5

Statistical Mechanics of Generalization

Manfred Opper and Wolfgang Kinzel[1]

with 18 figures

Synopsis. We estimate a neural network's ability to generalize from examples using ideas from statistical mechanics. We discuss the connection between this approach and other powerful concepts from mathematical statistics, computer science, and information theory that are useful in explaining the performance of such machines. For the simplest network, the perceptron, we introduce a variety of learning problems that can be treated exactly by the replica method of statistical physics.

5.1 Introduction

Neural networks learn from examples. This statement is obviously true for the brain, but artificial networks also adapt their "synaptic" weights to a set of examples. After the learning phase, the system has adopted some ability to generalize; it can make predictions on inputs which it has not seen before; it has learned a rule.

To what extent is it possible to understand learning from examples by mathematical models and their solutions? It is this question that we emphasize in this chapter. We introduce simple models and discuss their properties combining methods from statistical mechanics, computer science, and mathematics.

The simplest model for a neural network is the *perceptron*. It maps an N–dimensional input vector $\boldsymbol{\xi}$ to a binary variable $\sigma \in \{+1, -1\}$, and the function is given by an N–dimensional weight vector $\mathbf{w}$:

$$\sigma = \text{sign}\ (\mathbf{w} \cdot \boldsymbol{\xi})\,. \tag{5.1}$$

Motivated by real neurons, the components of $\mathbf{w}$ may be called *synaptic* weights; i. e., $w(i)$ is a measure of the strength of the influence of the neuron signal $\xi(i)$ to the output neuron σ.

[1]Physikalisches Institut, Universität Würzburg, D-97074 Würzburg, Germany.

For a given $\mathbf{w}$ this function separates the input space by a hyperplane into two parts, $\mathbf{w} \cdot \boldsymbol{\xi} > 0$ and $\mathbf{w} \cdot \boldsymbol{\xi} < 0$, and the hyperplane is normal to $\mathbf{w}$. But also for a given input $\boldsymbol{\xi}$, the space of weights $\mathbf{w}$ is divided into two parts with different outputs σ. Equation (5.1) gives a very limited class of all possible functions from R^N to ± 1. But this limitation is necessary for a good generalization, as we shall show later.

In the simplest case, the perceptron operates in two ways: in a learning and in a generalization phase. In the learning process, the network receives a set of $P = \alpha N$ many examples, i.e., input/output pairs $(\sigma_k, \boldsymbol{\xi}_k)$, $k = 1, ..., \alpha N$, which were generated by some unknown function $\sigma_k = F(\boldsymbol{\xi}_k)$. The weight vector $\mathbf{w}$ is adapted to these examples by some learning algorithm, i.e., the strengths of the synapses are changed when one or more examples are shown to the perceptron. Of course, the aim of learning is to map each pair correctly by Eq. (5.1), and the number of examples for which the network disagrees with the shown output, $\sigma_k \neq \text{sign}\ (\mathbf{w} \cdot \boldsymbol{\xi}_k)$, is the training error E:

$$E = \sum_{k=1}^{\alpha N} \theta\left(-\sigma_k\ \mathbf{w} \cdot \boldsymbol{\xi}_k\right). \tag{5.2}$$

θ is the step function, $\theta(x) = (\text{sign}\ x + 1)/2$. If the examples are generated by another perceptron with weights $\mathbf{w}_t$, then it is possible to obtain zero training error, $\varepsilon_t = 0$, for instance, by using the perceptron learning rule (see [1]).

After the learning phase, the perceptron has achieved some knowledge about the rule by which the examples were produced. Therefore, the network can make predictions on a new input vector $\boldsymbol{\xi}$ that it has not learned before. Let $(\sigma, \boldsymbol{\xi})$ be a new example that the network has *not* seen before. Then the probability that the perceptron gives the wrong answer, $\sigma \neq \text{sign}\ (\mathbf{w} \cdot \boldsymbol{\xi})$, is given by

$$\varepsilon = \overline{\theta\left(-\sigma\ \mathbf{w} \cdot \boldsymbol{\xi}\right)}, \tag{5.3}$$

where the bar means an average over all possible examples $(\sigma, \boldsymbol{\xi})$.

The calculation of the generalization error ε as a function of the fraction α of the learned examples is the main subject of this chapter. We call the learning network *student* and the example producing rule the *teacher*. Hence, ε is the probability of disagreement between student and teacher on a new input $\boldsymbol{\xi}$. $\varepsilon(\alpha)$ depends on the structure of student and teacher, on the structure of the examples, and on the learning algorithm.

From very general concepts one obtains bounds and relations between different generalization errors. Using methods of statistical mechanics developed from the theory of disordered solids (spin glasses), one obtains exact results on $\varepsilon(\alpha)$ for infinitely large networks ($N \to \infty$). Section 5.2 introduces general results, while the statistical mechanics approach is presented

in Sec. 5.3. Section 5.4 discusses scaling ideas, from which the asymptotic behavior of the generalization error can be understood in some cases. A variety of applications for perceptrons are reviewed in Sec. 5.5.

This chapter is not supposed to review the new field of generalization using neural networks. (For a review we recommend the article by Watkin, Rau, and Biehl [2].) But we want to give an introduction to the field with an emphasis on general results and on applications of our own group at Würzburg. We apologize for not referring to a large number of interesting and important results of our colleagues and friends.

5.2 General Results

The theory of learning in neural networks has benefitted from an interplay of ideas that come from various scientific fields; these include *computer science, mathematical statistics, information theory*, and *statistical physics.* In the following, we try to present some of these ideas. We review a variety of general results that can be obtained *without specifying a network architecture.*

5.2.1 Phase Space of Neural Networks

In this section we adress the problem of *noise-free learning* in networks with binary outputs. We assume that an ideal teacher network, with a vector of parameters $\mathbf{w}_t$, exists, who will give answers ($\oplus$ or $\ominus$) on[2] input vectors $\boldsymbol{\xi}$ without making mistakes.

Let us now look at the *phase space of all teachers* $\mathbf{w}_t$, described by a parameter vector $\mathbf{w}_t$, for fixed inputs $\boldsymbol{\xi}_1, \ldots \boldsymbol{\xi}_P$. *Before* knowing the teacher's correct answers to all of these inputs, a learner could partition the phase space into maximally 2^P cells or subvolumes, each cell σ corresponding to one of the 2^P possible labelings (= answers) $\sigma_k = \pm 1,\ k = 1, \ldots, P$. In general, a given type of neural network will not be able to produce all 2^P outputs on the given inputs. If the teacher network has a very complex architecture, we can assume that, by suitable choices of its parameters, more combinations of outputs, in other words, more cells in phase space, can be realized than for a less complex teacher. As we shall see in the next section, this number of cells plays an important role for the learner's ability to understand the teacher's problem.

After the teacher has given the answers, we know to which cell $\mathbf{w}_t$ belongs. In the so-called *consistent* learning algorithms, one trains a student network to respond *perfectly* to the P training inputs. In the following,

[2]In general, we do not assume that the dimensions of parameter space and input space are equal.

we assume that the student belongs to the *same class of networks as the teacher*. Thus, after learning, the student has parameters $\mathbf{w}_s$, which belong to the teacher's cell.

Will the probability of making a mistake on unknown inputs always become small when P grows large, whatever consistent algorithm we choose?

Surprisingly, the answer is yes, if the teacher has a *bounded complexity*. As a measure for this complexity, the so-called *Vapnik–Chervonenkis (VC)* dimension, which comes from mathematical statistics, has been introduced into computer science. We will try to review some of its basic ideas in the next section.

5.2.2 VC Dimension and Worst-Case Results

The *maximal* number of cells in the teacher's space is 2^P for P input vectors. But, due to the teacher's internal structure, the actual number of cells for a set of inputs may not grow exponentially fast in P. A combinatorial theorem, independently proved by Sauer [3] and Vapnik and Chervonenkis [4], gives an upper bound on this number: If d is the size of the *largest set* of inputs realizing *all* 2^d cells, then, for *any set* of $P > d$ inputs, the number $\mathcal{N}(P,d)$ of cells will only grow like a *polynomial* in P. d is called the *VC dimension*.

Formally, *Sauer's lemma* states:
$P \geq d \geq 1$:

$$\mathcal{N}(P,d) \leq \sum_{i=0}^{d} \binom{P}{i} \leq \left(\frac{eP}{d}\right)^d . \tag{5.4}$$

A sketch of the proof of Eq. (5.4) is given in Appendix 5.1. Equation (5.4) shows, that the VC–dimension plays a similar role as the *capacity* of the class of teacher networks. For $P >> d$, only an exponentially small fraction of input–output pairs can be stored in the net. For the perceptron, one has exactly $d = N$, the number of couplings. For general feedforward networks with N couplings and M threshold nodes, the bound $d \leq 2N \cdot \log_2(eM)$ was found in [5].

Using Sauer's lemma, Blumer, Ehrenfeucht, Haussler, and Warmuth [6] showed a worst-case result for the performance of *consistent* algorithms. To understand their result, consider the following learning scenario: After a student has learned a number of P independent random examples perfectly, he or she makes a prediction on an unknown input vector $\boldsymbol{\xi}$, which was drawn from the same distribution as the training examples. The student's probability of making a mistake on the random input $\boldsymbol{\xi}$ defines the generalization error ε. Different students (algorithms) will have different ε. In general, their performance will depend on the random training set, which makes ε a random variable. So we can define the probability $p(\varepsilon)$, that there exists a student, who learns the examples perfectly but makes

an error *larger than* ε. In [6] it was shown that, for $P > 8/\varepsilon$,

$$p(\varepsilon) \leq \mathcal{N}(2P, d) \cdot 2^{-\varepsilon P/2} \leq 2 \left(\frac{2eP}{d}\right)^d 2^{-\varepsilon P/2}. \tag{5.5}$$

Statistical physicists often discuss the *thermodynamic limit* $d, P \to \infty$, $\alpha = P/d$ fixed. In this limit, Eq. (5.5) means that *no errors* larger than

$$\varepsilon_{max} = \frac{2\ln(2e\alpha)}{\alpha}$$

will occur, whatever consistent student we choose.

Due to lack of space, we cannot sketch the proof of Eq. (5.5) here. A simpler theorem, which relates errors and the number of cells within the Bayesian framework of learning, will be proved in Sec. 5.2.4.

The power of the VC method lies in the fact that no specific assumption on the distribution of inputs, other than their independence, must be made. Further, the architecture of the teacher problem to be learned is characterized by *only a single number*, the VC dimension.

As a drawback of the worst-case results, one often finds that "typically," the error bounds are too pessimistic. In the next two sections, we will discuss a more optimistic learning scenario. We show what is gained if, besides the teacher's complexity, more prior knowledge, expressed by a probability distribution on the teacher's parameters, is available.

5.2.3 Bayesian Approach and Statistical Mechanics

The statistical mechanics approach to learning is closely related to concepts established in mathematical statistics and information theory [7, 8, 9, 10, 11]. To explain these connections, let us first briefly remind the reader of some ideas from *density estimation* in mathematical statistics.

A common problem in statistics is to infer a probability density, $\mathcal{P}_\theta(y)$, from a sample of P data values, $y^P \equiv y_1, \ldots, y_P$, independently drawn from this distribution. Here we assume that the class of distributions is known up to an unknown parameter θ. For example, assume $\mathcal{P}_\theta(y) = (2\pi)^{-1/2} \cdot e^{-(1/2)(y-\theta)^2}$, i.e., a Gaussian density, where θ, its mean, is unknown.

One approach to this problem is to estimate the value of θ first and then to approximate the unknown density by

$$\mathcal{P}_{\hat{\theta}}(y),$$

where $\hat{\theta}$ is the estimate. A well-known method is the *maximum likelihood estimation*, which uses a θ that makes the observed data most likely, i.e., which maximizes the *likelihood*

$$\prod_{k=1}^{P} \mathcal{P}_\theta(y_k). \tag{5.6}$$

For the Gaussian density, this leads to the simple arithmetic mean

$$\hat{\theta} = P^{-1} \sum_{k=1}^{P} y_k.$$

In the so-called *Bayesian approach* to density estimation, all prior knowledge (or lack of the same) of the unknown parameter is expressed by a prior distribution $p(\theta)$. For example, if the (Bayesian) statistician knows that the unknown mean of the Gaussian will not be too large or too small, say θ must be between -1 and $+1$, he or she could assume that θ is uniformly distributed in this interval. Rather than giving a single estimate of θ, as in the maximum likelihood case, the Bayesian calculates the *posterior distribution* $p(\theta|y^P)$, which represents his or her knowledge or uncertainty of the parameter after having observed the data values. This is derived by the *Bayes Formula*, which expresses the joint density $\mathcal{P}(y^P, \theta)$ of the data and the parameter in two ways using conditional densities:

$$\begin{aligned} \mathcal{P}(y^P, \theta) &= p(\theta) \cdot \prod_{k=1}^{P} \mathcal{P}_\theta(y_k) \\ \mathcal{P}(y^P, \theta) &= p(\theta|y^P) \cdot \mathcal{P}(y^P). \end{aligned} \tag{5.7}$$

The posterior density is then

$$p(\theta|y^P) = \frac{p(\theta) \cdot \prod_{k=1}^{P} \mathcal{P}_\theta(y_k)}{\mathcal{P}(y^P)}, \tag{5.8}$$

with the normalization

$$\mathcal{P}(y^P) = \int d\theta \prod_{k=1}^{P} \mathcal{P}_\theta(y_k) \cdot p(\theta). \tag{5.9}$$

Note that, for $p(\theta) = const$, the maximum of Eq. (5.8) is just the maximum likelihood estimate.

Then, if the Bayesian is asked to present an estimate of the unknown density, he or she will return the posterior averaged density

$$\mathcal{P}_{\theta,Bayes}(y) = \int d\theta \, p(\theta|y^P) \, \mathcal{P}_\theta(y), \tag{5.10}$$

which in general will not belong to the class of densities originally considered. Besides the most likely value of θ, this estimate includes neighboring values as well.

It can be shown that, if the parameter is actually distributed according to $p(\theta)$, then Eq. (5.10) gives the best approximation to the true density on average [4].

The justification of a prior probability for θ often has been questioned. Even if it is not satisfied, the posterior density Eq. (5.8) will, under some mild conditions, be highly peaked around the true value of θ for $P \to \infty$. The dependence on the actual shape of $p(\theta)$ will disappear asymptotically.

Let us now translate these ideas into the language of supervised learning. The data observed in a learning experiment are the examples consisting of P input–output pairs $\sigma^P \equiv \{(\sigma_1, \boldsymbol{\xi}_1), \ldots, (\sigma_P, \boldsymbol{\xi}_P)\}$. In general, we assume that there is a possibly stochastic relation between the inputs and the outputs, which can be expressed by a relation of the type

$$\sigma = F(\mathbf{w}_t, \boldsymbol{\xi}, \text{"noise"}). \tag{5.11}$$

$\mathbf{w}_t$ is a parameter vector representing an ideal classifier or teacher. In contrast to the previous section, we include the possibility that the observations may contain errors ("noise").

Using a neural network, which can implement functions of the type F (with "noise"=0!), the task of the learner is to find a student vector $\mathbf{w}_s$ that best explains the observed data. This can be understood as an estimation of the parameter $\mathbf{w}$ for the distribution

$$\mathcal{P}_{\mathbf{w}}(\sigma, \boldsymbol{\xi}) = \mathcal{P}_{\mathbf{w}}(\sigma|\boldsymbol{\xi}) \cdot f(\boldsymbol{\xi}), \tag{5.12}$$

where f is the density of the inputs and $\mathcal{P}_{\mathbf{w}}(\sigma|\boldsymbol{\xi})$ is the probability that, given an input $\boldsymbol{\xi}$, one observes an output σ.

The *statistical physics* approach to learning is closely related to the *Bayesian idea.* Based on the pioneering work of Gardner [12], one may study *ensembles* of neural networks to capture a "typical" behavior of their learning abilities. Such ensembles are defined by a Gibbs distribution,

$$p(\mathbf{w}|\sigma^P) = Z^{-1} \cdot p(\mathbf{w}) \cdot \exp\left(-\beta \sum_{k=1}^{P} E(\mathbf{w}; \sigma_k, \boldsymbol{\xi}_k)\right), \tag{5.13}$$

with partition function

$$Z = \int d\mathbf{w} \cdot p(\mathbf{w}) \cdot \exp\left(-\beta \sum_{k=1}^{P} E(\mathbf{w}; \sigma_k, \boldsymbol{\xi}_k)\right). \tag{5.14}$$

E is the *training energy* of the kth example and β^{-1} is the learning temperature in a stochastic learning algorithm. In $p(\mathbf{w})$, all constraints on the possible couplings are summarized.

Equation (5.13) has an interpretation as the *posterior distribution* [Eq. (5.8)] of coupling parameters if we identify

$$\begin{array}{rcc} p(\theta) \to & & p(\mathbf{w}) \\ \mathcal{P}_\theta(y^P) \to & \mathcal{P}_{\mathbf{w}}(\sigma^P) \propto \prod_{k=1}^{P} \exp\left(-\beta E(\mathbf{w}; \sigma_k, \boldsymbol{\xi}_k)\right) & \\ \mathcal{P}(y^P) \to & & \mathcal{P}(\sigma^P) \propto Z. \end{array} \tag{5.15}$$

As an example, let us consider a perceptron. We assume that the ideal classification $\sigma = \text{sign}(N^{-1/2}\mathbf{w}_t \cdot \xi)$ is inverted by *output noise*, i.e., $\sigma = \eta \cdot \text{sign}(N^{-1/2}\mathbf{w}_t \cdot \xi)$, where $\eta = -1$ with a probability $e^{-\beta}/1+e^{-\beta}$, and β^{-1} is the noise temperature. Fixing the inputs, the probability of observing P output labels is

$$\mathcal{P}_{\mathbf{w}}(\sigma^P) = \prod_{k=1}^{P} \left\{ \frac{\Theta(\sigma_k N^{-1/2}\mathbf{w} \cdot \boldsymbol{\xi}_k)}{1+e^{-\beta}} + \frac{e^{-\beta}\Theta(-\sigma_k N^{-1/2}\mathbf{w} \cdot \boldsymbol{\xi}_k)}{1+e^{-\beta}} \right\}$$

$$= (1+e^{-\beta})^{-P} \cdot \exp\left[-\beta \sum_{k=1}^{P} \Theta(-\sigma_k N^{-1/2}\mathbf{w} \cdot \boldsymbol{\xi}_k) \right]. \quad (5.16)$$

A second possibility of misclassification arises when the coupling parameters, or network *weights*, of the teacher are uncertain to some degree, i.e., $\mathbf{w}_t$ is replaced by $\mathbf{w}_t + \mathbf{v}$, where $\mathbf{v}$ is Gaussian with 0 mean and $\mathbf{v} \cdot \mathbf{v} = \beta^{-1}$. Now,

$$\mathcal{P}_{\mathbf{w}}(\sigma^P) = \prod_{k=1}^{P} H(-\beta^{1/2}\sigma_k N^{-1/2}\mathbf{w} \cdot \boldsymbol{\xi}_k), \quad (5.17)$$

where

$$H(x) = \int_x^\infty Dt$$

and

$$Dt = dt \cdot (2\pi)^{-\frac{1}{2}} \cdot \exp(-\tfrac{1}{2}t^2)$$

is the Gaussian measure.

To summarize, we obtain for the training energies

$$E(\mathbf{w}; \sigma, \boldsymbol{\xi}) = \begin{cases} \Theta(-\sigma N^{1/2}\mathbf{w} \cdot \boldsymbol{\xi}) & \text{for output noise} \\ -\beta^{-1} \ln\left(H(-\beta^{1/2}\sigma N^{-1/2}\mathbf{w} \cdot \boldsymbol{\xi})\right) & \text{for weight noise.} \end{cases} \quad (5.18)$$

The case of output noise also can be formulated for general neural networks and leads to a total training energy that is just the number of inputs, for which the noisy outputs σ and the student's answer disagree.

From the Bayesian viewpoint, the posterior distribution could be used to make predictions on new inputs $\boldsymbol{\xi}$ by calculating the output with the largest posterior probability. This is the *Bayes algorithm*, which, for binary outputs [cf. Eq. (5.10)], answers

$$\sigma = \text{sign}\left[\int d\mathbf{w}\, p(\mathbf{w}|\sigma^P) F(\mathbf{w}, \boldsymbol{\xi}, \text{"noise"} = 0) \right]. \quad (5.19)$$

Unfortunately, this represents a superposition of many neural networks, each given by a coupling vector $\mathbf{w}$. In general, this output cannot be realized by a single network of the same architecture, but requires a more complicated machine.

An algorithm that also uses the entire posterior density is the *Gibbs algorithm* [9, 10, 13], which draws a *a single* vector $\mathbf{w}$ *at random* according to the posterior Eq. (5.13). This is precisely what we would call the "typical" neural network in statistical physics.

