Stochastic Dynamics of On-Line Learning in Neural Networks

Stochastic Dynamics of On-Line Learning in Neural Networks

een wetenschappelijke proeve op het gebied van
de Natuurwetenschappen

Proefschrift

ter verkrijging van de graad van doctor
aan de Katholieke Universiteit Nijmegen,
volgens besluit van College van Decanen
in het openbaar te verdedigen
op vrijdag 26 januari 1996
des namiddags te 1.30 uur precies

door

Wilhelmus Adrianus Joseph Johannes Wiegerinck

geboren op 5 maart 1963 te Amsterdam

Promotor: Prof. Dr. C.C.A.M. Gielen

This work was supported by the Dutch Foundation for Neural Networks (SNN).

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Wiegerinck, Wilhelmus Adrianus Joseph Johannes

Stochastic dynamics of on-line learning in neural networks
/ Wilhelmus Adrianus Joseph Johannes Wiegerinck. - [S.l.
: s.n.]. - Ill.
Proefschrift Katholieke Universiteit Nijmegen. - Met lit.
opg. - Met samenvatting in het Nederlands.
ISBN 90-9009004-5
Trefw.: neurale netwerken / leerproces.

Aan alle geïnteresseerden.

Contents

Chapter 1

Introduction

Abstract

One of the most important features of neural networks is their ability to adjust to their environment by "learning". How neural networks learn is the subject of this thesis. The introduction of this thesis is organized as follows. First a general introduction to neural networks is given. This includes some ideas about the human brain – on which neural networks are inspired, a summary of what neural networks are about, and a short discussion why neural networks are interesting for application purposes. Subsequently, we discuss some aspects of neural network learning and we give a short review on current theories on neural network learning which served as the starting points of this thesis. Finally we will give an outline of the contents of the other chapters in this thesis, a general discussion of the main results and some suggestions for further research.

1.1 General introduction to neural networks

The brain

The human brain is a tremendously powerful information processing system, much more powerful than the largest and fastest digital computer. In particular, the brain is good in processing complex information which is needed to interact with its environment. For example, in a potential hostile environment the recognition of her mothers face within milliseconds is essential for a young child. On the other hand, being good in arithmetics – like digital computers are – will most probably not help her to survive. Another important aspect of the brain is that it is able to learn complex relations between various signals and symbols in its environment by "trial-and-error". As an example we mention the control of arm movements. The arm movements of a baby are rather erroneous at the beginning, but they increase in precision on the basis of very many trials as the child grows up. The child's growth poses an additional problem, since the growing of the limbs requires continuously adjustment of the control. However, by its continual adaptation the brain is excellently suited for such a task.

Although the brain is yet much too complex for a precise understanding, the commonly accepted explanation of the brain's powerful information processing capabilities is roughly as follows [11]. The human brain consists of about 10^{12} nerve cells or neurons, each of it connected by synapses to on average about 10^3 other neurons. It is nowadays commonly assumed that the neurons are the basic information processing elements of the brain, whereas the information transfer between the neurons is regulated by the synapses. Thus the synapses determine to a large extent how the neurons cooperate and consequently how the brain operates. The strength of the synaptic connections is not fixed; through modification of synapses, the brain adapts its functioning to its environment. In this way, the brain is continually learning and building up experience.

The huge number of in principle parallel processing neurons and the enormous amount of interconnectivity accounts for the tremendous information processing capabilities of the brain. The adaptive capabilities of the brain through synaptic modification on the basis of interaction with its environment accounts for the efficient information processing needed in relation to its environment.

Neural networks

Artificial neural networks – or simply, neural networks – are information processing systems which are based on the above described ideas about the brain. As such, the paradigm of computation by a neural network is completely different from the usual one based on a programmed instruction sequence which is used in conventional digital computers.

Most neural networks have two basic ingredients.

- A neural network consists of a number of units interconnected according to a certain architecture. The units or neurons are simple nonlinear processors, i.e. the output of a unit is a simple, but nonlinear function of its total input. The connections between the units allow the units to communicate and to exchange information. The strengths of the connections, or weights, are the parameters of the network. They determine how the network operates.

- The neural network can learn from examples – patterns from its environment. During learning, the weights of the neural network are adjusted to improve the network's performance in its environment.

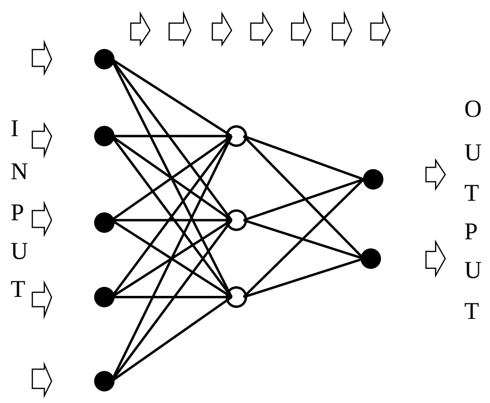

Figure 1.1 Multi-layered perceptron with 5 input units, three hidden units and two output units. The arrows symbolize the feedforward information flow. Note that this network has only one hidden layer.

With these two ingredients, artificial neural networks already show some brain-like behavior. The parallel architecture of neural networks – even if they are simulated in software on conventional computers – in combination with the nonlinearity in the neurons, makes neural networks robust and fault-tolerant. The adaptivity of the weights, in combination with the nonlinearity in the neurons, allows the neural network to learn complex relations just by examples. Excellent overviews of neural networks can be found in [27, 25].

An example of a neural network is the multi-layered perceptron [68]. The multi-layered perceptron consists of several layers of units. Each unit is in a feedforward manner connected to all the units in the next layer (see fig 1.1). The information flow is as follows. An input pattern is received by the network and its values are clamped on the input units. These values are subsequently passed to the units in the first hidden layer, after that they are multiplied by the weights between the input and hidden units in question. In other words, the input of each unit in the first hidden layer is a weighted sum of the values clamped on the input units. Next, the input of a hidden unit is processed by means of a so-called transfer function to form the output of this unit. In their turn, the outputs of the hidden units are fed into the next layer of hidden units, etc. Finally, the processed signals will reach the output units where they can be read off.

However, without learning the network's weights are not adapted to the problem at hand, and hence the network will respond to an input pattern with a completely erroneous output pattern. To find good weights, learning is needed. Most multi-layered perceptrons learn in a so-called supervised way. Input patterns in combination with the corresponding desired output patterns are presented (by a hypothetical supervisor) as examples (or 'training patterns') to the network. The adaptation of the weights is simply such that if the same input patterns would be presented the next time, the network would respond with output patterns more similar to the desired ones. In this way, the network learns the input-output relation by examples rather than that it is precisely programmed like conventional computer programs by an expert with explicit knowledge of this input-output relation.

Neural network research can be divided into two groups. One group of researchers studies neural networks as models for parts of the brain in order to understand neurobiological or psychological observations. The other group of researchers considers neural networks merely as alternative computing paradigms. Biological modeling is not a goal in itself, but rather is neurobiology useful as a source of inspiration for new powerful computing paradigms. In this context, neural networks can be compared with other nature-inspired algorithms, like genetic algorithms [21] and simulated annealing [41]. The previously mentioned brain-like behavior of neural networks – in particular the ability to learn complex, nonlinear relations by examples if a mathematical formulation of these relations is not available – can be useful to have on a computer. Neural networks are already applied in many real world applications; examples can be found in hydrological forecasting, chemical process industry, oil industry, agriculture, marketing, financial forecasting and many others [14, 13, 37]. This second type of research includes the work presented in this thesis.

1.2 Learning from examples

One of the main features of neural networks is their ability to learn from examples. A large part of this thesis is focussed on a popular learning strategy called on-line learning. In on-line learning, examples (combinations of input and desired output patterns in the above described supervised learning) from the environment are continually presented to the network at distinct time steps. At each time step a small adjustment of the network's weights is made on the basis of the pattern which is presented at that time step. This procedure is iterated as long as the network learns. The idea is that on a larger time scale the small adjustments sum up to a continuous adaptation of the network to the whole environment.

On-line versus batch-mode

On-line learning is opposed to batch-mode learning, where at each step the weights are adapted on the basis of *all* the patterns. Batch-mode learning is only an alternative if a finite set of examples (a so-called training set) is available. In fact, batch-mode learning is in most cases merely a classical optimization procedure, which can be found in any numerical analysis text book [62]. Most batch-mode learning rules try to minimize a cost function. The lower the cost, the better the performance of the neural network. In many supervised learning applications, for instance, the cost is the sum of the squared differences between the outputs produced by the network and the desired outputs of all training examples. A cost function can be viewed as a landscape: by varying the weights of the network the output responses of the network on the training set will be varied as well; as a result the cost will be increased or decreased. The aim is to minimize the costs, i.e. to find the lowest point in the landscape. A common technique, which includes the well-known backpropagation algorithm in batch-mode [68], is gradient descent. Gradient descent changes the weights at each learning step a little bit into the down-hill direction. The step size is regulated by the so-called learning parameter. The learning parameter plays an important role in the learning process, in particular in the more complex applications where the cost function often has deep valleys, flat plateaus, and many local minima. In deep, narrow valleys, the gradient descent has to be steered very carefully to remain nicely within the valley. With a too optimistic, large learning parameter, a step aimed to the down-hill direction may jump over the valley and may result in an undesired escape out of it. On flat plateaus, the down-hill direction is very unclear, and the gradient descent algorithm will get stuck. In particular, with a small learning parameter – necessary in relation with possible steep valleys – this effect can be severe. With local minima, the down-hill algorithm has a chance to direct to the wrong, non-optimal minimum.

With on-line learning, where minimization is not based on the whole training set but on individual examples, an additional problem concerns the conflicts between examples. With too large steps, the network will specialize too much on the example which happens to be presented at that learning step and will forget previously stored information. So with on-line learning, this is an additional reason to have small step sizes. In combination with the conflicts between the examples this makes on-line learning unfavorably slow. Nevertheless, on-line learning can be advantageous. The conflicts between the patterns can cause fluctuations in the learning process, which may just be beneficial to escape from plateaus or local minima. In particular with large redundant data-sets and complex, nonlinear problems on-line learning is often advised instead of batch-mode learning [6].

1.3 Current theories on learning

Neural network research, in particular the type of research which studies neural networks for applications, tends to be an area where many claims are made. For example, regular participants of major neural network conferences such as *ICANN* will probably have noticed a continual stream of propositions for new learning rules [1, 38, 50]. Most of them are adaptations of standard algorithms. The authors tested them on a single problem, for which the learning rule had been designed, and concluded that their rule gives an improvement with respect to some (usually not optimal) algorithm. For many of those new learning rules, a solid theoretical foundation or a general proof on the performance for particular types of problems is lacking.

Theoretical research on learning in neural networks is aimed to fill these gaps. This thesis focuses on two aspects of the theory. The one is on-line learning for general neural networks with nonlinear learning rules. The other is on-line learning for a particular network architecture, the single-layered perceptron, equipped with the perceptron learning rule.

On-line learning with nonlinear learning rules

In many theoretical approaches of on-line learning [2, 82, 29, 24, 17], it is assumed that the weight changes are a function of the current weights of the network and a randomly drawn example only. Under this assumption, on-line learning satisfies the so-called Markov property, since the probability for a certain weight change depends only on the weights before the learning step. The underlying Markov process, which describes the time evolution of the probability of the weights, is in general too difficult to solve. However, since the learning parameter is assumed to be small – to prevent overspecialization on single examples – an expansion can be made in the learning parameter [28]. This expansion (by Van Kampen [77]) results in equations which describe the evolution of the weights by a deterministic trajectory with small superimposed Gaussian fluctuations (cf. fig 1.2). This can be understood in the following way. With a small learning parameter, the evolution of the weights is so slow that the network sees the whole training set before its weights are significantly changed. This removes in lowest order the stochastic aspect of the learning process: the weights will follow a smooth deterministic (batch-mode) trajectory. In fact, in the lowest order approximation on-line learning is equivalent to batch-mode learning where a weight change is by definition made on the basis of the whole training set. Only by looking through a magnifying glass, the deterministic trajectory of on-line learning will show the small fluctuations due to the individual patterns which are randomly presented at each time step. Computer simulations of learning neural networks show that the equations resulting from Van Kampen's expansion provide an accurate description of the learning process [29].

Although the theory of on-line learning is very general, many on-line weight update rules do not satisfy the Markov property. A counterexample (see Chapter 2 and 3) is learning with correlated patterns where the pattern presented at a certain time step depends also on the previously presented patterns. In this case the probability for a certain weight change depends not only on the weights before the learning step, but also on the previously presented patterns. Another counterexample is learning with a so-called momentum term (see Chapter 4). In all these cases, the standard procedure of Van Kampen's system size expansion can no longer be

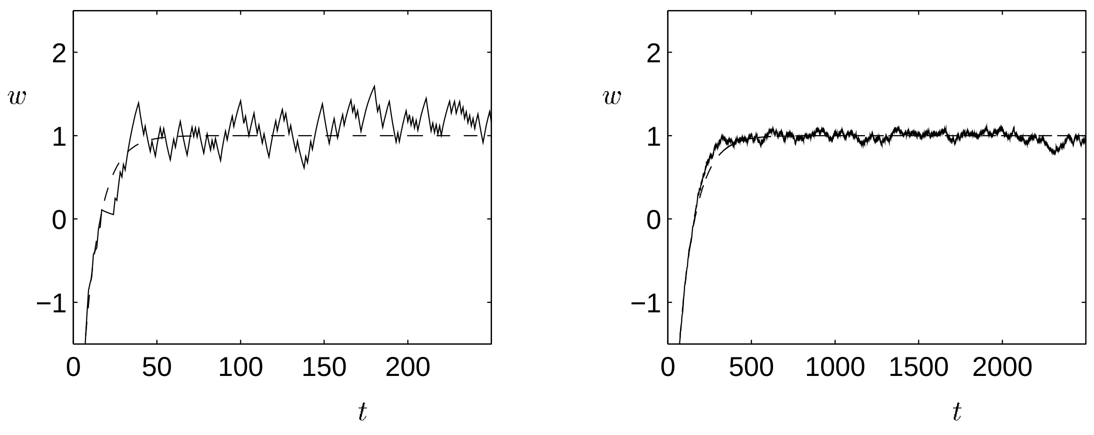

Figure 1.2 Typical weight evolutions in on-line learning (solid lines). In the right panel the learning parameter is 10 times smaller than in the left panel. Note that the time axis is scaled by a factor 10 in the right panel. In both panels, the dashed line represents the deterministic (batch-mode) approximation of the learning process. A smaller learning parameter reduces the fluctuations around the deterministic trajectory, but also the learning speed.

applied in a straightforward manner.

The perceptron

The above described theory on on-line learning is not related to a specific neural network architecture, or learning rule. In the last chapter, however, we concentrate on a particular type of neural network, the (single layered) perceptron. The perceptron is one of the classicals in the neural network history [65, 51]. It consists of an array of input units, directly connected to a binary output unit. If the total input of the output unit exceeds a certain threshold value, the output unit will be active. In the sixties, the perceptron has been successfully implemented by Rosenblatt *e.a.* to discriminate digits, among other things. Rosenblatt invented an on-line learning rule for the perceptron. For this rule he proved convergence; this means that if a solution exists, the network will find it in a finite number of steps.

The perceptron is one of the simplest, non-trivial neural networks. This made the perceptron a rewarding subject of many theoretical studies. Most of these studies concentrated on the statistical properties of a learned perceptron such as storage capacity of random patterns [19], and generalization as function of the number of examples provided by a teacher perceptron [70]. These studies considered the phase space of weights and used tools from statistical physics. We approached the perceptron from another side; using the dynamics of the weights, we tried to calculate properties of the weights directly.

