangle, \qquad \frac{dw_i}{d\alpha} = -\eta\langle g(x_i)\,\Delta\rangle \tag{3.13}$$

The averages are over the $(M \cdot K)$-dimensional Gaussian distribution (cf. Appendix A) of the $\{x_i, y_m\}$ which is determined through the correlations

$$\langle x_i x_j\rangle = Q_{ij}, \quad \langle x_i y_m\rangle = R_{im}, \quad \langle y_m y_n\rangle = \boldsymbol{B}_m \cdot \boldsymbol{B}_n \equiv T_{mn} \tag{3.14}$$

for overlapping architectures and

$$\langle x_i x_j\rangle = \delta_{ij} Q_{ij}, \quad \langle x_i y_m\rangle = \delta_{im} R_{im}, \quad \langle y_m y_n\rangle = \delta_{mn} T_{mn} \tag{3.15}$$

in the case of tree-like student and teacher networks.

As an example $g(z) = \mathrm{erf}\left(z/\sqrt{2}\right)$ will be chosen as a sigmoidal activation function in the following, but the results should not depend upon this choice crucially. The advantage is that all averages in (3.13) can be performed exactly [BS95, SS95a] and a numerical integration of the differential equations yields the evolution of the overlaps and hidden-to-output weights, and thus the learning curve $\epsilon_g(\alpha)$.

The averages in (3.13) require the evaluation of two types of multivariate Gaussian integrals. Terms proportional to η involve the three-dimensional integral

$$I_3 \equiv \langle g'(r) s g(t)\rangle, \tag{3.16}$$

where the argument r of g' is one the student's internal fields $\{x_i\}$, while both s and t can represent any $\{x_i, y_n\}$. Terms proportional to η^2 require the computation of the four-dimensional Gaussian integral

$$I_4 \equiv \langle g'(r) g'(s) g(t) g(u)\rangle, \tag{3.17}$$

where $r, s \in \{x_i\}$ while t and u can be any internal field $\{x_i, y_n\}$.

Using the generic integrals I_3 and I_4 the equations of motion of the macroscopic quantities $\{R_{im}, Q_{ij}, w_i\}$ can be presented in the compact form

$$\begin{aligned}
\frac{dR_{im}}{d\alpha} &= \eta w_i \left[\sum_{n=1}^{M} v_m I_3(i,m,n) - \sum_{j=1}^{K} w_j I_3(i,m,j)\right] \\
\frac{dQ_{ij}}{d\alpha} &= \eta w_i \left[\sum_{m=1}^{M} v_m I_3(i,j,m) - \sum_{k=1}^{K} w_k I_3(i,j,k)\right] \\
&\quad + \eta w_j \left[\sum_{m=1}^{M} v_m I_3(j,i,m) - \sum_{k=1}^{K} w_k I_3(j,i,k)\right] \\
&\quad + \eta^2 w_i w_j \left[\sum_{n=1}^{M}\sum_{m=1}^{M} v_n v_m I_4(i,j,n,m) - 2\sum_{k=1}^{K}\sum_{n=1}^{M} w_k v_n I_4(i,j,k,n)\right. \\
&\qquad \left. + \sum_{k=1}^{K}\sum_{l=1}^{K} w_k w_l I_4(i,j,k,l)\right]
\end{aligned}$$

$$\frac{dw_i}{d\alpha} = \frac{2}{\pi}\eta\left[\sum_{n=1}^{M} v_n \arcsin\left(\frac{R_{in}}{\sqrt{1+T_{nn}}\sqrt{1+Q_{ii}}}\right) - \sum_{j=1}^{K} w_j \arcsin\left(\frac{Q_{ij}}{\sqrt{1+Q_{ii}}\sqrt{1+Q_{jj}}}\right)\right]. \tag{3.18}$$

Arguments assigned to I_3 and I_4 are to be interpreted following the convention that indices $1 \leq i, j, k, l \leq K$ refer to the student network and $1 \leq m, n \leq M$ to the teacher network. For instance $I_3(i, n, j) \equiv \langle g'(x_i) y_n g(x_j) \rangle$, and the average is performed using the three-dimensional Gaussian in x_i, x_j, and y_n. Expressions for the generic integrals I_3 and I_4 will be given in Appendix A. For proofs and further properties of these integrals the reader is referred to [SS95a, SS95b].

The evaluation of the multivariate Gaussian integrals I_3 and I_4 is straightforward, however, it leads to rather lengthy results. For this reason only the equation of motion for the first layer student-teacher overlaps R_{im} will be given here explicitly:

$$\frac{dR_{im}}{d\alpha} = \frac{2}{\pi}\eta w_i \left[\sum_{n=1}^{M} v_n \frac{T_{mn}(1+Q_{ii}) - R_{im}R_{in}}{(1+Q_{ii})\sqrt{(1+Q_{ii})(1+T_{nn}) - R_{in}^2}} - \sum_{j=1}^{K} w_j \frac{R_{jm}(1+Q_{ii}) - R_{im}Q_{ij}}{(1+Q_{ii})\sqrt{(1+Q_{ii})(1+Q_{jj}) - Q_{ij}^2}}\right]. \tag{3.19}$$

The generic form of the equations of motion for Q_{ij} and w_i is similar, in particular the symmetry properties to be discussed below are the same. Their explicit form, however, is not necessary in the following in order to derive and discuss the essential properties of on-line backpropagation.

For the generalization error (3.2) one obtains after averaging over the joint distribution of the $\{x_i, y_m\}$

$$\epsilon_g = \frac{1}{\pi}\left[\sum_{i,k=1}^{K} w_i w_k \arcsin\left(\frac{Q_{ik}}{\sqrt{1+Q_{ii}}\sqrt{1+Q_{kk}}}\right) + \sum_{m,n=1}^{M} v_m v_n \arcsin\left(\frac{T_{nm}}{\sqrt{1+T_{mm}}\sqrt{1+T_{nn}}}\right) - 2\sum_{i=1}^{K}\sum_{n=1}^{M} w_i v_n \arcsin\left(\frac{R_{in}}{\sqrt{1+Q_{ii}}\sqrt{1+T_{nn}}}\right)\right]. \tag{3.20}$$

Note that this expression is invariant under a simultaneous inversion of the order parameters R_{im}, Q_{ij} ($i \neq j$), and w_i or an inversion of R_{im}, T_{nm} ($n \neq m$), and the corresponding second layer coupling v_m in the teacher network. Of course this inversion symmetry stems from the invariance of the outputs of the student and teacher networks (3.6,3.7) under an inversion of the microscopic quantities $\boldsymbol{J}_i$ and w_i in the student network or $\boldsymbol{B}_m$ and v_m in the teacher network.

Due to this inherent symmetry a single realization of backpropagation learning for the networks considered here will lead to a positive or negative R_{im} with equal probability. Consequently an ensemble average over several single realization would result in a zero value for R_{im}.

Therefore one has to remove this symmetry. This can be most easily done "by hand", for instance by multiplying R_{im} and w_i with the corresponding sign such that $R_{im} > 0$ for sufficiently large values of α.

For the overlapping architecture there is another inherent symmetry which is due to the invariance of the respective networks under a permutation of the different branches. This will be discussed further in Section 3.4 where the dynamics of such networks will be analyzed.

The equations of motion (3.13) describe the dynamics of the student network for an arbitrary number of hidden units K (restricted to $K \ll N$). Depending on the actual number of hidden units in the teacher network one can distinguish among three possible situations. For $K = M$ the learning task is completely realizable, i.e. the rule is perfectly learnable for the student. This situation will be considered exclusively in this work. An extension to unlearnable rules (e.g. $K < M$) and oversophisticated students ($K > M$) can be found in [BRW96, Wöh96, BRW].

3.2 Y-Architecture

As the simplest model for a two-layer neural network the "Y-shaped" network (Fig. 3.1) will be discussed first. This architecture consists of a simple perceptron with an output function $g(\cdot)$ in the first layer and a single coupling w ($K = 1$) in the second layer:

$$\sigma(\boldsymbol{\xi}) = wg(\boldsymbol{J} \cdot \boldsymbol{\xi}). \tag{3.21}$$

Since the focus is on learnable rules the teacher network is of the same architecture as the student network and is parametrized by the set $\{\boldsymbol{B}, v\}$:

$$\tau(\boldsymbol{\xi}) = vg(\boldsymbol{B} \cdot \boldsymbol{\xi}). \tag{3.22}$$

Although this network is nothing more than a toy model it comprises many features of larger architectures and the mathematical methods are most easily explained for the Y-network. In the following section the equations of motion for this type of network will be solved. The results will indicate some remarkable generic properties of on-line backpropagation in two-layer neural networks. These properties will be formally proven in Section 3.2.2 for a more general Y-network than (3.21).

3.2.1 Solution of the Equations of Motion

For the Y-network ($K = 1$) the averages in the equations of motion (3.13) can be carried out analytically for the choice $g(z) = \mathrm{erf}(z/\sqrt{2})$ as a transfer function:

$$\begin{aligned}
\frac{dR}{d\alpha} &= \frac{2}{\pi}\eta\frac{w}{1+Q}\left(v\frac{T(1+Q)-R^2}{\sqrt{(1+Q)(1+T)-R^2}} - w\frac{R}{\sqrt{1+2Q}}\right) \\
\frac{dQ}{d\alpha} &= \frac{4}{\pi}\eta\frac{w}{1+Q}\left(v\frac{R}{\sqrt{(1+Q)(1+T)-R^2}} - w\frac{Q}{\sqrt{1+2Q}}\right)
\end{aligned}$$

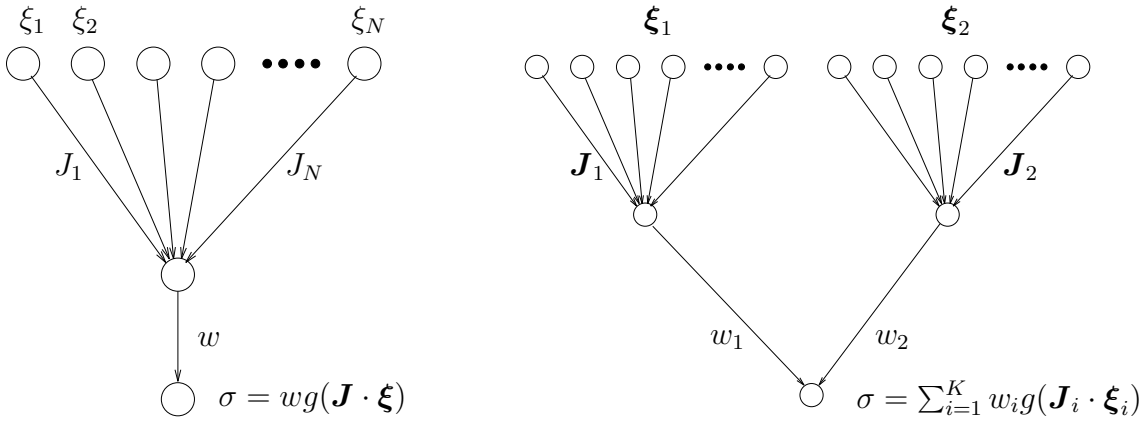


Figure 3.1: Illustration of the Y-network (left) and the tree-like architecture with $K = 2$ hidden units (right), showing the notation for units and weights. For the tree-like architecture each branch consists of N inputs.

$$+\frac{4}{\pi^2}\eta^2\frac{w^2}{\sqrt{1+2Q}}\left[v^2\arcsin\left(\frac{(1+2Q)T-2R^2}{(1+2Q)(1+T)-2R^2}\right)+w^2\arcsin\left(\frac{Q}{1+3Q}\right)\right.$$
$$\left.-2vw\arcsin\left(\frac{R}{\sqrt{1+3Q}\sqrt{(1+2Q)(1+T)-2R^2}}\right)\right]$$
$$\frac{dw}{d\alpha} = \frac{2}{\pi}\eta\left[v\arcsin\left(\frac{R}{\sqrt{1+T}\sqrt{1+Q}}\right)-w\arcsin\left(\frac{Q}{1+Q}\right)\right]. \tag{3.23}$$

For the generalization error (3.2,3.20) as a function of the order parameters R, Q, and w one obtains

$$\epsilon_g = \frac{1}{\pi}\left[w^2\arcsin\left(\frac{Q}{1+Q}\right)+v^2\arcsin\left(\frac{T}{1+T}\right)-2vw\arcsin\left(\frac{R}{\sqrt{1+T}\sqrt{1+Q}}\right)\right]. \tag{3.24}$$

Figure 3.2 shows the generalization error obtained by a numerical integration of the equations of motion (3.23) for several values of the learning rate η.

It is easy to check that the equations of motion (3.23) have a fixed point at the desired solution $R = Q = T$, $w = v$: Defining f–lim to mean

$$\text{f–lim}(...) = \lim_{R\to T, Q\to T, w\to v}(...), \tag{3.25}$$

one obtains from (3.23)

$$\text{f–lim}\,\frac{dR}{d\alpha} = 0\,,\ \text{f–lim}\,\frac{dQ}{d\alpha} = 0\,,\ \text{f–lim}\,\frac{dw}{d\alpha} = 0. \tag{3.26}$$

The second fixed point $-R = Q = T$, $-w = v$ stemming from the inversion symmetry of the dynamics can be discarded according the convention that one requires $R > 0$ asymptotically.

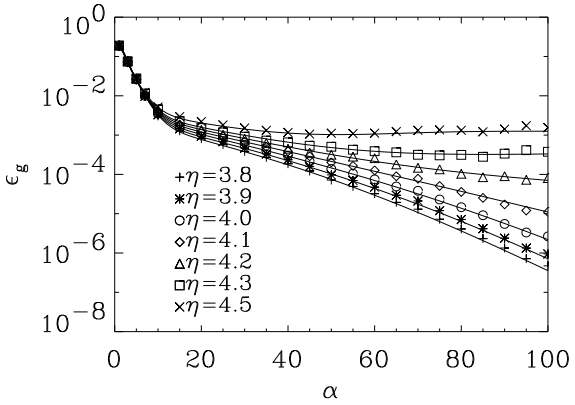


Figure 3.2: Generalization error of the Y-network for different learning rates η. The analytic results (full curves) are compared to simulations (symbols) for a network with $N = 100$ input units averaged over 10 runs (standard error bars would be approximately the size of the symbols). All curves are for initial conditions $R(0) = 0$, $Q(0) = 1.0$, and $w(0) = 0.5$. The teacher network is parametrized by $T = 2.0$ and $v = 1.0$.

Of course this analysis only shows the existence of the fixed point and does not reveal anything about its stability. In oder to check the stability and to solve for the asymptotics of the dynamical variables R, Q, and w the equations of motion (3.23) will be linearized for small deviations from the fixed point $R_\infty = Q_\infty = T, w_\infty = v$. Defining $\mathbf{V}^{(3)} = (R-T, Q-T, w-v)^\top$, the linearized equations of motion are of the form

$$\frac{d\mathbf{V}^{(3)}}{d\alpha} = \mathbf{m}^{(3)}\mathbf{V}^{(3)}, \tag{3.27}$$

where here and in the following the upper index denotes the dimension of the corresponding square matrix. By definition the elements of $\mathbf{m}^{(3)}$ are

$$\begin{aligned} m_{11} &= \text{f–lim}\,\frac{\partial}{\partial R}\frac{dR}{d\alpha}\,, m_{12} = \text{f–lim}\,\frac{\partial}{\partial Q}\frac{dR}{d\alpha}\,, m_{13} = \text{f–lim}\,\frac{\partial}{\partial w}\frac{dR}{d\alpha}\,, \\ m_{21} &= \text{f–lim}\,\frac{\partial}{\partial R}\frac{dQ}{d\alpha} \qquad \text{and so forth.} \end{aligned}$$

The asymptotics (3.27) is governed by the eigenvalues of the matrix $\mathbf{m}^{(3)}$. For nonsingular $\mathbf{m}^{(3)}$ the solution will be $\mathbf{V}^{(3)} \propto \exp(\lambda_{max}(\eta)\alpha)$, where $\lambda_{max}(\eta)$ denotes the largest eigenvalue of $\mathbf{m}^{(3)}$. This eigenvalue has to be negative in order to assure convergence to the fixed point $\mathbf{V}^{(3)} = \mathbf{0}$. If this is the case the asymptotical approach of the order parameters R, Q, and w towards their fixed point is also exponentially in α with the same time constant $|\lambda_{max}^{-1}|$.