This should be contrasted with a *maximum likelihood strategy*, which simply chooses a student vector [for $p(\mathrm{w}) = const$] that minimizes the training energy E. In the case of noisy outputs, the student would try to learn perfectly as many examples as it can, even if a fraction of them contains wrong classifications.

In general, perfect knowledge of the prior distribution of teachers will not be available. Nevertheless, the Gibbs distribution [Eq. (5.13)] is a natural device for defining learning algorithms, even if they are not optimally matched to the learning problem. We will discuss some examples in the section on perceptrons.

5.2.4 Information-Theoretic Results

In this section we explore in more detail what would happen if the Bayesian assumption was perfectly realized. That is, we assume that "nature" actually selects teacher problems at random, and that their prior distribution is completely known by the student.[3]

Learning more and more examples, the student's knowledge of the unknown teacher parameters grows. This knowledge gained by learning a new example is expressed in the so-called *information gain*. As was shown by Haussler, Kearns, and Shapire [13], this quantity can be related to the average error made by a student using the Gibbs and Bayes algorithms. Finally, using information theory and the VC approach, inequalities for errors can be derived. We restrict ourselves to the case of noise-free learning. A more general treatment can be found in [14].

We assume for this section that the inputs are fixed, so that the only randomness is in the choice of the teacher, and, for the Gibbs algorithm, in the choice of the student.

Having observed P classified inputs, we know that the teacher is constrained to one of the $\mathcal{N}(P, d)$ nonempty cells. Thus, the posterior density for the teacher's parameters is 0 outside the cell (see Fig. 5.1 for a perceptron) and equals

$$p(\mathbf{w})/V(\sigma^P) \tag{5.20}$$

inside the cell, where

[3]This a natural assumption for physics students, who, in preparing their exams, often use a catalog of the professor's questions from previous exams.

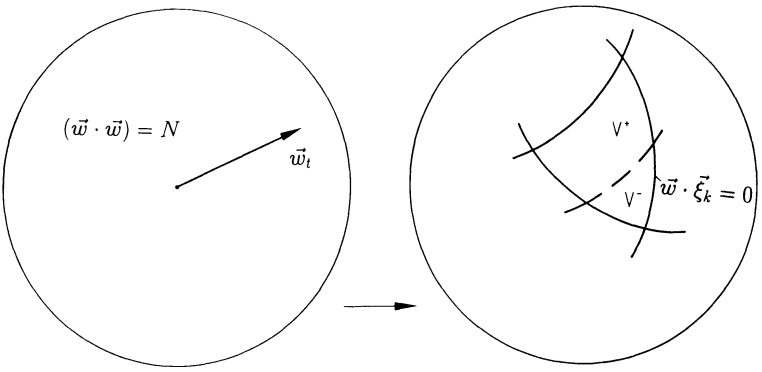


Fig. 5.1. Sketch of the phase space of weights for a perceptron. *Left:* Before learning, the vector $\mathbf{w}_t$ is completely unknown and assumed to be randomly distributed on the surface of an N-dimensional sphere. *Right:* After learning of P input–output examples $\boldsymbol{\xi}_k$, σ_k, the teacher $\mathbf{w}_t$ must be in a smaller cell of the phase space with boundaries given by the planes $\sigma_k \mathbf{w} \cdot \boldsymbol{\xi}_k = 0$, $k = 1, \ldots, P$. A new input (dashed line) divides the cell into two new subcells, V^+ and V^-, corresponding to the two possible answers.

$$V(\sigma^P) \equiv V_P = \int_{cell} p(\mathbf{w})\, d\mathbf{w} \tag{5.21}$$

is its (weighted) volume, satisfying $\sum_{\sigma_1 \ldots \sigma_P = \pm 1} V(\sigma^P) = 1$.

Let us begin with the Gibbs algorithm and fix the teacher for a moment. The learner chooses a vector $\mathbf{w}_s$ at random, with density [Eq. (5.20)]. If a new input is added, the cell is divided into two subcells (Fig. 5.1). If an output cannot be realized, we will formally assume a new cell with zero volume.

Let us compare the student's prediction on the new input with the teacher's answer. Both *agree only* if the student vector $\mathbf{w}_s$ is in the same cell as the teacher's. Averaging over $\mathbf{w}_s$, this will happen with probability

$$Y = \frac{V_{P+1}}{V_P}, \tag{5.22}$$

where V_{P+1} is the volume of the teacher's new cell. The probability of making a mistake thus is given by $1 - Y$.

The Bayesian prediction would weight the answers of the two subcells with their corresponding posterior probabilities and vote for the output

$$\sigma = \text{sign}[V^+ - V^-].$$

Thus, the answer of the largest cell wins. Since the Bayesian gives the answer with largest posterior probability, he or she will, on average, have

the lowest number of mistakes over all of the algorithms.[4] The Bayesian will only make a mistake if the teacher is in the smaller volume, i.e., if $Y < \frac{1}{2}$. To this algorithm we can assign the number

$$\Theta(1 - 2Y) \in \{0, 1\}, \tag{5.23}$$

which counts as a "1" when the algorithm makes a mistake.

Finally, by observing a new classified input, our uncertainty on the teacher's couplings will be reduced if the volume of the teacher's cell shrinks.[5] Formally, this corresponds to an *information gain*,

$$\Delta I = -[\ln(V_{P+1}) - \ln(V_P)] = -\ln(Y). \tag{5.24}$$

Obviously, Y, the volume ratio, is a random variable with respect to the random teacher and the inputs. Performing the average over the teacher only, simple and useful relations between the information gain and the probabilities of mistakes may be derived next.

Clearly, Y does not change if the teacher is moved inside a cell. Thus, we can average any function $F(Y)$ over the space of teachers, first by integrating over all teachers *inside* a cell, and then summing over all cells, labeled by their configuration σ^{P+1} of outputs:

$$\langle F(Y) \rangle = \sum_{\sigma_1 \ldots \sigma_{P+1} = \pm 1} V(\sigma^{P+1}) \cdot F(V(\sigma^{P+1})/V(\sigma^P)). \tag{5.25}$$

The factor $V(\sigma^{P+1})$ is the integral over the new cell [Eq. (5.21)]. Thus, outputs, which cannot be realized, are counted with zero weight.

We first will show the useful relation

$$\langle F(Y) \rangle = \langle YF(Y) + (1 - Y)F(1 - Y) \rangle. \tag{5.26}$$

Beginning with the right-hand side, and using the definition in Eq. (5.25), we fix the first P labels and sum over the σ_{P+1}. Let V^+ and V^- be the two possible subvolumes and $Y^+ = V^+/V(\sigma^P)$, $Y^- = 1 - Y^+$. The summation over σ_{p+1} gives a contribution

$$\begin{aligned} V^+[Y^+F(Y^+) + Y^-F(Y^-)] + V^-[Y^-F(Y^-) + Y^+F(Y^+)] = \\ V(\sigma^P)[Y^-F(Y^-) + Y^+F(Y^+)] = V^-F(Y^-) + V^+F(Y^+) \end{aligned} \tag{5.27}$$

---

[4] We always assume that the teacher actually was drawn from the assumed prior distribution.

[5] An interpretation of $-\ln(V_P)$ in terms of the *stochastic complexity* of Rissanen [15] has been discussed in [16]. Viewing the learning of the examples as an encoding of the outputs in the network's parameters $\mathbf{w}$, this quantity measures how many bits we need to describe the parameters if we use only a finite set of discrete values for the components of $\mathbf{w}$.

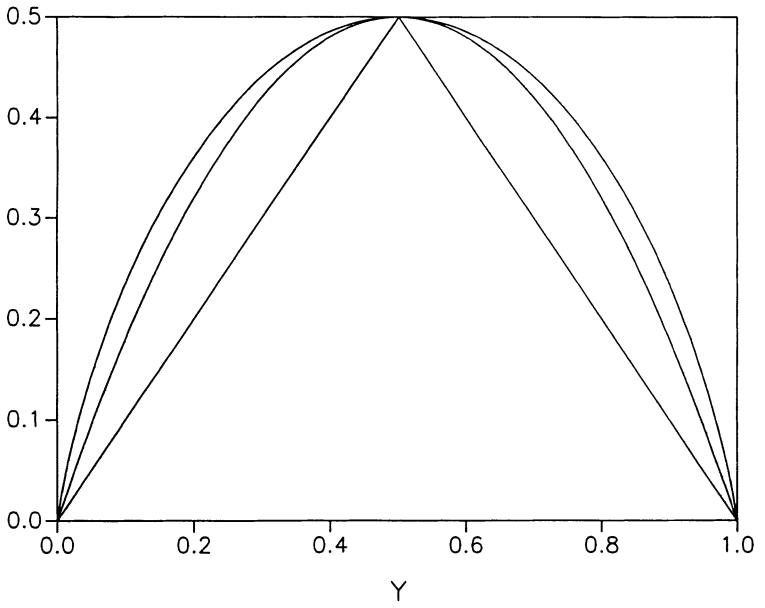


Fig. 5.2. Graphic demonstration of the inequalities $\min(Y, 1-Y)$ (*lower curve*) $\leq 2(Y-Y^2)$ (*middle curve*) $\leq -1/2\ln 2(Y\ln Y+(1-Y)\ln(1-Y))$ (*upper curve*).

to Eq. (5.26). Here we have used $V^+ + V^- = V(\sigma^P)$. Summing over the remaining labels, we obtain Eq. (5.26).

Using relation (5.26), the total probabilities of mistakes, in other words, the errors averaged over all teachers (but for fixed inputs), are given by

$$\begin{aligned} \varepsilon_{Gibbs} &= \langle 1-Y\rangle = 2\langle Y-Y^2\rangle \\ \varepsilon_{Bayes} &= \langle\Theta(1-2Y)\rangle = \langle\min(Y,1-Y)\rangle, \end{aligned} \tag{5.28}$$

where the first equality is from Eq. (5.22) and the second from Eq. (5.23). The average information gain is rewritten as

$$\langle\Delta I\rangle = -\langle\ln(Y)\rangle = -\langle Y\ln Y + (1-Y)\ln(1-Y)\rangle. \tag{5.29}$$

By comparing the three curves in Fig. 5.2, we conclude that

$$\begin{aligned} \varepsilon_{Gibbs} &\leq 2\varepsilon_{Bayes} \\ \varepsilon_{Gibbs} &\leq \frac{1}{2\ln(2)}\langle\Delta I\rangle. \end{aligned} \tag{5.30}$$

Although the random Gibbs algorithm is not optimal, its error is of the same order of magnitude as that of the optimal Bayes algorithm.

The second inequality (5.30) indicates that, in order to gain a lot of information on the teacher, a student should select inputs on which his or her

performance is bad. This can be utilized in the so-called query algorithms (see Sec. 5.5.4).

Using the VC method and Eq. (5.30), an estimate of the decrease of the generalization error for the Gibbs algorithm can be obtained [13].

Summing the second inequality (5.30) from $P = 0$ to $P = M - 1$, we can bound the average *cumulative* number of mistakes,

$$\sum_{P=0}^{M-1} \epsilon_{Gibbs}(P) \leq -\frac{1}{2\ln(2)} \sum_{\sigma_1 \ldots \sigma_M = \pm 1} V(\sigma^M) \ln(V(\sigma^M)), \tag{5.31}$$

where we have used the fact that the individual terms

$$\Delta I = -[\ln(V_{P+1}) - \ln(V_P)]$$

sum up to $-\ln(V_M) \equiv -\ln(V(\sigma^M))$. Since the volume of each cell equals its probability, the sum in the last expression equals the *entropy* of the distribution of outputs.

As is well known from information theory, the entropy is maximal if all probabilities are equal. In other words, this happens if the total unit volume of the phase space is equally divided under the $\mathcal{N}(M, d)$ cells. Thus we obtain the inequalities

$$\sum_{P=0}^{M-1} \epsilon_{Gibbs}(P) \leq \frac{1}{2\ln(2)} \ln\left(\mathcal{N}(M, d)\right) \leq \frac{1}{2\ln(2)} \cdot d(\ln(M/d) - 1). \tag{5.32}$$

The logarithmic growth in M indicates a faster decay of errors than the worst-case result $\epsilon(P) \simeq \ln(\alpha)/\alpha$, with $\alpha = P/d$. Rather, the estimate is consistent with a faster decay $\epsilon_{Gibbs}(P) \propto \alpha^{-1}$, asymptotically. In fact, using more refined techniques, it is shown in [13] that

$$\epsilon_{Gibbs}(P) \leq 2/\alpha. \tag{5.33}$$

Since this bound holds for arbitrary distributions of inputs, even very artificial ones, one might expect that, for "typical" distributions, learning might be even faster. Using the statistical mechanics approach, we will see, however, in the section on perceptrons, that the α^{-1} decay also holds for a natural distribution of inputs.

A greater speed of generalization only can be achieved if the asymptotic information gain from new new inputs can be enlarged. We will come back to this idea in Sec. 5.5.4.

5.2.5 Smooth Networks

Most parts of this chapter deal with networks that have binary outputs and the sign transfer function. Often in technical applications of neural nets, the transfer functions between in– and outputs are highly nonlinear, but they

nevertheless are *smooth* functions. This property is utilized in the so-called backpropagation algorithm [28], where a training energy is minimized via gradient descent. This requires the calculations of derivatives of the energy with respect to the coupling parameters.

It turns out that the asymptotic behavior of the generalization errors can be calculated easier than for binary outputs.

We assume a learning algorithm that is defined by the Gibbs ensemble

$$p(\mathbf{w}|\sigma^P) = Z^{-1} \cdot \exp\left(-\beta \sum_{k=1}^{P} E(\mathbf{w}; \sigma_k, \xi_k)\right). \tag{5.34}$$

We assume that σ_k is a function of the inputs and a teacher parameter vector. In the following we will not assume that the problem is completely learnable. Then the aim of a learner will be to find a network that minimizes the training energy *averaged over the space of all examples.* If P, the number of examples, grows large, we expect that the final state of the network converges to the optimal value $\mathbf{w}_0$, for which

$$\partial_i \, \overline{E(\mathbf{w}; \sigma, \xi)}_{\mathbf{w}=\mathbf{w}_0} = 0, \tag{5.35}$$

for all i. The bar denotes the average over the examples, and the derivative is with respect to the components $w(i)$.

The generalization error after learning P examples is

$$\varepsilon = \int d\mathbf{w} \; \overline{p(\mathbf{w}|\sigma^P) \; E(\mathbf{w}; \sigma, \xi)}. \tag{5.36}$$

We further expect that the posterior density is strongly peaked at its maximum $\hat{\mathbf{w}}$, the *maximum likelihood estimate.* The fluctuations around this value are, to the lowest order, Gaussian with zero mean and covariance:

$$\langle (w(i) - \hat{w}(i)) \; (w(j) - \hat{w}(j)) \rangle \simeq (\beta \; P)^{-1} \; (U^{-1})_{ij}, \tag{5.37}$$

where

$$U_{ij} = P^{-1} \; \partial_i \partial_j \; \sum_{k=1}^{P} E(\mathbf{w}; \sigma_k, \xi_k)_{\mathbf{w}=\hat{\mathbf{w}}} \simeq \partial_i \partial_j \; \overline{E(\mathbf{w}; \sigma, \xi)}. \tag{5.38}$$

Expanding Eq. (5.36) around ($\mathbf{w} = \hat{\mathbf{w}}$), and averaging over the Gaussian fluctuations in Eq. (5.37), we get

$$\varepsilon \simeq \overline{E(\hat{\mathbf{w}}; \sigma, \xi)} + \frac{1}{2} (\beta P)^{-1} \sum_{ij} U_{ij} (U^{-1})_{ij}. \tag{5.39}$$

The sum on the right-hand side simply equals N, the number of weights.

For P large, $\hat{\mathbf{w}}$ will be close to the optimum $\mathbf{w}^0$. To estimate the difference between $\hat{\mathbf{w}}$ and $\mathbf{w}^0$, we use the fact that $\mathbf{w} = \hat{\mathbf{w}}$ extremizes the learning energy, i.e., it fulfills

$$0 = P^{-1/2}\partial_i \sum_{k=1}^{P} E(\mathbf{w}; \sigma_k, \xi_k)_{\mathbf{w}=\hat{\mathbf{w}}} \simeq$$

$$\underbrace{P^{-1/2}\partial_i \sum_{k=1}^{P} E(\mathbf{w}; \sigma_k, \xi_k)_{\mathbf{w}=\mathbf{w}^0}}_{\gamma_i} + \sum_j U_{ij}\sqrt{P}(\hat{w}(j) - w(j)^0), \qquad (5.40)$$

where we have expanded to the first order at $\mathbf{w} = \mathbf{w}^0$. We also neglected the dependence of U_{ij} on $\mathbf{w}$. The first term, γ_i, is a sum of independent random variables and is, in the limit, Gaussian distributed. We find from Eq. (5.35) that $\overline{\gamma_i} = 0$ and

$$\overline{\gamma_i\, \gamma_j} \simeq \overline{\partial_i E \cdot \partial_j E} \equiv I_{ij}. \qquad (5.41)$$

Using this information, we can solve Eq. (5.40) to get

$$P\, \overline{(w(i) - w^0(i))\, (w(j) - w^0(j))} \simeq (U^{-1}\, I\, U^{-1})_{ij}. \qquad (5.42)$$

Finally, we expand the first term of Eq. (5.39) at $\mathbf{w}^0$ up to the second order in $(w(i) - w(i)^0)$; the first order clearly vanishes. Using Eq. (5.42), we get

$$\varepsilon \simeq \varepsilon_{min} + \frac{1}{2P} Tr(U^{-1}\, I) + \frac{N}{2\beta P}. \qquad (5.43)$$

ε_{min} is the minimal error achieved by the parameter $\mathbf{w}^0$.

This result has been shown in [17] using the replica method. In [18], a similar result has been proved using the analogy to density estimation in mathematical statistics. In this framework, the matrix I is proportional to the so-called *Fisher Information*, defined as

$$I_{ij} = \int dy\, \partial_i \ln(\mathcal{P}_\theta(y)) \cdot \partial_j \ln(\mathcal{P}_\theta(y)). \qquad (5.44)$$

Here we have used the terminology of Sec. 5.2.3 and we assumed that the parameter θ is a vector. I plays an important role in the asymptotics of statistical estimation procedures [4].

The result in Eq. (5.43) has the same $\propto P^{-1}$ behavior as the decay of the Gibbs errors in Eq. (5.33). It should be noted, however, that the definition (5.36) of the generalization error does not correspond to a binary classification problem like the ones treated in the previous sections. If we would force a smooth network to give "straight" answers $\oplus$ or $\ominus$, by clipping its outputs after training, the generalization error may be different. As we will see in Sec. 5.5.2, for the ADALINE algorithm, a slower performance $\varepsilon \propto 1/\sqrt{P}$ can be observed in such a case.

5.3 The Perceptron

5.3.1 Some General Properties

The perceptron shows many interesting features that distinguish it from other neural networks.

One of the oldest rigorous results for perceptrons is the number of possible output combinations or *cells*. Besides the estimate of Sauer's lemma, we know a precise result for the perceptron, given by Cover [19] in 1965: For any set of P inputs in general position,[6] one has *exactly*

$$\mathcal{N}(P,N) = 2\sum_{i=0}^{N-1}\binom{P-1}{i}, \tag{5.45}$$

where N is the number of weights. Equation (5.45) also yields $P = N$ for the largest number of input vectors with $\mathcal{N}(P,N) = 2^P$. Thus, the VC dimension equals N.

The independence of $\mathcal{N}(P,N)$ from the location of input vectors is no longer valid when we look at other networks. Perceptrons with binary weights already show large fluctuations for this quantity. Based on exact enumerations on small systems [20], but for many samples of random inputs, we have obtained lower bounds on the VC dimension d for this model. Finite-size scaling (see Fig. 5.3) indicates that for $N \to \infty$ we will have $d \simeq N/2$.

Another striking feature is the simple geometric picture (Fig. 5.4) of the perceptron's classification ability. In the space of the inputs, the vector of couplings defines a separating plane perpendicular to $\mathbf{w}$. Inputs on the side of this normal vector are classified as $\oplus$, while those on the other side are classified as $\ominus$. Perceptrons realize *linear-separable* functions.

As a consequence of this geometric picture, we can easily find the generalization error ε (= probability of making a mistake) when the inputs have a spherical distribution. Such a distribution can be realized from independent, normally distributed cartesian components $\xi(j)$, $j = 1,\ldots,N$, with density

$$f(\boldsymbol{\xi}) = (2\pi)^{-N/2}\exp(-\tfrac{1}{2}\boldsymbol{\xi}\cdot\boldsymbol{\xi}). \tag{5.46}$$

For fixed teacher and student, one finds

$$\varepsilon = \frac{1}{\pi}\arccos\left(\frac{\mathbf{w}_s\cdot\mathbf{w}_t}{|\mathbf{w}_s||\mathbf{w}_t|}\right). \tag{5.47}$$

Equation (5.47) will be used extensively in the following sections. Although this theorem can be derived by averaging over the Gaussian random variables, it is immediately clear from the geometric construction of Fig. 5.4.