1.4 Outline of this thesis

Chapter 2: On-line learning with time-correlated patterns

The goal of this chapter is to derive for small learning parameters the first and second order approximations of the stochastic dynamics of the weights in on-line learning with time-correlated patterns. Starting point of this chapter is the description of on-line learning with correlated patterns in terms of an evolution equation for the joint probability distribution of the network's weights and the presented pattern. To derive an evolution equation for the probability distribution of the weights only, we make use of perturbation theory – well known from quantum mechanics. First we put the learning parameter exactly equal to zero. This is viewed as the unperturbed system. In this system the probability distribution of the weights does not evolve at all (since the network does not learn with a learning parameter equal to zero!). Next we perturb the system, i.e. we give the learning parameter a small but nonzero value. Now, from a standard result of the so-called second-order perturbation theory with degenerate eigenvalues, we can immediately read off how the dynamics of the probability distribution of the weights is affected by the perturbation. To solve the perturbed dynamics is a hopeless task. However, Van Kampen's expansion can be applied straightforwardly and consequently the first and second order approximations of the stochastic weight dynamics are immediately found. The result of this chapter is that only the fluctuations are affected by the correlations between subsequent patterns. The batch-mode equation is unaffected, and appears in the same form as in the uncorrelated case.

This result can be understood within the same heuristic framework as presented in the previous section. With a small learning parameter, the evolution of the weights is so slow that the network sees the whole training set before its weights are significantly changed. This removes in lowest order the stochastic aspect of the learning process, including the correlations between the patterns since these correlations are merely of importance on a short time scale, and again leads to the batch-mode equation. However the fluctuations in the weights are affected by the pattern correlations, since the fluctuations are due to the individual patterns which are randomly – but correlated – presented at each time step.

Chapter 3: The effect of correlations on the learning process

The fact that the results of the previous chapter have been so easy to interpret with the above described heuristics has motivated us to continue the study of on-line learning with correlated patterns, but now in a way which is more related to these heuristics. This approach simplifies the calculations considerably. For instance, it allows us to calculate asymptotic quantities like the representation error and the prediction error. The representation error is the expected average error of the network with respect to the whole environment. It measures how well the environment is represented by the network after learning. The prediction error is the network's expected average error with respect to the next presented pattern. A remarkable relation between these two errors is that if one compares learning with and without correlations between the patterns, the gain in the predictability of the next pattern cancels in lowest order exactly the loss in representation of the whole environment and vice versa.

Until now we found that correlations had only a higher order effect on the learning process. In regular situations the learning process is dominated by the batch-mode force whereas correlations are only of a marginal importance. However, in the literature dramatic global differences have been reported between on-learning with correlated and uncorrelated patterns [52, 33]. The explanation is the following. In neural network learning one indeed often encounters situations where the batch-mode equation vanishes, namely on plateaus. In the absence of the batch-mode term, higher order terms will play an important role, since they may drive the weights away from the plateau. In this chapter we showed that with correlations between patterns, such higher order terms are much more likely to exist than in the absence of correlations. This is illustrated by an example of a multi-layered perceptron learning the symmetric tent map with

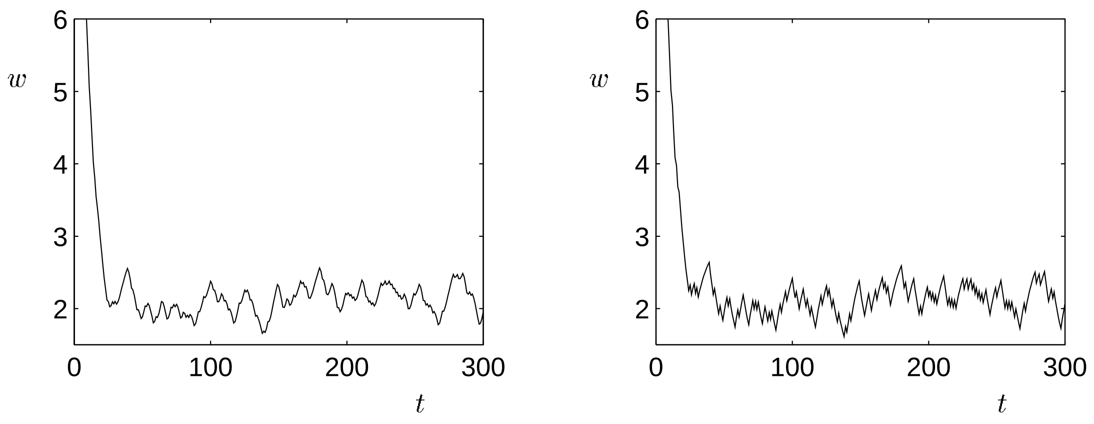

Figure 1.3 Typical weight evolution in on-line learning with momentum (left) and on-line learning without momentum (right). The initial conditions and the presentation of patterns were taken identical in both simulations. Note that although the trajectory in learning with momentum is more smooth, there is no gain in speed or reduction of the large-scale fluctuations. (The "ideal" asymptotic weight would be $w = 2$ in these simulations)

correlated and uncorrelated patterns. Exactly as quantitatively predicted by the theory of this chapter, the network with uncorrelated patterns gets stuck at the plateau, whereas the network with correlated patterns can escape and reaches the global minimum.

Chapter 4: On-line learning with a momentum term

On-line learning is a slow and noisy process. In an attempt to speed up the learning process while reducing fluctuations due to the randomness in the presentation of patterns, a so-called momentum term is sometimes added [68]. In this chapter, on-line learning with a momentum term is studied. On-line learning with a momentum term is not covered by the standard theory on on-line learning, since the weight changes do not only depend on the weights of the network and the presented pattern, but also on the weight-changes at the previous learning step. The dynamical structure of the learning process depends on the ratio between the learning parameter and the momentum parameter. We studied the two interesting regimes. In the one regime the process is a second order process with one time scale. This regime can be dealt with the usual Van Kampen expansion. In the other regime the process is of second order with two time scales. The changes in the weights are small, but the changes in the weight-changes are relatively large. Similarly to chapter 2 we first had to eliminate the fast variables (using perturbation theory) before we could apply Van Kampen's expansion. The main result of this chapter is that in both regimes the benefit of a momentum term in on-line learning is at least questionable (cf. fig 1.3).

Chapter 5: The connections of large perceptrons

Unlike the other chapters, this chapter restricts itself to a specific network architecture, namely the perceptron. The question of this chapter is: given a linearly separable task (i.e. the desired input/output relation is feasible by a perceptron) and given a training set of input vectors for the perceptron, can we calculate the weights for a perceptron that faithfully perform the task on this training set. To calculate the weights, one should in principle solve a set of coupled inequalities: for each input vector, the dot product of the weight vector and the input vector should be either positive or negative, in accordance with the desired output.

In this chapter we tried to solve this problem with another approach. We used the guaranteed convergence of the perceptron learning rule, implying that the asymptotic training result with the perceptron learning rule is a solution to the problem. It is indeed shown that solutions of the fixed point equations of the lowest order, deterministic batch-mode approximation of the (stochastic) perceptron learning rule indeed are solutions to the full problem. The difference is that these fixed points equations are easier to handle analytically, in particular for large networks (i.e. with many input units and connections).

We considered two cases. In the simplest case, the training set consists of *all* possible input vectors. For this case, we are able to calculate the weights as a series expansion in the system size. The leading term in this expansion turns out to be either the Hebb rule [26, 34] or the biased Hebb rule [3], depending on whether the presentation of the input vectors to the network during learning is assumed to be unbiased or biased.

As a more realistic case we considered a training set consisting of an extensive number of prototype patterns. These patterns are trained with noise. We find that the weights satisfy a self-consistent physically transparent set of non-linear equations . In the limit of small training noise the solution of these equations is shown to correspond exactly to the weights of maximal stability in the Gardner sense [19, 39].

1.5 Discussion and directions for further research

In this thesis we studied on-line learning in neural networks. In the chapters 2, 3 and 4 we viewed on-line learning as a process with two time scales. During learning, the weights of the neural network adapt slowly. However, this is always the result of interaction with much faster changing variables, e.g. patterns which might be completely different at each learning step. Separation of time scales and elimination of the fast variables made it possible to study the learning process in the case where the fast variables are correlated in time. This includes learning with correlated patterns and learning with a momentum term. In the latter case the fast variables are the changes in the weights, and subsequent weight changes are correlated.

The analysis in these chapters has led us to two conclusions. The first is that correlations in the pattern presentation can be very helpful in on-line learning. This has been observed previously by simulations by others [52, 33, 5], but it was never properly understood. The other one is that the benefits of a momentum term in on-line learning is doubtful. Although a momentum term is sometimes still recommended in on-line learning [27], this result has been known for linear learning rules [71, 72, 59]. Our analysis has extended this result for on-line learning with general nonlinear learning rules. Note that our analysis does not deal with a momentum term for batch-mode. In batch-mode, a momentum term is indeed beneficial [72] , but other techniques, like the conjugent gradient algorithm [62], are far more efficient [75].

There are many possible directions to extend on this framework in future studies. In chapter 3 we found that on the one hand, correlations can help to avoid plateaus, and on the other hand, correlations affect the asymptotic learning result. These effects may be utilized to devise time-dependent pattern selection techniques. For instance, one can think of a scheme starting with correlations designed to avoid plateaus and continuing in a later stage with correlations designed for the fine tuning around minima. Perhaps such schemes will relate to common sense, like the pedagogical idea that the presentation of patterns should start simple and gradually increase its complexity [54, 10, 49].

Another direction could be the study of recurrent neural networks. In recurrent networks, the state of the neurons at a certain moment in time does not only depend on the external stimulus at that moment but also on the state of the neurons at previous times. This can cause a behavior of the network that is richer than for example of a feedforward network, like the multi-layered perceptron. The framework described in this thesis might well be applied to study on-line learning in recurrent networks, with its bidirectional interaction between the rapidly changing neuron states and the slowly changing weights [61].

In Chapter 5 we focussed on the result of learning in the single-layered perceptron. Some of the results of this chapter – the Hebb rule gives a good representation of the weights of large perceptrons (with the complete training set) and some statistical properties of the weights (in the case of an extensive number of prototype patterns) – were in fact known in the literature [74, 56]. A difference is that in chapter 5 the Hebb rule emerged as a natural expression for the weight. In [74], the Hebb rule was taken as the starting point for the analysis which yielded the results. Another difference is that in chapter 5 these results have been derived using the dynamics of the perceptron, rather than statistical mechanical tools.

We restricted our study of the perceptron to continuously valued weights. An interesting class of perceptrons, however are the ones with binary ($\{-1,1\}$) or ternary ($\{-1,0,1\}$) weights. For instance, such weights are much easier to store or to implement in hardware. For these classes of weights the fixed-point theorem described in chapter 5 still holds. One could wonder if the fixed-point theorem could help to find the optimal weights in such cases.

Chapter 2

On-Line Learning with Time-Correlated Patterns

Abstract

Many of current theories on on-line learning in neural networks are based on the unrealistic assumption that subsequent patterns are uncorrelated. In this paper we study on-line learning with time-correlated patterns. For small learning parameters we derive a Fokker-Planck equation describing the evolution of the average network state and the fluctuations around this average. Correlations between subsequent patterns contribute to the diffusion term in this Fokker-Planck equation and thus affect the fluctuations in the learning process. Our results are valid for a general class of learning rules, including backpropagation and the Kohonen learning rule. Simulations with Oja's rule illustrate the theoretical results.

Adapted from: W. Wiegerinck and T. Heskes. On-line learning with time-correlated patterns, *Europhysics Letters*, 28: 451–455, 1994.

2.1 Introduction

On-line learning is a learning process for neural networks where the weights of the network are updated each time a training pattern $\vec{x}$ is presented to the network. For many learning processes this weight change can be written in the general form

$$\Delta \boldsymbol{w}(n) \equiv \boldsymbol{w}(n+1) - \boldsymbol{w}(n) = \eta\, \boldsymbol{f}(\boldsymbol{w}(n), \vec{x}) , \tag{2.1}$$

with $\boldsymbol{w}(n)$ the network state at iteration step n, η the learning parameter, and $\boldsymbol{f}(\cdot,\cdot)$ the learning rule. Examples can be found in supervised learning, e.g. backpropagation for multilayer perceptrons [68], where $\vec{x}$ stands for an input and desired output pair, as well as in unsupervised learning, e.g. Kohonen's self-organizing rule for topological feature maps [42], where $\vec{x}$ stands for the input vector.

If the patterns $\vec{x}$ are drawn at random from the training set, on-line learning (2.1) can be described by a first order Markov process, since the new network state $\boldsymbol{w}(n+1)$ is solely a function of the old state $\boldsymbol{w}(n)$ and the randomly drawn pattern $\vec{x}$. An evolution equation for the probability $P(\boldsymbol{w}, n)$ that at step n the network is in state $\boldsymbol{w}$ follows directly from (2.1). In recent years, theoretical studies of this evolution equation for small learning parameters η have provided a better understanding of on-line learning processes with uncorrelated patterns [64, 29, 63, 24, 28].

However, in biological learning as well as in real world applications subsequent patterns are correlated. Clearly, the above described theory based on random presentation of patterns is not valid in such cases. In this paper, we therefore study on-line learning (2.1) with time-correlated patterns, and we show that for small learning parameters the behavior of $P(\boldsymbol{w}, n)$ can still be analyzed. For the moment, we assume that the probability that a pattern $\vec{x}$ is presented to the network depends on its predecessor $\vec{x}'$ through a transition probability $\rho(\vec{x}|\vec{x}')$, i.e. that the patterns follow a first order Markov process. Later we will show that the final results hold for stationary Markov processes of any finite order.

With time-correlated patterns the dynamics in weight space is no longer Markovian, which makes it much less straightforward to derive an evolution equation for $P(\boldsymbol{w}, n)$. However, the joint probability $\hat{P}(\boldsymbol{w}, \vec{x}, n)$ that at step n the network is in state $\boldsymbol{w}$ and the presented pattern is $\vec{x}$ *does* follow a Markov process:

$$\begin{aligned} \Delta \hat{P}(\boldsymbol{w}, \vec{x}, n) &= \int d\boldsymbol{w}'\, d\vec{x}'\, \rho(\vec{x}|\vec{x}')\, \delta(\boldsymbol{w} - \boldsymbol{w}' - \eta \boldsymbol{f}(\boldsymbol{w}', \vec{x}'))\, \hat{P}(\boldsymbol{w}', \vec{x}', n) \\ &- \hat{P}(\boldsymbol{w}, \vec{x}, n) . \end{aligned} \tag{2.2}$$

The time scale for the dynamics of the weights $\boldsymbol{w}$ is inversely proportional to η, whereas the time scale for the dynamics the patterns $\vec{x}$ is completely independent of η. For $\eta \to 0$ this separation of time scales makes it possible to eliminate the fast variable $\vec{x}$ [76] and to derive a systematic expansion of the evolution equation for $P(\boldsymbol{w}, n) = \int d\vec{x}\, \hat{P}(\boldsymbol{w}, \vec{x}, n)$. In chapter 4 we use a similar method to analyze learning with momentum.