For sake of comparison it is useful to consider the case of fixed hidden-to-output couplings in the student network as well. In this case one has to restrict to $w \equiv v$ in order to keep the task learnable. It should be emphasized that in this case learning the rule should be a much simpler task for such a student, as the rule parameter v is assumed to be known *a priori*. The case of a

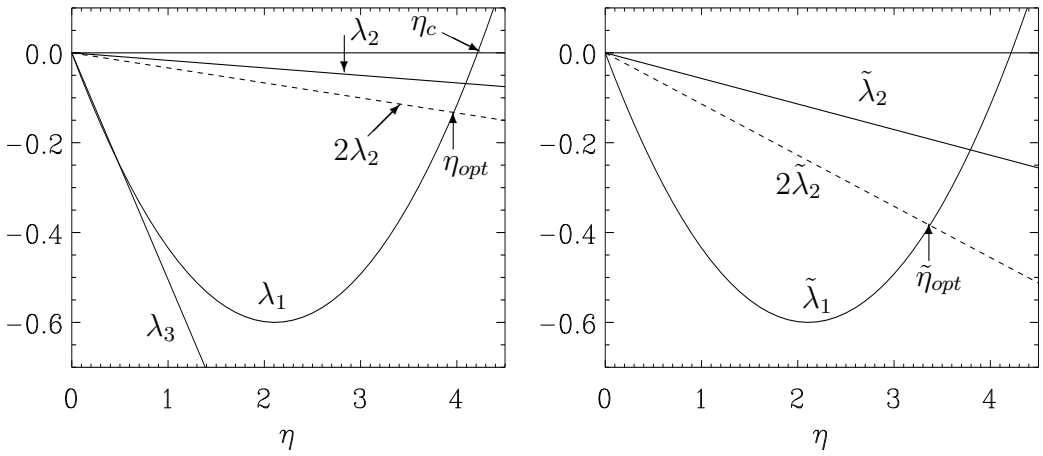


Figure 3.3: Left: Eigenvalues of the linearized system (3.27) governing the asymptotics of the generalization error of the Y-network. Right: Corresponding eigenvalue spectrum for an *a priori* known second layer. Teacher parameters $T = 2.0$, $v = 1.0$ for both figures. Note that the nonlinear eigenvalue is the same for both cases. Also the dominating linear eigenvalue is larger in the first case.

fixed hidden-to-output weight $w \equiv v$ has already been considered in [BS95] for the more special case of a normalized perceptron ($T = 1$).

Defining $\mathbf{V}^{(2)} = (R - T, Q - T)^\top$ correspondingly for the case of an *a priori* known $w \equiv v$, the asymptotics near the optimal solution $R_\infty = Q_\infty = T$ is governed by $d\mathbf{V}^{(2)}/d\alpha = \mathbf{m}^{(2)}\mathbf{V}^{(2)}$. Note that by construction the (2×2)-matrix $\mathbf{m}^{(2)}$ can be obtained from $\mathbf{m}^{(3)}$ by truncation of the last column and the last row. Both matrices can be diagonalized analytically for arbitrary parameters T and v. Denoting the eigenvalues of $\mathbf{m}^{(3)}$ by λ_i and those of $\mathbf{m}^{(2)}$ by $\tilde{\lambda}_i$ one gets

$$\begin{aligned}
\lambda_1 &= \tilde{\lambda}_1 = -\frac{4v^2\eta}{\pi\sqrt{1+2T}}\left(1 - \frac{\eta}{\pi}v^2\sqrt{\frac{1+2T}{1+4T}}\right), \\
\tilde{\lambda}_2 &= -\frac{2v^2\eta}{\pi(1+2T)^{3/2}}, \\
\lambda_{2,3} &= \frac{1}{2}\left(\tilde{\lambda}_2 + m_{33} \pm \sqrt{(\tilde{\lambda}_2 - m_{33})^2 + \frac{4m_{31}^2 T}{(1+T)^2}}\right),
\end{aligned} \tag{3.28}$$

where $m_{33} = -(2\eta/\pi)\arcsin(T/(1+T))$, $m_{31} = 2\eta v/(\pi\sqrt{1+2T})$. Note that $\lambda_1 = \tilde{\lambda}_1$ is quadratic in η whereas the remaining eigenvalues depend linearly on η. The eigenvalue spectra of $\mathbf{m}^{(2)}$ and $\mathbf{m}^{(3)}$ are shown in Figure 3.3.

The asymptotics of $\mathbf{V}^{(3)}$ is governed by $\max(\lambda_1, \lambda_2)$ which depends on η, see Fig. 3.3. For small η the asymptotic decay of $\mathbf{V}^{(3)}$ is proportional to $\exp(\lambda_2\alpha)$ whereas $\mathbf{V}^{(3)} \propto \exp(\lambda_1\alpha)$ for larger values of η. For η greater than the critical value

$$\eta_c = \frac{\pi}{v^2}\sqrt{\frac{1+4T}{1+2T}} \tag{3.29}$$

λ_1 becomes positive and the on-line backpropagation algorithm does not converge to the optimal solution. Therefore the range of learning rates that lead to perfect generalization is given by $\eta < \eta_c$. Note that as $\eta \stackrel{<}{\to} \eta_c$ the relaxation time $-1/\lambda_1$ diverges like $(\eta_c - \eta)^{-1}$. This *critical slowing down* is a well-known property of the dynamics near a critical point [Gar90]. Because of $\lambda_1 = \tilde{\lambda}_1$ the same holds true for $\mathbf{V}^{(2)}$, i.e. for learning with predetermined hidden-to-output couplings.

In addition, the identity $\lambda_1 = \tilde{\lambda}_1$ implies a rather remarkable result: *The critical learning rate is the same whether or not the hidden-to-output weights are adjustable.* A general proof of this property will be given in the next section and later this will be generalized to more complicated architectures with more than only one hidden unit.

For $\eta > \eta_c$, λ_1 becomes positive and $(R, Q, w) = (T, T, v)$ is not an attractive fixed point any longer. However, in this regime one can find a second fixed point of (3.23) with $R, Q \neq T$, $w \neq v$. Hence, perfect learning is not possible and $\epsilon_g(\alpha \to \infty) > 0$. This supercritical case will not be investigated in larger detail in this work, however. See [BS95] for a discussion of this regime for the case of a nonlinear perceptron.

While the convergence of the order parameters R, Q, and w is fastest for the learning rate satisfying $\lambda_1(\eta) = \lambda_2(\eta)$, the situation is different for the generalization error. Writing the generalization error (3.24) in terms of the asymptotic variables $\mathbf{V}^{(3)}$ one obtains as an expansion

$$\epsilon_g = \frac{2v^2}{\pi\sqrt{1+2T}}\Big(\frac{1}{2}V_2^{(3)} - V_1^{(3)}\Big) + \mathcal{O}\Big(V_i^{(3)}V_j^{(3)}\Big), \tag{3.30}$$

where the last term denotes terms that are higher order than linear in the components of $\mathbf{V}^{(3)}$. An analysis of the eigenvectors of $\mathbf{m}^{(3)}$ shows that the eigenvector related to λ_2 has the components $V_1^{(3)} = 1$ and $V_2^{(3)} = 2$. Therefore this mode cannot contribute to the asymptotic behaviour of the linear combination $(V_2^{(3)}/2 - V_1^{(3)})$. Consequently, for small η, the term linear in $V_i^{(3)}$ decays faster than $V_i^{(3)}V_j^{(3)}$, and the behaviour of ϵ_g depends on the actual order of the linear term compared to the nonlinear terms. If, for given η, $V_i^{(3)}V_j^{(3)}$ is larger than $(V_2^{(3)}/2 - V_1^{(3)})$, the generalization error decays proportional to $\exp(2\lambda_2\alpha)$, while $\epsilon_g \propto \exp(\lambda_1\alpha)$ if the linear term is larger than the next higher order term.

This change of the asymptotic behaviour of the generalization error happens at η_{opt} where λ_1 and $2\lambda_2$ coincide (see Fig. 3.3). Therefore, the fastest asymptotic decay of ϵ_g is achieved for η_{opt}. It should be emphasized, however, that this discussion only holds for the *asymptotic* behaviour. The initial decay of the generalization error can be faster for values of η different from η_{opt} as can be seen in Figure 3.2. This shows that the learning rate effectively has to depend on the dynamics as well, in order to achieve an optimal decay of the generalization error, i.e. $\eta = \eta(R, Q, w)$. This statement is also supported by the results for the truly optimal weight functions derived from a variational principle in Chapter 2.

For *a priori* known second layer coupling $w \equiv v$ the dynamics of generalization error and order parameters is qualitatively similar, one simply has to replace λ_2 by the corresponding eigenvalue $\tilde{\lambda}_2$. Since $\tilde{\lambda}_2 > \lambda_2$, however, the asymptotics is much faster in this case, at least for small learning rates η. This is not too surprising, since the *a priori* knowledge should be reflected by an acceleration of the learning process.

3.2.2 General Proof of the Asymptotic Properties

The last section revealed the rather remarkable result that the critical learning rate η_c is independent of the state of the second layer, i.e. whether or not the second layer coupling w already has adopted the value v of the corresponding teacher coupling. This has been shown to be true for the special choice of transfer function considered in the previous section. In this section it will be shown that this is a general result for any Y-network learning a realizable task.

The general input-output relation for the Y-network is of the form

$$\sigma(\boldsymbol{\xi}) = f(wg(\boldsymbol{J}\cdot\boldsymbol{\xi})) \tag{3.31}$$

with transfer functions $g(\cdot)$ and $f(\cdot)$ at the hidden layer and the output layer, respectively. Correspondingly, the teacher network realizes the functional relationship $\tau = f(vg(\boldsymbol{B}\cdot\boldsymbol{\xi}))$. The special case investigated in the previous section corresponds to a linear transfer function $f(\cdot) = \cdot$ at the output.

Applying the backpropagation learning rule (3.3) one obtains in the thermodynamic limit $N \to \infty$ deterministic differential equations for the order parameters R, Q, and w:

$$\frac{dR}{d\alpha} = -\eta\Big\langle \Gamma(\tilde{x},\tilde{y})\sqrt{T}\tilde{y}\Big\rangle_{P(\tilde{x},\tilde{y};R/\sqrt{QT})} \equiv f_1(R,Q,w) \tag{3.32}$$

$$\begin{aligned}\frac{dQ}{d\alpha} &= -2\eta\Big\langle \Gamma(\tilde{x},\tilde{y})\sqrt{Q}\tilde{x}\Big\rangle_{P(\tilde{x},\tilde{y};R/\sqrt{QT})} + \eta^2\Big\langle \Gamma^2(\tilde{x},\tilde{y})\Big\rangle_{P(\tilde{x},\tilde{y};R/\sqrt{QT})} \\ &\equiv f_2(R,Q,w).\end{aligned} \tag{3.33}$$

Here, the equation of motion for the order parameter w has been omitted since it will turn out to be irrelevant for the following considerations.[2]

In (3.32) the abbreviation

$$\Gamma(\tilde{x},\tilde{y}) = (\sigma(\tilde{x}) - \tau(\tilde{y}))f'\Big(wg(\sqrt{Q}\tilde{x})\Big)g'\Big(\sqrt{Q}\tilde{x}\Big)w \tag{3.34}$$

has been used. For convenience the internal fields have been normalized: $\tilde{x} = \boldsymbol{J}\cdot\boldsymbol{\xi}/\sqrt{Q}$ and $\tilde{y} = \boldsymbol{B}\cdot\boldsymbol{\xi}/\sqrt{T}$. Note that for a Gaussian distribution $P(\tilde{x},\tilde{y};R/\sqrt{QT})$ is of the same form (1.16) as for the perceptron considered in Section 1.3.

In order to proof the claimed result it is necessary to investigate the eigenvalue spectrum of the matrix $\mathbf{m}^{(3)}$ (cf. (3.27)) that determines the asymptotic solution of the order parameters R, Q, and w. In the following it will be shown that the matrix elements of $\mathbf{m}^{(3)}$ fulfill the following properties:

$$\begin{aligned}&\text{(A1)} \quad 2m_{13} = m_{23} \\ &\text{(A2)} \quad \kappa \equiv m_{22} - 2m_{12} = m_{11} - \frac{1}{2}m_{21}.\end{aligned}$$

Having shown this, application of a matrix theorem will lead to the desired result.

[2] This is of course not too surprising. If certain properties like η_c are to be independent of the state of the second layer, then the updating rule for the second layer coupling should not enter the proof.

By definition

$$m_{13} \equiv \text{f–lim}\,\frac{\partial}{\partial w} f_1(R,Q,w) \tag{3.35}$$

$$= -\eta\sqrt{T}\,\text{f–lim}\left\langle \tilde{y}\frac{\partial}{\partial w}\Gamma(\tilde{x},\tilde{y};R,Q,w)\right\rangle_{P(\tilde{x},\tilde{y};R/\sqrt{QT})}. \tag{3.36}$$

The expression on the right hand side is easily evaluated by first differentiating Γ with respect to w and then carrying out the f–lim, exploiting the fact that

$$\text{f–lim}\,P(\tilde{x},\tilde{y};R/\sqrt{QT}) = P(\tilde{x})\delta(\tilde{y}-\tilde{x}). \tag{3.37}$$

This holds true, since at the fixed point $R=Q=T$ the internal fields $\tilde{x},\tilde{y}$ of teacher and student network are completely correlated. One therefore obtains:

$$m_{13} = -\eta\sqrt{T}v\int d\tilde{x}P(\tilde{x})\tilde{x}g\Big(\sqrt{T}\tilde{x}\Big)g'\Big(\sqrt{T}\tilde{x}\Big)\Big[f'\Big(vg\Big(\sqrt{T}\tilde{x}\Big)\Big)\Big]^2. \tag{3.38}$$

Proceeding in the same way with m_{23} yields

$$m_{23} = \text{f–lim}\,\frac{\partial}{\partial w} f_2(R,Q,w)$$

$$= -2\eta\sqrt{T}v\int d\tilde{x}P(\tilde{x})\tilde{x}g\Big(\sqrt{T}\tilde{x}\Big)g'\Big(\sqrt{T}\tilde{x}\Big)\Big[f'\Big(vg\Big(\sqrt{T}\tilde{x}\Big)\Big)\Big]^2 \tag{3.39}$$

and therefore $2m_{13}=m_{23}$ as has been claimed in (A1).

In order to show assumption (A2) it is convenient to make a transformation to a new set of variables: $(R,Q)\to(\rho=R/\sqrt{QT},Q)$, where ρ is the normalized overlap of the teacher and student weight vectors of the first layer. Hence, one obtains for the right hand side of (A2)

$$m_{11}-\frac{1}{2}m_{21} \equiv \text{f–lim}\,\frac{\partial}{\partial R}\Big(f_1(R,Q,w)-\frac{1}{2}f_2(R,Q,w)\Big) \tag{3.40}$$

$$= \text{f–lim}\,\frac{1}{\sqrt{QT}}\frac{\partial}{\partial\rho}\Big(\tilde{f}_1(\rho,Q,w)-\frac{1}{2}\tilde{f}_2(\rho,Q,w)\Big), \tag{3.41}$$

where f–lim in terms of the new set of order parameters reads

$$\text{f–lim}(...) = \lim_{\rho\to1,Q\to T,w\to v}(...) \tag{3.42}$$

and $\tilde{f}_i$ is f_i expressed in terms of the new variable ρ.

Analogously one obtains for the left hand side of (A2)

$$m_{22}-2m_{12} = \text{f–lim}\,\frac{\rho}{Q}\frac{\partial}{\partial\rho}\Big(\tilde{f}_1(\rho,Q,w)-\frac{1}{2}\tilde{f}_2(\rho,Q,w)\Big)$$

$$+\,\text{f–lim}\,\frac{\partial}{\partial Q}\Big(\tilde{f}_2(\rho,Q,w)-2\tilde{f}_1(\rho,Q,w)\Big). \tag{3.43}$$

A straightforward but tedious calculation shows that the second term in the last equation vanishes. Consequently, assumption (A2) is true, if f–lim$[\partial(\tilde{f}_1 - \tilde{f}_2/2)/\partial\rho]$ exists. Since this expression essentially is defined in terms of the distribution $P(\tilde{x}, \tilde{y}; \rho)$ of the internal fields, the existence of the limit depends on the functional form of $P(\tilde{x}, \tilde{y}; \rho)$.