---

[6] Any subset of inputs containing no more than N input vectors is linearly independent.

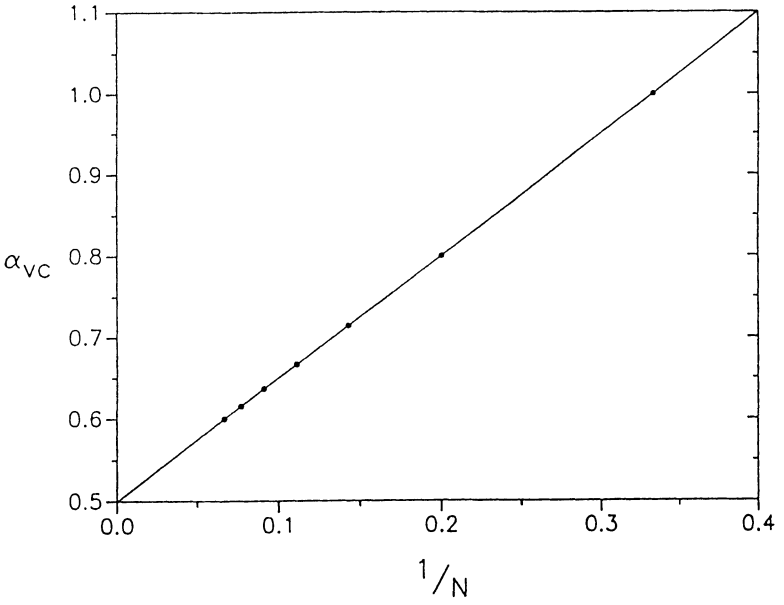


Fig. 5.3. VC dimension for the perceptron with binary weights. The curve gives a lower bound as a function of N, the number of weights. The data were obtained from large samples of input sets upon calculating the number of possible labelings by scanning all 2^N weight vectors.

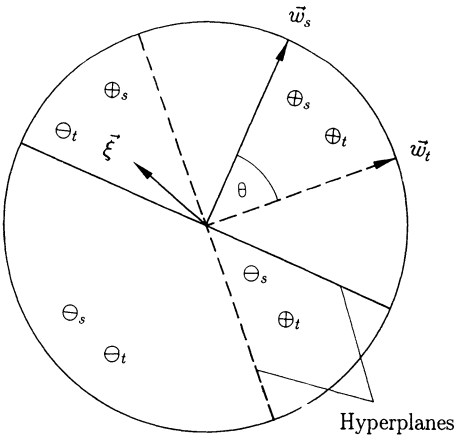


Fig. 5.4. For a perceptron, teacher $\mathbf{w}_t$ and student $\mathbf{w}_s$ determine separating planes in the input space. Inputs are mapped onto $\oplus$ if they are in the same half-space as $\mathbf{w}_{t,s}$. Thus, the generalization error equals the ratio $\varepsilon = \theta/\pi$ of area with different outputs and total areas.

Note that, for $N \to \infty$, any distribution with the same first two moments will give the above result. A popular choice is $\xi_k(j) = \pm 1$ with probability $\frac{1}{2}$.

5.3.2 Replica Theory

In this section we develop a general framework that will allow us to treat a variety of perceptron learning problems using the replica method.

Following Gardner's approach, we will consider a Gibbs ensemble of perceptrons defined by the distribution

$$p(\mathbf{w}_s|\sigma^P) = Z^{-1} \cdot p(\mathbf{w}_s) \cdot \exp\left(-\beta \sum_{k=1}^{P} E(\mathbf{w}_s; \sigma_k, \boldsymbol{\xi}_k)\right) \tag{5.48}$$

with partition function

$$Z = \int d\mathbf{w}\; p(\mathbf{w}) \cdot \exp\left(-\beta \sum_{k=1}^{P} E(\mathbf{w}; \sigma_k, \boldsymbol{\xi}_k)\right).$$

In the following we will keep the form of Eq. (5.48) rather general: We will only assume that E depends on the internal fields $N^{-1/2}\sigma_k \mathbf{w} \cdot \boldsymbol{\xi}_k$. Thus, we consider partition functions of the type

$$Z(\sigma^P) = \int d\mathbf{w}\; p(\mathbf{w}) \cdot \prod_{k=1}^{P} \Phi(N^{-1/2}\sigma_k \mathbf{w} \cdot \boldsymbol{\xi}_k), \tag{5.49}$$

with an arbitrary Φ.

We constrain the coupling vectors to the surface of a sphere, i.e.,

$$p(\mathbf{w}) = V_0^{-1} \delta(\mathbf{w} \cdot \mathbf{w} - q_0 N),$$

with $V_0 = e^{N/2(\ln 2\pi + 1)} \simeq \int \delta(\mathbf{w} \cdot \mathbf{w} - N) d\mathbf{w}$. Finally, $d\mathbf{w} = \prod_{j=1}^{N} dw(j)$ is the volume element in cartesian coordinates.

One of the basic assumptions of the statistical mechanics approach can be stated as follows: The free energy per coupling $\mathcal{F}$, defined by

$$\mathcal{F} = N^{-1} \ln Z(\sigma^P), \tag{5.50}$$

is a self-averaging quantity for $N \to \infty$ and most "natural" distributions of the random examples. This means that it equals its average

$$\mathcal{F} = N^{-1} \cdot \sum_{\sigma_1 \ldots \sigma_P = \pm 1} \overline{\mathcal{P}(\sigma^P) \ln Z(\sigma^P)} \tag{5.51}$$

for almost all realizations of the random examples. Here,

$$\mathcal{P}(\sigma^P) = \mathcal{P}(\sigma_1, \ldots, \sigma_P | \boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_P)$$

is the total probability over all teachers (and noise) that, *given the inputs*, the binary classifications σ^P will be observed. The bar denotes the average over the distributions of inputs. If Eq. (5.48) was actually the posterior distribution corresponding to a prior distribution of random teachers (see Sec. 5.2.3), we would have

$$\mathcal{P}(\sigma^P) = Z(\sigma^P)/\mathcal{C}, \tag{5.52}$$

where the normalization

$$\mathcal{C} = \sum_{\sigma_1 \ldots \sigma_P = \pm 1} Z(\sigma^P)$$

is assumed to be independent of the inputs. In general, we will not restrict ourselves to Eq. (5.52), but rather we will use the more general ansatz,

$$\mathcal{P}(\sigma^P) = Z_t(\sigma^P)/\mathcal{C}_t$$

$$Z_t(\sigma^P) = \int d\mathbf{w}_t \, p_t(\mathbf{w}) \cdot \prod_{k=1}^{P} \Phi_t(N^{-1/2}\sigma_k \mathbf{w}_t \cdot \boldsymbol{\xi}_k), \tag{5.53}$$

where Φ_t can be different from Φ and

$$p_t(\mathbf{w}) = V_0^{-1}\delta(\mathbf{w}_t \cdot \mathbf{w}_t - N).$$

To perform the average over the inputs, the *replica trick* is utilized:

$$\mathcal{F} = N^{-1} \cdot \sum_{\sigma_1 \ldots \sigma_P = \pm 1} \overline{\mathcal{P}(\sigma^P) \ln Z(\sigma^P)} = \lim_{n \to 0} \frac{\partial}{\partial n} N^{-1} \ln \Xi_n, \tag{5.54}$$

where

$$\Xi_n = \sum_{\sigma_1 \ldots \sigma_P = \pm 1} \overline{Z_t(\sigma^P) Z^n(\sigma^P)} \tag{5.55}$$

is the weighted and averaged n-times replicated partition function. Equation (5.55) will be calculated for integer n, and the result then will be continued to reals.

Since all inputs are assumed to be statistically independent and drawn from the same distribution, we get

$$\Xi_n = \int d\mathbf{w}_t \, p(\mathbf{w}_t) \prod_{a=1}^{n} d\mathbf{w}_a \, p(\mathbf{w}_a)$$

$$\times \left(\sum_{\sigma = \pm 1} \overline{\Phi_t(N^{-1/2}\sigma \mathbf{w}_t \cdot \boldsymbol{\xi}) \prod_{a=1}^{n} \Phi(N^{-1/2}\sigma \mathbf{w}_a \cdot \boldsymbol{\xi})} \right)^P . \tag{5.56}$$

For the inputs, we use the Gaussian distribution (5.46). Φ and Φ_t depend on $\boldsymbol{\xi}$ only via $u_a = N^{-1/2}\sigma \mathbf{w}_a \cdot \boldsymbol{\xi}$, $a = 1, \ldots, n$, and $u_{n+1} = u_t = N^{-1/2}\sigma \mathbf{w}_t \cdot \boldsymbol{\xi}$.

For fixed couplings, these are joint Gaussian random variables with 0 means and second moments $Q_{ab} = \overline{u_a u_b} = N^{-1}\mathbf{w}_a \cdot \mathbf{w}_b$. Thus, we have, for $P = \alpha N$,

$$N^{-1}\ln(\Xi_n) = N^{-1}\ln\int\prod_{a=1}^{n+1} d\mathbf{w}_a\, p_a(\mathbf{w}_a)\exp[\alpha N \mathcal{G}_1(n)] \tag{5.57}$$

with

$$e^{\mathcal{G}_1(n)} = 2\Phi_t(u_{n+1}\{Q\})\prod_{a=1}^{n}\Phi(u_a\{Q\}). \tag{5.58}$$

The average over the u_a can be performed with the help of the following basic assumption of mean–field theory.

For $N \to \infty$, the integrals over $\mathbf{w}_a$ will be dominated by regions in the phase space where the matrix elements Q_{ab} assume *nonfluctuating values*. These are the *order parameters*, which determine the macroscopic physics of the network. Assuming that *replica symmetry* is valid, the order parameters will obey $Q_{ab} = q$, for $a \neq b$ and $a, b \leq n$. Further, $Q_{a,n+1} = R$.

$q = N^{-1}\mathbf{w}_a \cdot \mathbf{w}_b$ is the typical overlap between any two student vectors $\mathbf{w}_a$ and $\mathbf{w}_b$, which are drawn randomly from the Gibbs distribution (5.48). Accordingly, $R = N^{-1}\mathbf{w}_t \cdot \mathbf{w}_s$ is the overlap between a teacher and a student. By Eq. (5.47), the knowledge of the order parameters enables us to obtain the generalization error via

$$\varepsilon = \frac{1}{\pi}\arccos(R/\sqrt{q_0}). \tag{5.59}$$

Using the replica-symmetric ansatz, the Gaussian fields can be constructed explicitly,

$$u_a = z_a(q_0 - q)^{1/2} - tq^{1/2} \tag{5.60}$$

for $a \leq n$ and

$$u_{n+1} = y(1 - R^2/q)^{1/2} - tR/q^{1/2}, \tag{5.61}$$

where z_a, y, t are independent Gaussian variables with variance 1. Obviously, these variables yield the correct second moments. Now the Gaussian average is easily performed, yielding

$$e^{\mathcal{G}_1(n)} = 2\int_{-\infty}^{\infty} Dt \int_{-\infty}^{\infty} Dy\, \Phi_t\left(y(1 - R^2/q)^{1/2} - tR/q^{1/2}\right) \times \left[\int_{-\infty}^{\infty} Dz\, \Phi\left(z\sqrt{q_0 - q} - t\sqrt{q}\right)\right]^n. \tag{5.62}$$

Again, $Dt = (2\pi)^{-1/2}e^{-1/2t^2}\,dt$ is the Gaussian measure. Using the saddle-point method, for $N \to \infty$, we finally get

$$\lim_{N\to\infty} N^{-1}\ln\Xi_n = \mathrm{E}xtr_{q,R}\,[\alpha\mathcal{G}_1 + \mathcal{G}_2].$$

The second term is an entropic term coming from the phase-space integral, where the order parameters q and R are fixed:

$$e^{N\mathcal{G}_2(n)} = V_0^{-(n+1)} \int \prod_{a=1}^{n+1} d\mathbf{w}_a \prod_{a \le b} \delta(\mathbf{w}_a \cdot \mathbf{w}_b - NQ_{ab}). \tag{5.63}$$

In replica symmetry, it is not hard to evaluate this expression, giving the result

$$\mathcal{G}_2 = \frac{n-1}{2} \ln(q_0 - q) + \frac{1}{2} \ln\left[(q_0 - q) + n(q - R^2)\right]. \tag{5.64}$$

Finally, performing the derivative with respect to n yields

$$\mathcal{F} = Extr_{q,R}\left[\alpha \mathcal{F}_1 + \mathcal{F}_2\right], \tag{5.65}$$

where

$$\begin{aligned} \mathcal{F}_1 \;=\; & \int_{-\infty}^{\infty} Dt \; \frac{\int_{-\infty}^{\infty} Dy \; \Phi_t\left(y(1 - R^2/q)^{1/2} - tR/q^{1/2}\right)}{\int_{-\infty}^{\infty} Dy \, \Phi_t(y)} \\ & \times \ln\left[\int_{-\infty}^{\infty} Dz \, \Phi\left(z\sqrt{q_0 - q} - t\sqrt{q}\right)\right] \end{aligned} \tag{5.66}$$

and

$$\mathcal{F}_2 = \frac{1}{2} \ln(q_0 - q) + \frac{1}{2} \frac{q - R^2}{(q_0 - q)}. \tag{5.67}$$

Extremizing the free energy in Eq. (5.65), we will get the physical values of the order parameters q and R, which in turn determine the generalization error ε.

5.3.3 Results for Bayes and Gibbs Algorithms

Before we come to specific deterministic learning algorithms, we will study the performance of the Gibbs algorithm for a perceptron. As in Secs. 5.2.3 and 5.2.4, we will assume that the prior distribution of the teacher is known to the student.

However, it should be noted that, for the spherical density of inputs in Eq. (5.46), by symmetry, the order parameters will not depend on the actual teacher. Thus, for this special density, the following results will hold not only on average, but also for any specific teacher. If noise is present in the teacher's classifications, we also will assume that the student will know the type of noise and its strength.

The interpretation of the Gibbs ensemble as the posterior distribution in the Bayesian sense simplifies the algebra. In this case we always have $\Phi = \Phi_t$.

Then, teacher and student will enter the replica theory in a completely symmetric way. The teacher is just another replica, so that, from the beginning, we can set $q = R$ and $q_0 = 1$.

Now Eq. (5.65) is replaced by

$$\mathcal{F} = Extr_q \left\{ \frac{\alpha}{A_0} \int_{-\infty}^{\infty} A(t;q) \ln [A(t;q)] + \frac{1}{2} \ln(1-q) + \frac{q}{2} \right\}, \quad (5.68)$$

where

$$A(t;q) = \int_{-\infty}^{\infty} Dz\, \Phi\left(z\sqrt{1-q} - t\sqrt{q}\right) \quad (5.69)$$

and

$$A_0 = \int_{-\infty}^{\infty} Dz\, \Phi(z).$$

It is interesting to note that this expression could have been derived by a slight modification of the standard replica trick, where we replace the limit $n \to 0$ by $n \to 1$. Setting $Z_t = Z$, we get

$$\mathcal{F} = \lim_{n \to 1} \frac{\partial}{\partial n} N^{-1} \ln \sum_{\sigma_1 \ldots \sigma_P = \pm 1} \overline{Z^n(\sigma^P)}. \quad (5.70)$$

We now give explicit expressions for noise-free and noisy teachers [see Eq. (5.18)]:

$$\Phi(u) = \begin{cases} \Theta(u) & \text{no noise} \\ \exp[-\beta\Theta(-u)] & \text{output noise} \\ H(-\beta^{-\frac{1}{2}}u) & \text{weight noise,} \end{cases} \quad (5.71)$$

leading to

$$A(t;q) = \begin{cases} H(\gamma t) & \text{no noise} \\ e^{-\beta} + (1-e^{-\beta})H(\gamma t) & \text{output noise} \\ H(\hat{\gamma} t) & \text{weight noise} \end{cases} \quad (5.72)$$

with $\gamma = \sqrt{q/1-q}$ and $\hat{\gamma} = \sqrt{q/1-q+1/\beta}$. For output noise, $A_0 = \frac{1}{2}(1+e^{-\beta})$, and $A_0 = \frac{1}{2}$ in the other cases. Calculating the order parameter q from Eq. (5.68), yields the Gibbs error as:

$$\varepsilon_{Gibbs} = \frac{1}{\pi} \arccos(q). \quad (5.73)$$

For noisy outputs, this is the probability that the student will find the *ideal* output of the teacher.

Solving the order parameter equation asymptotically for $\alpha \to \infty$, i.e., $q \to 1$, one obtains

$$\epsilon_{Gibbs} \simeq \begin{cases} 0.62 \cdot \alpha^{-1} & \text{without noise} \\ C_1(\beta) \cdot \alpha^{-1} & \text{output noise} \\ C_2(\beta) \cdot \alpha^{-1/2} & \text{weight noise} \end{cases} \quad (5.74)$$

C_1, C_2 are functions of the temperature. C_1 converges to the value 0.62 for $\beta \to \infty$, whereas C_2 goes to 0, indicating the crossover to the faster decay in the noise-free limit.

The decay $\propto \alpha^{-1}$ in the noise-free case is of the same order as the bound (5.33) discussed in Sec. 5.2.4. Remarkably, this asymptotic decay still persists if output noise is included. When the noise temperature grows large (i.e., $\beta \to 0$), the coefficient C_1 diverges like $4/\beta^2$.

The case of weight noise also has been studied in [8, 21]. However, the authors calculated the Gibbs error for a different algorithm, which uses the sum of mistakes [the *first* line in Eq. (5.18)] as the learning energy. With an optimized learning temperature, $\varepsilon_{Gibbs} \simeq \alpha^{-1/4}$ was found. With a 0 temperature learning, which corresponds to minimizing the training energy (maximum likelihood), the behavior is even worse. This shows that the generalization ability can be remarkably enhanced if more information on the teacher is included in the learning algorithm.

The error of the Bayes algorithm has been calculated in [9, 10]. We will give a different derivation by looking at the volume ratio,

$$Y = \frac{V(\sigma^{P+1})}{V(\sigma^P)}, \tag{5.75}$$

previously defined in Sec. 5.2.4. Equation (5.75) describes the reduction of the volume of the teacher's cell when a new input is learned. As was shown in Sec. 5.2.4, Y can be used to find Gibbs and Bayes errors, as well as the information gain.

Obviously, Y is an average of the function $\Theta(N^{-1/2}\sigma_{P+1}\mathbf{w}_t \cdot \boldsymbol{\xi}_{P+1})$ over all couplings of the teacher's old cell $V(\sigma^P)$. We can write

$$Y = \langle \Theta(N^{-1/2}\sigma_{P+1}\mathbf{w}_t \cdot \boldsymbol{\xi}_{P+1}) \rangle. \tag{5.76}$$

One of the basic assumptions of the replica-symmetric mean-field theory is the *clustering hypothesis* [22], which states that the thermodynamic fluctuations of different components $w_t(j)$ of $\mathbf{w}_t$ are uncorrelated in the thermodynamic limit. From the central limit theorem, we conclude that, for *fixed* input $\boldsymbol{\xi}_{P+1}$, the field $N^{-1/2}\sigma_{P+1}\mathbf{w}\cdot\boldsymbol{\xi}_{P+1}$ can be written as $N^{-1/2}\sigma_{P+1}\langle \mathbf{w} \rangle \cdot \boldsymbol{\xi}_{P+1} + v$, where the fluctuating part v is Gaussian distributed and has variance

$$\langle v^2 \rangle = N^{-1}(\langle \mathbf{w} \cdot \mathbf{w} \rangle - \langle \mathbf{w} \rangle^2) = 1 - q. \tag{5.77}$$

Here, we have again used the clustering hypothesis, yielding

$$q = N^{-1}\mathbf{w}_a \cdot \mathbf{w}_b = N^{-1}\langle \mathbf{w} \rangle^2. \tag{5.78}$$

Performing the average over v gives the expression

$$Y = H\left(\frac{N^{-1/2}\sigma_{P+1}\langle \mathbf{w} \rangle \cdot \boldsymbol{\xi}_{P+1}}{\sqrt{1-q}}\right), \tag{5.79}$$

which holds for a fixed input pattern *and* classification label in the thermodynamic limit! The Bayesian algorithm votes for that value of σ_{P+1} which gives the largest volume, in other words, the largest value for Y. By its definition, $H(x) = \int_x^\infty Dt > \frac{1}{2}$, if $x > 0$. This has the consequence that a student with coupling vector $\mathbf{w}_s = \langle \mathbf{w} \rangle$ will always make the optimal Bayes decision.