2.2 Perturbation theory

To write (2.2) in a form which is more convenient for algebraic manipulations, we introduce the left and right eigenfunctions $\Psi_i(\vec{x})$ and $\Phi_i(\vec{x})$ of the transition probability $\rho(\vec{x}|\vec{x}')$ with corresponding eigenvalues λ_i:

$$\int d\vec{x}\, \Psi_i(\vec{x})\, \rho(\vec{x}|\vec{x}') = \lambda_i \Psi_i(\vec{x}') , \quad \int d\vec{x}'\, \rho(\vec{x}|\vec{x}')\, \Phi_i(\vec{x}') = \lambda_i \Phi_i(\vec{x}) ,$$

$$\text{and} \quad \int d\vec{x}\, \Psi_i(\vec{x})\, \Phi_j(\vec{x}) \;=\; \delta_{ij} \;\; \forall_{i,j\geq 0} \; .$$

For convenience, we assume that the stationary distribution $\Phi_0(\vec{x})$ of the patterns is unique, i.e. we have $\lambda_0 = 1$, Re $\lambda_i < 1$ for $i \geq 1$ and normalisation $\Psi_0(\vec{x}) = 1$. The probability distribution $\hat{P}(\boldsymbol{w}, \vec{x}, n)$ can be decomposed in right eigenfunctions:

$$\hat{P}(\boldsymbol{w}, \vec{x}, n) \;=\; \sum_{i\geq 0} Q_i(\boldsymbol{w}, n)\, \Phi_i(\vec{x}) \; .$$

Substitution of this decomposition into (2.2), multiplication with $\Psi_i(\vec{x})$, and integration over $\vec{x}$ yields a set of equations for $Q_i(\boldsymbol{w}, n)$ [which are treated as components of the vector $\vec{Q}(\boldsymbol{w}, n)$]. Performing a Kramers-Moyal expansion [18] with respect to $\boldsymbol{w}$, this set of equations can be written

$$\Delta\, \vec{Q}(\boldsymbol{w}, n) \;=\; \left[\mathcal{H}\vec{Q}\right](\boldsymbol{w}, n) \;=\; \sum_{\alpha=0}^{\infty} \eta^{\alpha} \left[\mathcal{H}^{(\alpha)}\vec{Q}\right](\boldsymbol{w}, n) \, , \tag{2.3}$$

in which the components of the matrices $\mathcal{H}^{(\alpha)}$ are defined by the differential operators

$$\begin{aligned}
\mathcal{H}^{(0)}_{ij} \;&\equiv\; -(1-\lambda_i)\delta_{ij} \; , \\
\mathcal{H}^{(\alpha)}_{ij} \;&\equiv\; \frac{(-1)^{\alpha}}{\alpha!} \int d\vec{x} \left[\lambda_i \Psi_i(\vec{x}) \Phi_j(\vec{x}) \right. \\
&\qquad \left. \sum_{i_1,\ldots,i_\alpha} \frac{\partial}{\partial w_{i_1}} \cdots \frac{\partial}{\partial w_{i_\alpha}} f_{i_1}(\boldsymbol{w}, \vec{x}) \cdots f_{i_\alpha}(\boldsymbol{w}, \vec{x}) \right] \, , \quad \text{for } \alpha \geq 1.
\end{aligned}$$

Since the operator $\mathcal{H}$ is a series in the small parameter η, it is possible to analyze (2.3) using perturbation theory. Let us first consider the unperturbed ($\eta = 0$) system

$$\Delta\, \vec{Q}(\boldsymbol{w}, n) = \left[\mathcal{H}^{(0)}\vec{Q}\right](\boldsymbol{w}, n) \;\text{ with solution }\; Q_i(\boldsymbol{w}, n) = \lambda_i^n Q_i(\boldsymbol{w}, 0) \, .$$

The components Q_i with $i \geq 1$ will rapidly relax to zero. For large n, only the component Q_0 will remain. The eigenvalues of the perturbed system are equal to the eigenvalues $-(1-\lambda_i)$ of the unperturbed system plus terms of order η. So, if $\eta \ll \min_{i\geq 1}(1 - \text{Re}\ \lambda_i)$, we can still distinguish the invariant subspace in which $\vec{Q}$ rapidly relaxes to zero from the invariant subspace in which $\vec{Q}$ slowly evolves. To describe the evolution of $Q^{\rm s}$, defined as the projection of $\vec{Q}$ on the slow subspace, we can immediately use a standard result from second-order perturbation theory with degenerate eigenvalues [20],

$$\Delta Q^{\rm s}(\boldsymbol{w}, n) \;=\; \left[\eta \mathcal{H}^{(1)}_{00} + \eta^2 \mathcal{H}^{(2)}_{00} + \eta^2 \sum_{j>0} \frac{\mathcal{H}^{(1)}_{0j} \mathcal{H}^{(1)}_{j0}}{1-\lambda_j} + \mathcal{O}(\eta^3) \right] Q^{\rm s}(\boldsymbol{w}, n) \, .$$

For large n, when the projection of $\vec{Q}$ on the fast subspace has vanished, $Q^{\rm s}$ equals $Q_0 + \mathcal{O}(\eta^2)$. Substitution of $\mathcal{H}^{(1)}$, $\mathcal{H}^{(2)}$, and $Q_0(\boldsymbol{w}, n) = P(\boldsymbol{w}, n)$ then leads to

$$\begin{aligned}
\Delta P(\boldsymbol{w}, n) \;=\; & -\eta\, \boldsymbol{\nabla}_w^T \int d\vec{x}\, \Phi_0(\vec{x})\, \boldsymbol{f}(\boldsymbol{w}, \vec{x})\, P(\boldsymbol{w}, n) \, + \\
& + \frac{\eta^2}{2} \operatorname{Tr} \boldsymbol{\nabla}_w \boldsymbol{\nabla}_w^T \int d\vec{x}\, \Phi_0(\vec{x})\, \boldsymbol{f}(\boldsymbol{w}, \vec{x})\, \boldsymbol{f}^T(\boldsymbol{w}, \vec{x})\, P(\boldsymbol{w}, n) \, + \\
& + \eta^2 \sum_{i\geq 1} \frac{\lambda_i}{1-\lambda_i} \int d\vec{x}\, d\vec{x}' \left[\Phi_i(\vec{x})\, \Psi_i(\vec{x}')\, \Phi_0(\vec{x}') \right. \\
& \qquad \left. \times\; \boldsymbol{\nabla}_w^T \boldsymbol{f}(\boldsymbol{w}, \vec{x}) \boldsymbol{\nabla}_w^T \boldsymbol{f}(\boldsymbol{w}, \vec{x}')\, P(\boldsymbol{w}, n) \right] \,+\, \mathcal{O}(\eta^3) \, .
\end{aligned} \tag{2.4}$$

2.3 Van Kampen's expansion

To study this evolution equation in the limit $\eta \to 0$, we apply Van Kampen's expansion [77, 28]. We start with the Ansatz

$$\boldsymbol{w} = \boldsymbol{\phi}(t) + \sqrt{\eta}\boldsymbol{\xi} \,,$$

with rescaled time $t = \eta n$. This Ansatz says that the state of the network $\boldsymbol{w}$ can be described by a deterministic part $\boldsymbol{\phi}(t)$ plus a term of order $\sqrt{\eta}$ containing the fluctuations. The function $\Pi(\boldsymbol{\xi}, t) \equiv P(\boldsymbol{\phi}(t) + \sqrt{\eta}\boldsymbol{\xi}, t/\eta)$ is the probability in terms of the new variable ξ. From Van Kampen's expansion it immediately follows that the deterministic part $\boldsymbol{\phi}(t)$ has to satisfy the equation

$$\frac{d\boldsymbol{\phi}(t)}{dt} = \langle \boldsymbol{f}(\boldsymbol{\phi}(t), \vec{x}) \rangle_{\vec{x}} \,, \tag{2.5}$$

where the average $\langle \ldots \rangle_{\vec{x}}$ is over the pattern space. The evolution of $\Pi(\boldsymbol{\xi}, t)$ is governed by the Fokker-Planck equation

$$\frac{\partial \Pi(\boldsymbol{\xi}, t)}{\partial t} = \mathrm{Tr}\left[H(\boldsymbol{\phi}(t)) \boldsymbol{\nabla}_{\xi} \left[\boldsymbol{\xi}^T \, \Pi(\boldsymbol{\xi}, t)\right]\right] + \frac{1}{2}\mathrm{Tr}\left[\tilde{D}(\boldsymbol{\phi}(t)) \boldsymbol{\nabla}_{\xi} \boldsymbol{\nabla}_{\xi}^T \, \Pi(\boldsymbol{\xi}, t)\right] \,, \tag{2.6}$$

with the usual Hessian $H(\boldsymbol{w}) = \boldsymbol{\nabla}_w \left\langle \boldsymbol{f}^T(\boldsymbol{w}, \vec{x}) \right\rangle_{\vec{x}}$, but with the (new) effective difussion matrix

$$\tilde{D}(\boldsymbol{w}) \equiv C_0(\boldsymbol{w}) + \lim_{\epsilon \to 1} \sum_{m=1}^{\infty} \left[C_m(\boldsymbol{w}) + C_m^T(\boldsymbol{w})\right] \epsilon^m \,, \tag{2.7}$$

where the auto-correlation matrices C_m read

$$\begin{aligned} C_m(\boldsymbol{w}) &\equiv \sum_{i \geq 1} \lambda_i^m \int d\vec{x}\, d\vec{x}'\, \Phi_i(\vec{x})\, \Psi_i(\vec{x}')\, \Phi_0(\vec{x}')\, \boldsymbol{f}(\boldsymbol{w}, \vec{x})\, \boldsymbol{f}^T(\boldsymbol{w}, \vec{x}') \\ &= \left\langle \boldsymbol{f}(\boldsymbol{w}, \vec{x}(m))\, \boldsymbol{f}^T(\boldsymbol{w}, \vec{x}(0)) \right\rangle_{\vec{x}} - \left\langle \boldsymbol{f}(\boldsymbol{w}, \vec{x}) \right\rangle_{\vec{x}} \left\langle \boldsymbol{f}^T(\boldsymbol{w}, \vec{x}) \right\rangle_{\vec{x}} . \end{aligned} \tag{2.8}$$

In deriving (2.7) and (2.8) from (2.4), we used $\lambda_i/(1-\lambda_i) = \lim_{\epsilon \to 1} \sum_{m=1}^{\infty} \lambda_i^m \epsilon^m$, which is correct even if $\lambda_i \neq 1$ lies on the complex unit circle. Equations (2.7) and (2.8) constitute the main result. They explain how correlations in the training data affect, through the effective diffusion $\tilde{D}$, the fluctuations in the learning process. For uncorrelated patterns, i.e. if $\rho(\vec{x}|\vec{x}') = \rho(\vec{x})$, all auto-correlation matrices $C_m(\boldsymbol{w}) = 0$ for $m \geq 1$ and the effective diffusion matrix reduces to the usual diffusion matrix $C_0(\boldsymbol{w})$ for on-line learning with random pattern presentation [28]. To generalize our results to higher order Markov processes, we view a k-th order Markov process as a first order Markov process in the space of extended patterns $\{\vec{x}(m)\} \equiv \{\vec{x}(m), \ldots, \vec{x}(m-k+1)\}$. However, since $\boldsymbol{f}(\boldsymbol{w}, \{\vec{x}(m)\}) = \boldsymbol{f}(\boldsymbol{w}, \vec{x}(m))$, i.e. the weight update does only depend on the last pattern, the auto-correlation matrices in the pattern space are equal to the auto-correlation matrices in the original pattern space. Therefore the results are valid for stationary Markov processes of any finite order.

2.4 Simulations

To illustrate the theory, we simulate the nonlinear Oja learning rule [55]

$$\Delta \boldsymbol{w}(n) = \eta\, (\vec{x}^T \boldsymbol{w}(n))\, [\vec{x} - (\vec{x}^T \boldsymbol{w}(n))\, \boldsymbol{w}(n)]$$

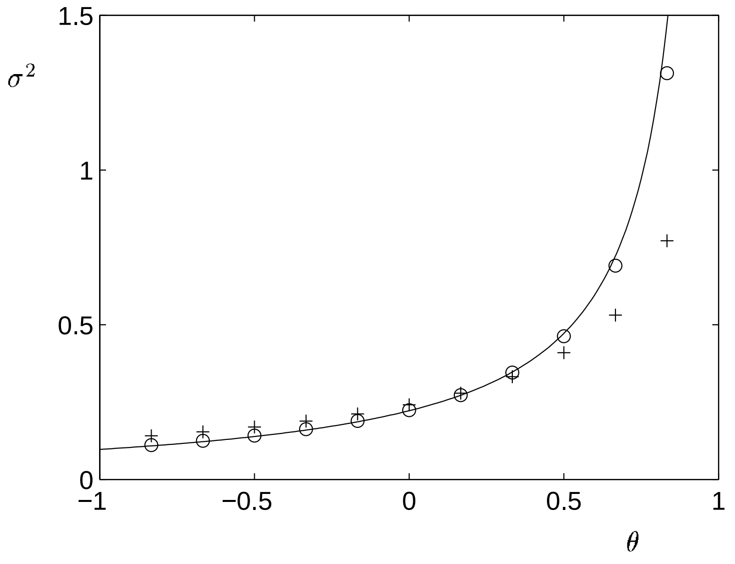

Figure 2.1 Rescaled asymptotic variance σ^2 as a function of flip correlation θ. Theoretical prediction (full curve) can be compared with the simulations for $\eta = 0.01$ ("o") and $\eta = 0.1$ ("+"). Standard error bars would be smaller than the size of the symbols.

in two dimensions, which searches for the principal component of the input correlation matrix $\langle \vec{x}\vec{x}^T \rangle_{\vec{x}}$. The absolute value $|x_i|$ is, independent of previous patterns, homogeneously distributed between 0 and l_i, with $l_1 = 2$ and $l_2 = 1$. The sign of x_i has a probability q_i to flip after each presentation, i.e. $\vec{x}$ follows a first order Markov process. The Fokker-Planck equation (2.6) predicts the asymptotic (rescaled) variance

$$\sigma^2 \equiv \eta^{-1} \left\langle (\boldsymbol{w} - \langle \boldsymbol{w} \rangle_{w(\infty)})^T (\boldsymbol{w} - \langle \boldsymbol{w} \rangle_{w(\infty)}) \right\rangle_{w(\infty)} = \left\langle \boldsymbol{\xi}^T \boldsymbol{\xi} \right\rangle_{\xi(\infty)} = \frac{2}{9} + \frac{\theta}{4(1-\theta)} \, .$$

with "flip correlation" $\theta \equiv (1 - 2q_1)(1 - 2q_2)$. Simulations are performed with an ensemble of 10 000 independently learning networks, initialized at $\boldsymbol{w}(0) = (1, 0)^T$. In figure 2.1 can be seen that the agreement between theory and simulations is better for smaller learning parameters and less correlations, as could be expected. Flip correlation $\theta < 0$ leads to a better sampling of the input space and thus to a smaller asymptotic variance than random pattern presentation ($\theta = 0$).

2.5 Discussion

In this chapter we studied on-line learning in networks with time-correlated patterns. Starting point in our analysis was the evolution of the joint probability distribution of weights and patterns. Using perturbation theory – with the learning parameter as perturbation parameter – we eliminated the patterns from this evolution equation. By applying Van Kampen's expansion (again valid for a small learning parameter) we derived a deterministic equation and a Fokker-Planck equation which together describe the evolution of the average network state and the fluctuations around this average. The results show that correlations between subsequent patterns affect the fluctuations in the learning process. Our approach is certainly not unique: in the context of stochastic approximation theory equivalent results have been obtained [8, 44]. In the next chapter, we will study other aspects of learning with time-correlated patterns. One of these is related to learning of (chaotic) time series, which is claimed to be easier if the patterns are presented in their natural order of appearance instead of completely random [52, 33]. This phenomenon will be explained.

Acknowledgements

We thank one of the referees for a useful suggestion.

Chapter 3

The Effect of Correlations on the Learning Process

Abstract

We study the dynamics of on-line learning for a large class of neural networks and learning rules, including backpropagation for multilayer perceptrons. In this paper, we focus on the case where successive examples are dependent, and we analyze how these dependencies affect the learning process. We define the representation error and the prediction error. The representation error measures how well the environment is represented by the network after learning. The prediction error is the average error which a continually learning network makes on the next example. In the neighborhood of a local minimum of the error surface, we calculate these errors. We find that the more predictable the example presentation, the higher the representation error, i.e. the less accurate the asymptotic representation of the whole environment. Furthermore we study the learning process in the presence of a plateau. Plateaus are flat spots on the error surface, which can severely slow down the learning process. In particular, they are notorious in applications with multilayer perceptrons. Our results, which are confirmed by simulations of a multilayer perceptron learning a chaotic time series using backpropagation, explain how dependencies between examples can help the learning process to escape from a plateau.

Adapted from: W. Wiegerinck and T. Heskes. How dependencies between successive examples affect on-line learning. Submitted to *Neural Computation*.

3.1 Introduction

The ability to learn from examples is an essential feature in many neural network applications [27, 25]. Learning from examples enables the network to adapt its parameters or weights to its environment without the need for explicit knowledge of that environment. This paper focusses on a popular learning procedure called on-line learning. In this learning procedure examples from the environment are continually presented to the network at distinct time steps. At each time step a small adjustment of the network's weights is made on the basis of the currently presented example. This procedure is iterated as long as the network learns. The idea is that on a larger time scale the small adjustments sum up to a continuous adaptation of the network to the whole environment.

In many applications the network has to be trained with a training set consisting of a finite number of examples. In these applications a strategy is often used where at each step a randomly selected example from the training set is presented. In particular with large training sets and complex environments successful results have been obtained with this strategy [9, 6]. Characteristic of this kind of learning is that successive examples are independent, i.e. that the probability to select an example at a certain time step is independent of its predecessors. Of course, successive examples in on-line learning do not need to be independent. For example, one can think of an application where the examples are obtained by on-line measurements of an environment. If these examples are directly fed into the neural network, it is likely that successive examples are correlated with each other.