Note that so far no assumptions have been made about distribution $P(\tilde{x}, \tilde{y}; \rho)$. Specializing now to a Gaussian distribution $P(\tilde{x}, \tilde{y}; \rho)$ of the form (1.16) the existence of f–lim$[\partial(\tilde{f}_1 - \tilde{f}_2/2)/\partial\rho]$ can be shown. In order to do that one formally carries out the derivative with respect to ρ in (3.41) and applies the relation[3]

$$\lim_{\rho\to 1} \frac{\partial}{\partial\rho} \int d\tilde{y} P(\tilde{x}, \tilde{y}; \rho) F(\tilde{y}) = P(\tilde{x})(\tilde{x}F'(\tilde{x}) - F''(\tilde{x})), \tag{3.44}$$

which finally leads to the result

$$\begin{aligned} \kappa &= -2\eta v^2 \int D\tilde{x} \Big[f'\Big(v g\Big(\sqrt{T}\tilde{x}\Big)\Big) g'\Big(\sqrt{T}\tilde{x}\Big)\Big]^2 \\ &\quad + \eta^2 v^4 \int D\tilde{x} \Big[f'\Big(v g\Big(\sqrt{T}\tilde{x}\Big)\Big) g'\Big(\sqrt{T}\tilde{x}\Big)\Big]^4. \end{aligned} \tag{3.45}$$

Hence, assumptions (A1) and (A2) have been proven for any input distribution that leads to a Gaussian of the internal fields $\tilde{x}$ and $\tilde{y}$.

Theorem 2 in Appendix B states that because of the validity of the conditions (A1) and (A2) κ is one of the eigenvalues of the matrix $\mathbf{m}^{(3)}$. Moreover, the matrix $\mathbf{m}^{(2)}$ describing the asymptotics of a Y-network with fixed and *a priori* known second layer coupling $w \equiv v$ has κ as an eigenvalue as well. Note that κ according to (3.45) is nonlinear in the learning rate η. This nonlinearity gives rise to a critical learning rate η_c such that $\kappa > 0$ for $\eta > \eta_c$. Hence, the backpropagation algorithm will not result in convergence to zero generalization error for a supercritical learning rate.

According to this matrix theorem the other eigenvalues of $\mathbf{m}^{(3)}$ ($\lambda_{2,3}$) and $\mathbf{m}^{(2)}$ ($\tilde{\lambda}_2$) are given by

$$\begin{aligned} \tilde{\lambda}_2 &= m_{11} + 2m_{12} \\ \lambda_{2,3} &= \frac{1}{2}\left(\tilde{\lambda}_2 + m_{33} \pm \sqrt{\Big(\tilde{\lambda}_2 - m_{33}\Big)^2 + 4m_{13}(m_{31} + 2m_{32})} \right). \end{aligned} \tag{3.46}$$

These eigenvalues do not depend on matrix elements of the second row of $\mathbf{m}^{(3)}$ which is obtained from the equation of motion for the order parameter Q and is the only one that is nonlinear in η. Therefore, κ is the only eigenvalue nonlinear in the learning rate.

These facts lead to the following important conclusion: The existence of a critical learning rate is exclusively determined by the first layer of a Y-network. It does not depend on whether the second layer coupling is adjustable or known beforehand. In the consecutive sections it will be shown that this result holds true for more complicated two-layer networks.

[3]The proof of (3.44) is rather straightforward by expanding $F(\tilde{y})$ in a Taylor series around $\rho\tilde{x}$.

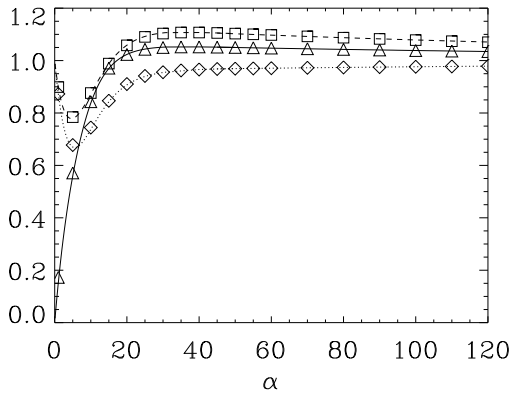


Figure 3.4: Time evolution of the order parameters ($\triangle$ R, $\square$ Q, $\diamond$ w) for a Y-network with nonlinear transfer function $g(z) = \mathrm{erf}(z/\sqrt{2})$ at both hidden and output node (3.31) learning a rule represented by a teacher network of the same architecture. The analytic results (curves) are compared to simulations (symbols) for a network with $N = 100$ inputs. The average is over 100 independent runs, error bars would be smaller than the size of the symbols. Initial conditions: $R(0) = 0$, $Q(0) = w(0) = 1$; parameters: $T = v = \eta = 1$.

It is worth stressing that the validity of the results derived in this section is not restricted to on-line backpropagation. It is not too difficult a task to generalize the findings to general on-line algorithms. For any algorithm

$$\begin{aligned} \boldsymbol{J}^{\mu+1} &= \boldsymbol{J}^{\mu} - \frac{\eta}{N}\Gamma(\boldsymbol{J}^{\mu}, w^{\mu}, \boldsymbol{\xi}\mu)\boldsymbol{\xi}^{\mu} \\ w^{\mu+1} &= w^{\mu} - \frac{\eta}{N}\Delta(\boldsymbol{J}^{\mu}, w^{\mu}, \boldsymbol{\xi}^{\mu})\, g(\boldsymbol{J}^{\mu}\cdot\boldsymbol{\xi}^{\mu}) \end{aligned} \tag{3.47}$$

the first layer properties will determine the critical learning rate. The weight functions Γ and Δ have to be differentiable and must satisfy the fixed point condition $\Gamma(\boldsymbol{B}, v, \boldsymbol{\xi}) = 0 = \Delta(\boldsymbol{B}, v, \boldsymbol{\xi})$.

Note that Eqs. (3.37,3.44) provide a convenient means for an analytical investigation of the learning curves. They allow to obtain the asymptotics of the order parameters by first expanding around the fixed point of the equations of motion and then averaging over the randomness of the inputs $\boldsymbol{\xi}$. This is often easier than carrying out the average over the weight functions Γ and Δ first and then making the expansion.

Finally, Figure 3.4 shows an example of a learning process in a Y-network with nonlinear transfer functions at both hidden and output node (3.21).

3.3 Nonoverlapping Architecture

In the nonoverlapping architecture each branch of the network receives a different part of the $(N \cdot K)$-dimensional input vector $\boldsymbol{\xi} = (\boldsymbol{\xi}_1, .., \boldsymbol{\xi}_K)$. Since the components of the vector $\boldsymbol{\xi}$ are assumed to be mutually uncorrelated the receptive fields in different branches of the student or teacher network are uncorrelated as well. Therefore the "off-diagonal" order parameters R_{in} and

Q_{ij} ($n \neq i \neq j$) do not carry any significant information and hence need not to be considered. The dynamics of learning is completely described by the equations of motion for the remaining dynamical variables R_{ii}, Q_{ii}, and w_i.

For sake of simplicity only the special symmetric case will be considered in the following where the weights in the teacher network satisfy

$$v_n = v \quad \text{and} \quad T_{mn} = T\delta_{mn} \quad \text{for all} \quad m, n = 1, 2, \ldots, M \ll N. \tag{3.48}$$

For large N, this corresponds to uncorrelated vectors B_n of length T with independent random components of zero mean and variance $\mathcal{O}(1/N)$. Note, however, that the following is easily extended to more general asymmetric settings though the explicit calculations may become quite lengthy.

Due to this symmetry, the time evolution rapidly leads to an equality of the order parameters corresponding to different branches of the student network (see Figure 3.5 for an example). Therefore, asymptotically the learning dynamics can be described in terms of only three variables $R = R_{ii}$, $Q = Q_{ii}$, $w = w_i$, regardless of the actual number of hidden units.

The resulting equations of motion then differ from those for the Y-network (3.23) only by the K-dependence in the equation of motion for Q:

$$\begin{aligned}
\frac{dQ}{d\alpha} &= \frac{4}{\pi}\eta\frac{w}{1+Q}\left(v\frac{R}{\sqrt{(1+Q)(1+T)-R^2}} - w\frac{Q}{\sqrt{1+2Q}}\right) \\
&\quad + \frac{4}{\pi^2}\eta^2\frac{w^2}{\sqrt{1+2Q}}\left[v^2\arcsin\left(\frac{(1+2Q)T-2R^2}{(1+2Q)(1+T)-2R^2}\right) + w^2\arcsin\left(\frac{Q}{1+3Q}\right)\right. \\
&\qquad \left. - 2vw\arcsin\left(\frac{R}{\sqrt{1+3Q}\sqrt{(1+2Q)(1+T)-2R^2}}\right)\right] \\
&\quad + \frac{4}{\pi^2}\eta^2\frac{w^2(K-1)}{\sqrt{1+2Q}}\left[v^2\arcsin\left(\frac{T}{1+T}\right) + w^2\arcsin\left(\frac{Q}{1+Q}\right)\right. \\
&\qquad \left. - 2vw\arcsin\left(\frac{R}{\sqrt{(1+T)(1+Q)}}\right)\right].
\end{aligned} \tag{3.49}$$

This equation reduces for $K = 1$ to the corresponding equation for the Y-network. Due to the similar form of the equations of motion, it is expected that the dynamics of the nonoverlapping machine essentially has the same properties as for the Y-network.

In order to solve the asymptotics of the dynamical variables R, Q, and w, the equations of motion (3.23,3.49) have to be linearized for small deviations from the fixed point $R_\infty = Q_\infty = T$, $w_\infty = v$. Defining $\mathbf{V}^{(3)} = (R - T, Q - T, w - v)^\top$ as for the Y-network, the linearization is again of the form

$$\frac{d\mathbf{V}^{(3)}}{d\alpha} = \mathbf{m}^{(3)}\mathbf{V}^{(3)}. \tag{3.50}$$

Also, for sake of comparison the case of fixed hidden-to-output weights will be considered, with $w \equiv v$ in order to keep the task learnable. As before, the asymptotics of the dynamics for this

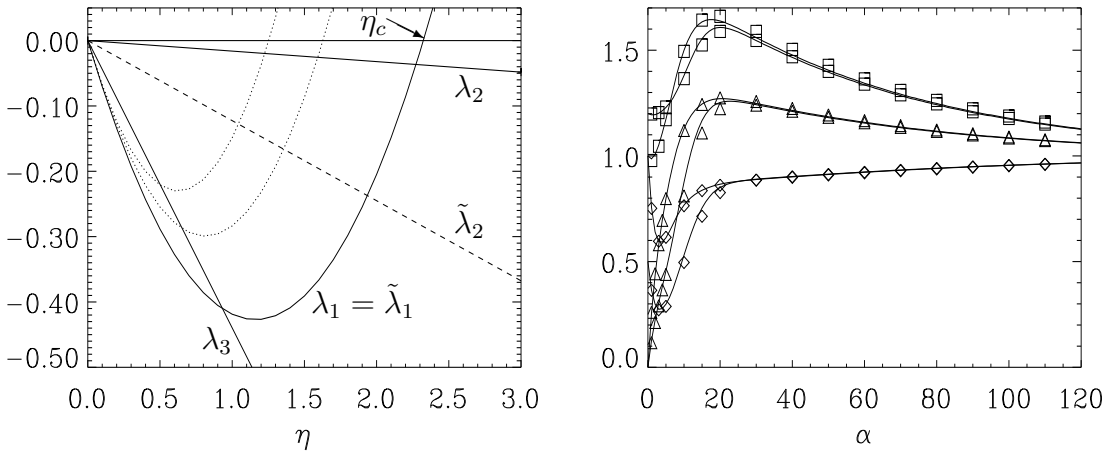


Figure 3.5: Learning in a tree-like architecture with two hidden units ($K = 2$) for parameters $T = v = 1$. Left: Graph of η-dependence of the eigenvalues of the linearized system (λ_i of $\mathbf{m}^{(3)}$, $\tilde{\lambda}_i$ of $\mathbf{m}^{(2)}$) governing the asymptotics of learning. The dotted lines show λ_1 for $K = 3$ and $K = 4$, respectively, demonstrating the decrease of η_c with increasing K. Right: Evolution of the dynamical variables for $\eta = 1$ ($\triangle$ R, $\square$ Q, $\diamond$ w). The symbols represent results obtained by a simulation of a system with $2N = 200$ input units averaged over 100 experiments. Error bars would be smaller than the symbols. Note that the dynamical variables in different branches become equal rapidly in spite of the asymmetric initial conditions $R_{11}(0)$= $R_{22}(0)$=0, $Q_{11}(0)$=1.2, $Q_{22}(0)$=$w_1(0)$=1.0, and $w_2(0)$=0.5.

case of *a priori* known hidden-to-output weights is contained in the matrix $\mathbf{m}^{(2)}$ which is easily obtained from $\mathbf{m}^{(3)}$ by truncating the third column and the third row.

It is straightforward to show that $\mathbf{m}^{(3)}$ again satisfies conditions (A1) and (A2) of the previous section. Thus, the eigenvalue spectra of $\mathbf{m}^{(2)}$ and $\mathbf{m}^{(3)}$ are easily calculated by application of Theorem 2 in Appendix B. One obtains for arbitrary parameters T and v

$$\begin{aligned}
\lambda_1 &= \tilde{\lambda}_1 = \frac{4v^4\eta^2}{\pi^2}\left(\frac{K-1}{1+2T} + \frac{1}{\sqrt{1+4T}}\right) - \frac{4v^2\eta}{\pi\sqrt{1+2T}}, \\
\tilde{\lambda}_2 &= -\frac{2v^2\eta}{\pi(1+2T)^{3/2}}, \\
\lambda_{2,3} &= \frac{1}{2}\left(\tilde{\lambda}_2 + m_{33} \pm \sqrt{(\tilde{\lambda}_2 - m_{33})^2 + \frac{4m_{31}^2 T}{(1+T)^2}}\right),
\end{aligned} \tag{3.51}$$

where $m_{33} = -(2\eta/\pi)\arcsin(T/(1+T))$, $m_{31} = 2\eta v/(\pi\sqrt{1+2T})$. The eigenvalues of $\mathbf{m}^{(3)}$ ($\mathbf{m}^{(2)}$) are denoted by λ_i ($\tilde{\lambda}_i$), as for the Y-network. Since the tree-like architecture reduces to a Y-network for $K = 1$ these eigenvalues are identical to (3.28) in this special case.

The asymptotics of $\mathbf{V}^{(3)}$ is governed by $\max(\lambda_1, \lambda_2)$ which depends on η, see Fig. 3.5. For small η the asymptotic decay of $\mathbf{V}^{(3)}$ is proportional to $\exp(\lambda_2\alpha)$ whereas $\mathbf{V}^{(3)} \propto \exp(\lambda_1\alpha)$ for

larger values of η. For η greater than the critical value

$$\eta_c = \frac{\pi/v^2}{(K-1)/\sqrt{1+2T} + \sqrt{1+2T}/\sqrt{1+4T}} \tag{3.52}$$

λ_1 becomes positive and the on-line backpropagation algorithm does not converge to the optimal solution. Therefore the range of learning rates that lead to perfect generalization is given by $\eta < \eta_c$. This range decreases with an increasing number of hidden units like $\eta_c \propto 1/K$ for large K. Because of $\lambda_1 = \tilde{\lambda}_1$ the same holds true for $\mathbf{V}^{(2)}$, i.e. for learning with predetermined hidden-to-output couplings investigated in [SS95a].

In addition, the identity $\lambda_1 = \tilde{\lambda}_1$ implies a result already known from the Y-network: The critical learning rate is the same whether or not the hidden-to-output weights are adjustable. Consequently one has to conclude that – at least for a finite number of hidden units K – the existence of a critical learning rate is a first layer effect.

In the regime of η small enough the asymptotics is governed by λ_2 and $\tilde{\lambda}_2$, respectively. Eq. (3.51) implies $\lambda_2 > \tilde{\lambda}_2$ for all values of T and v. An updating of the couplings in the second layer therefore leads to a slowing down of the asymptotic convergence. This is precisely what one would expect, since the adaptation of additional (hidden-to-output) couplings should decelerate the learning process. Furthermore, the eigenvalues λ_2 and $\tilde{\lambda}_2$ are independent of K. This means that the asymptotics of the dynamical variables is independent of the number of hidden units in the small η-regime.