This was first shown in [23] by means of a slightly different argument. It is nontrivial because, in general, the "Bayesian student" does not belong to the phase space of the teachers. Finally, to get the Bayes error, we simply have to find the average overlap between the student and a random teacher [Eq. (5.47)]:

$$\frac{\langle \mathbf{w}_t \rangle \cdot \langle \mathbf{w}_t \rangle}{|\langle \mathbf{w}_t \rangle| \cdot \sqrt{N}} = \sqrt{q}. \tag{5.80}$$

Hence,

$$\varepsilon_{Bayes} = \frac{1}{\pi} \arccos(\sqrt{q}). \tag{5.81}$$

Solving the order parameter equation (5.68), we get an asymptotic decay,

$$\varepsilon_{Bayes} \simeq 0.44 \cdot \alpha^{-1},$$

for large α. A comparison of Gibbs and Bayes errors is given in Fig. 5.5. Different algorithms to achieve the performance of the Bayes prediction can be found in [2, 9, 24].

We will end this section by calculating the density of Y. We first need the probabilities of the classification labels $\sigma_{P+1} = \pm 1$. These probabilities are proportional to the volumes of the two new cells. Thus, they simply equal $Y(\sigma_{P+1})$! Using that $N^{-1/2}\langle \mathbf{w} \rangle \cdot \boldsymbol{\xi}_{P+1}$ is Gaussian distributed with respect to the random input $\boldsymbol{\xi}_{P+1}$, with variance equal to $\langle \mathbf{w} \rangle^2 = q$, we find

$$f(Y) = 2 \int_{-\infty}^{\infty} Dt \, H(\gamma t) \, \delta(Y - H(\gamma t)). \tag{5.82}$$

Here, $\gamma = \sqrt{q/1-q}$ and $\delta(\cdot)$ is Dirac's δ–function. Equation (5.82) is depicted in Fig. 5.6 for different values of q. This density is also valid for discrete couplings as long as replica symmetry is exact. Figure 5.7 gives a result for $f(Y)$ obtained from simulations of perceptrons with binary weights. Here, the volumes of the cells were found by *counting* the number of discrete coupling vectors belonging to each cell.

The smooth behavior of $f(Y)$ somehow seems to contradict the VC results. Since the number of cells grows only like a polynomial in P, most of the cells will not be split into two pieces by adding a new input. Thus, $f(Y)$ should contain δ–functions at $Y = 0$ and $Y = 1$, corresponding to one new cell with the old volume, and one with 0 volume. This would in fact be true if all of the cells had the same volume. We conclude that those cells which are not cut into two pieces have neglectable volume (probability) in the thermodynamic limit.

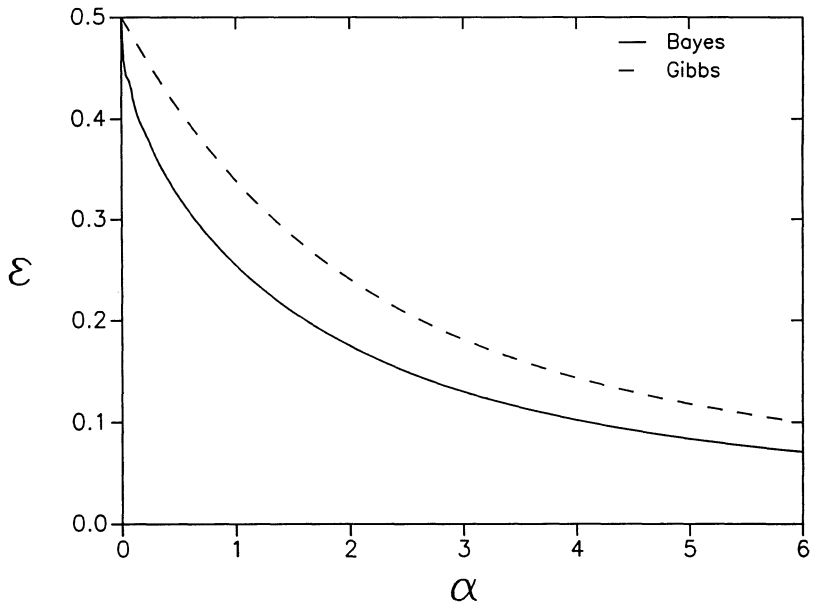


Fig. 5.5. Comparison of Gibbs and Bayes generalization errors.

5.4 Geometry in Phase Space and Asymptotic Scaling

The result of the replica theory for the Gibbs algorithm shows an asymptotic decay of the error $\varepsilon_{Gibbs} \simeq \alpha^{-1}$. The same power law was obtained as an upper bound from the VC theory in Sec. 5.2.4. While for the replica calculation a specific distribution of inputs was assumed, in the VC approach only the VC dimension of the network entered the theory. Thus, arises the question of whether the asymptotic scaling of the generalization error can be explained using only a few parameters of a network.

As we will see, simple geometric scaling arguments will bring us a step closer to this idea of universality. We begin with the perceptron. The phase space of all perceptrons is a simple manifold — the surface of a sphere. The generalization error,

$$\varepsilon = \frac{1}{\pi} \arccos(\mathbf{w}_s \cdot \mathbf{w}_t), \tag{5.83}$$

which is valid for normalized teacher and student vectors, is just the arclength of the shortest line (the geodesic) between $\mathbf{w}_s$ and $\mathbf{w}_t$, and ε is a natural distance between perceptron networks.

A second contribution to a geometry in phase space comes from the information theoretic results of Sec. 5.2.4. We remember that the average information gain for any new input is an upper bound for the Gibbs error

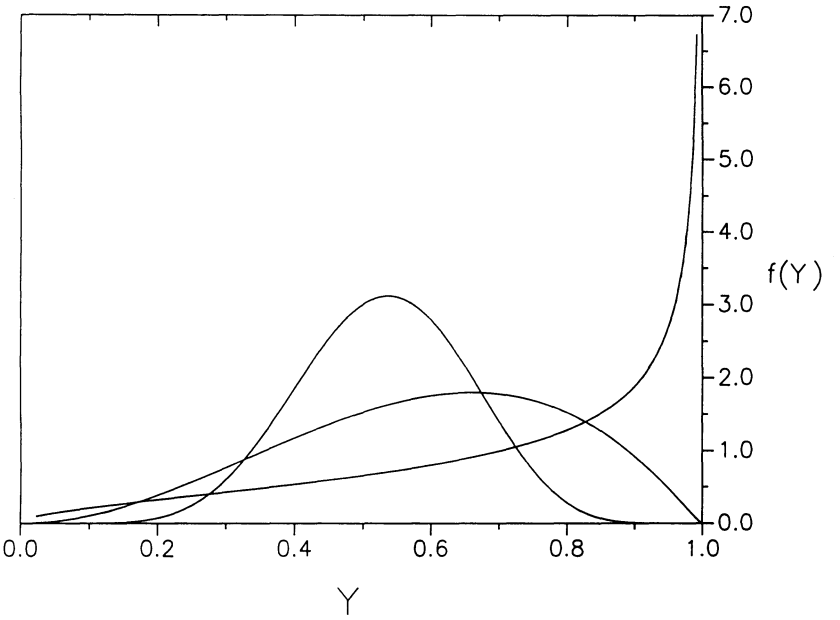


Fig. 5.6. Density of the volume ratio Y for $q = 0.1$ (bell–shaped curve), $q = 0.3$ (flatter curve), and $q = 0.7$ (curve peaked at 1).

on that input. Assuming that both quantities will be of the same order asymptotically, we set

$$\langle \Delta I \rangle = -\langle \ln(V_{P+1}/V_P) \rangle \simeq \varepsilon. \tag{5.84}$$

V_P is *the volume* of the teacher's cell and ε is, as we have shown, a *typical distance* in the cell. Since the number of couplings N is the dimension of the manifold, we expect that

$$V_P \simeq \varepsilon^N. \tag{5.85}$$

Then, with $P = \alpha N$, Eq. (5.84) can be written as

$$\varepsilon(\alpha) = -\frac{\partial}{\partial \alpha} \ln(\varepsilon(\alpha)), \tag{5.86}$$

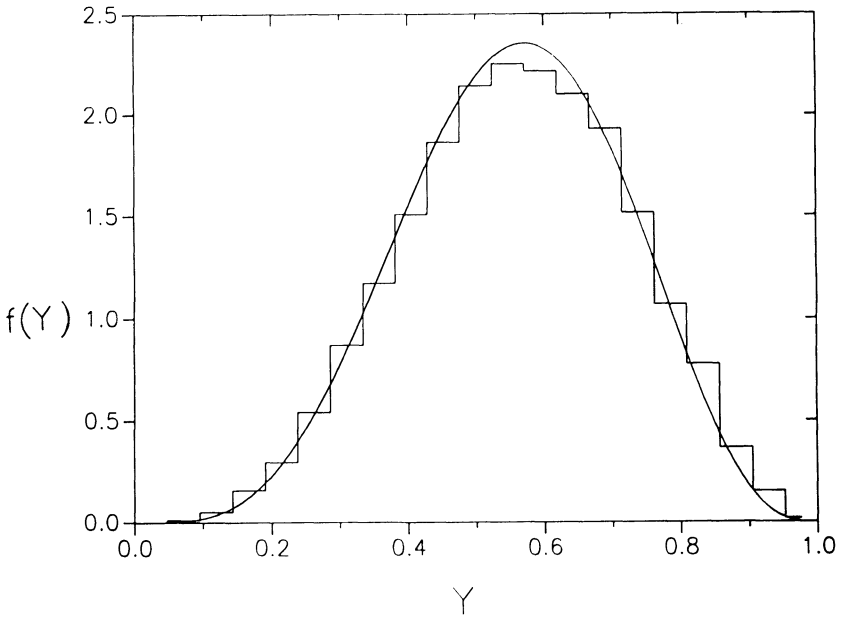
from which the asymptotic relation

$$\varepsilon(\alpha) \simeq \alpha^{-1}$$

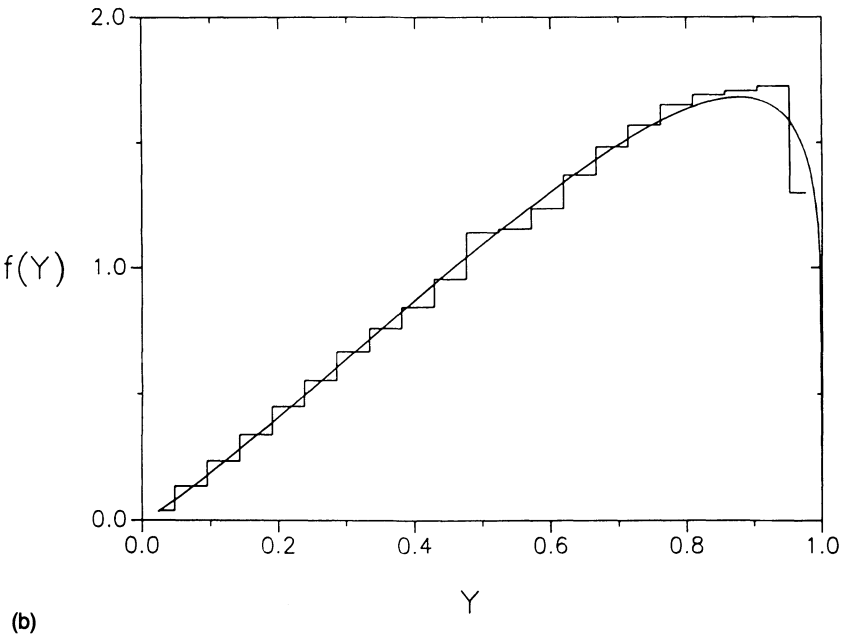
follows.

As a further consequence we see that, if the learner can select examples such that the asymptotic information gain becomes a constant for each new input, then a faster decay of the generalization error like

$$\varepsilon \simeq \exp(-\alpha N \langle \Delta I \rangle) \tag{5.87}$$



(a)



(b)

Fig. 5.7. Density of volume ratio Y from simulations of a perceptron with binary weights. (a) $P = 4, N = 14$. (b) $P = 16, N = 20$. The smooth curves are the theoretical predictions.

Table 5.1. Volumes in input space

	Ω_1	Ω_2	Ω_3	Ω_4	Ω_5	Ω_6	Ω_7	Ω_8
σ_a	-1	-1	-1	-1	1	1	1	1
σ_b	-1	-1	1	1	-1	-1	1	1
σ_c	-1	1	-1	1	-1	1	-1	1

is expected. In fact, such behavior is observed for query algorithms (see Sec. 5.5.4).

The interpretation of the generalization error as a distance between networks is no artefact of the perceptron. Generalizing Eq. (5.83) to arbitrary networks, we will show that the probability $\Delta(t, s)$ (over all inputs) that two networks with parameters $\mathbf{w}_t$ and $\mathbf{w}_s$ *do not give the same answer* defines a metric in the space of networks (of a given type). The only nontrivial part[7] is the triangular inequality. Consider three parameter vectors $\mathbf{w}_a$,$\mathbf{w}_b$,$\mathbf{w}_c$ and divide the *input space* into 8 sets with volumes $\Omega_1, \ldots, \Omega_8$, $\sum_i \Omega_i = 1$, according to the outputs $\sigma_{a,b,c}$ (see Table 5.1). Then, $\Delta(a, b)$ Probability of all $\boldsymbol{\xi}$, for which $\mathbf{w}_a$ and $\mathbf{w}_b$ have different outputs $= \Omega_3 + \Omega_4 + \Omega_5 + \Omega_6$. Similarly, $\Delta(b, c) = \Omega_2 + \Omega_3 + \Omega_6 + \Omega_7$ and $\Delta(a, c) = \Omega_2 + \Omega_4 + \Omega_5 + \Omega_7$. Thus,

$$\Delta(a, b) + \Delta(b, c) = \Omega_2 + 2\Omega_3 + \Omega_4 + \Omega_5 + 2\Omega_6 + \Omega_7 \geq \Delta(a, c).$$

This completes the proof of the triangular inequality.

So we can expect that the asymptotic scaling of the learning error based on the simple geometric picture is valid for more general types of networks or learning machines.

Based on similar ideas, an asymptotic scaling of the information gain $\langle \Delta I \rangle \simeq \alpha^{-1}$ for noise-free learning was predicted in [25]. Using this simple geometric picture, we now derive an asymptotic result for the Gibbs error in the case of learning with strong output noise [26]. This will be shown for *a general network*, where only the number N of free adjustable parameters enters the calculation.

We consider a teacher network with a noisy output,

$$\sigma = \eta \cdot F(\mathbf{w}_t, \boldsymbol{\xi}). \tag{5.88}$$

The teacher's ideal answer is inverted, i.e., $\eta = -1$, with probability $e^{-\beta}/1 + e^{-\beta}$ independent of the inputs. The task of the learner is to construct a deterministic, i.e., noise-free, student network $\mathbf{w}_s$,

$$\sigma = F(\mathbf{w}_s, \boldsymbol{\xi}), \tag{5.89}$$

[7] We neglect the possibility that two different parameters $\mathbf{w}_a$ and $\mathbf{w}_b$ will give the same outputs on *all* inputs.

who will be able to give the teacher's ideal answers [$\eta = +1$ in Eq. (5.88)]. We will use the Gibbs algorithm to construct such a network. This algorithm will draw a $\mathbf{w}_s$ randomly from the posterior distribution of the unknown teacher, after having seen P noisy examples. Using the ideas of Sec. 5.2.3, the students will have probability

$$p(\mathbf{w}_s|\sigma^P) = Z^{-1} \exp\left(-\beta \sum_{k=1}^{P} E(\mathbf{w}_s; \sigma_k, \boldsymbol{\xi}_k)\right), \tag{5.90}$$

where

$$Z(\sigma^P) = \int d\mathbf{w} \, \exp\left(-\beta \sum_{k=1}^{P} E(\mathbf{w}; \sigma_k, \boldsymbol{\xi}_k)\right). \tag{5.91}$$

$E(\mathbf{w}_s; \sigma_k, \boldsymbol{\xi}_k)$ equals 1 if $\sigma_k \neq F(\mathbf{w}_s, \boldsymbol{\xi}_k)$, i.e., if the student does not learn the outputs correctly.

By using the Gibbs algorithm, the student will not simply try to minimize his or her learning error, but instead will make mistakes on the observed labels with probability $e^{-\beta}/1+e^{-\beta}$. This is precisely the rate at which the teacher produces wrong outputs. Using the temperature β^{-1}, the student assumes a priori that a fraction of the teacher's answers are not correct.

Fixing teacher and student for a moment, the probability that the student's and the teacher's ideal answer disagree on a new input $\boldsymbol{\xi}$, i.e., that

$$F(\mathbf{w}_t, \boldsymbol{\xi}) \neq F(\mathbf{w}_s, \boldsymbol{\xi}), \tag{5.92}$$

is given by

$$\Delta(t,s) = 1 - \sum_{\sigma=\pm 1} \overline{E(\mathbf{w}_t; \sigma, \boldsymbol{\xi}) E(\mathbf{w}_s; \sigma, \boldsymbol{\xi})}. \tag{5.93}$$

Given the P outputs, the teacher and the student have the same distribution [Eq. (5.90)] by the definition of the algorithm. Using this fact, and weighting all possible output configurations with their probability [Eq. (5.52)], $\mathcal{P}(\sigma^P) = \mathcal{C}^{-1} \cdot Z(\sigma^P)$, we can average Eq. (5.93) over teachers and students:

$$\langle \Delta(t,s) \rangle =$$

$$\sum_{\sigma_1 \ldots \sigma_P = \pm 1} \mathcal{C}^{-1} \cdot Z(\sigma^P) \int d\mathbf{w}_t \, d\mathbf{w}_s \, \Delta(t,s) \cdot p(\mathbf{w}_t|\sigma^P) \cdot p(\mathbf{w}_s|\sigma^P). \tag{5.94}$$

The total Gibbs error is obtained by averaging this expression over the training inputs. This can be done with the replica method, in a form similar to Eq. (5.70). One finds, using Eqs. (5.90) and (5.91),

$$\varepsilon_{Gibbs} = \lim_{\substack{n \to 1 \\ \gamma \to 0}} \frac{\partial^2}{\partial \gamma \partial n} \ln \int \prod_{a=1}^{n} d\mathbf{w}_a \exp[-PG_n(\{\mathbf{w}_a\}) + \gamma \sum_{a \neq b} \Delta(a,b)] \tag{5.95}$$

with the replica Hamiltonian

$$G_n = -\ln[\sum_{\sigma=\pm 1} \exp(-\beta \sum_{a=1}^{n} E(\mathbf{w}_a, \sigma, \boldsymbol{\xi}))]. \tag{5.96}$$

This result has an interesting limit for strong noise, i.e., small β:

$$G_n(\{\mathbf{w}_a\}) = -\ln 2 + \frac{n\beta}{2} - \frac{\beta^2 n^2}{8} + \frac{\beta^2}{4} \sum_{a \neq b} \Delta(a,b) + \mathcal{O}(\beta^3). \tag{5.97}$$

Here we have made use of the fact that $(E(\mathbf{w}_a; \sigma, \boldsymbol{\xi}))^2 = E(\mathbf{w}_a; \sigma, \boldsymbol{\xi})$ and $\sum_{S\pm 1} E(\mathbf{w}_a; \sigma, \boldsymbol{\xi}) = 1$. Inserting this into Eq. (5.95), we get

$$\epsilon_{Gibbs} \simeq -\lim_{n \to 1} \frac{\partial^2}{\partial n \partial B} \ln \int \prod_{a=1}^{n} d\mathbf{w}_a \exp[-B \sum_{a \neq b} \Delta(a,b)], \tag{5.98}$$

where the derivative with respect to B has to be taken at $B = P\beta^2/4$. The phase-space integral in Eqs. (5.98) is the partition function for n classical "particles" at temperature B^{-1} interacting with the pair potential $\Delta(a,b)$.

If the number of examples P grows large, the effective temperature B^{-1} goes to 0 and the particles are close together at the minimum of the potential. In other words, $\Delta(t,s)$ vanishes, and we have perfect generalization!

To estimate the speed of generalization, we fix one of the couplings, e.g., $\mathbf{w}_n$. If all distances $\Delta(n,b)$ are small for large B, then the triangular inequality will enforce all other distances $\Delta(a,b)$ to be small as well.

Our basic assumption is that for small distances the manifold of parameters $\mathbf{w}$ is locally flat. In suitably chosen coordinates, with $\mathbf{w}_n$ at the origin, $\mathbf{w}_a \equiv w_a(i), \quad i = 1, \ldots, N$, the volume element ($\mathbf{w}_n$ is fixed)

$$d\mathbf{w}_a \simeq \prod_{i=1}^{N} dw_a(i) \tag{5.99}$$

is locally cartesian. Also, the distances $\Delta(a,b)$ are expected to be of the form $\Delta[\{w_a(i) - w_b(i)\}]$ for $w_a(i) - w_b(i) << 1$, and Δ should obey the "regular scaling" of a length,

$$B \cdot \Delta[\{w_a(i) - w_b(i)\}] = \Delta[\{B \cdot (w_a(i) - w_b(i)\}]. \tag{5.100}$$

Then we can simply scale the inverse temperature B out of Eq. (5.98) by using $Bw_a(i)$ as new coordinates. We get

$$\epsilon_{Gibbs} \simeq -\lim_{n \to 1} \frac{\partial^2}{\partial n \partial B} \ln[B^{-(n-1)N}] = \frac{N}{B}. \tag{5.101}$$

Setting $B = P\beta^2/4$, we get

$$\varepsilon_{Gibbs} \simeq \frac{4}{\beta^2 \alpha}. \tag{5.102}$$

This coincides with the known result in the case of the perceptron. Note, however, that in the present approach we have made no assumptions on the distribution of inputs and the special architecture of the network.