A related example is the use of neural networks for time-series prediction [46, 80, 83, 79, 35]. Essentially, the task of these networks is, given the last k data points of the time series, to predict the next data point of the time series. Each example consists of a data point and its k predecessors. There are two obvious ways to train a network "on-line" with these examples. In what we call "randomized learning", successively presented examples are drawn from the time series on arbitrary, randomly chosen times. This makes successively presented examples independent. In the other type of learning, which we call "natural learning", the examples are presented in their natural order, keeping their natural dependencies. In [52, 33] both types of example presentation are compared for the learning of a one-dimensional chaotic mapping. In their simulations natural learning performs significantly better than randomized learning. This phenomenon, and, more generally, how the presentation order of examples affects the process of on-line learning are the subject of this paper. Understanding these issues is not only interesting from a theoretical point of view, but it may also help to devise better learning strategies.

In this paper we study the dynamics of on-line learning with dependent examples from a general point of view. In section 3.2, we define the class of learning rules and the types of stochastic, yet dependent, example presentation which are analyzed in the rest of the paper. Because of the stochasticity in the presentation of examples, on-line learning is a stochastic process. However, since the weight changes at each time step are assumed to be small – in this paper the weight changes scale linearly with a small constant η, the so-called learning parameter – it is possible to give approximate deterministic descriptions of the learning process on a larger time scale. In lowest order, the learning process can be described by an ordinary differential equation (ODE). The fluctuations, i.e. the differences between the stochastic trajectory of the weights and the ODE are of order $\sqrt{\eta}$. These fluctuations are described by a covariance matrix. Besides an heuristic (re)derivation of the ODE and the equation for the fluctuations [1], section 3.3 also derives in the same heuristic framework an equation for a systematic bias. This bias, which

[1]these results already have been derived in Chapter 2. The ODE is given by the deterministic equation (2.5). The fluctuations are described by the Fokker-Planck equation (2.6). Other rigorous derivations – in the context of stochastic approximation theory – can be found in [8, 44]

is of order η, describes the lowest order difference between the mean value of the weights and their ODE trajectory. One could interpret the bias as a first order correction to the ODE. With these equations, we will study the effect of dependencies in the examples on the learning process. In section 3.4 we use these results to calculate how the presentation of examples affects asymptotic performances like the representation error and the prediction error. The representation error measures how well the environment is represented by the network after learning. The prediction error is the average error which a continually learning network makes on the next example. In section 3.5 we use the results of section 3.3 to study the effect of dependencies when the learning process is stuck on a so-called plateau in the error surface. Plateaus are frequently present in the error surface of multilayer perceptrons [36]. Using the results in this section the remarkable difference between randomized learning and natural learning, which has been mentioned in the previous paragraph, is explained. The last section gives a brief summary and discussion.

3.2 The Framework

In many on-line learning processes the weight change at learning step n can be written in the general form

$$\Delta \boldsymbol{w}(n) \equiv \boldsymbol{w}(n+1) - \boldsymbol{w}(n) = \eta \boldsymbol{f}(\boldsymbol{w}(n), \vec{x}(n)) , \tag{3.1}$$

with $\boldsymbol{w}(n)$ the network weights and $\vec{x}(n)$ the presented example at iteration step n. η is the learning parameter, which is assumed to be constant in this paper, and $\boldsymbol{f}(\cdot,\cdot)$ the learning rule. Examples satisfying (3.1) can be found in supervised learning such as backpropagation for multilayer perceptrons [81, 67], where the examples $\vec{x}(n)$ are combinations of input vectors $(x_1(n), \dots, x_k(n))$ and desired output vectors $(y_1(n), \dots, y_l(n))$, as well as in unsupervised learning such as Kohonen's self-organizing rule for topological feature maps [42], where $\vec{x}(n)$ stands for the input vector $(x_1(n), \dots, x_k(n))$. On-line learning in the general form (3.1) has been studied extensively [2, 64, 82, 29, 48, 58, 24, 63, 17]. Most papers on this subject restrict themselves to independent presentation of examples, i.e. the probability $p(\vec{x}, n)$ to present an example $\vec{x}$ at iteration step n is given by a probability distribution $\rho(\vec{x})$, independent of its predecessor. In this paper – similar to Chapter 2 – we incorporate dependencies between examples by assuming that the probability to present an example $\vec{x}$ depends on its predecessor $\vec{x}'$ through a transition probability $\tau(\vec{x}|\vec{x}')$, i.e. that $p(\vec{x}, n)$ follow a first-order stationary Markov process

$$p(\vec{x}, n+1) = \int d\vec{x}' \tau(\vec{x}|\vec{x}') p(\vec{x}', n). \tag{3.2}$$

Learning with independent examples is a special case with $\tau(\vec{x}|\vec{x}') = \rho(\vec{x})$. As we have argued in Chapter 2, the limitation to first-order Markov processes is not as severe as it might seem at first sight, since stationary Markov processes of any finite order k can be incorporated in the formalism by redefining the vectors $\vec{x}$ to include the last k examples. The Markov process is assumed to have a unique asymptotic or stationary distribution $\rho(\vec{x})$, i.e., we assume that we can take limits like

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} \phi(\vec{x}(n)) = \int d\vec{x} \rho(\vec{x}) \phi(\vec{x})$$

in which $\phi(\vec{x})$ is some function of the patterns. So $\rho(\vec{x})$ describes the (asymptotic) relative frequency of patterns. A randomized learning strategy therefore will select its independent examples from this stationary distribution. In this paper we will denote these long time averages with brackets $\langle \cdot \rangle_x$,

$$\langle \phi(\vec{x}) \rangle_x \equiv \int d\vec{x} \rho(\vec{x}) \phi(\vec{x})$$

and sometimes we use capitals, i.e. we define quantities like $\Phi \equiv \langle \phi(\vec{x}) \rangle_x$.

Many neural network algorithms, including backpropagation, perform gradient descent on a "local" cost or error function $e(\boldsymbol{w}, \vec{x})$,

$$\boldsymbol{f}(\boldsymbol{w}(n), \vec{x}(n)) \equiv -\boldsymbol{\nabla}_w e(\boldsymbol{w}(n), \vec{x}(n)) . \qquad (3.3)$$

The idea of this learning rule is that with a small learning parameter, the stochastic gradient descent [(3.1) and (3.3)] approximates deterministic gradient descent on the "global" error potential

$$E(\boldsymbol{w}) \equiv \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} e(\boldsymbol{w}, \vec{x}(n)) \qquad (3.4)$$

We restrict ourselves to learning with a cost function in order to compare performances between several types of pattern presentation (with equal stationary distributions), in particular in section 3.4 and 3.5. However, most derivations and results in this paper can be easily generalized to the general rule (3.1).

3.3 ODE Approximation and Beyond

The update rule for the weights (3.1) and the Markov process governing the presentation of examples (3.2) can be combined into one evolution equation for the joint probability $\hat{P}(\boldsymbol{w}, \vec{x}, n)$ that at step n example $\vec{x}$ is presented to the network with weight vector $\boldsymbol{w}$. This probability obeys the Markov process

$$\hat{P}(\boldsymbol{w}, \vec{x}, n+1) = \int d\boldsymbol{w}'\, d\vec{x}'\, \tau(\vec{x}|\vec{x}')\, \delta(\boldsymbol{w} - \boldsymbol{w}' - \eta \boldsymbol{f}(\boldsymbol{w}', \vec{x}'))\, \hat{P}(\boldsymbol{w}', \vec{x}', n) . \qquad (3.5)$$

We are interested in the learning process, i.e. in the evolution of the probability distribution of weights

$$P(\boldsymbol{w}, n) = \int d\vec{x}\, \hat{P}(\boldsymbol{w}, \vec{x}, n) .$$

With dependent examples, it is not possible to derive a self-supporting equation for the evolution of $P(\boldsymbol{w}, n)$ by direct integration over $\vec{x}$ in (3.5). However, in Chapter 2 we have shown that the evolution equation of $P(\boldsymbol{w}, n)$ can be expanded systematically in the small learning parameter η. The basic assumption for this expansion is that the dynamics of the weights, with typical time scale $1/\eta$, is much slower than the typical time scale of the examples.

In the following, we present a slightly different approach to approximate the evolution of the probability distribution of weights. This approach, based on [77], assumes that the distribution of weights, with initial form $P(\boldsymbol{w}, 0) = \delta(\boldsymbol{w} - \boldsymbol{w}(0))$, remains sharply peaked as n increases. We follow the heuristic treatment in [8] and average the learning rule over a "mesoscopic" time scale [24] which is much larger than the typical time scale of the example dynamics yet much smaller than the time scale on which the weights can change significantly. With the averaged learning rule we can directly calculate approximate equations for the mean $\overline{\boldsymbol{w}}(n)$ and the covariance matrix $\Sigma^2(n)$, which describe the position and the width of the peak $P(\boldsymbol{w}, n)$ respectively.

We iterate the learning step (3.1) M times, where M is a mesoscopic time scale, i.e. $1 \ll M \ll 1/\eta$, and obtain

$$\boldsymbol{w}(n+M) - \boldsymbol{w}(n) = \eta \sum_{m=0}^{M-1} \boldsymbol{f}(\boldsymbol{w}(n+m), \vec{x}(n+m)) . \qquad (3.6)$$

For the average $\overline{\boldsymbol{w}}(n) \equiv \langle \boldsymbol{w}(n) \rangle$ [brackets $\langle \ldots \rangle$ stand for averaging over the combined process (3.5)], we have the exact identity

$$\overline{\boldsymbol{w}}(n+M) - \overline{\boldsymbol{w}}(n) = \eta \sum_{m=0}^{M-1} \langle \boldsymbol{f}(\boldsymbol{w}(n+m), \vec{x}(n+m)) \rangle \; . \tag{3.7}$$

On the one hand, the mesoscopic time scale is much smaller than the time scale on which the probability distribution $P(\boldsymbol{w}, n)$ can change appreciably. Therefore, if the probability distribution $P(\boldsymbol{w}, n)$ is very sharply peaked, we can expand (3.7) around the mean $\overline{\boldsymbol{w}}(n)$

$$\overline{\boldsymbol{w}}(n+M) - \overline{\boldsymbol{w}}(n) = \eta \sum_{m=0}^{M-1} \langle \boldsymbol{f}(\overline{\boldsymbol{w}}(n), \vec{x}(n+m)) \rangle + \ldots$$

On the other hand, the mesoscopic time scale is much larger than the typical time scale of the Markov process governing the presentation of examples. Therefore we can approximate the sum

$$\frac{1}{M} \sum_{m=0}^{M-1} \langle \boldsymbol{f}(\overline{\boldsymbol{w}}(n), \vec{x}(n+m)) \rangle \approx \lim_{N \to \infty} \frac{1}{N} \sum_{m=0}^{N-1} \langle \boldsymbol{f}(\overline{\boldsymbol{w}}(n), \vec{x}(n+m)) \rangle \equiv \boldsymbol{F}(\overline{\boldsymbol{w}}(n)) \, . \tag{3.8}$$

Thus, in lowest order, the stochastic equation (3.7) can be approximated by the deterministic difference equation

$$\overline{\boldsymbol{w}}(n+M) - \overline{\boldsymbol{w}}(n) = \eta M \boldsymbol{F}(\overline{\boldsymbol{w}}(n)) \, .$$

For small ηM, the difference equation for the position of the peak turns into an ordinary differential equation (ODE). In terms of the rescaled continuous time t, with $t_n \equiv \eta n$ [we will use both notations $\boldsymbol{w}(n)$ and $\boldsymbol{w}(t)$], we obtain that the learning process is approximated by the ODE

$$\frac{d\overline{\boldsymbol{w}}(t)}{dt} = \boldsymbol{F}(\overline{\boldsymbol{w}}(t)) = -\boldsymbol{\nabla} E(\overline{\boldsymbol{w}}(t)) \, . \tag{3.9}$$

In this equation $E(\boldsymbol{w})$ is the global error potential defined in (3.4). In lowest order the weights do indeed follow the gradient of the global error potential. Dependencies in successively presented examples have no influence on the ODE (3.9): this equation only depends on the stationary distribution $\rho(\vec{x})$ of the examples. Corrections to the ODE arise when we expand (3.6)

$$\begin{aligned} \boldsymbol{w}(n+M) - \boldsymbol{w}(n) &= \eta \sum_{m=0}^{M-1} \boldsymbol{f}(\boldsymbol{w}(n+m), \vec{x}(n+m)) \\ &= \eta \sum_{m=0}^{M-1} \boldsymbol{f}(\boldsymbol{w}(n), \vec{x}(n+m)) \\ &\qquad - \eta \sum_{m=0}^{M-1} \mathsf{h}(\boldsymbol{w}(n), \vec{x}(n+m))(\boldsymbol{w}(n+m) - \boldsymbol{w}(n)) + \ldots \\ &= \eta \sum_{m=0}^{M-1} \boldsymbol{f}(\boldsymbol{w}(n), \vec{x}(n+m)) \\ &\qquad - \eta^2 \sum_{m=0}^{M-1} \mathsf{h}(\boldsymbol{w}(n), \vec{x}(n+m)) \sum_{l=0}^{m-1} \boldsymbol{f}(\boldsymbol{w}(n), \vec{x}(n+l)) + \ldots \end{aligned} \tag{3.10}$$

with the "local Hessian" $\mathsf{h}(\boldsymbol{w}, \vec{x}) \equiv \boldsymbol{\nabla}_w \boldsymbol{\nabla}_w^T e(\boldsymbol{w}, \vec{x})$. Averaging, expanding functions of the form $\langle \phi(\boldsymbol{w}(n), \vec{x}(n')) \rangle$ around the mean $\overline{\boldsymbol{w}}(n)$, and transformation to continuous time t, yields us a first approximation beyond the ODE (3.9)

$$\frac{d\overline{\boldsymbol{w}}(t)}{dt} = \boldsymbol{F}(\overline{\boldsymbol{w}}(t)) - \frac{1}{2}\mathsf{Q}(\overline{\boldsymbol{w}}(t)):\Sigma^2(t) - \eta\boldsymbol{B}(\overline{\boldsymbol{w}}(t)) \tag{3.11}$$

in which

$$\begin{aligned}
\Sigma^2(t) &\equiv \left\langle [\boldsymbol{w}(t) - \overline{\boldsymbol{w}}(t)][\boldsymbol{w}(t) - \overline{\boldsymbol{w}}(t)]^T \right\rangle , && (3.12.a)\\
Q_{\alpha\beta\gamma}(\boldsymbol{w}) &\equiv \frac{\partial^2 F_\alpha(\boldsymbol{w})}{\partial w_\beta\, \partial w_\gamma} , && (3.12.b)\\
(\mathsf{Q}:\Sigma^2)_\alpha &\equiv \sum_{\beta\gamma} Q_{\alpha\beta\gamma}\Sigma^2_{\beta\gamma} && (3.12.c)\\
\boldsymbol{B}(\boldsymbol{w}) &\equiv \lim_{N\to\infty}\sum_{n=1}^{N-1}\left[1 - \frac{n}{N}\right]\Big\langle [\mathsf{h}(\boldsymbol{w},\vec{x}(n)) - \mathsf{H}(\boldsymbol{w})] \\
&\qquad \times\ [\boldsymbol{f}(\boldsymbol{w},\vec{x}(0)) - \boldsymbol{F}(\boldsymbol{w})]\Big\rangle_x , && (3.12.d)\\
\mathsf{H}(\boldsymbol{w}) &\equiv \lim_{N\to\infty}\frac{1}{N}\sum_{n=0}^{N-1}\langle \mathsf{h}(\boldsymbol{w},\vec{x}(n))\rangle_x . && (3.12.e)
\end{aligned}$$

The first-order corrected evolution equation (3.11) for the mean $\overline{\boldsymbol{w}}(t)$ is influenced by the covariance matrix $\Sigma^2(t)$. The covariance matrix obeys

$$\begin{aligned}
\Sigma^2(n+M) - \Sigma^2(n) &= \left\langle [\boldsymbol{w}(n+M) - \boldsymbol{w}(n)][\boldsymbol{w}(n) - \overline{\boldsymbol{w}}(n)]^T \right\rangle \\
&+ \left\langle [\boldsymbol{w}(n) - \overline{\boldsymbol{w}}(n)][\boldsymbol{w}(n+M) - \boldsymbol{w}(n)]^T \right\rangle \\
&+ \left\langle [\boldsymbol{w}(n+M) - \boldsymbol{w}(n)][(\boldsymbol{w}(n+M) - \boldsymbol{w}(n)]^T \right\rangle \\
&- [\overline{\boldsymbol{w}}(n+M) - \overline{\boldsymbol{w}}(n)][\overline{\boldsymbol{w}}(n+M) - \overline{\boldsymbol{w}}(n)]^T .
\end{aligned}$$

Substitution of expansion (3.10), expansion of functions of the form $\langle\phi(\boldsymbol{w}(n), \vec{x}(n'))\rangle$ around the mean $\overline{\boldsymbol{w}}(n)$, and transformation to continuous time t, leads in lowest order to the approximation

$$\frac{d\Sigma^2(t)}{dt} = -\mathsf{H}(\overline{\boldsymbol{w}}(t))\Sigma^2(t) - \Sigma^2(t)\mathsf{H}(\overline{\boldsymbol{w}}(t)) + \eta\mathsf{D}(\overline{\boldsymbol{w}}(t)) \tag{3.13}$$

with the "diffusion" matrix

$$\mathsf{D}(\boldsymbol{w}) = \left\langle \lim_{N\to\infty}\frac{1}{N}\sum_{n=0}^{N-1}\sum_{m=0}^{N-1}[\boldsymbol{f}(\boldsymbol{w},\vec{x}(n)) - \boldsymbol{F}(\boldsymbol{w})]\,[\boldsymbol{f}(\boldsymbol{w},\vec{x}(m)) - \boldsymbol{F}(\boldsymbol{w})]^T \right\rangle_x . \tag{3.14}$$

From (3.13), we can see that $\Sigma^2(t)$ remains bounded if H is positive definite. In this case $\Sigma^2(t) = \mathcal{O}(\eta)$, which makes (3.13) with (3.11) a valid approximation [77]. In other words, since η is small, this justifies *a posteriori* the assumption that $P(\boldsymbol{w}, n)$ is sharply peaked. In other cases where the fluctuations do not remain bounded, the approximation is only applicable during a short period.