In order to obtain the asymptotic decay of ϵ_g, the generalization error has to be expanded up to second order in $\mathbf{V}^{(3)}$ ($\mathbf{V}^{(2)}$ respectively). As for the Y-network discussed in the previous section one finds that the generalization error decays like $\exp(2\lambda_2\alpha)$ for small η ($\exp(2\tilde{\lambda}_2\alpha)$ if the couplings in the second layer are fixed), while $\epsilon_g \propto \exp(\lambda_1\alpha)$ for large values of η. The change of the asymptotic decay occurs at η_{opt} defined by $\lambda_1 = 2\lambda_2$ ($\tilde{\eta}_{opt}$ by $\lambda_1 = 2\tilde{\lambda}_2$). With this learning rate the fastest asymptotic decrease of the generalization error can be achieved. Note that η_{opt} is much closer to η_c than $\tilde{\eta}_{opt}$, as can be seen in Figure 3.5 (e.g. for $v = T = 1$ one obtains $\tilde{\eta}_{opt} = (2/3)\eta_c$ [BS95, SS95b], whereas $\eta_{opt} \approx 0.96\eta_c$). Therefore it will be very difficult to tune the learning rate optimally in practical applications of on-line backpropagation.

Figure 3.5 also depicts the numerical solution of the full system of the equations of motion (3.23,3.49) together with simulations.

3.4 Overlapping Architecture

In this setting each hidden node receives information from all input units:

$$\sigma(\boldsymbol{\xi}) = \sum_{i=1}^{K} w_i g(\boldsymbol{J}_i \cdot \boldsymbol{\xi}). \tag{3.53}$$

Therefore a permutation symmetry is inherent to the problem, namely the i-th branch in the student network does not necessarily specialize on the i-th branch in the teacher network. Without loss of generality, however, one can relabel the dynamical variables such as if this were indeed

the case. Nevertheless, this permutation symmetry will turn out to be a dominant feature because it leads to a deceleration of learning.

As in the previous sections the main concern here is the analysis of the asymptotic properties of the backpropagation dynamics (3.3). For a perfect match of student and teacher architectures ($K = M$) the number of dynamical variables in the macroscopic equations of motion (3.13) is quadratic in the number of hidden units K. As for the tree-like architecture, the analysis can be further simplified by restricting to a symmetric architecture of the teacher network as in (3.48). Then the time evolution will rapidly lead to the equality of corresponding variables in different branches of the student network. Hence, for $\alpha \to \infty$ the student network can be described in terms of only five variables $R = R_{ii}$, $S = R_{im}$, $Q = Q_{ii}$, $C = Q_{ij}$, and $w = w_i$, regardless of the actual number of hidden units. This notation already implies that the permutation symmetry mentioned above has been taken into account by the convention that the branch in the student network which will specialize to the i-th teacher branch is labeled by the index i as well. Yet, even for a symmetric teacher architecture the equations of motion according to (3.18) are very lengthy, in particular those for Q and C which are nonlinear in η. For this reason the equations of motion will not be stated here explicitly.

Having restricted to a symmetric architecture the analysis proceeds similarly to the tree-like architecture, except for the fact that the asymptotics has to be described by the five dynamical variables R, S, Q, C, and w. A linearization for small deviations from the fixed point $R_\infty = Q_\infty = T, S_\infty = C_\infty = 0, w_\infty = v$ leads to

$$\frac{d\mathbf{V}^{(5)}}{d\alpha} = \mathbf{m}^{(5)}\mathbf{V}^{(5)}, \tag{3.54}$$

where $\mathbf{V}^{(5)} = (R - T, S, Q - T, C, w - v)^\top$.

As for the tree-like architecture, fixed hidden-to-output couplings $w \equiv v$ will also be investigated. This special case has been investigated recently in [SS95a, SS95b].[4] The linearization of the corresponding equations of motion reads

$$\frac{d\mathbf{V}^{(4)}}{d\alpha} = \mathbf{m}^{(4)}\mathbf{V}^{(4)}, \tag{3.55}$$

where $\mathbf{V}^{(4)} = (R - T, S, Q - T, C)^\top$. As before, $\mathbf{m}^{(4)}$ can be obtained from $\mathbf{m}^{(5)}$ by a truncation of the last column and the last row.

The eigenvalues λ_i of $\mathbf{m}^{(5)}$ and $\tilde{\lambda}_i$ of $\mathbf{m}^{(4)}$ can be computed analytically for arbitrary values of T and v. This is because the matrices $\mathbf{m}^{(4)}$ and $\mathbf{m}^{(5)}$ are conjugated, similarly to the corresponding matrices for the Y-network and the tree-like networks. See Appendix B for the matrix theorem. The expressions for the eigenvalues λ_i are rather lengthy, however, even for a particular choice of T and v. Therefore only the important features of the asymptotics will be discussed here. For the exact expressions for the eigenvalues the reader is referred to Appendix C.

[4]Strictly speaking, the results of [SS95a, SS95b] for large α are valid only for the tree-like architecture, not for the fully connected architecture. This is because it has been assumed there for simplicity that S and C can be neglected asymptotically compared to $R - T$ and $Q - T$.

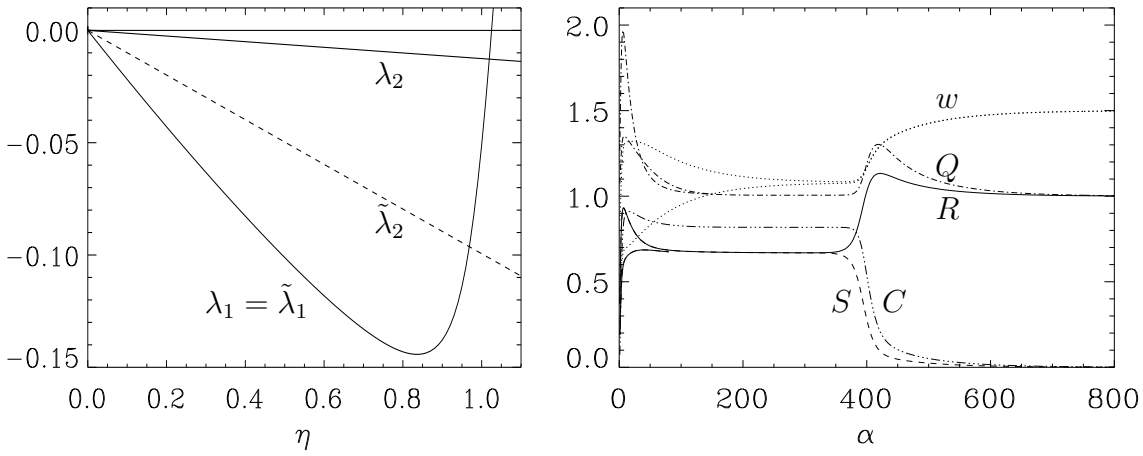


Figure 3.6: Learning in a two-layer network with a fully connected architecture with two hidden units ($K = 2$) for parameters $T = 1$, $v = 1.5$. Left: Eigenvalues that govern the asymptotics of the dynamical variables. Notation as in Figures 3.3 and 3.5. Right: Numerical solution for the dynamical variables for $\eta = 1$ and asymmetric initial conditions $R(0) = S(0) = C(0) = 0, Q(0) = w_1(0) = 1, w_2(0) = 0.5$.

The asymptotics is governed by the largest eigenvalues which are denoted by λ_1 and λ_2 for $\mathbf{m}^{(5)}$ and by $\tilde{\lambda}_1, \tilde{\lambda}_2$ for $\mathbf{m}^{(4)}$. The η-dependence of the dominating eigenvalue is qualitatively the same as for the tree-like architecture: The eigenvalues $\lambda_2, \tilde{\lambda}_2$ are linear in η, whereas $\lambda_1, \tilde{\lambda}_1$ depend nonmonotonically on η, cf. Fig. 3.6. Note that $\tilde{\lambda}_1$ is not polynomial in η in contrast to the claim of [SS95b]. Figure 3.6 shows the eigenvalues for a particular set of parameters.

The nonmonotonic $\lambda_1, \tilde{\lambda}_1$ give rise to a critical learning rate η_c, such that for $\eta \geq \eta_c$ no perfect generalization can be achieved. See Appendix C for the functional form of η_c. Moreover, $\lambda_1 = \tilde{\lambda}_1$ holds true as before, implying that the value of η_c is independent of an adaptation of couplings in the hidden-to-output layer. Even if one had an *a priori* knowledge of v_m the maximal learning rate η_c would be the same.

Again, one finds $\lambda_2 > \tilde{\lambda}_2$ for all values of T, v. The additional updating of the second layer makes the asymptotic convergence much slower. As can be seen in Figure 3.6, λ_2 is very close to zero. In contrast to the tree-like architecture both λ_2 and $\tilde{\lambda}_2$ are K-dependent, see Fig. 3.7.

The asymptotics of the generalization error is analogous to the nonoverlapping case: For $\eta < \eta_{opt}$ the generalization error decreases like $\epsilon_g \propto \exp(2\lambda_2\alpha)$ while $\epsilon_g \propto \exp(\lambda_1\alpha)$ for $\eta_{opt} < \eta < \eta_c$; η_{opt} is defined by $\lambda_1(\eta_{opt}) = 2\lambda_2(\eta_{opt})$. However, here the optimal learning rate η_{opt} is very close to η_c due to the nonmonotonicity of $\lambda_1 = \tilde{\lambda}_1$ and the K-dependence of λ_2 and $\tilde{\lambda}_2$. Therefore, the dynamics of the fully connected machine is essentially dominated by the eigenvalues λ_2 and $\tilde{\lambda}_2$, respectively.

The dependence of the critical learning rate η_c and the eigenvalues λ_2 and $\tilde{\lambda}_2$ on the system parameter K, T, and v is shown in Figure 3.7. Most interesting is the dependence on the number of hidden units K: The critical learning rate decreases like $\eta_c \sim 1/K$ for large values of K. Since $\lambda_2 \propto \eta$ and $\eta < \eta_c$ in order to reach perfect generalization, this implies that effectively

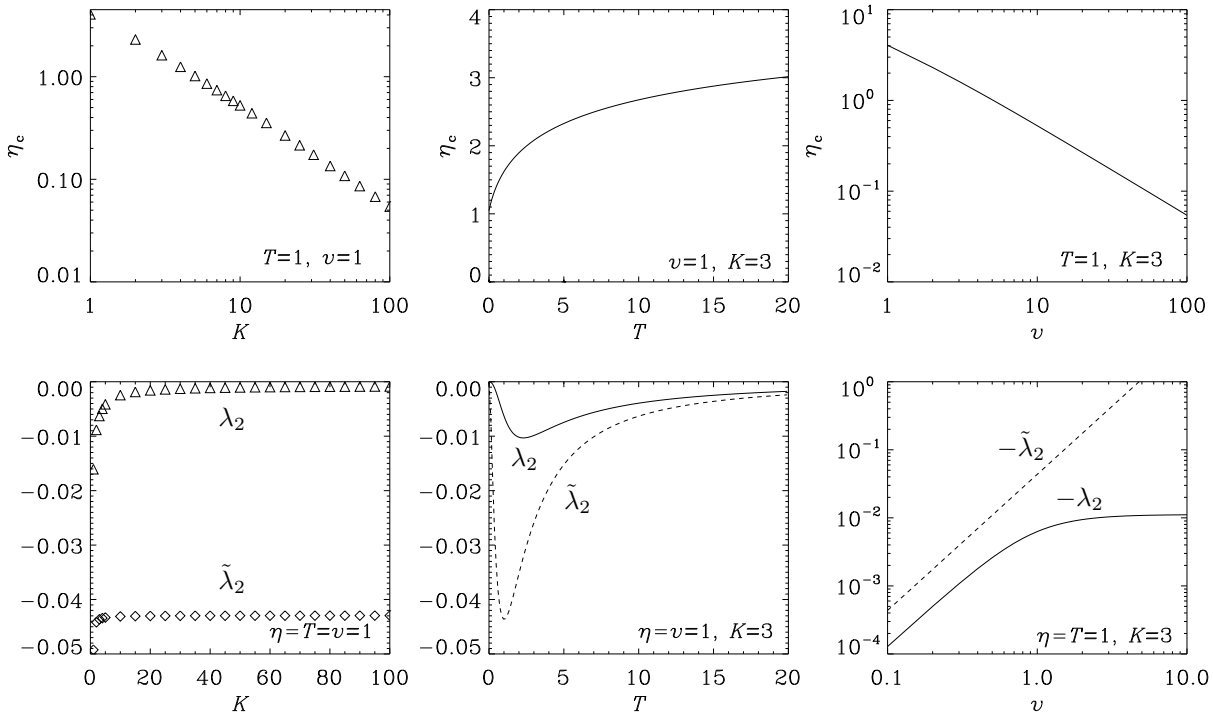


Figure 3.7: Dependence of the critical learning rate η_c and the eigenvalues λ_2 and $\tilde{\lambda}_2$ on the system parameters K, v, and T. See text for a detailed discussion.

$\lambda_2 \sim 1/K$. Therefore, the convergence of the generalization error becomes slower with an increasing number of hidden units K. This gives a hint on the speed of convergence of the backpropagation algorithm in the interesting limit $K = \mathcal{O}(N)$, cf. Cybenko's Theorem, Section 3.1.2.

Furthermore, Figure 3.7 shows the dependence on the norm of the teacher vectors T. An interesting limit is $T \to \infty$ since there the transfer functions become sign-functions, i.e. each input branch approaches a thresholded perceptron. Of course, the backpropagation algorithm is not properly defined any more for such a transfer function as it is not differentiable. Nevertheless it is important to understand the scaling with large values of T. As $T \to \infty$ the critical learning rate η_c remains finite. This would suggest that the order parameters still are approaching their fixed point values exponentially fast in this limit. Since $Q \to T$, however, the increasing norm of the student weight vectors causes an effective diminution of the step size in the update of $\boldsymbol{J}_i$ (3.8). Stated differently, the change of the normalized student weight vectors scales with η/Q and hence is decreasing. The appropriate renormalized order parameters $R/\sqrt{QT}$ and Q/T have a critical learning rate η_c/T which vanishes in the limit $T \to \infty$. Based on these arguments one expects an algebraic decay of the order parameters in this limit.

The v-dependence of the critical learning rate, $\eta_c \propto 1/v^2$, is due to the fact that η_c depends on the properties of the first layer only. Therefore its value is completely determined by the microscopic equations of motion for the couplings $\boldsymbol{J}_i$, Eq. (3.8). From this equation it can be

seen that v^2 can be scaled out by defining $\eta_{scal} = \eta v^2$ and $w_{scal} = w/v$, where the latter quantity is 1 for *a priori* known hidden-to-output weights. Consequently, the (unscaled) quantities η_c and $\tilde{\lambda}_2$ scale with $1/v^2$ and v^2, respectively. This is not the case for λ_2, however, since the equations of motion for the second layer couplings w_i are not invariant under the above scaling transformation.

Figure 3.6 shows the numerical solution of the equations of motion (3.13) for the fully connected architecture. Note the plateaus at intermediate values of α, which result from the existence of symmetric fixed points of (3.13). A complete discussion of plateau states can be found in [BRW96, Wöh96].

The very reason for the existence of plateaus in the learning curves is the inherent permutation symmetry of the fully connected machine mentioned above. At the onset of learning no branch in the student vector has specialized to a branch in the teacher network, yet. Each student vector $\boldsymbol{J}_i$ is adapting to some sort of weighted mean of the teacher vectors $\boldsymbol{B}_n$ which represents the plateau configuration. The time α required to leave such a plateau configuration strongly depends on the symmetry of the configuration and the initial values of the student network, see [BRW96, Wöh96].

3.5 Rescaling the Dynamics of the Second Layer

The analysis of the previous two sections has revealed that the quantities which are characteristic for the asymptotics of the learning dynamics exhibit certain scaling properties with respect to the system parameters. For instance, the critical learning rate η_c has been shown to scale with the number of hidden units K like $1/K$ for both overlapping and nonoverlapping architecture. This suggests to incorporate such scaling properties into the theoretical description. In this section this will be done for the learning rate. The focus will be on the learning rate governing the step size in the second layer. The analysis will lead to interesting and nontrivial results.