Since exact replica calculations for multilayer networks become technically very involved, we expect that the geometric approach will provide a useful alternative, at least in asymptotic regions. It would be interesting to establish a connection with the VC results.

5.5 Applications to Perceptrons

In this section we discuss several applications of the statistical mechanics of generalization. In particular, we concentrate on the simplest case: the teacher as well as the student are simple one–layer perceptrons, with one input layer $\boldsymbol{\xi}$, one weight layer $\mathbf{w}_t$ or $\mathbf{w}_s$, respectively, and one output bit σ. As before, we normalize the teacher weight vector to $\mathbf{w}_t \cdot \mathbf{w}_t = N$:

$$\sigma = \text{ sign } \left(\frac{1}{\sqrt{N}} \, \mathbf{w}_{t/s} \cdot \boldsymbol{\xi} \right) . \tag{5.103}$$

The student tries to learn a set of $\alpha N = P$ input–output examples $\sigma_k, \boldsymbol{\xi}_k \, , k = 1, \, ..., \alpha N$, given by the teacher network. In the following, several learning rules are considered; in addition, the structure of the teacher may be different from that of the student, or it may even change with time. It turns out that the simplest case — perceptron learns from perceptron — already shows many interesting phenomena.

The advantage of simplicity is the fact that one obtains exact mathematical relations, for example, the generalization error ε as a fuction of the number αN of learned examples. Furthermore, the simple structure is always a part of more complex networks, and from understanding the perceptron it may be possible to derive results for multilayer networks.

5.5.1 Simple Learning: Hebb Rule

The learning rule that easily can be analyzed [27] is the Hebb rule: At each presentation of a new example $(\sigma_k, \boldsymbol{\xi}_k)$ the product of input and output bits is added to the corresponding weight,

$$\vec{w_s} \, (t+1) = \vec{w_s}(t) + \frac{1}{\sqrt{N}} \, \sigma_k \, \boldsymbol{\xi}_k \, . \tag{5.104}$$

If each example is presented once, and if the initial weight vector is 0, then the final weights are given by

$$\vec{w_s} = \frac{1}{\sqrt{N}} \sum_{k=1}^{P} \sigma_k \, \boldsymbol{\xi}_k \, . \tag{5.105}$$

Note that σ_k is given by the teacher,

$$\sigma_k = \text{sign}\left(\frac{1}{\sqrt{N}}\,\mathbf{w}_t \cdot \boldsymbol{\xi}_k\right). \tag{5.106}$$

Now we study the case of random inputs $\boldsymbol{\xi}_k$. We are interested in the generalization error ε, which, following Eq. (5.47), is given by the overlaps $R = \mathbf{w}_t \cdot \mathbf{w}_s/N$ and $q_0 = \mathbf{w}_s \cdot \mathbf{w}_s/N$:

$$\varepsilon = \frac{1}{\pi}\arccos\left(\frac{R}{\sqrt{q_0}}\right). \tag{5.107}$$

At each step of presenting a new example $(\sigma_k, \boldsymbol{\xi}_k)$ the teacher–student overlap $R = \mathbf{w}_t \cdot \mathbf{w}_s/N$ changes by an amount ΔR given by

$$\Delta R = \frac{1}{N}\,\frac{1}{\sqrt{N}}\;\text{sign}\;(\mathbf{w}_t \cdot \boldsymbol{\xi}_k) \qquad \mathbf{w}_t \cdot \boldsymbol{\xi}_t = \frac{1}{N}\frac{|\mathbf{w}_t \cdot \boldsymbol{\xi}_k|}{\sqrt{N}}\,. \tag{5.108}$$

However, for different input patterns $\boldsymbol{\xi}_k$, the variable $u = \mathbf{w}_t \cdot \boldsymbol{\xi}_k/\sqrt{N}$ is Gaussian distributed (u = sum of independent random numbers) with

$$\overline{u} = 0 \qquad \text{and} \qquad \overline{u^2} = \frac{1}{N}\,\mathbf{w}_t \cdot \mathbf{w}_t = 1\,. \tag{5.109}$$

Hence, with $\overline{|u|} = \sqrt{2/\pi}$, on average, the teacher–student overlap changes by the amount $\Delta R = \sqrt{2/\pi}/N$, which gives

$$R = \sqrt{\frac{2}{\pi}}\,\alpha\,. \tag{5.110}$$

The square of Eq. (5.104) gives the change of the student–student overlap, and one has

$$\begin{aligned}\Delta q_0 &= \frac{1}{N}\left(\frac{2}{\sqrt{N}}\,\sigma_k\,\boldsymbol{\xi}_k \cdot \mathbf{w}_s(t) + 1\right)\\ &= \frac{1}{N}\left(2\;\text{sign}\;(u)\cdot z + 1\right).\end{aligned} \tag{5.111}$$

The variable $z = \mathbf{w}_s(t) \cdot \boldsymbol{\xi}_k/\sqrt{N}$ is again Gaussian distributed with

$$\overline{z^2} = q_0 \qquad \text{and} \qquad \overline{zu} = R\,.$$

The correlations between z and u are taken into account by the substitution $z = Ru + \sqrt{q_0 - R^2}\,t$ with $\overline{t^2} = 1$ and $\overline{tu} = 0$. One obtains for the average of Δq_0

$$\Delta q_0 = \frac{1}{N}\left(2R\,\overline{|u|} + 1\right) = \Delta R^2 + \frac{1}{N}\,. \tag{5.112}$$

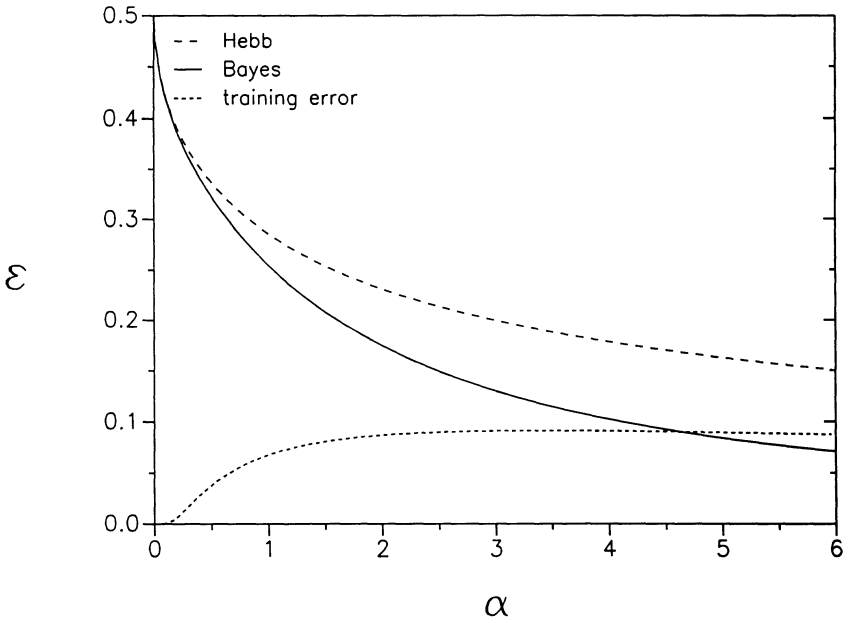


Fig. 5.8. Generalization error for Hebbian learning. The other two curves are the Bayesian error and the Hebbian training error.

This gives

$$q_0 = \alpha + R^2, \tag{5.113}$$

and, as the final result,

$$\varepsilon = \frac{1}{\pi} \arccos\left(\sqrt{\frac{2}{\pi}} \frac{\alpha}{\sqrt{\alpha + \frac{2}{\pi}\alpha^2}}\right) = \frac{1}{\pi} \arctan\left(\sqrt{\frac{\pi}{2\alpha}}\right) . \tag{5.114}$$

Hebbian learning also may be considered as a drifting random walk of $\mathbf{w}_s$ in an N-dimensional vector space [2]. The component of $\mathbf{w}_s$ in the direction of the teacher increases like $\sqrt{2/\pi}\alpha$ while, perpendicular to the teacher, the student performs a random walk with mean-square displacement α. The ratio of the two lengths determines $\tan(\pi\varepsilon)$, according to Fig. 5.4.

Figure 5.8 shows the generalization error ε [Eq. (5.114)] as a function of the number of learned examples α. If only a finite number of examples has been stored ($\alpha = 0$), the network cannot generalize, and one has $\varepsilon = 0.5$. But if the number of examples is of the order of the number of weights, ε decreases. For large values of α, Eq. (5.114) gives

$$\varepsilon \propto 1/\sqrt{\alpha} . \tag{5.115}$$

Hence, asymptotically, the Hebbian rule is worse than the Bayesian optimum $\varepsilon \simeq 0.44/\alpha$. Nevertheless, it is surprising that the rule gives a rea-

sonably low error ε. That is, the Hebbian network cannot learn perfectly; its training error

$$\varepsilon_t = \overline{\theta\left[-\left(\mathbf{w}_t \cdot \boldsymbol{\xi}_k\right)\left(\mathbf{w}_s \cdot \boldsymbol{\xi}_k\right)\right]} \tag{5.116}$$

is nonzero for any $\alpha > 0$. With the Gaussian variables u and t as before, one has

$$\varepsilon_t = \overline{\theta\left[-u\left(Ru + \sqrt{q_0 - R^2}\, t + (u)\right)\right]}, \tag{5.117}$$

which gives [27]

$$\varepsilon_t = \frac{1}{2} - \int_0^\infty Du\ erf\left(u\sqrt{\frac{\alpha}{\pi}} + \frac{1}{\sqrt{2\alpha}}\right) . \tag{5.118}$$

Hence, for $\alpha \simeq 5$, one finds a maximal training error of about 10%, which appears to be rather large. Nevertheless, the Hebbian network is able to generalize reasonably well.

5.5.2 Overfitting

If one has a cost function E that depends continuously on the weight vector $\mathbf{w}_s$, then a learning rule may be defined as a gradient descent in the N-dimensional weight space:

$$\mathbf{w}_s\ (t+1) = \mathbf{w}_s(t) - \gamma\ \nabla\ E\left(\mathbf{w}_s(t)\right). \tag{5.119}$$

In many applications, the cost function is defined as the quadratic deviation between student and teacher output. In a multilayer feedforward network with continuous activation functions, the gradient rule is called *error backpropagation* [28].

Unfortunately, a gradient cannot be defined for binary student output. But one may try to learn the binary teacher output by a linear student network, minimizing the cost function

$$E = \sum_k \left(\frac{1}{\sqrt{N}}\, \sigma_k \mathbf{w}_s \cdot \boldsymbol{\xi}_k - 1\right)^2 \tag{5.120}$$

with σ_k given by the teacher network, $\sigma_k = \text{sign}\left(\mathbf{w}_t \cdot \boldsymbol{\xi}_k / \sqrt{N}\right)$. This gives the learning algorithm

$$\mathbf{w}_s(t+1) = \mathbf{w}_s(t) + \frac{\gamma}{\sqrt{N}}\left(1 - \frac{1}{\sqrt{N}}\, \sigma_k\ \mathbf{w}_s(t)\ \boldsymbol{\xi}_k\right) \sigma_k\ \boldsymbol{\xi}_k\ . \tag{5.121}$$

This algorithm has been studied for more than 30 years [29]; it is called ADALINE. For attractor networks it improves the storage capacity for random patterns from $\alpha_c = 0.14$ (Hebbian weights) to $\alpha_c = 1$ [30].

For $E = 0$, Eq. (5.120) gives αN many linear equations for the N unknown coefficients of $\mathbf{w}_s$:

$$\frac{1}{\sqrt{N}} \mathbf{w}_s \, \boldsymbol{\xi}_k = \operatorname{sign}\left(\frac{1}{\sqrt{N}} \mathbf{w}_t \cdot \boldsymbol{\xi}_k\right); \qquad k = 1, ..., \alpha N \, . \tag{5.122}$$

If the input patterns $\boldsymbol{\xi}_k$ are linearly independent, one can solve this equation for $\alpha < 1$. But, for $\alpha > 1$, it is obvious that Eq. (5.122) cannot be fulfilled; although the rule is realizable, the ADALINE algorithm cannot learn it perfectly. The training error E_t increases for $\alpha > 1$ to a nonzero value.

Although the learning algorithm is defined by the linear network, its training and generalization errors still are defined by the nonlinear network $\sigma = \operatorname{sign}(\mathbf{w} \cdot \boldsymbol{\xi})$. Both of the errors can be calculated analytically using the replica method of Sec. 5.3.2 [31]. Using the Gibbs weight $\exp[-\beta E]$, one finds the properties of the stationary state of the weight vector $\mathbf{w}_s$ (i.e. after having learned for infinitely many timesteps t) from the limit $\beta \to \infty$.

In Eq. (5.66) we replace $\Phi(u)$ by

$$\Phi(u) = \sqrt{\beta} \exp\left[-\tfrac{1}{2} \, \beta \, (u-1)^2\right] \tag{5.123}$$

and Φ_t by

$$\Phi_t(u) = \Theta(u). \tag{5.124}$$

Then we solve the saddle-point equations for the order parameters q and R. For the limit $\beta \to \infty$, we have to consider two cases:

$\alpha < 1$: In this case, one has $E = 0$, and the length $\sqrt{q_0}$ of the student $\mathbf{w}_s$ is a free parameter that is maximized by $q_0 \to q$. One finds

$$R = \alpha \sqrt{\frac{2}{\pi}}; \qquad q_0 = \frac{\alpha - R^2}{1 - \alpha} \, . \tag{5.125}$$

$\alpha > 1$: One has only one minimum of E, and q converges to q_0 automatically. However, the quantity $\beta(q_0 - q)$ remains finite and nonzero. One finds

$$R = \sqrt{2/\pi}; \qquad q_0 = \frac{1 + \frac{2}{\pi}(\alpha - 2)}{\alpha - 1} \, . \tag{5.126}$$

These equations show that the length of the student vector diverges when α approaches the value 1. This means — since the overlap R between the teacher and the student remains finite — that the generalization error ε increases to $1/2$. *At $\alpha = 1$, the network cannot generalize, although it has learned perfectly!*

The linear network tries to learn a nonlinear problem; for $\alpha \leq 1$, it does so by increasing the length of the weight vector. This gives a low performance of generalization; this effect has been named *overfitting* [27]. For $\alpha > 1$, the network cannot learn perfectly, and its generalization error decreases. Figure 5.9 compares the ADALINE rule with other learning rules.

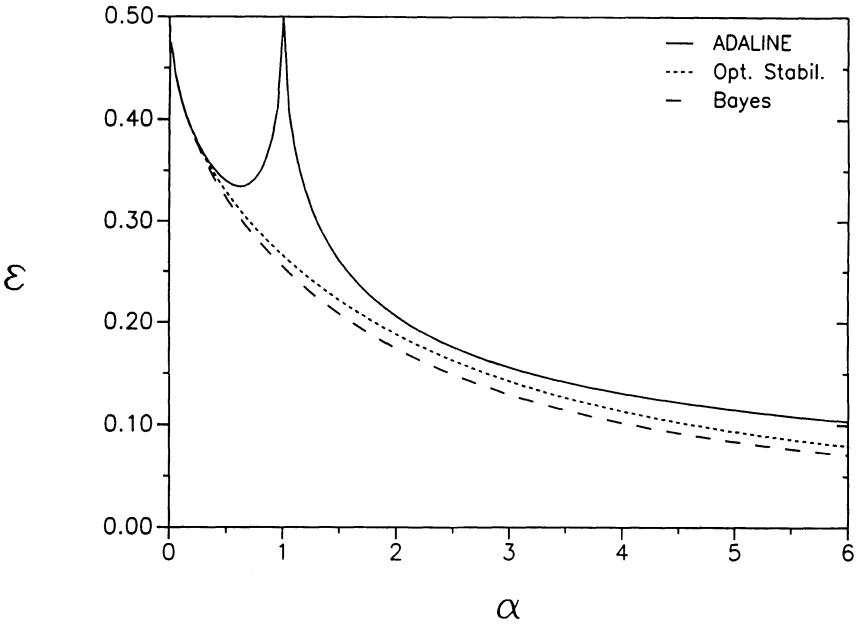


Fig. 5.9. Comparison of generalization errors for ADALINE learning, perceptron learning with optimal stability, and the optimal Bayes prediction.

For small α values, Eq. (5.125) agrees with the Hebbian rule equations (5.110) and (5.113). In fact, the ADALINE algorithm only adds an additional weight to the Hebbian term $\sigma_k \, \xi_k/\sqrt{N}$ that is small for small α. For large α, one finds

$$\varepsilon \propto 1/\sqrt{\alpha}, \tag{5.127}$$

which is again the result of the Hebbian network. Note that in both cases the training error is nonzero.

It is interesting to note that the results for the order parameters [Eqs. (5.125) and (5.126)] can be obtained without the replica method, by an explicit calculation of the coupling vectors [60]. This will be shown in Appendix 5.2.

Finally, we want to mention that the *dynamics* of ADALINE learning can be solved exactly, in contrast to nonlinear learning rules [32, 33, 34, 35]. It can easily be shown that the dynamics is governed by the spectrum of eigenvalues of the matrix

$$B_{ij} = \frac{1}{N} \sum_{k=1}^{P} \xi_k(i) \, \xi_k(j). \tag{5.128}$$

B measures correlations between different input bits; note that at each input unit i the P different training vectors define a P–dimensional vector,

and Eq. (5.128) gives the product of those vectors. B is a kind of random matrix; its spectrum is a distorted semicircle between the values $(1 \pm \sqrt{\alpha})^2$ with an additional degenerate eigenvalue 0 for $\alpha < 1$, [32, 1]. One finds that for $\alpha \to 1$ the longest relaxation time diverges like $|\sqrt{\alpha} - 1|^{-2}$. Hence, one obtains a critical slowing down at the transition to perfect learning.

5.5.3 Maximal Stability

The simple perceptron $\mathbf{w}_s$ has learned an example $\boldsymbol{\xi}_k$ if

$$\operatorname{sign}\left(\frac{1}{\sqrt{N}}\,\mathbf{w}_t \cdot \boldsymbol{\xi}_k\right) = \operatorname{sign}\left(\frac{1}{\sqrt{N}}\,\mathbf{w}_s \cdot \boldsymbol{\xi}_k\right) . \tag{5.129}$$

Its ability to generalize is related to the fact that the sign function maps *similar* input vectors $\boldsymbol{\xi}$ to the *same* output bit σ_k. But from the above equation it is obvious that this property is optimal if the quantity

$$\Delta_k = \frac{\sigma_k}{\sqrt{N}}\,\mathbf{w}_s \cdot \boldsymbol{\xi}_k \tag{5.130}$$

is as large as possible (for fixed norm $\mathbf{w}_s \cdot \mathbf{w}_s/N$). The quantity

$$\Delta = \min_k \frac{\sigma_k \sqrt{N}\,\mathbf{w}_s \cdot \boldsymbol{\xi}_k}{|\mathbf{w}_s|} \tag{5.131}$$

is called the *stability* of the perceptron, and a good learning algorithm should maximize the stability Δ. For attractor networks, a similar relation is assumed between the stability and the size of the basin of attraction [12].