The diffusion $\mathsf{D}(\boldsymbol{w})$ can be expressed as the sum of an independent and a dependent part:

$$\mathsf{D}(\boldsymbol{w}) = \mathsf{C}_0(\boldsymbol{w}) + \lim_{N\to\infty}\sum_{n=1}^{N-1}\left[1 - \frac{n}{N}\right]\left[\mathsf{C}_n(\boldsymbol{w}) + \mathsf{C}_n^T(\boldsymbol{w})\right] \equiv \mathsf{C}_0(\boldsymbol{w}) + \mathsf{C}_+(\boldsymbol{w}) \tag{3.15}$$

where we have defined the auto-correlation matrices

$$\mathsf{C}_n(\boldsymbol{w}) \equiv \left\langle [\boldsymbol{f}(\boldsymbol{w}, \vec{x}(n)) - \boldsymbol{F}(\boldsymbol{w})]\,[\boldsymbol{f}(\boldsymbol{w}, \vec{x}(0)) - \boldsymbol{F}(\boldsymbol{w})]^T \right\rangle_x . \tag{3.16}$$

For on-line learning with random sampling, there are no dependencies between subsequent weight changes, so $\mathsf{C}_+(\boldsymbol{w}) = 0$ and consequently the diffusion $\mathsf{D}(\boldsymbol{w})$ reduces to $\mathsf{C}_0(\boldsymbol{w})$ [see e.g. [28]].

The set of equations (3.11) and (3.13) for $\overline{\boldsymbol{w}}$ and Σ^2 forms a self-supporting first approximation beyond the ODE approximation (3.9). It is not necessary to solve (3.11) and (3.13) simultaneously. Since the covariance Σ^2 appears in (3.11) as a correction it suffices to compute Σ^2 from (3.13) using the ODE approximation for $\overline{\boldsymbol{w}}$. Following [77] we set $\overline{\boldsymbol{w}} = \boldsymbol{w}_{\text{ODE}} + \boldsymbol{u}$, and solve

$$\begin{aligned}
\frac{d\boldsymbol{w}_{\text{ODE}}(t)}{dt} &= \boldsymbol{F}(\boldsymbol{w}_{\text{ODE}}(t)) && \text{(3.17.a)}\\
\frac{d\Sigma^2(t)}{dt} &= -\mathsf{H}(\boldsymbol{w}_{\text{ODE}}(t))\Sigma^2(t) - \Sigma^2(t)\mathsf{H}(\boldsymbol{w}_{\text{ODE}}(t)) + \eta\mathsf{D}(\boldsymbol{w}_{\text{ODE}}(t)) && \text{(3.17.b)}\\
\frac{d\boldsymbol{u}(t)}{dt} &= -\mathsf{H}(\boldsymbol{w}_{\text{ODE}}(t))\boldsymbol{u}(t) - \frac{1}{2}\mathsf{Q}(\boldsymbol{w}_{\text{ODE}}(t)) : \Sigma^2(t) - \eta\boldsymbol{B}(\boldsymbol{w}_{\text{ODE}}(t)) . && \text{(3.17.c)}
\end{aligned}$$

The first two equations (3.17.a) and (3.17.b) are equivalent to the results derived in Chapter 2, or in the literature [8, 44]. The ODE (3.17.a) approximates in lowest order the dynamics of the weights. The covariance matrix $\Sigma^2(t)$ which obeys (3.17.b), describes the stochastic deviations $\boldsymbol{w}(n) - \boldsymbol{w}_{\text{ODE}}(t_n)$ between the weights and the ODE approximation. These fluctuations are typically of order $\sqrt{\eta}$. (Their "square" $\Sigma^2(t)$ is of order η). In Chapter 2, these fluctuations are described by the Fokker-Planck equation (2.6). In [8, 44] a Wiener process is derived to describe these fluctuations. In the next section we will study how these fluctuations affects some asymptotic error measures.

The last equation (3.17.c) describes a bias $\boldsymbol{u}$ between the mean $\overline{\boldsymbol{w}}$ and the ODE approximation $\boldsymbol{w}_{\text{ODE}}$. The dynamics of the bias consists of two driving terms. The first one is the interaction between the nonlinearity of the learning rule Q and the fluctuations described by Σ^2. This term can be understood in the following way: if a random fluctuation into one direction in weight space does not result in the same restoring effect as a random fluctuation into the opposite direction, then random fluctuations will obviously result in a netto bias effect. The other driving term in (3.17.c) is $\boldsymbol{B}$ [see (3.12.d)]. This term is only due to the dependencies of the examples. Since the two driving terms are typically of order η, the bias term is also typically of order η, and is therefore neglected in regular situations. However in section 3.5 it will be shown (and this will be supported by simulations) that there are situations where this bias term is of crucial importance.

As an approximation, the set of coupled equations (3.17.a,b,c) is equally valid as the coupled set (3.11) and (3.13). However, in (3.17.a,b,c) the hierarchical structure of the approximations (ODE approximation, fluctuations, bias...) is more clear.

The influence of the example presentation on the evolution of the weight distribution is twofold. On the one side, dependencies between examples affect the covariance Σ^2 through the diffusion term D [see (3.14)]. On the other side, they affect the mean value through the vector $\boldsymbol{B}$, and indirectly through the covariance Σ^2. For independent examples, D reduces to C_0 [see (3.16)] and $\boldsymbol{B} = 0$ exactly.

Finally we want to stress that the essential assumption for the validity of (3.17) is that the weights can be described by their average value with small superimposed fluctuations. In other words, the approximation (3.17) is locally valid. This is the case if the Hessian H is positively definite. In other cases the approximation is only valid for short times [77]. In the analysis of the next two sections we tacitly assume this local validity.

3.4 Representation Error and Prediction Error

In this section we show how dependencies between successive examples influence the asymptotic performance of the network. In the asymptotic situation, the weights are assumed to remain concentrated around a minimum $\boldsymbol{w}^*$ of the global error $E(\boldsymbol{w})$. We consider two measures of network performance: the "representation error" E_{repr} and the "prediction error" E_{pred}. The meaning of this terminology slightly differs from its usual meaning in most neural network literature. The representation error

$$E_{\text{repr}} \equiv \lim_{n\to\infty} \langle\langle e(\boldsymbol{w}(n), \vec{x})\rangle_x\rangle = \lim_{n\to\infty} \langle E(\boldsymbol{w}(n))\rangle \tag{3.18}$$

is the expectation of the asymptotic global error $E(\boldsymbol{w}(\infty))$ [cf. (3.4)] made by the network. It is a useful measure to compare different example presentation techniques if the goal is minimization of the local cost function $e(\boldsymbol{w}, \vec{x})$ in an environment given by a probability distribution $\rho(\vec{x})$. In the context of time series E_{repr} measures how well the asymptotic network state is expected to represent the whole time series. The prediction error

$$E_{\text{pred}} \equiv \lim_{n\to\infty} \langle e(\boldsymbol{w}(n), \vec{x}(n))\rangle \tag{3.19}$$

is the error that the network in its final stage of learning is expected to make on the *next* example of the time-series. E_{pred} measures the error locally in time, in contrast to the more global measure E_{repr}.

The weights are assumed to be concentrated around a minimum $\boldsymbol{w}^*$ of the global error $E(\boldsymbol{w})$. This implies

$$\boldsymbol{\nabla} E(\boldsymbol{w}^*) = 0 \, .$$

The fluctuations Σ^2 and the bias $\boldsymbol{u} = \langle \boldsymbol{w}(\infty)\rangle - \boldsymbol{w}^*$ satisfy in lowest order the fixed point equations of (3.17.b) and (3.17.c)

$$\begin{aligned} \mathsf{H}(\boldsymbol{w}^*)\Sigma^2 + \Sigma^2\mathsf{H}(\boldsymbol{w}^*) &= \eta \mathsf{D}(\boldsymbol{w}^*) \\ \mathsf{H}(\boldsymbol{w}^*)\boldsymbol{u} &= -\frac{1}{2}\mathsf{Q}(\boldsymbol{w}^*):\Sigma^2 - \eta\, \boldsymbol{B}(\boldsymbol{w}^*)\, . \end{aligned}$$

With the techniques used in the previous section we calculate the two error measures up to $\mathcal{O}(\eta)$. To obtain the representation error (3.18) we expand $E(\boldsymbol{w})$ around its minimum $\boldsymbol{w}^*$,

$$\begin{aligned} E_{\text{repr}} = \lim_{n\to\infty} \langle E(\boldsymbol{w}(n))\rangle &= E(\boldsymbol{w}^*) + \frac{1}{2}\operatorname{Tr}\left[\mathsf{H}(\boldsymbol{w}^*)\Sigma^2\right] + \dots \\ &= E(\boldsymbol{w}^*) + \frac{\eta}{4}\operatorname{Tr}\left[\mathsf{D}(\boldsymbol{w}^*)\right] + \dots\, . \end{aligned}$$

To calculate the prediction error (3.19), we apply a time-averaging procedure similar to the one used in Section 3.3. Given weight vector $\boldsymbol{w}(n)$ before the first learning step, the local error over the next M examples is

$$\begin{aligned} \frac{1}{M}\sum_{m=0}^{M-1} e(\boldsymbol{w}(n+m), \vec{x}(n+m)) &= \frac{1}{M}\sum_{m=0}^{M-1} e(\boldsymbol{w}(n), x(n+m)) \\ &\quad - \frac{\eta}{M}\sum_{m=0}^{M-1}\sum_{l=0}^{m-1} \boldsymbol{f}^T(\boldsymbol{w}(n), \vec{x}(n+m))\boldsymbol{f}(\boldsymbol{w}(n), \vec{x}(n+l)) + \dots\, . \end{aligned}$$

For a mesoscopic timescale M we obtain, using the definitions (3.15) and (3.16),

$$E_{\text{pred}} = \lim_{n\to\infty}\frac{1}{M}\sum_{m=0}^{M-1}\langle e(\boldsymbol{w}(n+m), \vec{x}(n+m))\rangle = E_{\text{repr}} - \frac{\eta}{2}\operatorname{Tr}\left[\mathsf{C}_+(\boldsymbol{w}^*)\right] + \dots\, .$$

For randomized learning, the prediction error and representation error are equal: $E_{\rm pred} = E_{\rm repr} \equiv E_{\rm ran}$. Using $\mathsf{D}(\boldsymbol{w}^*) = \mathsf{C}_0(\boldsymbol{w}^*)$, we obtain

$$E_{\rm ran} = E(\boldsymbol{w}^*) + \frac{\eta}{4}\,\mathrm{Tr}\,[\mathsf{C}_0(\boldsymbol{w}^*)] + \dots .$$

If we compare the representation and prediction error with dependent examples to the error with independent examples (assuming that the weight distribution is concentrated around the same minimum $\boldsymbol{w}^*$), we see that, up to order η, the profit in prediction exactly cancels the loss in representation and vice versa:

$$\frac{E_{\rm pred} + E_{\rm repr}}{2} = E_{\rm ran} + \dots .$$

In the context of strategies to select examples, this implies that a strategy that yields a larger prediction error will most likely lead to a smaller representation error. Depending on whether successive weight changes are, roughly speaking, positively or negatively correlated, the prediction error is smaller or larger than the representation error, respectively. This is nicely illustrated by the following simple example.

We consider a process where the examples can take two values $x = \pm 1$ with transition probability

$$\tau(x|x') = (1-q)\delta_{x,x'} + q\delta_{x,-x'} ,$$

i.e. there is a probability $0 < q \leq 1$ to flip the sign of the input. The stationary distribution $\rho(x)$ is given by

$$\rho(x) = \frac{1}{2}\left(\delta_{x,-1} + \delta_{x,1}\right) . \tag{3.20}$$

A one-dimensional "weight" w tries to minimize the squared distance between the presented example and the weight, i.e. the local error is

$$e(w,x) = \frac{1}{2}(w-x)^2 \tag{3.21}$$

and the corresponding update rule [cf. (3.1) and (3.3)] is

$$\Delta w = \eta\,(x-w) .$$

The global error $E(w)$ [cf. (3.4)] is obtained by averaging (3.21) over the stationary distribution (3.20),

$$E(w) = \frac{1}{4}(1-w)^2 + \frac{1}{4}(-1-w)^2 = \frac{1}{2} + \frac{1}{2}w^2 ,$$

and has a minimum $E(w^*) = 1/2$ for $w^* = 0$. To compute the performance measures (3.18) and (3.19) for our simple unsupervised example as a function of the flip probability q, we first calculate the auto-correlations $\mathsf{C}_m(w^*)$ [cf. (3.16)] in the minimum $w^* = 0$:

$$\mathsf{C}_m(0) = \langle x(m)x(0)\rangle_x = (1-2q)^m \text{ and thus } \mathsf{C}_0 = 1 \text{ and } \mathsf{C}_+ = \frac{1-2q}{q} .$$

Up to $\mathcal{O}(\eta)$ we obtain

$$E_{\rm pred} = \frac{1}{2} + \frac{3q-1}{4q}\eta \text{ and } E_{\rm repr} = \frac{1}{2} + \frac{1-q}{4q}\eta .$$

For flip probability $q < 1/2$ we have better prediction than representation, for $q > 1/2$ better representation than prediction ($q = 1/2$ corresponds to randomized learning). This is what we could expect: the larger the flip probability, the better the overall sampling of the input space for the problem at hand (finding the average input) and thus the better the representation. However, the larger the flip probability, the more difficult to predict the next example for the network that has just been adapted to the current example.

3.5 Plateaus

In comparing the asymptotic performance of networks trained with dependent and independent examples in the previous section, we assumed that with small η, both types of learning lead to the same (local) minimum of the global error $E(\boldsymbol{w})$ [see (3.4)]. This is not unreasonable if the learning process is initiated in the neighborhood of this minimum. A minimum is a stable equilibrium point of the ODE dynamics (3.9), i.e. the eigenvalues of the Hessian $\mathsf{H}(\boldsymbol{w})$ [see (3.12.e)] are strictly positive. In the neighborhood of a minimum, the ODE force $\boldsymbol{F}(\boldsymbol{w})$ [see (3.8)] is the dominating factor in the dynamics. Perturbations due to the higher order corrections are immediately restored by the ODE force.