3.5.1 Optimal Learning Rate in the Second Layer

The analysis of the previous sections has shown that some of the asymptotic properties of on-line backpropagation are completely independent of the dynamics of the hidden-to-output weights w. In particular the critical learning rate η_c will be the same irrespective of whatever dynamics has been chosen for the second layer. This is because the coefficients m_{3i} of the asymptotic expansion of the equations of motion for the second layer degrees of freedom do not enter the expression for η_c, see e.g. (3.28,3.29). In contrast, the eigenvalue λ_2, which determines the speed of convergence for most choices of the learning rate, depends on these coefficients and therefore on the dynamics of w.

In the following, this influence of the w-dynamics will be investigated in more detail. To this end, the microscopic equations of motion (3.8) will be changed in such a way that the learning rates in the first and second layer are different:

$$\boldsymbol{J}^{\mu+1} \;=\; \boldsymbol{J}^{\mu} - \frac{\eta_1}{N}\Gamma^{\mu}\boldsymbol{\xi}^{\mu}$$

$$w^{\mu+1} = w^{\mu} - \frac{\eta_2}{N}\Delta^{\mu} g(x^{\mu}), \tag{3.56}$$

where

$$\Delta = \sigma(\boldsymbol{\xi}) - \tau(\boldsymbol{\xi}) = w g(x) - v g(y) \quad \text{and} \quad \Gamma = w\, g'(x)\, \Delta. \tag{3.57}$$

Here, restriction has been made to the Y-network in order to simplify the analysis which can be easily extended to larger networks, however.

Despite the change of the second layer dynamics, Theorem 2 in Appendix B can still be applied because the conditions for its validity are independent of the last row of $\mathbf{m}^{(3)}$, i.e. independent of the equation of motion for w. Therefore, one obtains the same results for the eigenvalue λ_1 and the critical learning rate $\eta_{1,c}$ as in Section 3.2:

$$\lambda_1 = -\frac{4v^2\eta_1}{\pi\sqrt{1+2T}}\left(1 - \frac{\eta_1}{\pi}v^2\sqrt{\frac{1+2T}{1+4T}}\right), \tag{3.58}$$

$$\eta_{1,c} = \frac{\pi}{v^2}\sqrt{\frac{1+4T}{1+2T}}. \tag{3.59}$$

The other eigenvalues are given by

$$\lambda_{2,3} = \frac{1}{2}\left(\tilde{\lambda}_2 + m_{33}(\eta_2) \pm \sqrt{\left(\tilde{\lambda}_2 - m_{33}(\eta_2)\right)^2 + 4m_{13}(m_{31}(\eta_2) + 2m_{32}(\eta_2))}\right) \tag{3.60}$$

according to Eq. (B.6). See Section 3.2.1 for the definition of $\tilde{\lambda}_2$ and m_{ij}. In (3.60) the dependence on the second layer learning rate η_2 has been denoted explicitly. Of course it enters only through the matrix elements m_{3i}. Since the equation of motion for the hidden-to-output weight w is linear in η_2 these matrix elements are linear in η_2 as well.

Of course, the critical learning rate $\eta_{1,c}$ is independent of the learning rate in the second layer as has been generally shown in Section 3.2.2. Note, however, while $\eta_1 < \eta_{1,c}$ is required in order to achieve convergence to zero generalization error, no such restriction is imposed on the second layer learning rate η_2. Consequently, η_2 can be adjusted in order to decrease the asymptotic relaxation time $1/|\lambda_2|$.

An analysis of λ_2 as a function of η_2 shows that the relaxation time is monotonically decreasing as η_2 increases. Hence, the fastest relaxation is achieved in the limit $\eta_2 \to \infty$, where λ_2 becomes

$$\begin{aligned}\lambda_2 &= \tilde{\lambda}_2\left(1 + \frac{m_{13}}{\tilde{\lambda}_2}\lim_{\eta_2\to\infty}\frac{m_{31}+2m_{32}}{|m_{33}|}\right) \\ &= -\frac{2\eta v^2}{\pi(1+2T)^{3/2}}\left(1 - \frac{T}{(1+T)^2}\frac{\sqrt{1+2T}}{\arcsin\left(\frac{T}{1+T}\right)}\right).\end{aligned} \tag{3.61}$$

Needless to say, the resulting asymptotic relaxation time $1/|\lambda_2|$ is faster than before, Eq. (3.28), where the change of weights in each layer has been scaled with the same learning factor η/N.

In conclusion, the above analysis shows that different learning rates for the different layers of the student network can accelerate the learning process. This is achieved by increasing the

learning rate η_2 governing the dynamics of the weights in the second layer. Formally, the optimal value for η_2 is diverging.

However, what does it mean that the step size η_2 in the equation of motion for w (3.56) is divergent? How can w converge to v in this limit? In order to answer these questions the equations of motion (3.56) will be reanalyzed using well-known techniques from nonequilibrium statistical physics.

3.5.2 Adiabatic Elimination of Second Layer Couplings

An increase of η_2 in the microscopic equations of motion (3.56) leads to a significant change of the dynamics of the student network as soon as $\eta_2 = \mathcal{O}(N)$. Then the change of the second layer weight w becomes of order $\mathcal{O}(1)$ while the change of the first layer couplings $\boldsymbol{J}$ is still $\mathcal{O}(1/N)$ per time step, i.e. per training example. In this case the update rule for the student's weights reads

$$\begin{aligned} \boldsymbol{J}^{\mu+1} &= \boldsymbol{J}^{\mu} - \frac{\eta_1}{N}\Gamma^{\mu}\boldsymbol{\xi}^{\mu} \\ w^{\mu+1} &= w^{\mu} - \eta_2^{scal}\Delta^{\mu}g(x^{\mu}), \end{aligned} \tag{3.62}$$

where the scaled learning rate η_2^{scal} is defined via $\eta_2 = \eta_2^{scal}N$. Γ and Δ have the same definitions as in (3.57).

Consider the number of training steps required in order to change any weight by a certain amount: On average this number will be much smaller for w than for J_i, because of the different order of step size made per training example. Compared to a macroscopic time scale such as $\alpha = \mu/N$ where the relative change in J_i is $\mathcal{O}(1)$, w changes on a much faster time scale ($\sim d\alpha = 1/N$).

Since one is only interested in the evolution of (3.62) on the macroscopic time scale α one has to express (3.62) in terms of this time scale. Assume that the microscopic dynamics is such that $(w^{\mu+1} - w^{\mu})/(1/N)$ is finite. In the thermodynamic limit this expression is given by $dw/d\alpha$, i.e. the change of w on the macroscopic (slow) time scale α. The assumption that $dw/d\alpha$ is finite is well supported by a numerical simulation of the system (3.62), see Fig. 3.8.

Expressed in terms of the macroscopic variables R and Q the equations of motion read

$$\begin{aligned} \frac{dR}{d\alpha} &= \eta_1 w g'(x)(vg(y) - wg(x))y \\ \frac{dQ}{d\alpha} &= 2\eta_1 w g'(x)(vg(y) - wg(x))x + \eta_1^2[wg'(x)(vg(y) - wg(x))]^2 \\ \frac{dw}{d\alpha} &= \gamma^{-1}\eta_2^{scal}g(x)(vg(y) - wg(x)), \end{aligned} \tag{3.63}$$

where the parameter $\gamma = 1/N$ has been introduced in order to indicate the different time scales. Note that (3.63) is a system of stochastic differential equations. In order to obtain typical results, (3.63) has to be averaged over the internal fields x and y which comprise the randomness in the presentation of examples $\boldsymbol{\xi}$. This average leads to the following equations for the *mean values* of

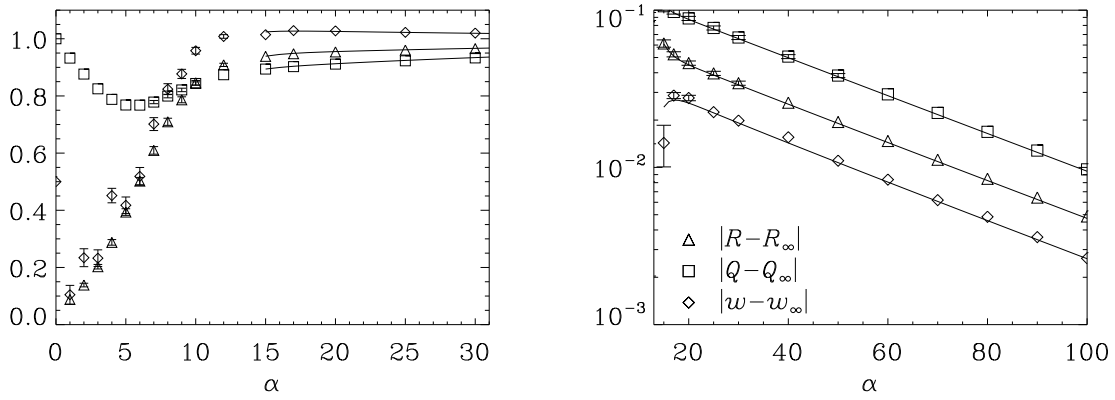


Figure 3.8: Learning curves for the update rule (3.62) where the change of the hidden-to-output weight w per learning step is $\mathcal{O}(1)$. Symbols ($\triangle$ R, $\square$ Q, $\diamond$ w) show the results of a simulation for a Y-network with input dimension $N = 100$, averaged over 89 independent runs ($T = v = 1$, $\eta_1 = \eta_2^{scal} = 1.3$, initial conditions $R(0) = \mathcal{O}(1/\sqrt{N})$, $Q(0) = 1$, $w(0) = 0.5$). Note that w is quickly relaxing to its equilibrium value ($w \simeq R$ for small α) although it is far off equilibrium initially ($w(0) = 0.5$). Right: Deviation of R, Q, w from the asymptotic fixed point $R_\infty = Q_\infty = T$, $w_\infty = v$. The approach to the fixed point is exponential in α consistent with the solution of the macroscopic equations of motion (solid). Note that for an integration of (3.64-3.66) one cannot use the above initial condition since it is not an equilibrium configuration. Therefore one has to take the mean values of the simulated R and Q at a finite α where the system is in equilibrium and the fluctuations are sufficiently small. Here $\alpha = 15$ has been chosen.

R, Q, and w:

$$\frac{dR}{d\alpha} = \frac{2}{\pi}\eta_1 \frac{w}{1+Q}\left(v\frac{T(1+Q)-R^2}{\sqrt{(1+Q)(1+T)-R^2}} - w\frac{R}{\sqrt{1+2Q}}\right) \tag{3.64}$$

$$\begin{aligned}\frac{dQ}{d\alpha} &= \frac{4}{\pi}\eta_1 \frac{w}{1+Q}\left(v\frac{R}{\sqrt{(1+Q)(1+T)-R^2}} - w\frac{Q}{\sqrt{1+2Q}}\right) \\ &\quad + \frac{4}{\pi^2}\eta_1^2 \frac{w^2}{\sqrt{1+2Q}}\left[v^2 \arcsin\left(\frac{(1+2Q)T-2R^2}{(1+2Q)(1+T)-2R^2}\right) + w^2 \arcsin\left(\frac{Q}{1+3Q}\right)\right. \\ &\quad \left. - 2vw \arcsin\left(\frac{R}{\sqrt{1+3Q}\sqrt{(1+2Q)(1+T)-2R^2}}\right)\right]\end{aligned} \tag{3.65}$$

$$\frac{dw}{d\alpha} = \frac{2}{\pi}\frac{\eta_2^{scal}}{\gamma}\left[v \arcsin\left(\frac{R}{\sqrt{1+T}\sqrt{1+Q}}\right) - w \arcsin\left(\frac{Q}{1+Q}\right)\right]. \tag{3.66}$$

Of course, these equations differ from those for the Y-network in Section 3.2 only by the different time scale of the equations of motion for w, cf. Eq. (3.23).

The goal is to find a solution of (3.63) in the limit $\gamma \to 0$, consistent with the observation that w changes on a times scale measured by γ. An appropriate method to do this is known as adiabatic elimination [Gar90]. It relies on the assumption that the mean value of the fast variable

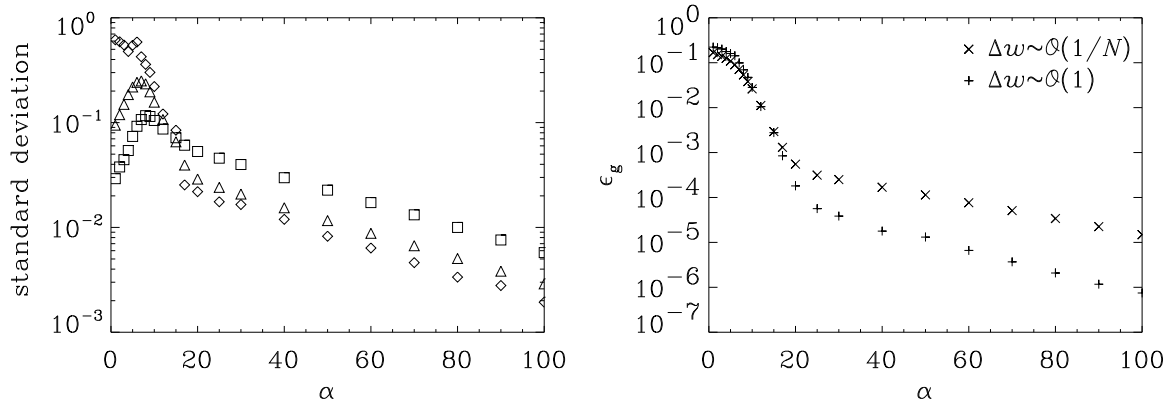


Figure 3.9: Left: Fluctuations of R, Q, w as a function of α (simulation results, notation and parameters as in Fig. 3.8). Note the large fluctuations in w initially. While the fluctuations in R and Q are finite size effects those of w do not vanish as $N \to \infty$, cf. Fig. 3.10. Right: Simulation results for the generalization error $(+)$. For comparison the generalization error one would obtain if the change of w scales with $1/N$ is also shown $(\times)$. As can be seen a scaling of $\mathcal{O}(1)$ leads to a decrease of the generalization error with a faster time constant $1/|\lambda_2|$.

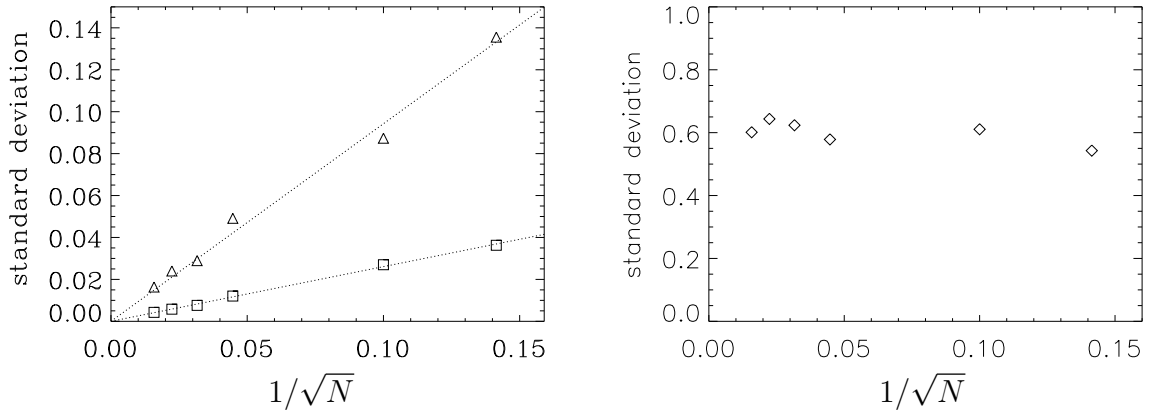


Figure 3.10: Finite size analysis of the dynamical variables R $(\triangle)$, Q $(\square)$, w $(\diamond)$. Shown are the standard deviations as a function of the system size for $\alpha = 1$. Clearly, R and Q become selfaveraging in the thermodynamic limit $N \to \infty$ while the fluctuations of w remain finite in this limit. Averages are over 100 runs, other parameters as in Figure 3.8.

has a stable equilibrium value w_{eq} at each macroscopic time. Then w will decay very rapidly to this equilibrium value because of its very fast relaxation time of order γ. The equilibrium value w_{eq} generally depends on the slow variables R, Q and is given by the zero of the r.h.s. of (3.66):

$$w_{eq}(\alpha) = v \frac{\arcsin\left(\frac{R(\alpha)}{\sqrt{1+T}\sqrt{1+Q(\alpha)}}\right)}{\arcsin\left(\frac{Q(\alpha)}{1+Q(\alpha)}\right)}. \tag{3.67}$$

Note that w_{eq} does not depend on the learning rate η_2^{scal}. This is not surprising since the quick relaxation of w to equilibrium is due to the difference of the time scales α and α/γ and does not depend on the absolute value of η_2^{scal}. It is worth stressing that w_{eq} is that value of w that minimizes the generalization error (3.24). That is, at each instance α the student network minimizes the generalization error with respect to w. Therefore it only remains to minimize ϵ_g with respect to R and Q. Effectively, w has been eliminated.