Equation (5.131) can be related to quadratic optimization with boundary conditions:

$$\begin{gathered}\text{Minimize } \mathbf{w} \cdot \mathbf{w} \\ \text{with the conditions } \tfrac{\sigma_k}{\sqrt{N}}\,\mathbf{w} \cdot \boldsymbol{\xi}_k \geq 1 \\ \text{for all patterns } \boldsymbol{\xi}_k .\end{gathered}$$

It turns out that the optimal perceptron $\mathbf{w}_s$ classifies the training examples into two classes [36]: One set of patterns is right at the boundary $\Delta_k = 1$, and the second set is in the interior of the allowed region. But only the first set has to be learned by the perceptron, namely, one has

$$\mathbf{w}_s = \frac{1}{\sqrt{N}} \sum_k x_k\,\boldsymbol{\xi}_k \tag{5.132}$$

with coefficients x_k that are 0 for the second set. The number $\alpha_{eff}N$ of the examples belonging to the first set can be calculated by the replica method; it is shown in Fig. 5.10 as a function of αN many random examples $\boldsymbol{\xi}_k$. α_{eff} remains smaller than 1, even for $\alpha \to \infty$. Only $\alpha_{eff}N$ many examples

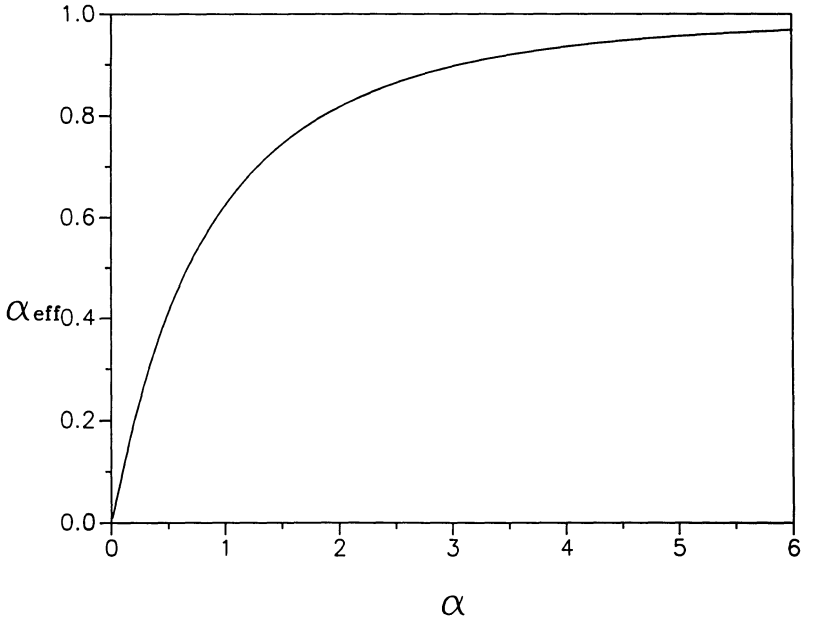


Fig. 5.10. Effective number of inputs per weight to be learned by the perceptron with optimal stability.

have to be learned by the network; and for these the ADALINE learning rule is sufficient, leading to $E = 0$ in Eq. (5.120). For large α, most of the added training examples are useless since they give $\Delta_k \geq 1$ and do not change the student $\mathbf{w}_s$ (they are not learned). Of course, this is related to the fact that the generalization error is small.

Optimal stability has a surprising geometric implication: Consider a set of αN many random points ξ on the unit hypersphere in αN-dimensional space. Label each point black or white randomly. Then a two-dimensional projection of the points looks like Fig. 5.11(a). For $N \to \infty$ and $\alpha < 2$, a perceptron with optimal stability $\Delta > 0$ exists (Sec. 5.3.1). This means that there is a weight vector $\mathbf{w}_s$, and a two–dimensional projection on a plane containing $\mathbf{w}_s$ looks like Fig. 5.11(b). Now black points are separated from white ones and there is a gap Δ between the two clouds. Precisely at the boundary planes of the gap there are $\alpha_{eff} N$ many points. Hence, just by rotating the cloud of random points one can find a view where the black and white points are clearly separated.

There is an interesting general relation between α_{eff} and ε, which holds independently of the distribution of the inputs. Consider the case where a $P+1$st example (ξ, σ) is added to the training set of $P = \alpha N$ inputs. If we run our algorithm on this new, enlarged set, the coupling vector of the P input problem is only changed if

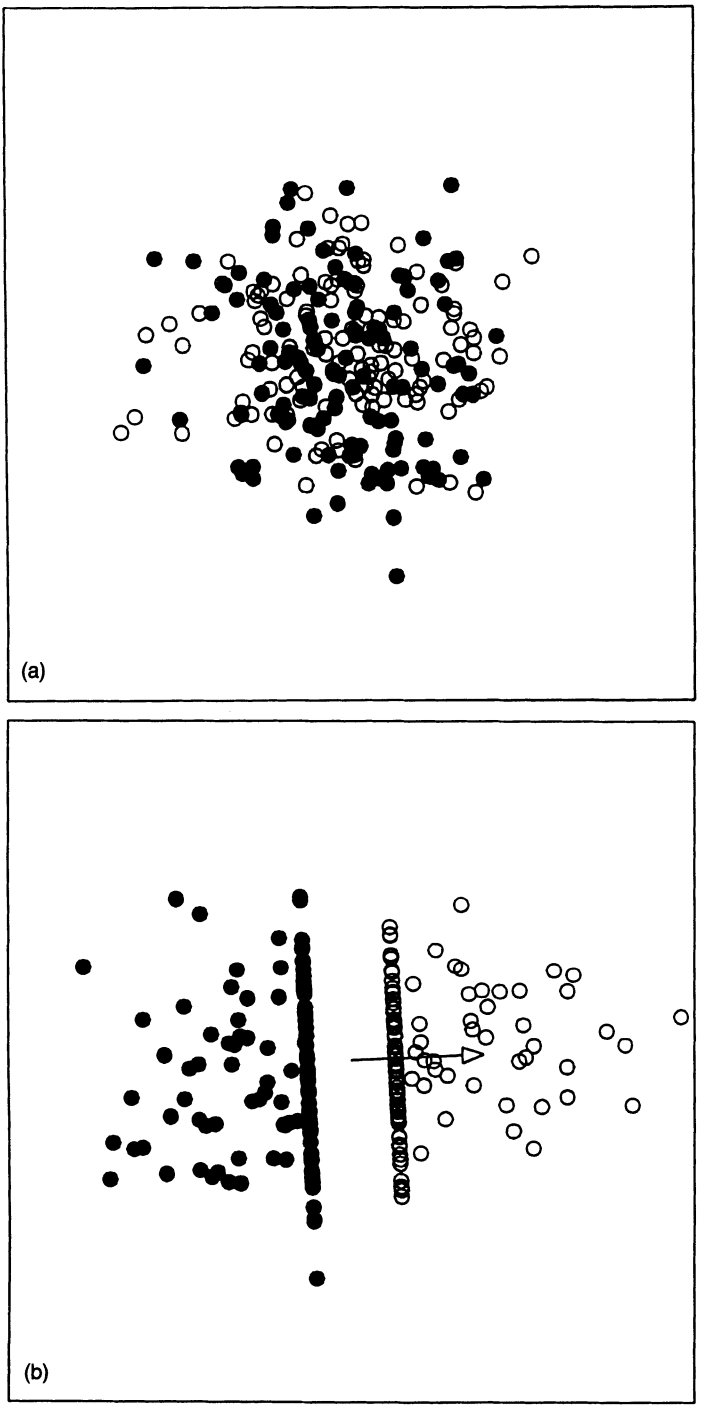


Fig. 5.11. Separating random inputs with a perceptron of maximal stability. The classifications are random (no teacher). (a) Projection onto a random plane. (b) Projection onto a plane containing $\mathbf{w}_s$. The arrow shows the student vector $\mathbf{w}_s$.

$$N^{-1/2}\sigma \mathbf{w} \cdot \boldsymbol{\xi} < 1,$$

where $\mathbf{w}$ is the vector of the old couplings. If, on the other hand,

$$N^{-1/2}\sigma \mathbf{w} \cdot \boldsymbol{\xi} \geq 1,$$

the old couplings also provide optimal stability for the $P+1$ pattern system. If this happens, then the new pattern is *uncorrelated* to $\mathbf{w}$.

Let $\mathcal{P}_0$ be the probability over the distribution of the new input for this event. It turns out that $\mathcal{P}_0$ and the probability G for a *correct* generalization on the new input are rather similar:

$$\begin{aligned} \mathcal{P}_0 &= \Pr(N^{-1/2}\sigma \mathbf{w} \cdot \boldsymbol{\xi} \geq 1) \\ G &= \Pr(N^{-1/2}\sigma \mathbf{w} \cdot \boldsymbol{\xi} \geq 0). \end{aligned} \tag{5.133}$$

Since $1 > 0$, it is clear that $P_0 \leq G$, so that we have for the generalization error

$$\varepsilon = 1 - G \leq 1 - \mathcal{P}_0. \tag{5.134}$$

Now, $\alpha\mathcal{P}_0$ is the average, relative number of patterns that need not be learned explicitly. Conversely, $\alpha_{eff} = \alpha(1 - \mathcal{P}_0)$ is the average fraction of patterns that must be learned. Since we always have $\alpha_{eff} \leq 1$, we get from Eq. (5.134)

$$\alpha\varepsilon = \alpha(1 - G) \leq \alpha_{eff} \leq 1. \tag{5.135}$$

Thus, we will always have $\varepsilon \leq 1/\alpha$.

There are several algorithms that are guaranteed to find the optimal perceptron for αN many random examples $\boldsymbol{\xi}_k$. Unfortunately, one cannot classify the examples according to Eq. (5.132) *in advance*; hence, one has to learn all of them instead of a fraction α_{eff}/α of them, which becomes very small for $\alpha \to \infty$.

One algorithm (Minover [38]) is an extension of the standard Rosenblatt [37] rule; another faster algorithm (Adatron [39]) is related to the quadratic optimization discussed above. But algorithms derived from standard optimization theories also have been developed [40].

All of these algorithms converge to the perceptron with maximal stability. Its properties have been calculated in [31] using the replica method of Gardner [12] introduced in Sec. 5.3.2. Now the function Φ of Eq. (5.66) is

$$\Phi(u_a) = \Theta(u_a - \kappa), \tag{5.136}$$

and $\Phi_t(u) = \Theta(u)$. Maximizing κ shrinks the volume in student space $\mathbf{w}_s$ to a single point, and the overlap $q = \mathbf{w}_a \cdot \mathbf{w}_b/N$ approaches the square of the norm, i.e., $q \to q_0 = \mathbf{w}_s \cdot \mathbf{w}_s/N$.

Figure 5.9 shows the generalization error as a function of the size α of the training set. ε decreases monotonically and behaves like

$$\varepsilon \simeq \frac{0.50}{\alpha} \tag{5.137}$$

for $\alpha \to \infty$. Hence, asymptotically, the perceptron with optimal stability can generalize much better than the Hebbian or ADALINE rule (for which $\varepsilon \simeq 1/\sqrt{\alpha}$). It performs only slightly worse than the Bayesian lower bound of Sec. 5.3.3, although this difference means that maximal stability does not imply optimal generalization.

5.5.4 QUERIES

In the previous applications of the statistical mechanics of neural networks only random input patterns were considered. However, it seems obvious that the student network can improve its generalization performance if it selects input patterns according to its present state of knowledge. In particular, if the fraction α of learned examples is large, a new random plane is unlikely to cut the (small) version space into two parts of roughly the same size; hence, the gain of information about the teacher is very small (see Sec. 5.2.4).

Much more information can be obtained if the student selects a question according to its present state [41]. For the simple perceptron, a good choice seems to be a pattern $\boldsymbol{\xi}_k$ that is perpendicular to the weight vector $\mathbf{w}_s$. Such a pattern is at the border of knowledge; tiny chances of $\mathbf{w}_s$ produce different answers.

For the simplest learning rule, e.g., the Hebbian algorithm discussed in Sec. 5.5.1, one easily obtains a differential equation for the overlap R and the length q_0, which determine the generalization error. Equation (5.111) gives $\Delta q_0 = 1/N$ since $\boldsymbol{\xi}_k \cdot \mathbf{w}_s = 0$ by construction. Hence, one has

$$q_0 = \alpha \, . \tag{5.138}$$

But $\overline{|\mathbf{w}_t \cdot \boldsymbol{\xi}_k|}$ of Eq. (5.108) also can be easily calculated. With $|\mathbf{w}_t| = \sqrt{N}$, the component of $\mathbf{w}_t$ perpendicular to $\mathbf{w}_s$ has a length $\sqrt{N}\sin\theta$, where θ is the angle between the teacher and the student vectors. If $\boldsymbol{\xi}_k$ is chosen randomly in the plane perpendicular to $\mathbf{w}_s$, then $\mathbf{w}_t \, \boldsymbol{\xi}_k$ is Gaussian distributed with variance

$$\begin{aligned} \overline{(\mathbf{w}_t \cdot \boldsymbol{\xi}_k)^2} &= N \sin^2 \theta \\ &= N(1 - \cos^2 \theta) \, . \end{aligned} \tag{5.139}$$

With $\cos\theta = R/\sqrt{q_0} = R/\sqrt{\alpha}$, one finds

$$\frac{1}{\sqrt{N}} \overline{|\mathbf{w}_t \, \boldsymbol{\xi}_k|} = \sqrt{\frac{2}{\pi}} \sqrt{1 - \frac{R^2}{\alpha}}, \tag{5.140}$$

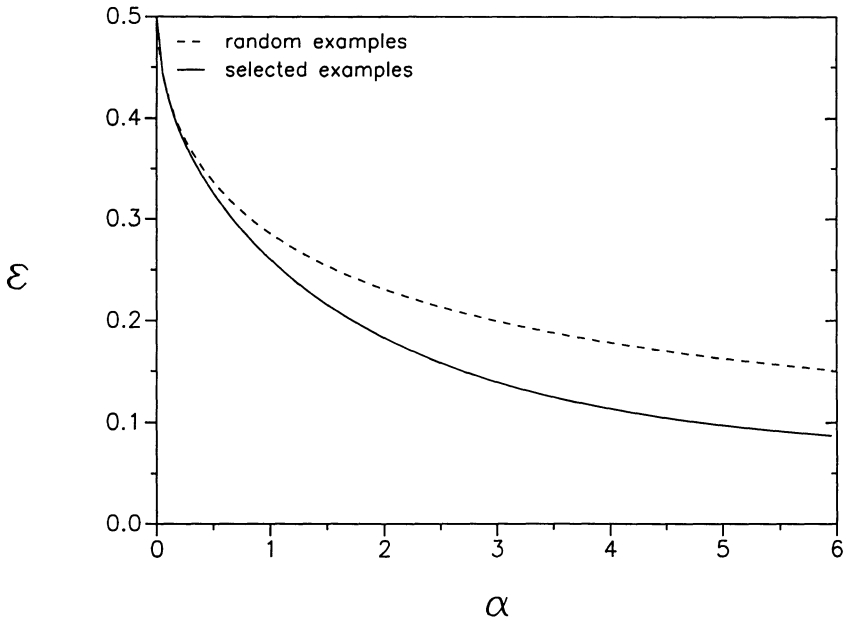


Fig. 5.12. Comparison of learning with selected and random inputs, using Hebb's rule.

which gives, with Eq. (5.108),

$$\frac{dR}{d\alpha} = \sqrt{\frac{2}{\pi}} \sqrt{1 - \frac{R^2}{\alpha}} \,. \tag{5.141}$$

The solution $R(\alpha)$ determines the generalization error $\varepsilon(\alpha)$ by $\varepsilon = \arccos(R/\sqrt{\alpha})/\pi$.

Figure 5.12 compares the results of random and selected examples. Although the generalization error is lower for "intelligent" questions, the asymptotic decay for large values of α is $\varepsilon \propto 1/\sqrt{\alpha}$ for both cases.

This is different if the whole set of examples is relearned after a new pattern was selected. Then the perceptron with maximal stability gives an exponential decay of the generalization error with an increasing fraction α of the number of learned examples [41, 42]. For *random* patterns, the Bayesian bound of Sec. 3.3.3 as well as the optimal perceptron give $\varepsilon \propto 1/\alpha$. Hence, in this case, selected examples give much better performance.

For more complicated networks it may be difficult to find patterns at the border of knowledge: An algorithm has been investigated that uses the principle of maximal disagreement between several students as a selection process [43]. Several students are trained on the same set of examples by an algorithm that selects students randomly in the version space. Then an algorithm starts which selects a new example for the training set: Many

random input vectors are presented to the students, and one is chosen on which the students disagree most. This problem has been solved using the replica theory. For large α, the gain of information becomes constant, yielding an exponential decay of the generalization error; this even holds for only two students.

Selecting examples according to the weight vector $\mathbf{w}_s$ (or several vectors $\mathbf{w}_s$) may not be the best way of selecting examples. If the student learns a new example that is perpendicular to all of the previous ones, the generalization error is much lower than for the examples perpendicular to the actual $\mathbf{w}_s$ (α) [2]. However, this algorithm works only for $\alpha < 1$.

5.5.5 Discontinuous Learning

If one increases the number of examples, one expects that the generalization error of a network continuously decreases to its minimal possible value for $\alpha \to \infty$. If the rule is completely learnable, then the asymptotic error is 0, at least for perfect learning. However, a different behavior is observed for perceptrons with binary weights: For small α, ε decreases; but at a critical value α_c, ε jumps discontinuously to a lower value that is 0 for a realizable rule [44, 45]. This transition occurs even for high-temperature learning. In this case, it can be easily described analytically, since one does not need replicas [47]. We consider the case where both the teacher and the student are simple perceptrons with binary weights $\mathbf{w}_s, \mathbf{w}_t \in \left\{1/\sqrt{N},\, -1/\sqrt{N}\right\}^N$. The student learns a set of αN many examples $(\boldsymbol{\xi}_k,\, \sigma_k)$ from the teacher, and the training algorithm is a Monte Carlo procedure. After learning each weight vector, $\mathbf{w}_s$ occurs with probability

$$p(\mathbf{w}_s) \;\propto\; \exp\left[-\beta \sum_k \Theta\left[-(\mathbf{w}_t \cdot \boldsymbol{\xi}_k)(\mathbf{w}_s \cdot \boldsymbol{\xi}_k)\right]\right] . \tag{5.142}$$

For high temperatures, $T = 1/\beta$, the free energy f per synapse of the thermal equilibrium after learning is only a function of the overlap R between the teacher and the student:

$$-\beta f = -\frac{\alpha\beta}{\pi} \arccos\, R - \frac{1-R}{2} \ln \frac{1-R}{2} - \frac{1+R}{2} \ln \frac{1+R}{2} . \tag{5.143}$$

The first term is the generalization error, and the second term is the entropy of Ising variables with magnetization R. Note that T and α appear only as

$$\alpha_{eff} = \alpha/T . \tag{5.144}$$

Hence, in the limit $T \to \infty$, the network has to learn $\alpha \to \infty$ many examples. The minimum of $f(R)$ gives the equation that determines the overlap R:

$$R = \tanh \frac{\alpha\beta}{\pi\sqrt{1-R^2}} . \tag{5.145}$$

Solving these equations, one finds three different regimes of α_{eff}:

1. For $\alpha_{eff} < 1.7$, a state with $R < 1$ is the minimum of f. The generalization error decreases from $\varepsilon = \frac{1}{2}$ at $\alpha_{eff} = 0$ to $\varepsilon \simeq 0.2$ at $\alpha_{eff} = 1.7$.

2. Between $1.7 < \alpha_{eff} < 2.1$, the state with $R < 1$ is a local minimum, only; the state $R = 1$ has lower free energy. Hence, the system has a first-order transition to perfect generalization.

3. For $\alpha_{eff} > 2.1$, the metastable state with $R < 1$ disappears.

Note that for large α the network collapses to its ground state at *high temperatures*! To understand this, consider a small deviation $\delta R = 1 - R$ from the state of perfect generalization. The energy increases like $E \propto N\sqrt{\delta R}$. This increase cannot be compensated for by the entropy increase $\delta S \propto N(\delta R)\ln(\delta R)$. Hence, the state $R = 1$ is always a local minimum of $f(R)$; and if the initial state of the student is identical to the teacher, then no Monte Carlo algorithm can move the student out of this state of perfect generalization. Increasing the complexity of the student network by using a multilayer architecture with binary weights leads to even more phases and discontinuous transitions of the generalization error [46].

At zero temperature, i.e., for perfect learning, the first-order transition for the binary weight perceptron occurs at $\alpha_c = 1.245$ [45]. Approaching the transition from below, $\alpha \to \alpha_c$, the entropy S obtained from the number of weight vectors $\mathbf{w}_s$ that learn αN many examples perfectly goes to 0. S has been calculated by the replica method.

Figure 5.13 shows the phase diagram of the binary perceptron obtained from replica calculation including replica symmetry breaking (RSB) [47]. In addition to the three phases discussed above, there is a spin-glass phase where a solution with one-step RSB exists; this solution is metastable. The spin-glass phase indicates a complex space of students $\{\mathbf{w}_s\}$ who learn perfectly. Its implications for a dynamics of the binary weight perceptron are still unclear. A direct treatment of the dynamics gives new types of freezing transitions [48].

How many questions does one have to ask in order to obtain a complete knowledge about the N unknown weights of the student $\mathbf{w}_s$? For binary weights, one needs at least N questions; hence, the minimal possible number of patterns for which a transition to perfect generalization occurs is N. This gives a lower bound

$$\alpha_c > 1 \; . \tag{5.146}$$

Therefore, learning random patterns with a transition $\alpha_c = 1.245$ is not the optimal way of asking questions. A better strategy seems to be learning patterns at the border of knowledge, as was discussed in Sec. 5.5.4. In fact, a replica calculation gives $\alpha_c \simeq 1.14$ [42].