In this section, however, we will consider so-called "plateaus". Plateaus are flat spots on the global-error surface. They are often the cause of the extreme long learning times and/or the bad convergence results in multilayer perceptron applications with the backpropagation algorithm [36]. On a plateau, the gradient of E is negligible and H has some positive eigenvalues but also some zero eigenvalues. Plateaus can be viewed as indifferent equilibrium points of the ODE dynamics. Even with small η, the higher order terms can make the weight vector move around in the subspace of eigenvectors of H with zero eigenvalue without being restored by $\boldsymbol{F}$. In other words, in these directions the higher order terms may give a larger contribution to the dynamics than the ODE term. Since the higher order terms are related to dependencies between the examples, on plateaus the presentation order of examples might significantly affect the learning process.

The effect of different example presentations in learning on a plateau will be illustrated by the following example. We consider the tent map

$$y(x) \;=\; 2\left[\frac{1}{2} - |x - \frac{1}{2}|\right], \qquad 0 \le x \le 1\,,$$

which we view as a dynamical system producing a chaotic time series $x(n+1) = y(x(n))$ [69]. To model this system we use a two-layered perceptron with one input unit, two hidden units and a linear output,

$$z(\boldsymbol{w}; x) \;=\; v_0 + \sum_{\beta=1}^{2} v_\beta \tanh(w_{\beta 1} x + w_{\beta 0})\,.$$

We train the network with input-output pairs $\vec{x} = \{x, y(x)\}$ by on-line backpropagation [67]

$$\Delta \boldsymbol{w} \;=\; -\eta \boldsymbol{\nabla}_w e(\boldsymbol{w}, \vec{x})$$

with the usual squared error cost function

$$e(\boldsymbol{w}, \vec{x}) \;=\; \frac{1}{2}\left[y(x) - z(\boldsymbol{w}; x)\right]^2 .$$

We compare two types of example presentation. With natural learning, examples are presented according to the sequence generated by the tent map, i.e.

$$\vec{x}(0) = \{x(0), y(x(0))\}\,,\; \vec{x}(1) = \{x(1), y(x(1))\}\,,\; \ldots$$
$$\ldots\,,\; \vec{x}(n) = \{x(n), y(x(n))\}\,, \ldots$$

with $x(n+1) = y(x(n))$ (and $x(0)$ randomly drawn from the interval $[0,1]$). With randomized learning, at each iteration step an input x is drawn according to the stationary distribution $\rho(x)$, i.e. homogeneously from the interval $[0,1]$, the corresponding output $y(x)$ is computed, and the pair $\{x, y(x)\}$ is presented to the network. In both cases we initialize with the same

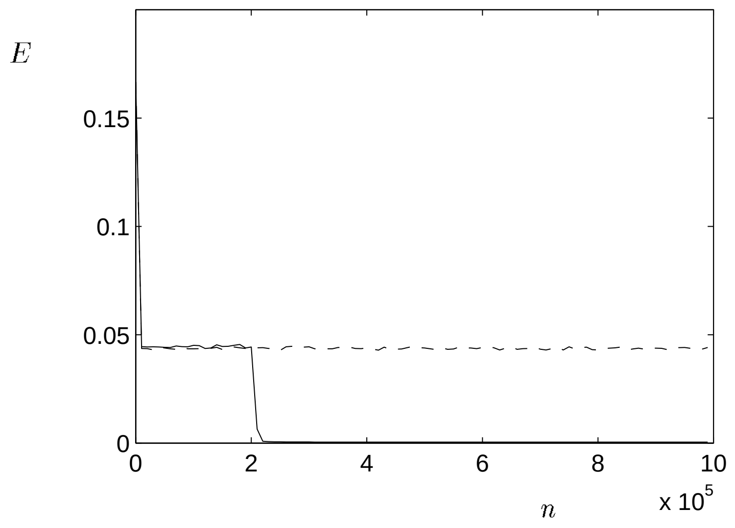

Figure 3.1 Typical global error E of natural learning (full curve) and of randomized learning (dashed curve). Simulation performed with a single network. Learning parameter $\eta = 0.1$. Weights initialization: $\epsilon = 10^{-4}$. Data points are plotted every 10^4 iterations.

small weights, $-\epsilon < v_\gamma, w_{\beta\alpha} < \epsilon$. Small random weights are often recommended to prevent early saturation of the weights [47].

As reported earlier [33], simulations show a dramatic difference between the two learning strategies in their performance learning the tent map (cf. fig 3.1 and fig 3.2). To understand this difference, we will study the weight dynamics by local linearizations. In the neighborhood of a point $\boldsymbol{w}^*$ in weight space the ODE (3.9) can be approximated by

$$\frac{d\overline{\boldsymbol{w}}(t)}{dt} = \boldsymbol{F}(\boldsymbol{w}^*) - \mathsf{H}(\boldsymbol{w}^*)[\overline{\boldsymbol{w}}(t) - \boldsymbol{w}^*]\,. \tag{3.22}$$

The weights are initialized at $\boldsymbol{w}(0) = \mathcal{O}(\epsilon)$, with $\epsilon \approx 0$. The linearization (3.22) around $\boldsymbol{w}^* = \boldsymbol{w}^{(0)} = 0$ yields an approximation of the weight dynamics during the initial stage of learning,

$$\frac{d}{dt}\begin{pmatrix} \overline{v}_0 \\ \overline{v}_\beta \\ \overline{w}_{\beta 0} \\ \overline{w}_{\beta 1} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ 0 \\ 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{3} & -\frac{1}{6} \\ 0 & -\frac{1}{3} & 0 & 0 \\ 0 & -\frac{1}{6} & 0 & 0 \end{pmatrix} \begin{pmatrix} \overline{v}_0 \\ \overline{v}_\beta \\ \overline{w}_{\beta 0} \\ \overline{w}_{\beta 1} \end{pmatrix} \tag{3.23}$$

with $\beta = 1, 2$. From (3.23), we see that $\overline{v}_0$ quickly converges to $\overline{v}_0 = 1/2$ on a time scale where the other weights hardly change (cf. fig 3.3). In other words, during this stage the network just learns the average value of the target function. This is a well-known phenomenon: backpropagation tends to select the gross structures of its environment first.

After the initial stage, (3.23) does not provide a good approximation any more. The linearization (3.22) of the ODE around the new point $\boldsymbol{w}^* = \boldsymbol{w}^{(1)} = (v_0^{(1)} = 1/2, v_\beta^{(1)} = 0, w_{\beta\alpha}^{(1)} = 0)$,

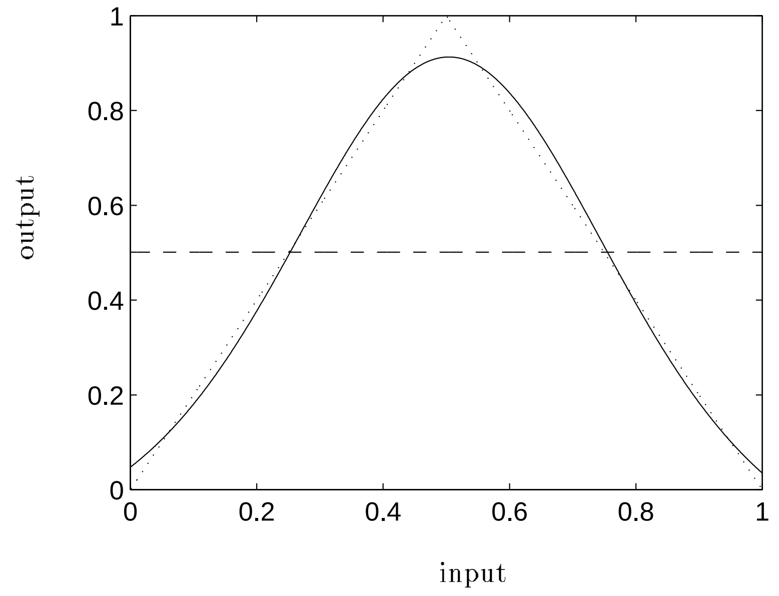

Figure 3.2 Typical network result after 10^6 iteration steps of natural learning (full curve) and randomized learning (dashed curve). The target function is the tent map (dotted curve). For simulation details, see caption of fig 3.1.

(with $\alpha = 0, 1$ and $\beta = 1, 2$), describes the dynamics of the weights during the next stage,

$$\frac{d}{dt}\begin{pmatrix} \overline{v}_0 \\ \overline{v}_\beta \\ \overline{w}_{\beta 0} \\ \overline{w}_{\beta 1} \end{pmatrix} = -\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} \overline{v}_0 - \frac{1}{2} \\ \overline{v}_\beta \\ \overline{w}_{\beta 0} \\ \overline{w}_{\beta 1} \end{pmatrix}$$

with $\beta = 1, 2$. At this stage, $\boldsymbol{F} = 0$, while the Hessian H has one positive eigenvalue ($\lambda = 1$) and further only zero eigenvalues. In other words, at $\boldsymbol{w}^{(1)}$ the weights are stuck on a plateau.

To find out whether the weights can escape the plateau, we have to consider the contributions of the higher η corrections to the weight dynamics (3.11) and (3.13). Linearization of this set of equations around $\boldsymbol{w}^{(1)}$ yields

$$\begin{aligned} \frac{d\overline{\boldsymbol{w}}(t)}{dt} &= -\mathsf{H}(\boldsymbol{w}^{(1)})[\overline{\boldsymbol{w}}(t) - \boldsymbol{w}^{(1)}] \\ &\quad - \frac{1}{2}\left\{\mathsf{Q}(\boldsymbol{w}^{(1)}) + \nabla\mathsf{Q}(\boldsymbol{w}^{(1)})[\overline{\boldsymbol{w}}(t) - \boldsymbol{w}^{(1)}]\right\}:\Sigma^2(t) \\ &\quad - \eta\left\{\boldsymbol{B}(\boldsymbol{w}^{(1)}) + \nabla\boldsymbol{B}(\boldsymbol{w}^{(1)})[\overline{\boldsymbol{w}}(t) - \boldsymbol{w}^{(1)}]\right\} && (3.24) \\ \frac{d\Sigma^2(t)}{dt} &= -\mathsf{H}(\boldsymbol{w}^{(1)})\Sigma^2(t) - \Sigma^2(t)\mathsf{H}(\boldsymbol{w}^{(1)}) + \eta\mathsf{D}(\boldsymbol{w}^{(1)})\,. && (3.25) \end{aligned}$$

At $\boldsymbol{w}^{(1)}$, the (v_0, v_0) component is the only nonzero component for both the Hessian H and the diffusion D (for randomized learning as well as for natural learning). From (3.25), it thus follows that $\Sigma^2_{v_0,v_0}$ is the only nonzero component of the covariance matrix. So only in in this direction there will be fluctuations. However these the fluctuations will be restored, due to the positive (v_0, v_0) component of the Hessian. Moreover, since $Q(\boldsymbol{w}^{(1)}))_{v_0 v_0 w}$ [see (3.12.b)] and its derivatives vanish for all w, the covariance matrix Σ^2 does not couple with the (linearized) weight dynamics, and (3.24) reduces to the autonomous equation

$$\frac{d\overline{\boldsymbol{w}}(t)}{dt} = -\mathsf{H}(\boldsymbol{w}^{(1)})[\overline{\boldsymbol{w}}(t) - \boldsymbol{w}^{(1)}] - \eta\left\{\boldsymbol{B}(\boldsymbol{w}^{(1)}) + \boldsymbol{\nabla B}(\boldsymbol{w}^{(1)})[\overline{\boldsymbol{w}}(t) - \boldsymbol{w}^{(1)}]\right\} .$$

With natural learning, straightforward calculations yield $\boldsymbol{B}(\boldsymbol{w}^{(1)}) = 0$ and $\boldsymbol{\nabla B}(\boldsymbol{w}^{(1)}) = 0$, except for the components

$$\nabla b_{v_\beta v_\beta} = \frac{1}{216}, \quad \nabla b_{v_\beta w_{\beta 1}} = \frac{1}{18}, \quad \nabla b_{w_{\beta 1} v_\beta} = \frac{1}{18},$$
$$\nabla b_{w_{\beta 1} w_{\beta 0}} = \frac{1}{108}, \quad \nabla b_{w_{\beta 1} w_{\beta 1}} = \frac{1}{216}.$$

with $\beta = 1, 2$. Concentrating on the dynamics of $\overline{v}_\beta$, and $\overline{w}_{\beta 1}$, we thus obtain the linear system

$$\frac{d}{dt}\begin{pmatrix} \overline{v}_\beta \\ \overline{w}_{\beta 1} \end{pmatrix} = -\frac{\eta}{216}\begin{pmatrix} 1 & 12 \\ 12 & 1 \end{pmatrix}\begin{pmatrix} \overline{v}_\beta \\ \overline{w}_{\beta 1} \end{pmatrix} \qquad (3.26)$$

with $\beta = 1, 2$. This system has one negative eigenvalue λ_- and one positive eigenvalue $\lambda_+ = \eta\frac{11}{216}$. Along the direction of the eigenvector $(1, -1)$ corresponding to the positive eigenvalue, the weights will move away from $\boldsymbol{w}^{(1)}$ (cf. fig 3.3). Thus, natural learning escapes from the plateau, and reaches the global minimum (cf. fig 3.1 and fig 3.2). On the other hand, for randomized learning $\boldsymbol{B} = 0$ identically. This means that the weights of a randomized learning network are not helped by the higher η corrections and therefore cannot escape the plateau (cf. fig 3.1 and fig 3.2).

Figure 3.3 shows that the predictions computed with the theory agree well with the simulations of the neural network learning the tent map, and therefore we conclude that the difference in performance of the two learning strategies is well explained by the theory.

The analysis of this section - supported by the simulations - shows that if the learning process suffers from a plateau, then dependencies can help learning by a nonzero $\boldsymbol{B}$ term (3.12.d) with some positive eigenvalues. Of course, the magnitude of these eigenvalues and the direction of the corresponding eigenvectors depends strongly on the problem at hand, i.e. $\boldsymbol{B}$ is probably not for every problem directed towards a global minimum. But the fact remains that a nonzero $\boldsymbol{B}$ term can make the weights move *away* from the plateau, which facilitates an escape, resulting in a lower error. On the other hand, if a nonzero $\boldsymbol{B}$ does not make the weights wander away, or if it does not lead to an escape, the performance of dependent learning is probably still not worse than the performance of randomized learning, which also would get stuck on the plateau.