Formally, this elimination is done by replacing w in (3.64,3.65) with w_{eq}. The resulting equations of motion then depend on the slow variables only and can be integrated easily. See Figure 3.8 for an example. For $\alpha \to \infty$ the dynamics can be solved analytically as before by linearizing the equations of motion. The resulting (2×2)-matrix has the eigenvalues

$$\begin{aligned}
\lambda_1 &= -\frac{4v^2\eta_1}{\pi\sqrt{1+2T}}\left(1 - \frac{\eta_1}{\pi}v^2\sqrt{\frac{1+2T}{1+4T}}\right) \\
\lambda_2 &= -\frac{2\eta v^2}{\pi(1+2T)^{3/2}}\left(1 - \frac{T}{(1+T)^2}\frac{\sqrt{1+2T}}{\arcsin\left(\frac{T}{1+T}\right)}\right),
\end{aligned} \tag{3.68}$$

which are exactly the same as those obtained in (3.58,3.61) of the previous section for an optimized second layer learning rate. For λ_1 this is trivial since this quantity is exclusively determined by the first layer. The equality of λ_2, however, leads to the conclusion that the diverging learning rate η_2 of the previous section indeed corresponds to a step size of $\mathcal{O}(1)$ in the second layer dynamics. Consequently, making the step size of the hidden-to-output weights of $\mathcal{O}(1)$ while keeping that of the first layer of $\mathcal{O}(1/N)$ accelerates the learning process.

Strictly speaking, however, the convergence has been shown in mean only, not in variance. That is, it still has to be checked whether the fluctuations of R, Q, and w vanish in the thermodynamics limit. This has to be the case at least for the desired final state $R = Q = T$, $w = v$. Otherwise the weights of the teacher network will not represent a stable configuration of the student network.

From the microscopic dynamics of w (3.62) it is immediately clear that w is not a selfaveraging quantity. Its fluctuations are of $\mathcal{O}(1)$. How about R and Q? Do the fluctuations of w make them being not selfaveraging any more? Simulations suggest that R and Q are still selfaveraging quantities, cf. Fig. 3.10.

This can also be made plausible the following way: Imagine the stochastic differential equa-

tions (3.63) can be cast into the form of Langevin equations:[5]

$$\begin{aligned}
\frac{dR}{d\alpha} &= h_R^{drift}(R,Q,w) + \sqrt{h_R^{diff}(R,Q,w)}\chi_R \\
\frac{dQ}{d\alpha} &= h_Q^{drift}(R,Q,w) + \sqrt{h_Q^{diff}(R,Q,w)}\chi_Q \\
\frac{dw}{d\alpha} &= \gamma^{-1} h_w^{drift}(R,Q,w) + \gamma^{-1/2}\sqrt{h_w^{diff}(R,Q,w)}\chi_w,
\end{aligned} \tag{3.69}$$

where the right hand sides have been split into drift and diffusion terms with Langevin forces χ_i ($\langle \chi_i \chi_j \rangle = \delta_{ij}$). Of course the drift terms are exactly the right hand sides of (3.64-3.66). In the Langevin equation for w the dependence on γ has been written explicitly in order to point out the different time scale.

Consider the case of a fixed hidden-to-output weight w (not necessarily $w = v$). Clearly, R and Q are selfaveraging in this case, i.e. their fluctuations are of order $\mathcal{O}(1/\sqrt{N})$. Therefore the diffusion terms h_R^{diff} and h_Q^{diff} (3.69) have to be of order $\mathcal{O}(1/N)$. Hence, R and Q are selfaveraging even for fluctuating w.

Therefore in the thermodynamic limit, the system (3.69) reduces to a stochastic differential equation for w which is adiabatically coupled to the deterministic differential equations for R and Q. Such a system is known to have an asymptotic stationary distribution of R, Q, and w which is a Gaussian with mean $h_w^{drift}(R,Q,w)$ and variance $h_w^{diff}(R,Q,w)$ [Gar90]. According to the definition of the diffusion, $\lim_{d\alpha\to 0}[(w(\alpha+d\alpha) - w(\alpha))^2/d\alpha]$, one obtains from (3.62)

$$h_w^{diff}(R,Q,w) = (\eta_2^{scal})^2 \Big\langle [\Delta(x,y;R,Q,w)g(x)]^2 \Big\rangle_{P(x,y;R,Q)}. \tag{3.70}$$

This expression is quadratic in w and vanishes at the fixed point: $h_w^{diff}(T,T,v) \equiv 0$. Consequently $(h_w^{diff})^{1/2}$ approaches 0 at the same rate as R, Q, and w_{eq} (see also Fig. 3.9) and therefore w is indeed convergent in mean *and* variance.

The generalization of the results obtained here for the Y-network to more complicated two-layer networks with a finite number of hidden units K is easy in principle. Nevertheless, the explicit calculation may become quite complicated, in particular for fully connected architectures. In the case of $N \gg K > 1$ one can scale the change in w with $1/K$ and define $\gamma = K/N$. As long as $\gamma \ll 1$ the equations of motion of the hidden-to-output weights can be eliminated adiabatically in the same manner as above. The resulting increase in the speed of convergence might be viewed as a theoretical explanation of the phenomenological rule that it is advantageous to scale the weight change with the inverse of the number of weights per node [HKP91].

Finally, an interesting extension of the concepts of this section will be discussed. Here it has been shown that in two-layer networks with a finite number of hidden units the speed of the learning progress can be increased by putting the dynamics of the hidden-to-output weights on a faster time scale. Similar results can be obtained if one puts other degrees of freedom on a faster time scale. Consider for example on-line backpropagation for the most general two-layer

[5] That is one assumes that the stochasticity of the r.h.s of (3.63) is completely described by the first two moments.

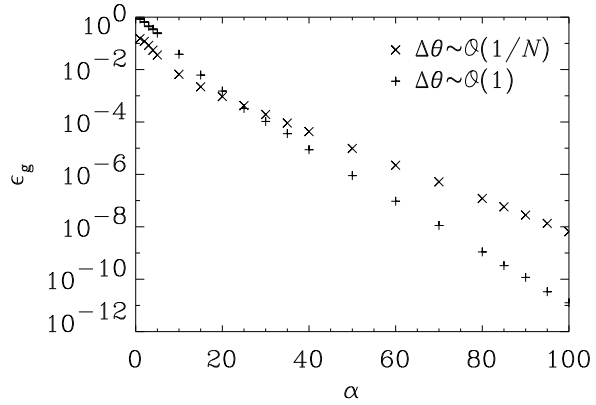


Figure 3.11: Simulation results for the generalization error of a perceptron $\sigma(\boldsymbol{\xi}) = g(\boldsymbol{J} \cdot \boldsymbol{\xi} + \theta)$ with threshold θ learning a realizable rule. Two different types of on-line backpropagation have been used in order to train the perceptron: For the first type (represented by $\times$) the update of the threshold θ has been chosen to scale with $1/N$ while it is of $\mathcal{O}(1)$ in the second case (represented by $+$). Clearly, an update of $\mathcal{O}(1)$ leads to better generalization asymptotically. Parameters: $T = \theta_t = 1$, the learning rate for the weights $\boldsymbol{J}$ and the threshold is 1.3, system size $N = 100$, $\theta(0) = 0.01$. Results have been obtained by an average over 100 independent runs.

network (3.5) where the microscopic equation of motion for the additional degrees of freedom θ_j is given by

$$\theta_j^{\mu+1} = \theta_j^{\mu} - \frac{\eta_3}{N} \frac{\partial}{\partial \theta_j} \epsilon(\boldsymbol{W}, \boldsymbol{\xi}). \tag{3.71}$$

In this case, a scaling of the respective learning rate $\eta_3 = \eta_3^{scal} N$ will also lead to a faster overall convergence. This is exemplified in Figure 3.11 for the perceptron

$$\sigma(\boldsymbol{\xi}) = g(\boldsymbol{J} \cdot \boldsymbol{\xi} + \theta) \tag{3.72}$$

learning a realizable rule $\tau(\boldsymbol{\xi}) = g(\boldsymbol{B} \cdot \boldsymbol{\xi} + \theta_t)$.

A general understanding which microscopic degrees of freedom can be profitably put on a faster time scale has still to be gained, however.

3.6 Conclusions

Unlike in Chapter 2 of this work the aim of this chapter has not been to derive optimal algorithms. This is a difficult task in general since the generalization error (3.20) is a nonmonotonic function of the order parameters. Therefore, it is not possible to simply replace the maximization of the change of the generalization error per learning step by an extremization of the change of the order parameters as in Chapter 2. Only for the Y-network defined in Section 3.2 the optimization could be performed easily since the internal field of the teacher network can be directly accessed in this case, see [KC95] for details.

The main concern of this chapter has been to elucidate how the dynamics of multi-layer networks can be described with tools from physics. This is of importance since so far generalization properties could be analyzed for very simplified models of two-layer networks only, such as committee and parity machines, see e.g. [Sch93, Opp94, CEK$^+$97]. Here, it has been shown how to incorporate the dynamics of the second layer degrees of freedom into the physical analysis. This is of interest since such networks have a larger complexity. Moreover, they are commonly used in practical applications of neural networks.

In a first step, this goal has been achieved by considering each single hidden-to-output coupling as an order parameter, i.e. as a quantity which is selfaveraging in the thermodynamic limit $N \to \infty$. This is possible because the weight change of these couplings has been taken to be of order $\mathcal{O}(1/N)$.

It has been shown how the increase of the network's complexity is reflected by a decrease of the asymptotic relaxation time. Other properties, however, have been found to be completely independent of the second layer dynamics. For instance the critical learning rate depends on the structure of the first layer only. This holds true irrespective of the dynamics of the second layer, whether or not on-line backpropagation is used for updating these weights.

It should be emphasized that these results have been obtained based on certain symmetry assumptions on the rule to be implemented. This has been done in order to make the analysis feasible. Therefore, it is not *a priori* clear whether the above results will carry over to more asymmmetric settings.

In a second step the weight change in the second layer has been taken to be of $\mathcal{O}(1)$, i.e. by a factor of N larger than in the first setting. Although the second layer couplings are not selfaveraging any more then, the theoretical analysis can still be performed. This is because the degrees of freedom in the second layer change on a much faster time scale. Hence, they can be eliminated adiabatically. Over all, the weight change of $\mathcal{O}(1)$ results in an increase of the network's speed of convergence.

It is expected that these features will carry over to the optimal on-line algorithm, yet to be found. For the case of a finite number of hidden units K, the second layer weights should assume their instantaneous equilibrium value which can be used to eliminate these degrees of freedom. The dynamical system then effectively reduces to a two-layer network with fixed hidden-to-output couplings. Therefore it should be sufficient to construct optimal algorithms for such a network.

The situation becomes different, however, in the limit of an extensive number of hidden units. Here, a theoretical description, irrespective of the algorithms used, has still to be found.

Chapter 4

Discussion and Directions for Further Research

In this work the generalization properties of on-line learning processes in neural networks have been investigated. On-line learning has become a central research topic in the physics of neural networks in recent years. In contrast to more traditional learning problems which can be treated by methods of equilibrium statistical mechanics, the learning processes investigated here represent dynamical systems. Hence, an approach with methods of nonequilibrium statistical mechanics is expected to be appropriate in order to calculate the generalization properties exactly.

The main focus of this work has been twofold: The construction of optimal training algorithms and the theoretical description of learning processes in multi-layer neural networks.

In the first part it has been shown how learning algorithms can be derived from very fundamental principles. This is done by means of a variational principle which is at the basis of many theories in physics. The resulting algorithms have two important properties. First, they are optimal, i.e. there is no on-line algorithm leading to a typically higher generalization ability after being trained with αN examples. Second, the features of the algorithm needed to reach such an optimal generalization ability can be calculated and, hence, have not to be postulated *ad hoc*.

In this work, optimal algorithms have been derived for perceptron learning where the training data is corrupted by noise. This investigation is of crucial importance since it has not been known before whether a concept can be learnt on-line in spite of noise in the data. For the perceptron investigated here the answer is positive. Meanwhile this has been shown to be true for more complicated machines as well [CEK$^+$97].

Moreover, the analysis reveals interesting critical phenomena: The calculated optimal feature algorithms require knowledge about the characteristics of the noise process. These characteristics are unknown in general and consequently have to be estimated. However, the convergence properties of the algorithms depend critically on these estimators. Depending on their values there are phases where perfect generalization is possible or impossible. Recent investigations [CEK$^+$97] have shown that the resulting phase diagrams are universal, i.e. independent of the network architecture, to a large extent.

These promising results suggest further research in this direction. The concept of optimal

feature algorithms is in principle extendible to other learning situations such as more complicated architectures and input distributions. However, new analytical tools have to be developed. For instance, so far not very much is known about how to handle input distributions relevant in applications such as correlated inputs or time series. The inherent dependence of optimal algorithms on the characteristics of the learning problem further requires to search for efficient methods to estimate these characteristics.

A general research goal is to find methods which enable one to calculate the generalization properties of multi-layer networks. So far, properties could be calculated only for very simple two-layer networks where the hidden-to-output weights are fixed. Since this limits the complexity of the networks an extension to variable hidden-to-output weights is desirable. Here, it has been shown how this can be accomplished for two-layer networks trained by the on-line version of the ubiquitous backpropagation algorithm.

For two-layer networks one has to distinguish between two limiting cases, depending on the number of hidden units compared to the number of input units. For a small number of hidden units this work has shown how to incorporate the dynamics of the second layer weights into the theoretical description. The essential result is that these degrees of freedom can be put on a much faster time scale than those in the first layer.

As has been pointed out in the text, the application of this concept is not limited to hidden-to-output weights. Preliminary results [Rie] show that it can be used to speed up convergence of other learning systems as well. In this context it is of general interest to find out which quantities can be put on a fast dynamics with the profit of a lower generalization error.

The second limiting case is characterized by an extensive number of hidden units. It is interesting from several points of view. First, it is relevant for the understanding of learning processes in multi-layer networks already used in applications today. Furthermore, these networks are of general interest since they represent universal approximators, i.e. they can realize virtually every task with arbitrary precision. However, it is still an open question how long (on the time scale α) it will typically take in order to reach a desired accuracy. Finally, these systems might be relevant as simple models for other physical systems with disorder, such as glasses.

It is therefore desirable to extend the theoretical description presented here to such networks. Some general problems one will encounter become immediately clear from this work. The number of order parameters required for a macroscopic description scales with the number of hidden units K which is extensive in this case. For short: There are too many "order parameters". Therefore one has to search for a way to carry out the thermodynamic limit appropriately. This might be achievable by identifying a finite set of proper order parameters for which the equations of motion can be written in a closed form.

Once a way to describe such networks has been found one can continue with the program presented in the first part of this work, namely to derive optimal algorithms, test their robustness with respect to noise, etc. Ultimately this might open the door to a general theory of on-line learning.