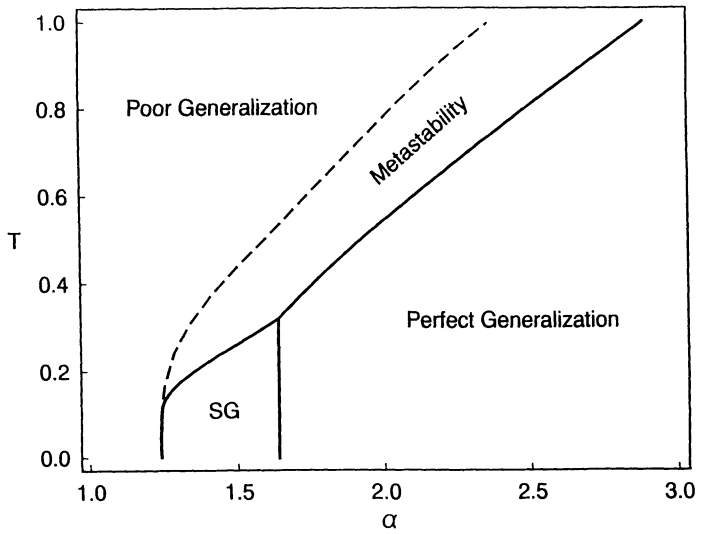


Fig. 5.13. Phase diagram for the perceptron with binary weights (taken from [47]). To the left of the dashed line, the equilibrium state has $R < 1$. To the right, the state of perfect generalization ($R = 1$) is the absolute minimum of the free energy. Between the dashed line and the solid spinodal line, the $R < 1$ state is metastable. In the region marked by "SG," a one-step replica symmetry breaking predicts a metastable spin-glass phase.

For $T = 0$, all results so far have been obtained by phase-space calculations. This means that one calculates the volume of all students who learn perfectly. However, a practicable training algorithm does not exist yet. In fact, finding a $\mathbf{w}_s$ may be an NP–hard problem of combinatorial optimization [48, 49], at least for $\alpha < 1.63$ (the upper limit of the spin-glass phase), for which an algorithm converging in a time that is a polynomial in N does not exist. Then even simulated annealing does not yield perfect learning.

5.5.6 Learning Drifting Concepts

In the previous sections the examples were given by a rule (= teacher) defined by a perceptron with a stable weight vector $\mathbf{w}_t$. All of the examples were learned iteratively, that is, the training algorithm was repeated for all of the examples until it converged.

But neural networks also may be useful for situations where the rule slowly changes with time, and the network tries to follow the changes by learning only the most recent examples. Hence, the teacher continuously changes his opinion and the student tries to adapt to such a dynamic process by learning the examples and, if possible, predicting $\mathbf{w}_t$ for the next time step.

In the simplest case, the teacher vector $\mathbf{w}_t$ is performing a random walk

in the N-dimensional space [50, 51] with

$$\mathbf{w}_t(t+1) \cdot \mathbf{w}_t(t) = 1 - \frac{\eta}{N}, \tag{5.147}$$

where η is a measure of the *drift velocity*. The student learns only one example $(\boldsymbol{\xi}, \sigma)$ with $\sigma = \text{sign}\ (\mathbf{w}_t \cdot \boldsymbol{\xi})$ given by the teacher at time t. The learning rule uses only information about the output bit σ and the field $h(t)$ of the student. One defines

$$\mathbf{w}_s(t+1) = \mathbf{w}_s(t)\left(1 - \frac{\lambda}{N}\right) + \frac{1}{\sqrt{N}} f\ (\sigma(t), h(t))\ \sigma(t)\ \mathbf{w}_s(t)\ . \tag{5.148}$$

f is a function that has to be optimized, and $h(t)$ is the field generated by the student, $h(t) = (1/\sqrt{N})\ \mathbf{w}_s(t) \cdot \boldsymbol{\xi}$; λ gives an additional weight decay which reduces the length of the student vector $\mathbf{w}_s$.

Again we need the overlaps $R = \mathbf{w}_t \cdot \mathbf{w}_s/N$ and $q_0 = \mathbf{w}_s \cdot \mathbf{w}_s/N$ to determine the generalization error $\varepsilon = (1/\pi)\ \arccos\ \left(R/\sqrt{q_0}\right)$. But, since only the latest example is learned, one obtains simple differential equations for $R(t)$ and $q_0(t)$, in analogy to Sec. 5.5.1. The changes of R and q are given by

$$\begin{aligned} \Delta R &= \frac{1}{N}\left[f(\sigma, h(t))\frac{\mathbf{w}_t \cdot \boldsymbol{\xi}}{\sqrt{N}}\ \sigma_t - (\lambda + \eta)\ R\right] \\ \Delta q_0 &= \frac{2}{N}\left[f(\sigma, h(t))\ \sigma\ h(t) + \frac{1}{2} f^2(\sigma, h(t)) - \lambda q_0\right]\ . \end{aligned} \tag{5.149}$$

These equations have to be averaged over different examples $\boldsymbol{\xi}$ and different random walks of the teacher $\mathbf{w}_t$. For random examples one obtains

$$\begin{aligned} \frac{dR}{d\alpha} &= \overline{f(\sigma, h)\ \frac{\mathbf{w}_t\ \boldsymbol{\xi}}{\sqrt{N}}\ \sigma} - (\lambda + \eta) R \\ \frac{dq_0}{d\alpha} &= \overline{2 f(\sigma, h) \frac{\mathbf{w}_t \cdot \boldsymbol{\xi}}{\sqrt{N}} + \frac{1}{2} f^2(\sigma, h)} - 2\lambda q_0\ . \end{aligned} \tag{5.150}$$

The fields $\mathbf{w}_s \cdot \boldsymbol{\xi}$ and $\mathbf{w}_t \cdot \boldsymbol{\xi}$ are correlated Gaussian variables that allow an easy calculation of the average values $\overline{(\ldots)}$ similar to Sec. 5.5.1. The "time t" has been replaced by αN, the number of learned examples.

As before, the simplest learning rule is the Hebbian one, with $f = 1$. In this case, one finds without decay ($\lambda = 0$):

$$\begin{aligned} R(\alpha) &= \sqrt{\frac{2}{\pi}}\ \frac{1}{\eta}\left(1 - e^{-\eta\alpha}\right) \\ q_0(\alpha) &= \left(1 + \frac{4}{\pi\eta}\right)\alpha + \frac{4}{\pi\eta^2}\ e^{-\eta\alpha} + \left(1 - \frac{4}{\pi\eta^2}\right)\ . \end{aligned} \tag{5.151}$$

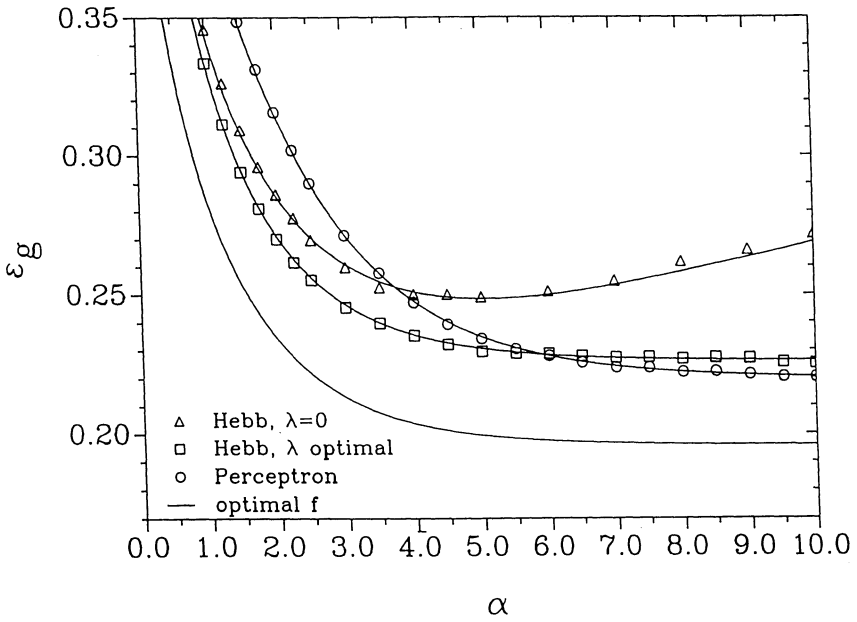


Fig. 5.14. Generalization errors for the learning of drifting concepts, using Hebbian learning (with and without weight decay), the perceptron algorithm, and the on-line algorithm with optimal f given in [53].

Figure 5.14 shows the generalization error $\varepsilon(\alpha)$ given by these equations. It has a minimum at some α value but then increases to $\varepsilon = 1/2$. Hence, the student cannot generalize if he or she has learned too much!

The reason for this surprising feature is the fact that the Hebbian couplings have equal strengths for all of the examples. But, since the teacher changes his or her direction, the examples produced some time ago destroy the most recent information that is important for generalization.

In fact, a weight decay $\lambda > 0$ produces *forgetting* [52]; hence, the error $\varepsilon(\alpha)$ decreases to a stationary value $\varepsilon(\infty)$ that can be minimized with respect to λ; the result is shown in Fig. 5.15. The minimal asymptotic error increases with small drift parameters η as

$$\varepsilon_{opt}(\infty) \simeq \frac{1}{\pi^{3/4}}\, \eta^{1/4} \,. \tag{5.152}$$

A better training algorithm is the perceptron learning rule [1], with $f(\sigma, h) = \theta(\kappa - \sigma h/q_0)$. Now, $\varepsilon(\infty)$ can be minimized with respect to the two parameters κ and λ. One finds [50, 51] for small η values

$$\varepsilon(\infty) \simeq 0.51\, \eta^{1/3} \,. \tag{5.153}$$

The same power of η is found if the learner knows $\varepsilon(\alpha)$ and uses this information to derive an optimal function $f(\sigma, h)$ [53].

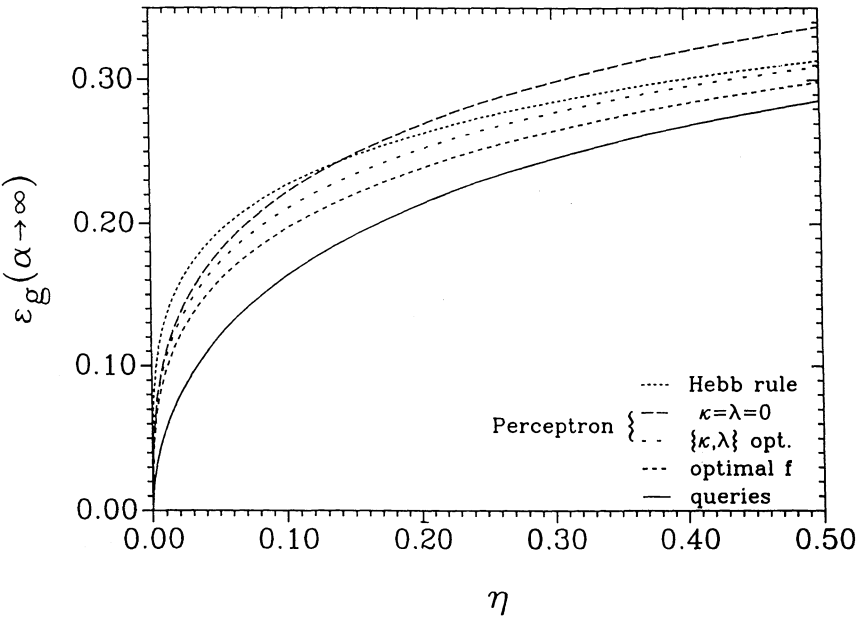


Fig. 5.15. Stationary value $\varepsilon(\infty)$ for learning of drifting concepts using the learning algorithms mentioned in Fig. 5.15. Also, the effect of queries is included.

An additional improvement is obtained by selecting examples as in Sec. 5.5.4. For the Hebb rule with optimal decay, one finds

$$\varepsilon(\infty) = \frac{1}{\pi} \arccos \frac{1}{\sqrt{1+\eta\pi}} \simeq \sqrt{\frac{\eta}{\pi}} \,. \tag{5.154}$$

Using $\varepsilon(\alpha)$ with an optimal f, the generalization error decays exponentially fast to the same asymptotic error [Eq. (5.154)].

Of course, a random walk cannot be predicted by definition. But for deterministic changes of the teacher or for biased random walks it should be possible to predict future actions of the teacher by studying the history of the presented examples. The statistical mechanics of such problems still have to be formulated.

5.5.7 Diluted Networks

In the previous examples the student had the same structure as the teacher. But it may be interesting to study cases where the student has to deduce the *structure* of the teacher from the set of presented examples. A simple case is the *diluted* teacher: Both teacher and student are simple perceptrons, but a certain fraction f of the couplings is erased. This means that the teacher has a fixed set of weights that are equal to 0, and the student also has a

fixed fraction of 0 weights, but he or she is allowed to choose which bonds are to be erased.

Hence, the student has additional dynamic variables $\mathbf{c} \in \{0,1\}^N$, which are multiplied with the weights $\mathbf{w}_s \in R^N$. For this problem, the perceptron of optimal stability can be calculated using the phase-space integral of Gardner, but now with the additional discrete variables $\mathbf{c}$ [54]. One obtains the generalization error ε as a function of α, f_s, f_t, where f_s and f_t are the fraction of nonzero bonds of the student and teacher, respectively. One finds that ε has a minimum as a function of f_s, and for large α this minimum approaches $f_s \simeq f_t$.

Again, the replica calculation does not provide us with a learning algorithm. Finding the optimal configuration $\mathbf{c}$ is presumably an NP–hard problem of combinatorial optimization, similar to the binary perceptron. Therefore, a practicable algorithm does not exist, yet. However, one might guess that, by learning the complete network and by erasing the weak bonds, one may obtain a good approximation of the optimal perceptrons. In fact, this is the case for attractor networks ($\hat{=}$ random teacher) [55].

A fast and effective dilution algorithm is given by the Hebbian couplings:

$$c_i = 0 \quad \text{if} \quad \left| \frac{1}{N} \sum_{\nu=1}^{\alpha N} \xi_i^\nu \, \sigma^\nu \right| < s, \tag{5.155}$$

where s is determined by f_s. For this fixed dilution vector $\mathbf{c}$, the remaining weights are determined by the standard algorithms for the perceptron of optimal stability [1].

The generalization error ε has been calculated analytically [54]. The order parameters now are defined by

$$q = \frac{1}{N f_s} \sum_{i=1}^{N} c_i w_a(i) \; w_b(i)$$

$$R = \frac{1}{N \sqrt{f_s f_t}} \sum_{i=1}^{N} c_i w_t(i) \; w_s(i). \tag{5.156}$$

R and q determine ε as usual. Figure 5.16 shows the result. For $f_s < f_t$, the target rule is unrealizable; the student cannot reproduce the teacher perfectly, even for $\alpha \to \infty$. For $f_s > f_t$, the student has too many degrees of freedom, which deteriorates his or her ability to generalize. Hence, ε has a maximum that approaches $f_s \to f_t$ for a large fraction α of learned examples.

Note that f_s is a fixed parameter. What remains is to find an algorithm that determines the optimal dilution fraction f_s of the student. By such a learning rule the student would be able to explore the structure of the teacher.

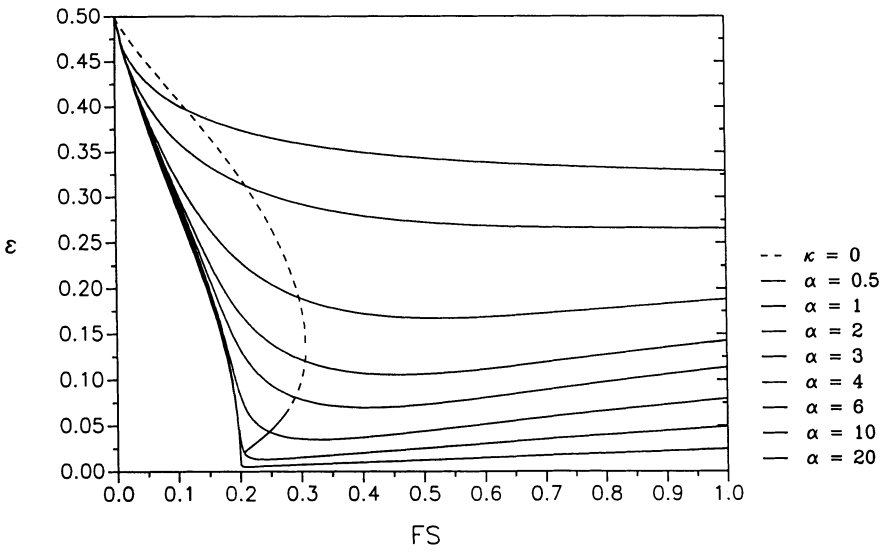


Fig. 5.16. Generalization error as a function of student dilution f_s for a teacher with dilution $f_t = 0.2$. The dashed curve separates training with errors (*left*) and without errors (*right*).

5.5.8 Continuous Neurons

Up to now we have mainly discussed output neurons with the step transfer function sign (x). The teacher as well as the student is a network with binary output $\sigma \in \{+1, -1\}$. But functions with continuous output also are interesting. First, they model a continuous firing rate as a function of excitation potential for real neurons; and, second, tasks for neurocomputers may involve analog signals, and learning rules like gradient descent work only for continuous functions [56].

In the context of statistical mechanics, continuous neurons have been studied for a simple perceptron [57]. The teacher and the student are perceptrons with weight vectors $\mathbf{w}_t$ and $\mathbf{w}_s$, respectively. But now the output signal is given by

$$\sigma = \tanh\left(\frac{\gamma}{\sqrt{N}}\,\mathbf{w}\cdot\boldsymbol{\xi}\right), \tag{5.157}$$

where γ is a parameter that measures the degree of nonlinearity of the transfer function. An increase of the student's length $q_0 = \mathbf{w}_s \cdot \mathbf{w}_s/N$ can be compensated for by a decrease of the slope γ_s of Eq. (5.157). Hence, only the product $\gamma_s^2\, q_0$ has a physical meaning, and the student has the freedom to adjust its slope γ_s during learning. In [57], $q_0 = 1$ was chosen.

Learning again is expressed as minimizing a cost function E, which is

chosen as the quadratic deviation

$$E = -\sum_k \left[\tanh\left(\frac{\gamma_t}{\sqrt{N}}\,\mathbf{w}_t \cdot \mathbf{w}_k\right) - \tanh\left(\frac{\gamma_s}{\sqrt{N}}\,\mathbf{w}_s \cdot \boldsymbol{\xi}_k\right)\right]^2 . \quad (5.158)$$

By defining

$$\sigma_k = \left(\frac{\gamma_t}{\sqrt{N}}\mathbf{w}_t \cdot \boldsymbol{\xi}_k\right), \quad (5.159)$$

one observes that $E = 0$ implies $\tilde{E} = 0$ for the function

$$\tilde{E} = -\sum_k \left[\sigma_k - \frac{\gamma_s}{\sqrt{N}}\,\mathbf{w}_s \cdot \boldsymbol{\xi}_k\right]^2 . \quad (5.160)$$

This is just the cost function for a linear network! For $\alpha < 1$, $\tilde{E} = 0$ gives less equations than unknowns, and the solution with minimal γ_s (corresponding to minimal norm) is given by the pseudoinverse as in Sec. 5.5.2. Using the replica method, one finds [57]

$$\gamma_s = \sqrt{\alpha}\gamma_t, \qquad R = \sqrt{\alpha}\,. \quad (5.161)$$

For $\alpha > 1$, $E = \tilde{E} = 0$ gives perfect generalization with $\gamma_s = \gamma_t$ and $R = 1$.

The generalization error ε can be defined by

$$\varepsilon = \frac{1}{2}\overline{\left[\tanh\left(\frac{\gamma_t}{\sqrt{N}}\,\mathbf{w}_t \cdot \boldsymbol{\xi}\right) - \tanh\left(\frac{\gamma_s}{\sqrt{N}}\,\mathbf{w}_s \cdot \boldsymbol{\xi}\right)\right]^2}, \quad (5.162)$$

i.e., the quadratic deviation between the answers of the teacher and the student for random patterns.

One finds for $\alpha < 1$

$$\varepsilon = \frac{1}{2}\int_{-\infty}^{\infty} Dx \int_{-\infty}^{\infty} Dy \left[\tanh(\gamma_t x) - \tanh\left(\gamma_t\sqrt{\alpha(1-\alpha y)} + \gamma_t \alpha x\right)\right]^2 \quad (5.163)$$

while $\varepsilon = 0$ for $\alpha > 1$. Figure 5.17 shows the results $\varepsilon(\alpha)$ for different teacher slopes γ_t. Surprisingly, for $\gamma_t > 1.33$, the generalization error increases with α when only a small number α of examples has been learned.

5.5.9 Unsupervised Learning

In the previous sections, a teacher function presented answers to random inputs to a student network. Hence, the teacher classified the input patterns.