Another situation occurs if randomized learning *does* escape from the plateau, e.g. as a result of the fluctuations. In such a case dependent pattern presentation probably does not harm either - since similar fluctuations would also enhance escaping from the plateau with dependent patterns - unless the presentation order reduces the fluctuations too much! For instance, in the example of section 3.4 the fluctuations are reduced if the examples are negatively correlated ($q > 1/2$). As a more realistic example consider a problem with a fixed training set of P examples. A commonly used incremental learning strategy presents in each epoch of P learning steps each example only once [25]. In other words, the patterns are arranged in a randomly ordered sequence $[\vec{x}(1), \ldots, \vec{x}(P)]$. It is obvious that this sequence-based or cyclic learning introduces dependencies between the examples. Moreover, the subsequent examples are negatively correlated. This follows from the fact that the probability to find identical subsequent examples is on average at

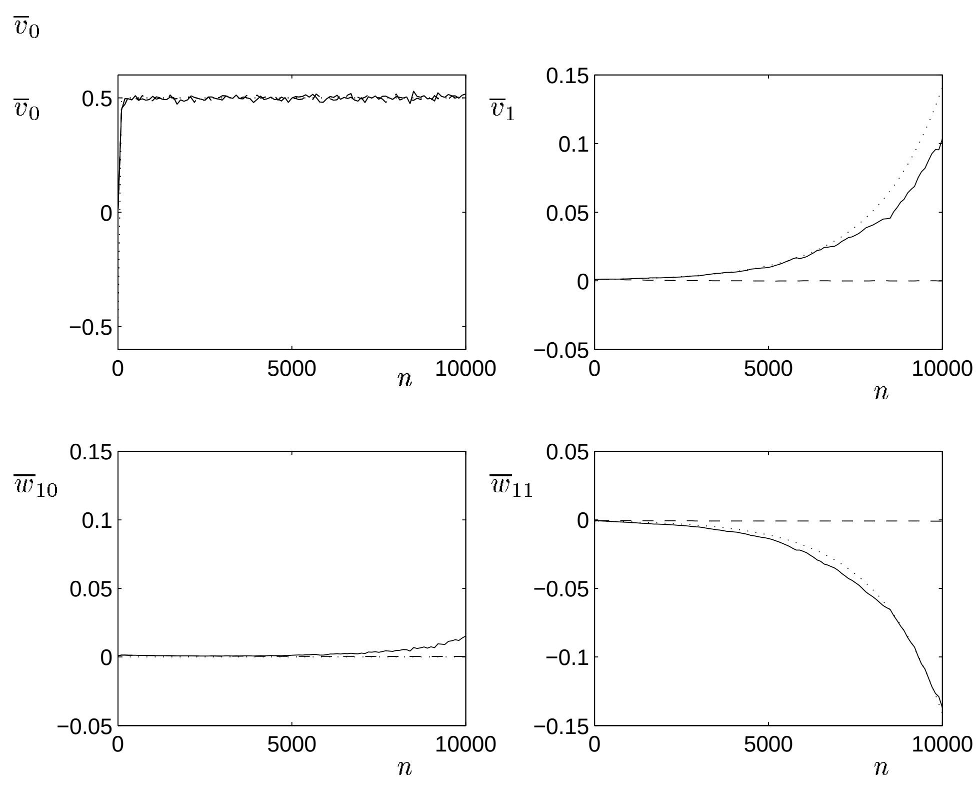

Figure 3.3 Weights obtained by simulations for natural learning (solid curves) and randomized learning (dashed curves) as functions of the number of iterations. Averaged over 100 iterations and an ensemble of 20 networks. The theoretical predictions computed with (3.26) are plotted as dotted curves.

least order P smaller in cyclic learning than in randomized learning. Indeed, it can be shown analytically that the leading term of the fluctuations completely vanishes in cyclic learning [32]. As a consequence randomized learning has a much larger chance to escape from a plateau than cyclic learning.

In conclusion, we recommend natural learning (with positive correlations) if the problem at hand suffers from a plateau. However, artificial dependencies introduced to reduce fluctuations are in such a case not advisable.

3.6 Summary and Discussion

This paper presents a quantitative analysis for on-line learning with dependent examples in a very general form. The analysis is based on two essential ingredients. One is the separation between the time scales of the example presentation and the weight dynamics. On the time scale needed for a representative sampling of the environment the weight changes must be negligible. A separation of time scales, which can be achieved using a small learning parameter, is essential in on-line learning to prevent over-specialization on single examples. The other essential ingredient is the assumption that the weights can be described by their average value with small superimposed fluctuations. In other words, the theory is locally valid, and may therefore not be suited for quantitative computations of global properties of the learning process, such as the stationary distribution of weights or the escape time out of a local minimum. However, even a local theory can be useful to understand some aspects of global properties [17]. Our study of learning on plateaus is an example of a local analysis of on-line learning which accounts for huge, non-local effects (section 3.5).

In section 3.3 we heuristically derived the first terms in a hierarchy of deterministic differential equations approximating the stochastic learning process. The leading term, the ODE term, only contains information of the stationary distribution of the examples. Dependencies between successive examples do not enter until the first correction to the ODE term. This implies that in general, when the ODE term is dominant, learning with dependent examples and learning with randomized examples are alike. The dependencies between examples merely act as corrections on the learning process, both in the fluctuations and in the bias. A rigorous derivation of the leading term, the ODE term, and of the Wiener process describing the fluctuations, can be found in [8, 44]. To our knowledge, a rigorous derivation of higher order terms, such as the bias term in (3.17.c) has not been studied before.

In section 3.4 we focussed on the asymptotic convergence of the learning process in terms of representation error and prediction error. The representation error is the expected average error of the network with respect to the whole environment. It measures how well the environment is represented by the network after learning. The prediction error is the network's expected average error with respect to the next presented example. It can be viewed as a measure for the irregularity of the example presentation. A remarkable relation between representation and prediction error is that the more predictable the examples, the larger the representation error.

In section 3.5 we studied on-line learning with a plateau. Plateaus are flat spots on the error surface which can severely slow down the learning process. In particular, backpropagation for multilayer perceptrons often suffers from plateaus. On a plateau the ODE contribution vanishes. The higher order terms, which contain the dependencies, therefore dominate the learning process. Simulations of a multilayer perceptron with backpropagation learning the tent map demonstrate that dependencies between successive examples can dramatically improve the final learning result. This phenomenon is explained by our analysis, which evidences that randomized learning gets stuck on a plateau, whereas the dependencies in natural learning cause the escape from the plateau. Predictions computed with the theory agree well with the simulations. At the end of this section we motivated our conjecture that if backpropagation suffers from plateaus then dependencies (with positive correlations) in example presentation can be helpful, and at least will not do any harm.

For convenience, the paper has been restricted to learning with a constant learning parameter in a stationary environment, i.e. the transition probability $\tau(\vec{x}|\vec{x}')$ between successive examples is independent of time. The theory can be extended straightforwardly to learning with time-dependent learning parameters $\eta(t)$ in a changing environment [8, 30], i.e. with a time-dependent transition probability $\tau(\vec{x}|\vec{x}';t)$, as long as the time scales of the learning parameter and the changing environment are large compared to the time scale of the learning process, and as long as this last time scale remains large compared to the time scale of the example presentation. As a consequence, time-dependent example selection techniques [54, 10, 49], possibly combined with a time-dependent learning parameter, may be devised and evaluated analytically. For instance, one can think of a scheme starting with dependencies designed to avoid plateaus and continuing in a later stage with dependencies designed for the fine tuning around minima. Perhaps such schemes will relate to common sense, like the pedagogical idea that the presentation of examples should start simple and gradually increase its complexity.

In fact, as long as the three previously mentioned time scales remain separated, the theory may also include weight-dependent transition probabilities $\tau(\vec{x}|\vec{x}';\boldsymbol{w},t)$ [8]. The vector $\vec{x}$ does not neccessarily represent an example. It may have components describing other fast variables.

For instance, in the next chapter we use fast variables to study learning with momentum, where the adaptation rule does not satisfy (3.1). Other obvious candidates for fast variables in neural network theory may be rapidly changing neuron states in recurrent networks. Thus, our framework may be applied to the analysis of the joint dynamics of neurons and weights [61].

In conclusion, the techniques for the local approximation of stochastic processes with separate time scales prove to be powerful tools for the analysis of on-line learning in neural networks.

Acknowledgments

We thank the referees for their useful suggestions.

Chapter 4

On-Line Learning with a Momentum Term

Abstract

We study on-line learning with momentum term for nonlinear learning rules. Through introduction of auxiliary variables, we show that the learning process can be described by a Markov process.

For small learning parameters η and momentum parameters α close to 1, such that $\gamma = \eta/(1-\alpha)^2$ is finite, the time scales for the evolution of the weights and the auxiliary variables are the same. In this case Van Kampen's expansion can be applied in a straightforward manner. We obtain evolution equations for the average network state and the fluctuations around this average. These evolution equations depend (after rescaling of time and fluctuations) only on γ: all combinations (η, α) with the same value of γ give rise to similar behavior.

The case α constant and η small requires a completely different analysis. There are two different time scales: a fast time scale on which the auxiliary variables equilibrate and a slow time scale for the change of the weights. By projection on the space of slow variables the fast variables can be eliminated. We find that for small learning parameters η and finite momentum parameters α learning with momentum is equivalent to learning without momentum term with rescaled learning parameter $\tilde{\eta} = \eta/(1-\alpha)$.

Simulations with the nonlinear Oja learning rule confirm the theoretical results.

Adapted from: W. Wiegerinck, A. Komoda and T. Heskes. Stochastic dynamics of learning with momentum in neural networks, *Journal of Physics A*, 27: 4425–4437, 1994.

4.1 Introduction

4.1.1 Background

On-line learning stands for learning in artificial neural networks where a weight change takes place each time a training pattern x is drawn at random from the total training set and is presented to the network. This weight change can be written in the general form

$$\Delta w(n) \equiv w(n+1) - w(n) = \eta \, f(w(n), x) \, , \tag{4.1}$$

with $w(n)$ the network state at iteration step n, η the so-called learning parameter, and $f(\cdot,\cdot)$ the learning rule. Because of the random presentation of patterns x, on-line learning as described by (4.1) is a stochastic process. The probability to be in a certain network state w can be shown to obey a master equation. In recent years, theoretical studies of this master equation have provided a better understanding of on-line learning processes [64, 29, 63, 24, 31, 28].

When the last weight change is added to the learning rule (4.1), the weight change takes the form

$$\Delta w(n) = \eta \, f(w(n), x) + \alpha \, \Delta w(n-1) \, , \tag{4.2}$$

with α the so-called momentum parameter. Equation (4.2) describes on-line learning with momentum term. The incorporation of this momentum term is frequently applied to backpropagation [68] with the intention to speed up learning with essentially no increase in computational complexity (see e.g. [27] for references). The backpropagation learning rule, like most learning rules in neural network literature, is nonlinear in the weights w. Theoretical studies, however, have been mainly focussing on the linear LMS algorithm with momentum updating [71, 72]. In this paper we will consider the effect of the momentum term on general nonlinear learning rules and ask ourselves whether incorporation of the momentum term leads to an improvement of the performance of on-line learning rules.

4.1.2 Framework

Equation (4.2) describes a second-order process. It can be turned into a (first order) Markov process through the introduction of the auxiliary variable $\mu(n) \equiv \Delta w(n-1)$:

$$\left\{ \begin{array}{rcl} \Delta w(n) & = & \eta \, f(w(n), x) + \alpha \, \mu(n) \\ \Delta \mu(n) & = & \eta \, f(w(n), x) + (\alpha - 1) \, \mu(n) \, . \end{array} \right.$$

With definitions $q \equiv (1-\alpha)\mu/\eta$, $\epsilon \equiv 1 - \alpha$, and $\gamma \equiv \eta/(1-\alpha)^2$, we can rewrite this to

$$\left\{ \begin{array}{rcl} \Delta w & = & \gamma\epsilon \, [(1-\epsilon) \, q + \epsilon \, f(w, x)] \\ \Delta q & = & \epsilon \, [f(w, x) - q] \, . \end{array} \right. \tag{4.3}$$

We are interested in the evolution of the probability $P(w, q, t)$ for the system to be in state (w, q) at time t. With Poisson-distributed time intervals between succeeding learning steps, this probability $P(w, q, t)$ obeys the master equation [7, 29]

$$\frac{\partial P(w,q,t)}{\partial t} = \int dw' \, dq' \, [T(w, q \mid w', q') \, P(w', q', t) - T(w', q' \mid w, q) \, P(w, q, t)] \, , \tag{4.4}$$

with transition probability

$$\begin{aligned} T(w, q \mid w', q') = \Big\langle & \delta(w - w' - \gamma\epsilon[(1-\epsilon) \, q + \epsilon \, f(w', x)]) \\ & \times \, \delta(q - q' - \epsilon \, [f(w', x) - q']) \Big\rangle_x \, , \end{aligned}$$

where $\langle\cdot\rangle_x$ denotes an average over the set Ω of training patterns. Averages with respect to the probability density $P(w, m, t)$ will be indicated by $\langle\cdot\rangle_\Xi$ or, more explicitly, by $\langle\cdot\rangle_{\Xi(t)}$. The master equation (4.4) is the starting point of the theoretical analysis presented in this paper. For notational convenience we treat the weight vector w as a one-dimensional variable. Generalization to higher dimensions is straightforward and has no influence on the basic ideas presented in this paper.

4.1.3 Outline

In section 2 we will study the system (4.3) for finite γ in the limit of very small ϵ, i.e., for small learning parameters η and momentum parameters α close to 1. In this case the time scales of the equations for the weight w and the auxiliary variable q are of the same order. We can immediately apply Van Kampen's expansion [77] to the master equation (4.4) and obtain evolution equations for the average weight w and the fluctuations around this average.

The situation ϵ finite and γ small, which corresponds to finite momentum parameters α (not close to 1) and (again) small learning parameters η, will be considered in section 3. Now the evolution of the auxiliary variable q takes place on a much faster time scale than the evolution of the weight w. Through projection of the master equation (4.4) on the "slow" space of the weight w, the fast variable q can be eliminated, resulting again in (approximate) evolution equations for the weight w.

In section 4 we check our theoretical results with simulations of the nonlinear Oja learning rule [55]. The main results are summarised and discussed in section 5.

4.2 Equal time scales

4.2.1 Van Kampen's expansion

In this section we will study the two-dimensional system (4.3) for small values of ϵ and finite values of γ, i.e., in the limits $\eta \to 0$ and $\alpha \to 1$ with a constant ratio $\gamma = \eta/(1-\alpha)^2$. The master equation (4.4) can be approximated for small parameters ϵ using Van Kampen's expansion. Basically (see [77, 31, 28] for a more detailed description of Van Kampen's expansion), this expansion is based on the assumption that the stochastic process (4.3) can be viewed as a deterministic trajectory with (small) superimposed fluctuations of order $\sqrt{\epsilon}$. Starting from the Ansätze

$$w = \phi + \sqrt{\epsilon}\,\xi \qquad \text{and} \qquad q = \psi + \sqrt{\epsilon}\,\chi\,,$$

Van Kampen's expansion yields evolution equations for the deterministic variables ϕ and ψ, and for the average and (co)variance of the noise terms ξ and χ.

After rescaling time with $\gamma\epsilon$ (we define a new time $\tau \equiv \gamma\epsilon t$), we obtain the deterministic equations

$$\left\{ \begin{array}{rcl} \dot{\phi} & = & \psi \\ \gamma\,\dot{\psi} & = & f_1(\phi) - \psi\,, \end{array} \right. \tag{4.5}$$

with drift $f_1(\phi)$, the first moment of the learning rule $f(\phi, x)$. For later purposes we give the general definition of the k-th jump moment:

$$f_k(\phi) \equiv \left\langle f^k(\phi, x) \right\rangle_x \,. \tag{4.6}$$

The evolution of the averages of the noise terms follows

$$\gamma \frac{d}{d\tau} \begin{pmatrix} \langle\xi\rangle_\Xi \\ \langle\chi\rangle_\Xi \end{pmatrix} = -A(\phi) \begin{pmatrix} \langle\xi\rangle_\Xi \\ \langle\chi\rangle_\Xi \end{pmatrix},$$

with

$$A(\phi) = \begin{pmatrix} 0 & -\gamma \\ -f_1'(\phi) & 1 \end{pmatrix}, \tag{4.7}$$

where the prime denotes differentiation of the function with respect to its argument. All learning networks are initialized at the same weight configuration, i.e., $w(0) = \phi(0)$ for all networks in the ensemble Ξ. This immediately implies $\langle\xi\rangle_{\Xi(t)} = \langle\chi\rangle_{\Xi(t)} = 0$ for all later times t. From (4.5) we then derive that the average network state $\langle w\rangle_\Xi = \phi$ obeys the second-order differential equation

$$\gamma\,\ddot{\phi} + \dot{\phi} - f_1(\phi) = 0\,.$$

The evolution of the covariance matrix

$$\Sigma^2 \equiv \begin{pmatrix} \langle\xi^2\rangle_\Xi & \langle\xi\chi\rangle_\Xi \\ \langle\xi\chi\rangle_\Xi & \langle\chi^2\rangle_\Xi \end{pmatrix}$$

is governed by

$$\gamma\,\frac{d}{d\tau}\Sigma^2 = -A(\phi)\,\Sigma^2 - \Sigma^2\,A(\phi) + D(\phi,\psi)\,, \tag{4.8}$$

with diffusion matrix

$$D(\phi,\psi) \equiv \begin{pmatrix} \gamma^2\,\psi^2 & \gamma\,\psi\,[f_1(\phi) - \psi] \\ \gamma\,\psi\,[f_1(\phi) - \psi] & f_2(\phi) - 2\,\psi\,f_1(\phi) + \psi^2 \end{pmatrix}.$$

The *a priori* Ansatz in Van Kampen's expansion is that the noise terms ξ and χ are of order 1. From (4.7) and (4.8), we see that this is valid for short times t and in regions of weight space where the real parts of the eigenvalues of the matrix $A(\phi)$ are positive, i.e., where $f_1'(\phi) < 0$. The same conditions hold for the validity of Van Kampen's expansion of the plain learning process (4.1) [31, 28].