Appendix A

Multivariate Gaussian Distribution

If $\boldsymbol{z} = (z_1, z_2, \ldots, z_k)^\top$ is a vector of k Gaussian random variables, the corresponding multivariate probability density function can be written

$$P(\boldsymbol{z}) = \frac{1}{(2\pi)^{k/2}\sqrt{\mathrm{Det}\,\mathcal{C}}} \exp\left(-\frac{1}{2}(\boldsymbol{z} - \langle \boldsymbol{z} \rangle)^\top \mathcal{C}^{-1} (\boldsymbol{z} - \langle \boldsymbol{z} \rangle)\right) \tag{A.1}$$

with covariance matrix

$$\mathcal{C}_{ij} = \langle z_i z_j \rangle - \langle z_i \rangle \langle z_j \rangle. \tag{A.2}$$

For instance in the case of weight noise (Section 2.4) the elements of the vector $\boldsymbol{z}$ are the internal fields, $\boldsymbol{z} = (h_J, h_B, \widetilde{h_B})^\top$, with zero mean and covariance (cf. Eq. (2.28))

$$\mathcal{C} = \begin{pmatrix} 1 & \rho & \omega\rho \\ \rho & 1 & \omega \\ \omega\rho & \omega & 1 \end{pmatrix}. \tag{A.3}$$

For the two-layer networks with K hidden units considered in Chapter 3, the Gaussian variables are the internal fields of each student and teacher branch in the input-to-hidden layer: $\boldsymbol{z} = (x_1, \ldots, x_K, y_1, \ldots, y_K)^\top$. The covariance matrix is

$$\mathcal{C} = \begin{pmatrix} Q & R \\ R^\top & T \end{pmatrix}, \tag{A.4}$$

where the elements of the matrices R and Q are the order parameters and the elements of T are the correlations between different teacher branches, cf. Eqs. (3.14,3.15).

Multivariate Gaussian averages over functions of l components of $\boldsymbol{z}$ are performed using the l-dimensional covariance matrix C which results from projecting the full covariance matrix $\mathcal{C}$ onto the relevant subspace. For instance, for the three-dimensional average $I_3(i,j,k) = \langle g'(z_i) z_j g(z_k) \rangle$ the corresponding matrix is:

$$C = \begin{pmatrix} \mathcal{C}_{ii} & \mathcal{C}_{ij} & \mathcal{C}_{ik} \\ \mathcal{C}_{ji} & \mathcal{C}_{jj} & \mathcal{C}_{jk} \\ \mathcal{C}_{ki} & \mathcal{C}_{kj} & \mathcal{C}_{kk} \end{pmatrix}. \tag{A.5}$$

The two multivariate integrals (3.16,3.17) needed in Chapter 3 can be performed analytically for $g(x) = \mathrm{erf}(x/\sqrt{2})$:

$$\begin{aligned} I_3(i,j,k) &\equiv \langle g'(z_i) z_j g(z_k) \rangle = \frac{2}{\pi} \frac{1}{\sqrt{\Lambda_3}} \frac{C_{23}(1+C_{11}) - C_{12}C_{13}}{1+C_{11}} \\ I_4(i,j,k,l) &\equiv \langle g'(z_i) g'(z_j) g(z_k) g(z_l) \rangle = \frac{4}{\pi^2} \frac{1}{\sqrt{\Lambda_4}} \arcsin\left(\frac{\Lambda_0}{\sqrt{\Lambda_1 \Lambda_2}} \right), \end{aligned} \tag{A.6}$$

where I_3 (I_4) is given in terms of the elements of the corresponding three-(four-)dimensional covariance matrix and

$$\begin{aligned} \Lambda_0 &= \Lambda_4 C_{34} - C_{23}C_{24}(1+C_{11}) - C_{13}C_{14}(1+C_{22}) + C_{12}C_{13}C_{24} + C_{12}C_{14}C_{23} \\ \Lambda_1 &= \Lambda_4(1+C_{33}) - C_{23}^2(1+C_{11}) - C_{13}^2(1+C_{22}) + 2C_{12}C_{13}C_{23} \\ \Lambda_2 &= \Lambda_4(1+C_{44}) - C_{24}^2(1+C_{11}) - C_{14}^2(1+C_{22}) + 2C_{12}C_{14}C_{24} \\ \Lambda_3 &= (1+C_{11})(1+C_{33}) - C_{13}^2 \\ \Lambda_4 &= (1+C_{11})(1+C_{22}) - C_{12}^2. \end{aligned} \tag{A.7}$$

See [SS95b] for the proof of these formulae and the general expressions for multivariate Gaussian integrals.

Appendix B

Matrix Theorems

This appendix contains the matrix theorems that have been exploited in Chapter 3. Only the proof for matrices of dimension 3 will be given in some detail. Though more tedious, the proof for matrices of dimension 5 can be carried out similarly to the three-dimensional case.

Theorem 2 *Let* $\mathbf{m}$ *be a square matrix of dimension* 3 *obeying* $b = m_{23}/m_{13}$ *and* $\not{\mathbf{m}}$ *the conjugated square matrix of dimension* 2 *that is obtained out of* $\mathbf{m}$ *by truncating the third column and row. If*

$$m_{11} - \frac{1}{b} m_{21} = m_{22} - b\, m_{12} \equiv \lambda_1 \tag{B.1}$$

then

1. λ_1 *is an eigenvalue of both* $\mathbf{m}$ *and* $\not{\mathbf{m}}$*,*
2. *for nondegenerate* λ_1*, the eigenvectors of* $\mathbf{m}$ *corresponding to the eigenvalues different from* λ_1 *are of the form* $(1, b, z_{2,3})^\top$*,*
3. *for* $\not{\mathbf{m}}$*, the eigenvector corresponding to the eigenvalue different from* λ_1 *is* $(1, b)^\top$*.*

In other words, given (B.1)*, the matrices* $\mathbf{m}$ *and* $\not{\mathbf{m}}$ *have one eigenvalue in common and the eigenvectors corresponding to the remaining eigenvalues are identical when projected onto the* (x, y)*-plane.*

Proof: Exploiting (B.1) it is straightforward to cast the characteristic polynomial of $\mathbf{m}$

$$\chi_3(\lambda) = \begin{vmatrix} m_{11} - \lambda & m_{12} & m_{13} \\ m_{21} & m_{22} - \lambda & b\, m_{13} \\ m_{31} & m_{32} & m_{33} - \lambda \end{vmatrix} \tag{B.2}$$

into the form

$$\chi_3(\lambda) = (\lambda_1 - \lambda)[(m_{33} - \lambda)(\not{\chi}_2 - \lambda) - m_{13}(m_{31} + b\, m_{32})] \tag{B.3}$$

where $\not\lambda_2 = m_{11} + b\, m_{12}$. From (B.3) it is trivial to see that the characteristic polynomial of $\not{\mathbf{m}}$ is

$$\chi_2(\lambda) = (\lambda_1 - \lambda)(\not\lambda_2 - \lambda) \tag{B.4}$$

which proofs the first statement.

The second statement is most easily shown by inserting $(1, b, z_{2,3})^\top$ into the corresponding eigenvalue equation. The equations for the first two components read

$$\begin{aligned} m_{11} + b\, m_{12} + z_{2,3}\, m_{13} &= \lambda_{2,3} \\ m_{21} + b\, m_{22} + z_{2,3}\, b\, m_{13} &= \lambda_{2,3}\, b \end{aligned} \tag{B.5}$$

where $\lambda_{2,3}$ denotes the eigenvalues of $\mathbf{m}$ different from λ_1:

$$\lambda_{2,3} = \frac{1}{2}\Big(\not\lambda_2 + m_{33} \pm \sqrt{(\not\lambda_2 - m_{33})^2 + 4m_{13}(m_{31} + b\, m_{32})}\Big). \tag{B.6}$$

Multiplying the first equation of (B.5) by b and taking the difference yields

$$b\Big(m_{11} - \frac{1}{b}m_{21} - m_{22} + b\, m_{12}\Big) = 0$$

which is true for any b since the term in parentheses vanishes due to supposition (B.1).

The third statement follows trivially from the eigenvalue equation for $\not{\mathbf{m}}$. □

Theorem 3 *Let* $\mathbf{m}$ *be a square matrix of dimension* 5 *obeying*

$$\frac{m_{15}}{m_{25}} = \frac{m_{35}}{m_{45}} = \frac{1}{2} \tag{B.7}$$

and $\not{\mathbf{m}}$ *the conjugated square matrix of dimension* 4 *that is obtained out of* $\mathbf{m}$ *by truncating the fifth column and row. If*

$$\begin{aligned} m_{11} - \frac{1}{2}m_{21} &= m_{22} - 2m_{12} \equiv A \\ m_{33} - \frac{1}{2}m_{43} &= m_{44} - 2m_{34} \\ m_{31} - \frac{1}{2}m_{41} &= m_{42} - 2m_{32} \equiv B \\ m_{13} - \frac{1}{2}m_{23} &= m_{24} - 2m_{14} \equiv C \end{aligned} \tag{B.8}$$

then

$$\lambda_{1,3} = \frac{1}{2}\Big(A + m_{44} \pm \sqrt{(A - m_{44})^2 + 4BC}\Big) \tag{B.9}$$

is an eigenvalue of both $\mathbf{m}$ *and* $\not{\mathbf{m}}$.

After an application of this theorem the remaining three (two) eigenvalues of $\mathbf{m}$ ($\not{\mathbf{m}}$) are easily obtained by solving for the roots of the characteristic polynomial by standard methods.

Appendix C

Overlapping Architecture

The eigenvalue spectrum of the linearized equations of motion (3.54,3.55) for the fully connected architecture can be obtained in a straightforward manner by application of Theorem 3 in Appendix B.

$$\begin{aligned}
\lambda_{1,3} = \tilde{\lambda}_{1,3} &= -\frac{1}{2}\Big(\mathcal{A} \mp \sqrt{\mathcal{A}^2 - 4\mathcal{B}}\Big) \\
\tilde{\lambda}_{2,4} &= -\frac{\eta v^2}{\pi}\left(\frac{K-2}{1+T} + \frac{2\,(1+T)}{(1+2\,T)^{3/2}}\right. \\
&\qquad \left. \mp \sqrt{\left(\frac{K-2}{1+T} + \frac{2\,T}{(1+2\,T)^{3/2}}\right)^2 + \frac{4\,(K-1)}{(1+T)^4}}\right) \\
\lambda_2 &= \frac{a}{3} + \frac{2^{1/3}}{3}\,\Big(a^2 + 3\,b\Big)\,d^{-1/3} + \frac{d^{1/3}}{2^{1/3}\cdot 3} \\
\lambda_4 &= -4\frac{\eta v^2}{\pi}\left(\frac{1}{\sqrt{1+2T}} - \frac{1}{1+T}\right) \\
\lambda_5 &= -4\frac{\eta v^2}{\pi}\left(\frac{1}{\sqrt{1+2T}} + \frac{K-1}{1+T}\right)
\end{aligned} \tag{C.1}$$

The critical learning rate η_c is given by the zero of λ_1 with respect to η:

$$\eta_c = -\frac{\pi}{2v^2\mathcal{B}_2}\left(\mathcal{B}_1 + \sqrt{\mathcal{B}_1^2 - 4\mathcal{B}_0\mathcal{B}_2}\right). \tag{C.2}$$

For sake of compactness, the following abbreviations have been used:

$$\begin{aligned}
a &= -\frac{2\eta v^2}{\pi}\left(\frac{K-2}{1+T} + \frac{2\,(1+T)}{(1+2\,T)^{3/2}}\right) - \frac{2\eta}{\pi}\,\arcsin\left(\frac{T}{1+T}\right) \\
b &= \frac{4\,\eta^2\,(K-1)\,T\,v^2}{\pi^2\,(1+T)^2} + \frac{4\,\eta^2\,T\,v^2}{\pi^2\,(1+T)^2\,(1+2\,T)} \\
&\quad -\frac{4\,\eta^2\,v^4}{\pi^2\,(1+T)^4}\left(\frac{(K-2)\,(1+T)^3}{(1+2\,T)^{3/2}} + \frac{T^2\,\left(2+4\,T+T^2\right)}{(1+2\,T)^2} - (K-2)\right)
\end{aligned}$$

$$
\begin{aligned}
& -\frac{4\,\eta^2\,v^2}{\pi^2}\left(\frac{K-2}{1+T}+\frac{2\,(1+T)}{(1+2\,T)^{3/2}}\right)\arcsin\left(\frac{T}{1+T}\right)\\
c \;=\;& \frac{8\,\eta^3\,(K-1)\,T^3\,v^4}{\pi^3\,(1+T)^4\,(1+2\,T)^{3/2}}\\
& -\frac{2\,\eta^3\,v^4}{\pi^3\,(1+T)\,\sqrt{1+2\,T}}\,T\left(\frac{4\,(K-1)}{(1+T)^3}+\frac{4}{1+T}-\frac{8}{1+2\,T}-\frac{4\,(K-2)}{(1+T)^2\,\sqrt{1+2\,T}}\right)\\
& -\frac{8\,\eta^3\,v^4}{\pi^3\,(1+T)^4}\left(\frac{(K-2)\,(1+T)^3}{(1+2\,T)^{3/2}}+\frac{T^2\,\left(2+4\,T+T^2\right)}{(1+2\,T)^2}-(K-2)\right)\arcsin\left(\frac{T}{1+T}\right)\\
d \;=\;& \left(2\,a^3+9\,a\,b+27\,c+\sqrt{(2\,a^3+9\,a\,b+27\,c)^2-4\,(a^2+3\,b)^3}\right)\\
\mathcal{A} \;=\;& \frac{4\eta v^2}{\pi}\left(\frac{K-2}{1+T}+\frac{2}{\sqrt{1+2\,T}}\right)\\
& -\frac{4\eta^2 v^4}{\pi^2}\left(\frac{(K-3)\,(K-2)}{(1+T)^2}+\frac{1+K}{1+2\,T}+\frac{4\,(K-2)}{(1+T)\,\sqrt{1+2\,T}}+\frac{1}{\sqrt{1+4\,T}}\right)\\
\mathcal{B} \;=\;& \frac{\eta^2 v^4}{\pi^2}\mathcal{B}_0+\frac{\eta^3 v^6}{\pi^3}\mathcal{B}_1+\frac{\eta^4 v^8}{\pi^4}\mathcal{B}_2\\
\mathcal{B}_0/16 \;=\;& -\frac{K-1}{(1+T)^2}+\frac{1}{1+2\,T}+\frac{K-2}{(1+T)\,\sqrt{1+2\,T}}\\
\mathcal{B}_1/16 \;=\;& \frac{-2}{(1+2\,T)^{3/2}}-\frac{(K-3)\,(K-2)}{(1+T)^2\,\sqrt{1+2\,T}}-\frac{1}{\sqrt{1+2\,T}\,\sqrt{1+4\,T}}\\
& +(K-2)\,(K-1)\left(-\frac{1}{(1+T)\,(1+2\,T)}+\frac{2}{(1+T)^2\,\sqrt{1+2\,T}}\right)\\
& +(K-1)\left(-(1+2\,T)^{-3/2}+\frac{4}{(1+T)^{3/2}\,\sqrt{1+3\,T}}\right)\\
& -(K-2)\left(\frac{4}{(1+T)\,(1+2\,T)}+\frac{1}{(1+T)\,\sqrt{1+4\,T}}\right)\\
\mathcal{B}_2/16 \;=\;& \frac{2\,(K-1)}{(1+2\,T)^2}+\frac{4\,(K-2)\,(K-1)}{(1+T)\,(1+2\,T)^{3/2}}-\frac{4\,(K-2)\,(K-1)}{(1+T)^{3/2}\,\sqrt{1+2\,T}\,\sqrt{1+3\,T}}\\
& +\frac{(K-3)\,(K-2)}{(1+T)^2\,\sqrt{1+4\,T}}+\frac{2}{(1+2\,T)\,\sqrt{1+4\,T}}+\frac{4\,(K-2)}{(1+T)\,\sqrt{1+2\,T}\,\sqrt{1+4\,T}}\\
& -\frac{4\,(K-1)}{(1+T)(1+3T)}+\frac{(K-3)\,(K-2)\,(K-1)}{(1+T)^2(1+2T)}-\frac{(K-2)^2\,(K-1)}{(1+T)^2(1+2T)}.
\end{aligned}
$$

Appendix D

Selfaveraging Properties

Throughout this work selfaveraging properties have been assumed about certain quantities in the course of the theoretical analysis. Here, some theoretical justification will be given for these assumptions. Consider a quantity O which is $\mathcal{O}(1)$ and whose time evolution on the microscopic time scale μ is given by

$$O^{\mu+1} = O^{\mu} + \frac{1}{N} F(O^{\mu}, \boldsymbol{\xi}^{\mu}). \tag{D.1}$$

For instance, O could represent the teacher-student overlap R for perceptron learning. For sake of simplicity only systems which can be described by a single order parameter O will be considered here.