However, sometimes one would like to find a classification of input patterns *without* knowing the answer of a teacher; hence, the input patterns

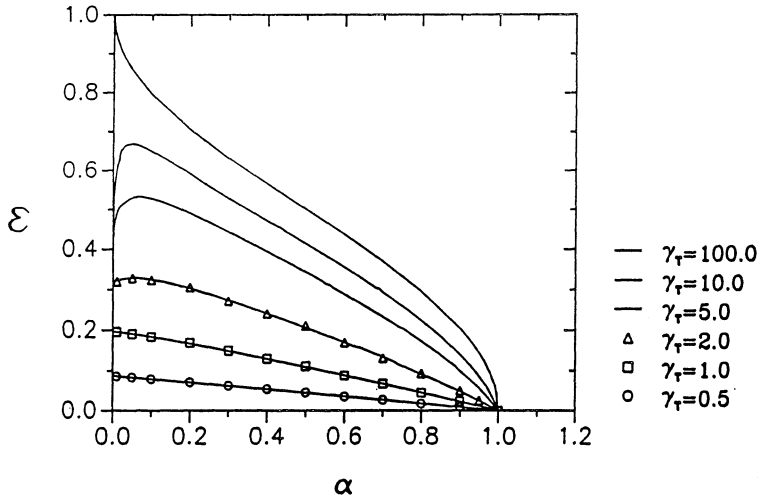


Fig. 5.17. Generalization errors for student and teacher with a "tanh($\gamma\cdot$)" transfer function. γ_T is the gain factor of the teacher.

$\boldsymbol{\xi}_k$ have a structure that the student network has to find out. Recently this problem of unsupervised learning was studied in the framework of the statistical mechanics of simple perceptrons [58].

The inputs are no longer completely random, but they have an internal structure defined by a teacher vector $\mathbf{w}_t$: The patterns $\boldsymbol{\xi}_k$ belong to two "clouds" with respect to the overlap to the teacher. The distribution of $u = \mathbf{w}_t \cdot \boldsymbol{\xi}_k/\sqrt{N}$ is a double Gaussian, that is, each peak has a width 1 and the two peaks are separated by $2\rho/\sqrt{N}$ with a parameter $\rho = 0(1)$. Note that the patterns have only a very weak overlap $\rho/\sqrt{N}$ with the teacher vector $\mathbf{w}_t$. In contrast to the previous problems, the student does not know the sign of u.

Learning again is expressed as minimizing a cost function E, and statistical mechanics of the phase space of all students $\mathbf{w}_s$ is used to calculate the overlap $R = \mathbf{w}_t \cdot \mathbf{w}_s/N$ after having learned αN many examples $\boldsymbol{\xi}_k$ taken from the double peak distribution.

Two cost functions have been considered:

$$E_A = -\frac{1}{N}\sum_k (\mathbf{w}_s \cdot \boldsymbol{\xi}_k)^2 \tag{5.164}$$

$$E_B = -\sum_k \theta(\kappa - |\mathbf{w}_s \cdot \boldsymbol{\xi}_k|/\sqrt{N}) \; . \tag{5.165}$$

The first corresponds to principal component analysis [56] and the second to finding the perceptron of maximal stability κ, i.e., one maximizes κ with $E_B = 0$, if possible.

In both cases, one finds a critical value α_c below which the student cannot generalize ($R = 0$). Only if the number of learned examples is larger

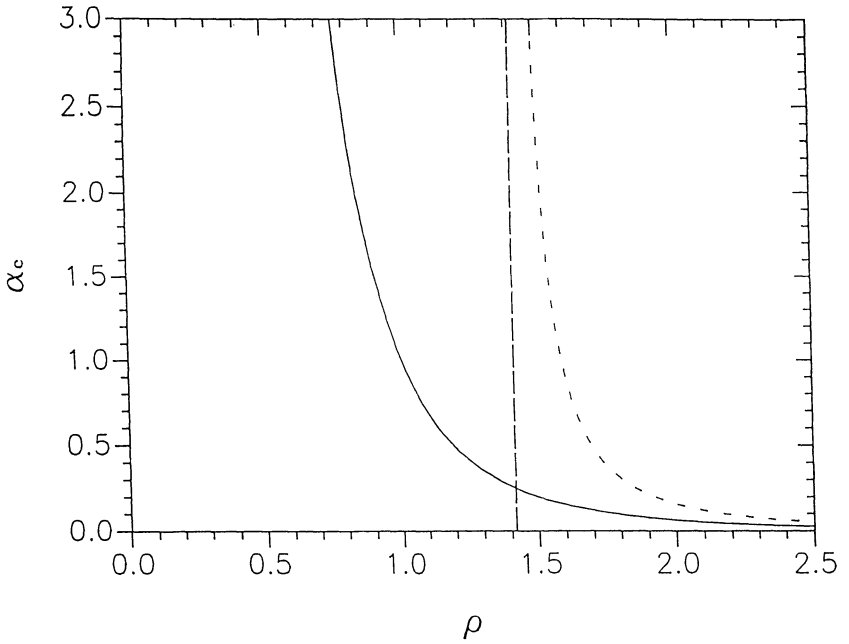


Fig. 5.18. Critical number of inputs/weight, below which unsupervised learning is impossible for: principal component analysis (solid curve) and maximal stability (dashed curve). The vertical line is $\rho = \sqrt{2}$.

than $\alpha_c N$ can the student develop an overlap with the teacher direction. Of course, the sign of the classification cannot be deduced, since it is not shown by the teacher (unsupervised learning). Figure 5.18 shows the critical number α_c as a function of ρ, which measures how strong the double peak structure of the cloud of patterns shows up. For the first case, α_c diverges with $\rho > 0$, and one finds

$$\alpha_c = \rho^{-1/4} \ . \tag{5.166}$$

But, surprisingly, the perceptron with maximal stability cannot generalize if the distinction ρ of the two classes of input patterns is smaller than $\rho_c = \sqrt{2}$. And, even for $\alpha \to \infty$, the generalization error does not decrease to 0, but one has

$$R(\alpha \to \infty) = \pm \sqrt{1 - 2/\rho^2} \qquad \text{for} \quad \alpha > \sqrt{2} \ . \tag{5.167}$$

5.6 Summary and Outlook

We set out to convince the reader that the study of simple mathematical models is a promising way to understand at least part of a neural network's

abilities to learn from examples. Thus, in the first part of this chapter we tried to review a few of the basic theoretical ideas and tools which are currently discussed in the computer science and statistical physics literature on neural networks.

The *Vapnik–Chervonenkis* method, well known in theoretical computer science, is able to bound the generalization error using only a single parameter of the class of networks, rather than their complete architecture.

The statistical physicist's tools, which mainly are based on the *replica method*, are designed for very large nets and allow for the exact calculation of learning curves in a variety of circumstances. Here, however, one is practically restricted to simple architectures and some hopefully "natural" probability distributions for the examples to be learned.

In the second part of the chapter we concentrated on the statistical physicist's methods and presented a variety of learning problems that can be treated *exactly* for a special network, the perceptron, which is far from being a toy model. Although there is a great interest to study more complicated, multilayer nets [2], the amount of recent results for perceptrons suggests that there are still more interesting facts to be discovered for this machine.

We found a rich structure of *learning curves* that may not be easily recovered within the VC framework. This stems from the fact that problems such as *overfitting, discontinuous learning*, or *intelligent dilution* are essentially related to either specific learning algorithms or specific features of the network architecture. On the other hand, comparing the VC predictions and the concrete learning curves for the perceptron, we found that the VC bounds match the correct order of magnitude for the typical *asymptotic* behavior in many cases. Thus, it seems that the asymptotic region can be estimated correctly by using only a *few parameters* of a neural network.

It would be a challenge to combine statistical physics methods based on the replica trick and the VC techniques. Such an approach may be helpful and important in treating multilayer nets when the complex structure of the network's phase space makes exact replica calculations a hard task.

Acknowledgments. We thank Andreas Mietzner for assistance and acknowledge support from the Deutsche Forschungsgemeinschaft and the Volkswagenstiftung.

Appendix 5.1: Proof of Sauer's Lemma

As can be easily seen, the first inequality of Eq. (5.4) is proved if we can show the following *theorem*:

Consider a sequence of inputs $\boldsymbol{\xi}^P = \boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_P$. If there is an integer d,

such that the number $\mathcal{N}(\boldsymbol{\xi}^P)$ of cells or output configurations fulfills

$$\mathcal{N}(\boldsymbol{\xi}^P) > \sum_{i=0}^{d} \binom{P}{i}, \tag{5.168}$$

then we can find a subsequence of these inputs, of length $d+1$, for which

$$\mathcal{N}(\boldsymbol{\xi}^{d+1}) = 2^{d+1}.$$

The proof is by induction on P and d. It is easy to see that the theorem holds $d = 0$. It also holds for any $P \leq d$ because, in this case, the premise (5.168) can never be fulfilled: The sum of binomials is then $\geq 2^P$. But $\mathcal{N}(\boldsymbol{\xi}^P)$ must always be $\leq 2^P$.

Let the assertion be true for all $d \leq d_0$ and all numbers of inputs. Now, assume that the theorem is also true for $d = d_0 + 1$ and for $P < P_0$ inputs. We then will show that it holds for all P.

We add a $P_0 + 1$st input $\boldsymbol{\xi}$ and assume the premise (5.168):

$$\mathcal{N}(\boldsymbol{\xi}^{P_0+1}) > \sum_{i=0}^{d_0+1} \binom{P_0+1}{i}. \tag{5.169}$$

If, on the *first* P_0 *inputs*, we had $\mathcal{N}(\boldsymbol{\xi}^{P_0}) > \sum_{i=0}^{d_0+1} \binom{P_0}{i}$, then, by the induction assumption, the theorem is true.

So let us discuss the other case:

$$\mathcal{N}(\boldsymbol{\xi}^{P_0}) \leq \sum_{i=0}^{d_0+1} \binom{P_0}{i}.$$

We divide the old cells (those for the first P_0 inputs) into two groups: a group M_2, which contains cells that will split into *two subcells* on presenting the new input, i.e., both outputs are possible on $\boldsymbol{\xi}$. The remaining cells, i.e., those which do not split, are contained in M_1. Obviously,

$$\begin{aligned} \mathcal{N}(\boldsymbol{\xi}^{P_0+1}) &= |M_1| + 2|M_2| \\ \mathcal{N}(\boldsymbol{\xi}^{P_0}) &= |M_1| + |M_2| \leq \sum_{i=0}^{d_0+1} \binom{P_0}{i}. \end{aligned} \tag{5.170}$$

The bars denote the number of cells in the groups. Now, we study two possibilities: If $|M_2| \leq \sum_{i=0}^{d_0} \binom{P_0}{i}$, then, by Eq. (5.170), we would have

$$\mathcal{N}(\boldsymbol{\xi}^{P_0+1}) \leq \sum_{i=0}^{d_0+1} \binom{P_0}{i} + \sum_{i=0}^{d_0} \binom{P_0}{i} = \sum_{i=0}^{d_0+1} \binom{P_0+1}{i}, \tag{5.171}$$

by a standard addition theorem for binomials. But this contradicts our condition (5.169). So we are left with the second possibility:

$$|M_2| > \sum_{i=0}^{d_0} \binom{P_0}{i}.$$

By the induction assumption we can find a subsequence of length $d_0 + 1$ out of the first P_0 inputs, such that the teachers of the cells in M_2 produce 2^{d_0+1} cells. Since these are able to give *both* possible answers on the new input $\boldsymbol{\xi}$, we have constructed a subsequence of length $d_0 + 2$ with 2^{d_0+2} output combinations. This completes the proof.

Appendix 5.2: Order Parameters for ADALINE

For $\alpha < 1$ it is well known [59] that the coupling vector can be explicitly written as

$$\mathbf{w}_s = N^{-1/2} \sum_{kl} \sigma_k (C^{-1})_{kl} \boldsymbol{\xi}_l \tag{5.172}$$

with $C_{kl} = N^{-1} \boldsymbol{\xi}_k \cdot \boldsymbol{\xi}_l$. The length of the coupling vector is then

$$q_0 = N^{-1} \mathbf{w} \cdot \mathbf{w} = N^{-1} \sum_{kl} \sigma_k (C^{-1})_{kl} \sigma_l. \tag{5.173}$$

The basic idea is to calculate the order parameters from an average over the teacher. Technically, it is useful to choose Gaussian distributed teacher vectors with density

$$g(\mathbf{w}_t) = (2\pi)^{-N/2} \cdot \exp(-\tfrac{1}{2} \mathbf{w}_t \cdot \mathbf{w}_t).$$

This realizes a homogeneous distribution on the surface of a sphere. The outputs are then $\sigma_k = \text{sign}(u_k)$, where the fields $u_k = N^{-1/2} \mathbf{w}_t \cdot \boldsymbol{\xi}_k$ are Gaussian variables with

$$\langle u_k \, u_l \rangle = C_{kl}. \tag{5.174}$$

For random inputs, C_{kl} is typically of order $N^{-1/2}$ for $k \neq l$, and

$$\langle \sigma_k \sigma_l \rangle = \begin{cases} 1 & \text{for } k = l \\ \frac{2}{\pi} C_{kl} + \mathcal{O}(\frac{1}{N}) & \text{for } k \neq l. \end{cases} \tag{5.175}$$

One can show [1] that for random inputs and $N \to \infty$,

$$N^{-1} \sum_k (C^{-1})_{kk} = \frac{\alpha}{1 - \alpha}. \tag{5.176}$$

Using this equation, and inserting Eq. (5.175) into Eq. (5.173), we get

$$q_0 = \frac{\alpha - \frac{2}{\pi} \alpha^2}{1 - \alpha}. \tag{5.177}$$

Finally, for the second order parameter, we get

$$R = N^{-1}\langle \mathbf{w}_t \cdot \mathbf{w}_s \rangle = N^{-1} \sum_{kl} (C^{-1})_{kl} \langle u_k \ \mathrm{sign}(u_l) \rangle = \alpha \sqrt{\frac{2}{\pi}}. \quad (5.178)$$

The case $\alpha > 1$ can be treated by the same method. We will not give the details here [60]. We only mention that $\mathbf{w}_s$ is the minimum of the quadratic learning error

$$\sum_k (\sigma_k - N^{-1/2} \mathbf{w} \cdot \boldsymbol{\xi}_k)^2. \quad (5.179)$$

Taking the gradient, we get explicitly for the ith component

$$w_s(i) = \sum_k (B^{-1})_{ij} f_j, \quad (5.180)$$

with

$$B_{ij} = N^{-1} \sum_k \xi_k(i) \xi_k(j)$$

and

$$f_j = N^{-1/2} \sum_k \sigma_k \xi_k(j).$$

Again, the order parameters can be claculated by averaging over the teacher vector.

REFERENCES

[1] W. Kinzel, M. Opper (1991) Dynamics of learning, In: *Physics of Neural Networks*, J. L. van Hemmen, E. Domany, K. Schulten (Eds.) (Springer-Verlag, New York), p. 149

[2] T.L.H. Watkin, A. Rau, M. Biehl (1993) *Rev. Mod. Phys.* **65**:499

[3] N. Sauer (1972) *J. Comb. Theory A* **13**:145

[4] V.N. Vapnik (1982) *Estimation of Dependences Based on Empirical Data* (Springer-Verlag, New York)

[5] E. Baum, D. Haussler (1989) *Neural Comput.*. **1**(1):151–160

[6] A. Blumer, A. Ehrenfeucht, D. Haussler, M.K. Warmuth (1989) *J. Assoc. Comp. Mach.* **36**:929

[7] E. Levin, N. Tishby, S. Solla (1989) A statistical approach to learning and generalization in neural networks, In: *Proc. 2nd Workshop on Computational Learning Theory* (Morgan Kaufmann)

[8] G. Gyorgyi, N. Tishby (1990) Statistical theory of learning a rule, In: *Neural Networks and Spin Glasses*, (World Scientific)

[9] M. Opper, D. Haussler (1991) *Phys. Rev. Lett.* **66**:2677

[10] M. Opper, D. Haussler (1991) In: *IVth Annual Workshop on Computational Learning Theory (COLT91)* (Santa Cruz, 1991) (Morgan Kaufmann, San Mateo, CA), pp. 75–87

[11] D. Haussler, M. Kearns, M. Opper, R.E. Schapire (1991) Estimating average — Case learning curves using Bayesian, statistical physics and VC dimension methods, In: *Neural Information Processing (NIPS 91)*

[12] E. Gardner (1988) *J. Physics A* **21**:257–270

[13] D. Haussler, M. Kearns, R. Schapire (1991) In: *IVth Annual Workshop on Computational Learning Theory (COLT91)* (Santa Cruz, 1991) (Morgan Kaufmann, San Mateo, CA), pp. 61–74

[14] D. Haussler, A. Barron (1992) How well do Bayes methods work for on-line prediction of $\{+1,-1\}$ values? In: *Proc. Third NEC Symposium on Computation and Cognition* (SIAM, Philadelphia, PA)

[15] J. Rissanen (1986) *Ann. Stat.* **14**:1080

[16] R. Meir, J.F. Fontanari (1993) *Proc. IVth International Bar-Ilan Conference on Frontiers in Condensed Matter Physics*, published in *Physica A* **200**:644

[17] H. Sompolinsky, N. Tishby, H.S. Seung (1990) *Phys. Rev. Lett.* **65**:1683

[18] S. Amari, N. Murata (1993) *Neural Computation* **5**:140

[19] T.M. Cover (1965) *IEEE Trans. El. Comp.* **14**:326–334

[20] G. Stambke (19XX) diploma thesis

[21] G. Gyorgyi (1990) *Phys. Rev. Lett.* **64**:2957

[22] M. Mezard, G. Parisi, M.A. Virasoro (1987) *Spin Glass Theory and Beyond*, Lecture Notes in Physics, 9 (World Scientific)

[23] T.L.H. Watkin (1993) *Europhys. Lett.* **21**:871

[24] R. Meir, J.F. Fontanari (1992) *Phys. Rev. A* **45**:8874

[25] S. Amari (1993) *Neural Networks* **6**:161

[26] M. Opper, D. Haussler, in preparation

[27] F. Vallet, J. Cailton, P. Refregier (1989) *Europhys. Lett.* **9**:315–320

[28] D.E. Rumelhart, J.L. McClelland, eds. (1986) *Parallel Distributed Memory* (MIT Press, Cambridge, MA)

[29] B. Widrow, M.E. Hoff (1960) WESCON Convention, Report IV, 96

[30] I. Kanter, H. Sompolinsky (1987) *Phys. Rev. A* **35**:380

[31] M. Opper, W. Kinzel, J. Kleinz, R. Nehl (1990) *J. Phys. A* **23**:L581

[32] M. Opper (1989) *Europhys. Lett.* **8**:389

[33] A.J. Hertz, A. Krogh, G.I. Thorbergsson (1989) *J. Phys. A* **22**:2133

[34] A. Krogh, J. Hertz (1991) In: *Advances in Neural Information Processing Systems III* (Morgan Kaufmann, San Mateo, CA)

[35] Y. LeCun, I. Kanter, S. Solla (1991) *Phys. Rev. Lett.* **66**:2396

[36] M. Opper (1988) *Phys. Rev. A* **38**:3824

[37] F. Rosenblatt (1961) *Principles of Neurodynamics — Perceptrons and the Theory of Brain* (Spartan Books, Washington DC)

[38] W. Krauth, M. Mezard (1987) *J. Phys. A* **20**:L745

[39] J. Anlauf, M. Biehl (1989) *Europhys. Lett.* **10**:687

[40] P. Ruján (1993) *J. de Phys. (Paris) I* **3**:277

[41] W. Kinzel, P. Ruján (1990) *Europhys. Lett.* **13**:473

[42] T.L.H. Watkin, A. Rau (1992) *J. Phys. A* **25**:113

[43] H.S. Seung, M. Opper, H. Sompolinsky (1992) In: *Vth Annual Workshop on Computational Learning Theory (COLT92)* (Pittsburgh 1992) pp. 287–294 (Assoc. for Computing Machinery, New York)

[44] E. Gardner, B. Derrida (1989) *J. Phys. A* **22**:1983

[45] G. Gyorgyi (1990) *Phys. Rev. A.* **41**:7097

[46] H. Schwarze, M. Opper, W. Kinzel (1992) *Phys. Rev. A* **46**:6185

[47] H. Seung, H. Sompolinsky, N. Tishby (1992) *Phys. Rev. A* **45**:6056

[48] H. Horner (1992) *Z. Phys. B* **87**:371

[49] H.K. Patel (1993) *Z. Physik B* **91**:257

[50] M. Biehl, H. Schwarze (1992) *Europhys. Lett.* **20**:733

[51] M. Biehl, H. Schwarze (1993) *J. Phys. A* **26**:2561

[52] M. Biehl (19XX) diploma thesis, University of Giessen

[53] O. Kinouchi, N. Caticha (1992) *J. Phys. A* **25**:6243

[54] P. Kuhlmann, K.R. Müller (1994) *J. Phys. A* **27**:3759

[55] R. Garces, P. Kuhlmann, H. Eissfeller (1992) *J. Phys. A* **25**:L1335

[56] J. Hertz, A. Krogh, R.G. Palmer (1991) *Introduction to the Theory of Neural Computation* (Addison–Wesley, Reading, MA)

[57] S. Bös, W. Kinzel, M. Opper (1993) *Phys. Rev. E* **47**:1384

[58] M. Biehl, A. Mietzner (1993) *Europhys. Lett.* **24**:421

[59] T. Kohonen (1988) *Self Organisation and Associative Memory* (Springer-Verlag, Berlin)

[60] M. Opper (1995) in preparation