For the following it is assumed that the function F in (D.1) is differentiable. Furthermore, the system is initially prepared in a fixed state $\overline{O^0}$, hence $P(O^0) = \delta(O^0 - \overline{O^0})$. If the dynamics (D.1) happens to possess several fixed points due to some symmetries of the system, it will be implied in the following that this symmetry has been removed by mapping all equivalent fixed point configurations to a single one. The random examples $\boldsymbol{\xi}$ will be assumed to be independent at different time steps:

$$P(\boldsymbol{\xi}^0, \ldots, \boldsymbol{\xi}^{\mu}) = \prod_{\nu=0}^{\mu} P(\boldsymbol{\xi}^{\nu}). \tag{D.2}$$

Finally it will be assumed that the statistics of O at any time can be sufficiently described by its first two moments, i.e. its distribution is a Gaussian:

$$\mathcal{P}(O) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\frac{(O-\overline{O})^2}{\sigma^2}\right). \tag{D.3}$$

According to (D.1) the variance $\sigma^2 = \langle O^2 \rangle - \langle O \rangle^2$ satisfies the recursion relation

$$\begin{aligned}(\sigma^{\mu+1})^2 - (\sigma^{\mu})^2 &= \frac{2}{N}[\langle F(O^{\mu}, \boldsymbol{\xi}^{\mu}) O^{\mu} \rangle - \langle F(O^{\mu}, \boldsymbol{\xi}^{\mu}) \rangle \langle O^{\mu} \rangle] \\ &\quad + \frac{1}{N^2}\Big[\Big\langle F^2(O^{\mu}, \boldsymbol{\xi}^{\mu}) \Big\rangle - \langle F(O^{\mu}, \boldsymbol{\xi}^{\mu}) \rangle^2\Big].\end{aligned} \tag{D.4}$$

The average is over the whole sequence of examples, i.e. over the distribution (D.2). Since examples $\boldsymbol{\xi}$ are independent at different time instances one can alternatively average over the

latest example $\boldsymbol{\xi}^\mu$ and over O^μ which comprises the randomness of $\left\{\boldsymbol{\xi}^0,\ldots,\boldsymbol{\xi}^{\mu-1}\right\}$ according to (D.1).

In the following it will be shown that the $(1/N)$-term on the r.h.s. of (D.4) is actually $\mathcal{O}(1/N^2)$. Therefore, the change of σ^2 per microscopic time step is $\mathcal{O}(1/N^2)$. Consequently, after a macroscopic number of time steps $\mu=\alpha N$, the variance of O can at most be $\mathcal{O}(1/N)$ which vanishes in the thermodynamic limit $N\to\infty$. Hence, O is indeed a selfaveraging quantity.

The proof will be accomplished by means of a complete induction. After presentation of the first example one has because of the initially sharp distribution of O:

$$\begin{aligned}(\sigma^1)^2-\underbrace{(\sigma^0)^2}_{=0} &= \frac{1}{N}\Big(\Big\langle F(O^0,\boldsymbol{\xi}^0)O^0\Big\rangle_{O^0,\xi^0}-\Big\langle F(O^0,\boldsymbol{\xi}^0)\Big\rangle_{O^0,\xi^0}\overline{O^0}\Big)+\mathcal{O}(1/N^2)\\ &= \frac{1}{N}\Big(\Big\langle F(\overline{O^0},\boldsymbol{\xi}^0)\Big\rangle_{\xi^0}\overline{O^0}-\Big\langle F(\overline{O^0},\boldsymbol{\xi}^0)O^0\Big\rangle_{\xi^0}\overline{O^0}\Big)+\mathcal{O}(1/N^2)\\ &= \mathcal{O}(1/N^2).\end{aligned}\tag{D.5}$$

Hence, after one time step the variance of O is $\mathcal{O}(1/N^2)$.

After μ time steps the first term of the r.h.s. of (D.4) reads (the index μ is omitted for simplicity):

$$\langle F(O,\boldsymbol{\xi})O\rangle-\langle F(O,\boldsymbol{\xi})\rangle\langle O\rangle=\int dO\mathcal{P}(O)O\langle F(O,\boldsymbol{\xi})\rangle_\xi-\overline{O}\int dO\mathcal{P}(O)\langle F(O,\boldsymbol{\xi})\rangle_\xi.\tag{D.6}$$

For the induction, the variance of O at time step μ is assumed to be $\sigma^2=\mu\cdot\mathcal{O}(1/N^2)=\alpha\cdot\mathcal{O}(1/N)$. Defining $x=(O-\overline{O})/\sigma$ one obtains for the last equation

$$\begin{aligned}&\int Dx(\overline{O}+\sigma x)\Big\langle F(\overline{O}+\sigma x,\boldsymbol{\xi})\Big\rangle_\xi-\overline{O}\int Dx\Big\langle F(\overline{O}+\sigma x,\boldsymbol{\xi})\Big\rangle_\xi\\ =\;&\overline{O}\Big\langle F(\overline{O},\boldsymbol{\xi})\Big\rangle_\xi\int Dx+\sigma\Big[\Big\langle F(\overline{O},\boldsymbol{\xi})\Big\rangle_\xi+\overline{O}\Big\langle F'(\overline{O},\boldsymbol{\xi})\Big\rangle_\xi\Big]\int Dx\,x\\ &-\overline{O}\Big\langle F(\overline{O},\boldsymbol{\xi})\Big\rangle_\xi\int Dx-\sigma\,\overline{O}\Big\langle F'(\overline{O},\boldsymbol{\xi})\Big\rangle_\xi\int Dx\,x+\mathcal{O}(\sigma^2)\\ =\;&\mathcal{O}(\sigma^2).\end{aligned}\tag{D.7}$$

The expansion in σ is justified because of the assumption $\sigma^2=\mathcal{O}(1/N)$. Equations (D.4, D.6,D.7) therefore lead to the result

$$(\sigma^{\mu+1})^2-(\sigma^\mu)^2=\frac{1}{N}\mathcal{O}\Big((\sigma^\mu)^2\Big)+\mathcal{O}(1/N^2)=\mathcal{O}(1/N^2),\tag{D.8}$$

which completes the induction, $(\sigma^{\mu+1})^2=(\mu+1)\mathcal{O}(1/N^2)\sim\mathcal{O}(1/N)$, and proves that O is selfaveraging for all α in the thermodynamic limit $N\to\infty$ under the above assumptions.

Appendix E

Notation

$$
\begin{aligned}
\operatorname{sign}(x) &= \begin{cases} 1 & \text{for } x > 0 \\ -1 & \text{for } x < 0 \end{cases} \\
\Theta(x) &= \begin{cases} 1 & \text{for } x \geq 0 \\ 0 & \text{for } x < 0 \end{cases} \\
Dx &= \frac{dx}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right) \\
\Phi(x) &= \int_{-\infty}^{x} Dz = \frac{1}{2}\left[1 + \operatorname{erf}\left(x/\sqrt{2}\right)\right] \\
\Phi^{-1}(x) &= 1/\Phi(x) \\
\Phi^{(-1)}(x) & \quad \text{inverse } \Phi\text{-function}
\end{aligned}
$$

Bibliography

[Ama67] S. Amari. A Theory of Adaptive Pattern Classifiers. *IEEE Trans. Elect. Comput., EC-16*, 299, 1967.

[BR94] M. Biehl and P. Riegler. On-line Learning with a Perceptron. *Europhys. Lett.* **28**, 525, 1994.

[BRS95] M. Biehl, P. Riegler, and M. Stechert. Learning from Noisy Data: An Exactly Solvable Model. *Phys. Rev. E* **52**, R4624, 1995.

[BRW] M. Biehl, P. Riegler, and C. Wöhler. In preparation.

[BRW96] M. Biehl, P. Riegler, and C. Wöhler. Transient dynamics of on-line learning in two-layered neural networks. *J. Phys. A* **29**, 4769, 1996.

[BS93] M. Biehl and H. Schwarze. Learning Drifting Concepts with Neural Networks. *J. Phys. A* **26**, 2651, 1993.

[BS95] M. Biehl and H. Schwarze. Learning by on-line gradient descent. *J. Phys. A* **28**, 643, 1995.

[BSS95] N. Barkai, H.S. Seung, and H. Sompolinsky. On-line Learning of Dichotomies. In G. Tesauro, D. Touretzky, and T.K. Leen, editors, *Advances in Neural Information Processing Systems 7*, Morgan Kaufmann, San Francisco, 1995.

[CC95] M. Copelli and N. Caticha. On-line learning in the committee machine. *J. Phys. A* **28**, 1615, 1995.

[CEK+97] M. Copelli, R. Eichhorn, O. Kinouchi, M. Biehl, R. Simonetti, P. Riegler, and N. Caticha. Noise Robustness in Multilayer Neural Networks. *Europhys. Lett.* **37**, 432, 1997.

[CKC96] M. Copelli, O. Kinouchi, and N. Caticha. Equivalence between learning in noisy perceptrons and tree committee machines. *Phys. Rev. E* **53**, 6341, 1996.

[Cop95] M. Copelli. Determinação variacional de algoritmos de aprendizagem em redes neurais. Master's thesis, Universidade de São Paulo, 1995.

[CR95] Y. Chauvin and D.E. Rumelhart, editors. *Backpropagation: Theory, Architectures, and Applications*. Erlbaum, Hillsdale, NJ, 1995.

[Cyb89] G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Math. Control Signals Systems* **2**, 303, 1989.

[Gar90] C.W. Gardiner. *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*. Springer, Berlin, 1990.

[Gol80] H. Goldstein. *Classical Mechanics*. Addison-Wesley, Reading, MA, second edition, 1980.

[Grä97] J. Gräwe. Lernen anhand verrauschter Beispieldaten. Diploma thesis, Universität Würzburg, 1997.

[GT90] G. Györgyi and N. Tishby. Statistical Theory of Learning a Rule. In W.K. Theumann and R. Köberle, editors, *Neural Networks and Spin Glasses*, World Scientific, Singapore, 1990.

[Hes94] T. Heskes. The use of being stubborn and introspective. In J. Dean, H. Cruse, and H. Ritter, editors, *Proceedings of the ZiF Conference on Adaptive Behavior and Learning*, Universität Bielefeld. 1994.

[HK91] T. Heskes and B. Kappen. Learning processes in neural networks. *Phys. Rev. A* **44**, 2718, 1991.

[HK93] T. Heskes and B. Kappen. On-line learning processes in artificial neural networks. In *Mathematical Foundations of Neural Networks*. Elsevier, Amsterdam, 1993.

[HKP91] J.A. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, CA, 1991.

[Hop92] J.J. Hopfield. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proc. Natl. Acad. Sci. USA* **79**, 2554, 1992.

[KC92a] O. Kinouchi and N. Caticha. Biased learning in Boolean perceptrons. *Physica A* **185**, 411, 1992.

[KC92b] O. Kinouchi and N. Caticha. Optimal generalization in perceptrons. *J. Phys. A* **25**, 6243, 1992.

[KC93] O. Kinouchi and N. Caticha. Lower bounds on generalization errors for drifting rules. *J. Phys. A* **26**, 6161, 1993.

[KC95] O. Kinouchi and N. Caticha. On-line versus off-line learning in the linear perceptron: A comparative study. *Phys. Rev.* **E 52**, 2878, 1995.

[KR90] W. Kinzel and P. Ruján. Improving a Network Generalization Ability by Selecting Examples. *Europhys. Lett.* **13**, 473, 1990.

[KS96] J.W. Kim and H. Sompolinsky. On-line Gibbs Learning. *Phys. Rev. Lett.* **76**, 3021, 1996.

[MPV87] M. Mézard, G. Parisi, and M. Virasoro. *Spin Glas Theory and Beyond*. World Scientific, Singapore, 1987.

[OH91a] M. Opper and D. Haussler. Calculation of the Learning Curve of Bayes Optimal Classification Algorithm for Learning a Perceptron with Noise. In L.G. Valiant and M.K. Warmuth, editors, *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, Morgan Kaufmann, San Mateo, CA, 1991.

[OH91b] M. Opper and D. Haussler. Generalization Performance of Bayes Optimal Classification Algorithm for Learning a Perceptron. *Phys. Rev. Lett.* **66**, 2677, 1991.

[OK96] M. Opper and W. Kinzel. Statistical Mechanics of Generalization. In E. Domany, J.L. van Hemmen, and K. Schulten, editors, *Physics of Neural Networks III*. Springer, New York, 1996.

[Opp] M. Opper. Private communication.

[Opp94] M. Opper. Learning and generalization in a two-layer neural network: the role of the Vapnik-Chervonenkis dimension. *Phys. Rev. Lett.* **72**, 2113, 1994.

[Opp96] M. Opper. Online versus Offline Learning from Random Examples: General Results. *Phys. Rev. Lett.* **77**, 4671, 1996.

[RB95] P. Riegler and M. Biehl. On-line backpropagation in two-layered neural networks. *J. Phys. A* **28**, L507, 1995.

[RBSM96] P. Riegler, M. Biehl, S.A. Solla, and C. Marangi. On-line learning from clustered input examples. In M. Marinaro and R. Tagliaferri, editors, *NEURAL NETS WIRN VIETRI-95, Proceedings of the 7th Italian Workshop on Neural Nets*, Word Scientific, Singapore, 1996.

[Rie] P. Riegler. In preparation.

[Rie94] P. Riegler. Verallgemeinern in neuronalen Netzen beim Einschritt–Lernen. Diploma thesis, Universität Würzburg, 1994.

[Sch93] H. Schwarze. Learning a rule in a multilayer neural network. *J. Phys. A* **26**, 5781, 1993.

[SS95a] D. Saad and S.A. Solla. Exact solution for on-line learning in multilayer neural networks. *Phys. Rev. Lett.* **74**, 4337, 1995.

[SS95b] D. Saad and S.A. Solla. On-line learning in soft committee machines. *Phys. Rev. E* **52**, 4225, 1995.

[SST92] H.S. Seung, H. Sompolinsky, and N. Tishby. Statistical Mechanics of Learning from Examples. *Phys. Rev. A* **45**, 6056, 1992.

[Ste95] M. Stechert. Lernen mit einem Perceptron bei verrauschten Trainingsmustern. Diploma thesis, Universität Würzburg, 1995.

[Val89] F. Vallet. The Hebb Rule for Learning Linearly Separable Boolean Functions: Learning and Generalization. *Europhys. Lett.* **8**, 747, 1989.

[vK92] N.G. van Kampen. *Stochastic processes in physics and chemistry*. Elsevier, Amsterdam, 1992.

[WH94] W. Wiegerinck and T. Heskes. On-Line Learning with Time-Correlated Patterns. *Europhys. Lett.* **28**, 451, 1994.

[Wöh96] C. Wöhler. Plateauzustände beim Einschritt-Lernen in neuronalen Netzwerken. Diploma thesis, Universität Würzburg, 1996.

[WRB93] T.L.H. Watkin, A. Rau, and M. Biehl. The Statistical Mechanics of Learning a Rule. *Rev. Mod. Phys.* **65**, 499, 1993.

Acknowledgements

Numerous people have contributed to this work, through discussions or just by listening to my thoughts. There are too many to mention all of them by name, but each of them can be sure of my gratitude.

My special thanks are to my colleagues in Prof. Dr. Wolfgang Kinzel's group at Würzburg University. I truly have enjoyed working there during the past three years. The scientific as well as cultural atmosphere has provided a stimulating environment. In particular, I owe a debt of gratitude to Prof. Dr. Wolfgang Kinzel who encouraged, supported, and supervised this work. Also, it has always been delightful to discuss physics with him.

More than anybody else Dr. Michael Biehl was engaged in this work. The blackboard in his office was the place where many an idea arose while an innumerable number of crayons passed away. The other blackboard which needs to be mentioned is the one of Dr. Georg Reents. His attitude to question many a detail was of invaluable help.

I also would like to acknowledge fruitful and stimulating discussions with a number of scientists outside Würzburg, especially with Prof. Dr. Sara A. Solla, Prof. Dr. Nestor Caticha, Mauro Copelli, and Dr. Manfred Opper. I would like to thank Dr. Sebastian Seung for the possibility to conduct research at Bell Laboratories, Lucent Technologies. Without doubt our discussions about machine learning have influenced this work. My special thanks are to Prof. Dr. Krzysztof Wódkiewicz who introduced me to scientific research and who has been a great advisor ever since.

Last but not least, I would like to express my thanks to my parents and to Elke for their support and encouragement.