4.2.2 Scaling properties

Let us take a closer look at the evolution equations for the average network state and the fluctuations around this average. With definitions

$$\sigma_1 \equiv \frac{1}{\gamma}\left\langle\xi^2\right\rangle_\Xi\,,\quad \sigma_2 \equiv \frac{1}{\gamma}\langle\xi\chi\rangle_\Xi\,,\quad\text{and}\quad \sigma_3 \equiv \left\langle\chi^2\right\rangle_\Xi\,,$$

the evolution equations (4.5) and (4.8) can be rewritten to

$$\left\{\begin{array}{rcl} \dot{\phi} - \psi &=& 0 \\ f_1(\phi) - \psi &=& \gamma\,\dot{\psi} \\ \dot{\sigma}_1 - 2\,\sigma_2 - \psi^2 &=& 0 \\ f_1'(\phi)\,\sigma_1 - \sigma_2 + \sigma_3 + \psi\,[f_1(\phi) - \psi] &=& \gamma\,\dot{\sigma}_2 \\ -2\,\sigma_3 + f_2(\phi) - 2\,\psi\,f_1(\phi) + \psi^2 &=& \gamma\,\dot{\sigma}_3 - 2\,\gamma\,f_1'(\phi)\,\sigma_2\,. \end{array}\right. \tag{4.9}$$

In this set of coupled differential equations, γ is the only remaining parameter. Suppose we know, through calculations or simulations, $\phi(\tau)$ and $\sigma_1(\tau)$ for a particular value of $\gamma = \eta/(1-\alpha)^2$. Then for all combinations (η,α) with this particular γ, the average weight and fluctuations at time t follow from (recall our definitions of time τ and variance σ_1)

$$\langle w\rangle_{\Xi(t)} = \phi(\tilde{\eta}\,t) \quad\text{and}\quad \left\langle w^2 - \langle w\rangle^2\right\rangle_{\Xi(t)} = \tilde{\eta}\,\sigma_1(\tilde{\eta}\,t)\,, \tag{4.10}$$

with "rescaled learning parameter" $\tilde{\eta} \equiv \eta/(1-\alpha)$. This rescaled learning parameter regulates the trade-off between speed and accuracy: a twice as large rescaled learning parameter leads to a twice as fast time scale, but also doubles the fluctuations in the weights. In section 4 we will describe simulations with the nonlinear Oja learning rule to check these scaling properties.

For small γ we can further simplify the set of equations (4.9). There are two different time scales: a slow time scale for the evolution of ϕ and σ_1 and a fast time scale for the evolution of ψ, σ_2 and σ_3. If we neglect all terms of order γ, we can eliminate the fast variables ψ, σ_2 and σ_3 and obtain

$$\left\{ \begin{array}{lcl} \dot{\phi} & = & f_1(\phi) \\ \dot{\sigma}_1 & = & 2\, f_1'(\phi)\, \sigma_1 \, + \, f_2(\phi) \, . \end{array} \right. \tag{4.11}$$

The same set of equations is obtained if Van Kampen's expansion is applied to the plain learning rule (4.1) with rescaled learning parameter $\tilde{\eta} = \eta/(1-\alpha)$ (see e.g. [31, 28]). Similar results have been reported in earlier studies on linear learning rules [71, 72, 59]. In the next section we will generalize these results to nonlinear learning rules for any finite value of the momentum parameter α, i.e., not close to 1. There we will go the other way around: first we will have to eliminate the fast variable q and only then we can apply Van Kampen's expansion.

4.3 Different time scales

4.3.1 Perturbation theory

In this section we will study the master equation (4.4) for small values of γ and finite values of ϵ, i.e., for small learning parameters η and momentum parameters α not close to 1. In these limits, we cannot approximate the master equation by Van Kampen's expansion as in section 2. However, as the results for $\gamma \ll 1$ obtained in the previous section suggest, there are two different time scales in the master equation. In the long time limit, we can try to eliminate the fast variable q and then obtain (a series expansion of) an evolution equation for $P(w,t) \equiv \int dq\, P(w,q,t)$. Our approach is very loosely based on the "adiabatic elimination of fast variables" in the theory of stochastic processes [23, 18].

Our starting point is the Kramers-Moyal expansion with respect to w'

$$\frac{\partial P(w,q,t)}{\partial t} = \left\{ \sum_{n=0}^{\infty} \frac{(-\gamma\epsilon)^n}{n!} \frac{\partial^n}{\partial w^n} \int dq' \, \Big\langle \left(\epsilon f(w,x) + (1-\epsilon) q'\right)^n \right.$$
$$\left. \times \; \delta\left(q - \{\epsilon f(w,x) + (1-\epsilon)q'\}\right) \Big\rangle_{\Omega} P(w,q',t) \right\} \; - \; P(w,q,t) \; , \tag{4.12}$$

which is a completely equivalent representation of the master equation (4.4) (see e.g. [77, 18]). From this Kramers-Moyal expansion we will derive evolution equations for the moments

$$Q_k(w,t) \equiv \int dq\, q^k P(w,q,t) \; , \qquad\qquad k = 0, \ldots, \infty.$$

Note that the moment vector $\vec{Q}(w,t)$ is just a different representation of the probability distribution $P(w,q,t)$ and that $Q_0(w,t) = P(w,t)$. Multiplying (4.12) by q^k and integrating over q yields (recall our definitions $\epsilon = 1-\alpha$ and $\tilde{\eta} = \eta/(1-\alpha)$).

$$\frac{\partial Q_k(w,t)}{\partial t} = \sum_{n=0}^{\infty} \frac{(-\tilde{\eta})^n}{n!} \frac{\partial^n}{\partial w^n} \sum_{l=0}^{n+k} \binom{n+k}{l} (1-\alpha)^{n+k-l} f_{n+k-l}(w) \alpha^l Q_l(w,t)$$
$$-Q_k(w,t), \tag{4.13}$$

with the jump moments $f_k(w)$ defined in (4.6). We can write equation (4.13) as a formal evolution equation (for notational convenience we suppress the w and t dependence):

$$\frac{\partial}{\partial t}\vec{Q} = \mathcal{H}\vec{Q} \tag{4.14}$$

with

$$\mathcal{H} = \sum_{n=0}^{\infty} \tilde{\eta}^n \mathcal{H}^{(n)} , \tag{4.15}$$

in which the matrices $\mathcal{H}^{(n)}$ are defined component wise by the operators

$$\mathcal{H}^{(n)}_{ij} = \left[\frac{(-1)^n}{n!} \frac{\partial^n}{\partial w^n} \sum_{l=0}^{n+i} \binom{n+i}{l} (1-\alpha)^{n+i-l} f_{n+i-l} \alpha^l \delta_{lj} \right] - \delta_{n0}\delta_{ij} , \tag{4.16}$$

for $i, j = 0, 1, \ldots, \infty$. The fact that the operator $\mathcal{H}$ can be written as a series in the small parameter $\tilde{\eta}$ [equation (4.15)] gives us the possibility to treat the system (4.14) using perturbation theory.

Let us first consider the unperturbed ($\tilde{\eta} = 0$) system

$$\frac{\partial}{\partial t}\vec{Q} = \mathcal{H}^{(0)}\vec{Q} . \tag{4.17}$$

From the triangular form of $\mathcal{H}^{(0)}$, we immediately find its degenerate eigenvalues

$$\lambda^{(0)}_\kappa = -(1-\alpha^\kappa) , \qquad \kappa = 0, 1, \ldots \infty .$$

We define $V^{(0)}_\kappa$ as the subspaces of eigenvectors with eigenvalue $\lambda^{(0)}_\kappa$, and $\mathcal{P}^{(0)}_\kappa$ as the orthogonal projectors (i.e., $\mathcal{P}^{(0)}_\kappa \mathcal{P}^{(0)}_\mu = \delta_{\kappa\mu}\mathcal{P}^{(0)}_\mu$) on the subspaces $V^{(0)}_\kappa$. These projectors commute with $\mathcal{H}^{(0)}$, i.e.,

$$\mathcal{P}^{(0)}_\kappa \mathcal{H}^{(0)} = \mathcal{H}^{(0)}\mathcal{P}^{(0)}_\kappa = -(1-\alpha^\kappa)\mathcal{P}^{(0)}_\kappa . \tag{4.18}$$

The projection $[\mathcal{P}^{(0)}_\kappa \vec{Q}](w,t)$ is called a "mode." From (4.18) it follows that the evolution of a mode is governed by

$$\frac{\partial}{\partial t}[\mathcal{P}^{(0)}_\kappa \vec{Q}] = \mathcal{H}^{(0)}[\mathcal{P}^{(0)}_\kappa \vec{Q}] = -(1-\alpha^\kappa)[\mathcal{P}^{(0)}_\kappa \vec{Q}] .$$

Since the modes are independent, the solution of the unperturbed system (4.17) is the sum of the solution of the modes:

$$\vec{Q}(w,t) = \sum_{\kappa=0}^{\infty} e^{-(1-\alpha^\kappa)t} [\mathcal{P}^{(0)}_\kappa \vec{Q}](w,0) .$$

The modes with $\kappa \neq 0$ will rapidly relax to equilibrium. We call these modes the fast modes. For large t, only the slow mode, i.e., the one with $\kappa = 0$, will remain. We write $\mathcal{P}^{(0)}$ as the projector on the slow mode, i.e., $\mathcal{P}^{(0)} \equiv \mathcal{P}^{(0)}_{\kappa=0}$. Consequently, the projector on the fast modes is $\mathbf{1} - \mathcal{P}^{(0)}$. So, for large t the fast modes will be equilibrated,

$$\left[\mathbf{1} - \mathcal{P}^{(0)}\right] \vec{Q} = 0 , \tag{4.19}$$

and only the dynamics on the slow mode remains:

$$\frac{\partial}{\partial t}\mathcal{P}^{(0)}\vec{Q} = \mathcal{H}^{(0)}\mathcal{P}^{(0)}\vec{Q} . \tag{4.20}$$

It is illustrative to see how we can arrive at an evolution equation for $P(w,t)$ from equation (4.20) using properties of the projector $\mathcal{P}^{(0)}(w)$ and the operator $\mathcal{H}(w)$. In the appendix it is shown that the projector $\mathcal{P}^{(0)}(w)$ has components

$$\mathcal{P}^{(0)}_{ij}(w) = v_i(w)\delta_{0j} \, ,$$

where the vector $\vec{v}(w)$ obeys $\mathcal{H}^{(0)}(w)\vec{v}(w) = 0$ and $v_0(w) = 1$. Using the constraint (4.19), we can express all components $Q_k(w,t)$ in terms of the zeroth component $Q_0(w,t)$: $Q_k(w,t) = v_k(w)Q_0(w,t)$. This corresponds to elimination of the fast variable q and can be compared with the elimination of the variables ψ, σ_2, and σ_3 in equation (4.9). Equation (4.20) now reduces to an equation for $Q_0(w,t) = P(w,t)$ only:

$$\frac{\partial}{\partial t}P(w,t) \;=\; 0 \, .$$

This equation makes the rather trivial statement that in the unperturbed system ($\tilde{\eta} = 0$) no learning takes place.

For the perturbed system (4.14) we follow the same line of reasoning as for the unperturbed system (4.17). The starting point of perturbation theory is the assumption that the eigenvalues and eigenvectors of the perturbed system can be written as an expansion in the perturbation parameter $\tilde{\eta}$. We define V_κ as the subspaces spanned by the eigenvectors corresponding to eigenvalues of which the unperturbed value is $\lambda^{(0)}_\kappa$, and $\mathcal{P}_\kappa$ as the orthogonal projectors on these subspaces V_κ. As in the unperturbed case, we decompose the perturbed system into modes.

The eigenvalues with $\kappa = 0$ are of order $\tilde{\eta}$, whereas the eigenvalues with $\kappa \neq 0$ are equal to $-(1-\alpha^\kappa)$ plus terms of order $\tilde{\eta}$. So, if $\tilde{\eta} \ll 1-\alpha$, the eigenvalues with $\kappa = 0$ are much smaller in absolute value than the eigenvalues with $\kappa \neq 0$, and we can still distinguish the slow mode from the fast modes. Again, we use the abbreviation $\mathcal{P} = \mathcal{P}_{\kappa=0}$ for the projector on the slow mode. For large t, the fast modes will be equilibrated, i.e.,

$$[\mathbf{1} - \mathcal{P}]\vec{Q} = 0 \, , \tag{4.21}$$

and only the dynamics on the slow mode remains:

$$\frac{\partial}{\partial t}\mathcal{P}\vec{Q} = \mathcal{H}\mathcal{P}\vec{Q} \, . \tag{4.22}$$

Due to the constraint (4.21), all the components Q_k of $\vec{Q}$ are determined once Q_0 is known. In other words, we can use the constraint (4.21) to derive a dynamical equation for $Q_0(w,t) = P(w,t)$ from equation (4.22). In this way, the fast variable q is eliminated from the master equation.

Since the operator $\mathcal{H}$ is only known in the form of a series expansion, the best we can achieve is a series expansion of the evolution equation for Q_0 in powers of $\tilde{\eta}$. In order to obtain this series expansion we only have to consider one of the components of (4.22). The zeroth component

$$\frac{\partial}{\partial t}(\mathcal{P}\vec{Q})_0 = (\mathcal{H}\mathcal{P}\vec{Q})_0$$

is the most obvious choice. The unknown quantities in this equation are both $\vec{Q}$ and the projector $\mathcal{P}$. Writing $\mathcal{P}$ as a series

$$\mathcal{P} = \sum_{n=0}^{\infty} \tilde{\eta}^n \mathcal{P}^{(n)} \tag{4.23}$$

we can subtract the desired components of $\mathcal{P}^{(n)}$ from the properties $\mathcal{P}^2 = \mathcal{P}$ and $\mathcal{HP} = \mathcal{PH}$. Using (4.21) and (4.23) we can then express the components Q_k with $k \neq 0$ in terms of the zeroth component Q_0 and derive an evolution equation for $Q_0 = P$ to arbitrary order in $\tilde{\eta}$. In appendix 4.A it is shown that this expansion yields

$$\begin{aligned} \frac{\partial}{\partial t}P(w,t) &= -\tilde{\eta}\frac{\partial}{\partial w}f_1(w)P(w,t) + \frac{\tilde{\eta}^2}{2}\frac{\partial^2}{\partial w^2}f_2(w)P(w,t) \\ &+ \frac{\tilde{\eta}^2\alpha}{1-\alpha}\left[\frac{\partial^2}{\partial w^2}f_1(w)^2P(w,t) - \frac{\partial}{\partial w}f_1(w)\frac{\partial}{\partial w}f_1(w)P(w,t)\right] + \mathcal{O}(\tilde{\eta}^3)\,. \end{aligned} \tag{4.24}$$

4.3.2 Van Kampen's expansion

To study (4.24) in the limit $\tilde{\eta} \to 0$, we apply Van Kampen's expansion [77]. We start with the Ansatz

$$w = \phi(\tau) + \sqrt{\tilde{\eta}}\zeta, \tag{4.25}$$

where $\tau = \tilde{\eta}t$ and $\phi(\tau)$ is a function to be determined (compare with section 2.1). Note that the constraint (4.21) and thus the evolution equation (4.24) are valid for times $t = \mathcal{O}(1/\tilde{\eta})$, i.e., for $\tau = \mathcal{O}(1)$. The function $\Pi(\zeta,\tau)$ is the probability P in terms of the new variable ζ:

$$\Pi(\zeta,\tau) \equiv P(\phi(\tau) + \sqrt{\tilde{\eta}}\zeta, \tau/\tilde{\eta})\,.$$

From Van Kampen's expansion it immediately follows that the deterministic part $\phi(\tau)$ has to satisfy the equation

$$\frac{d\phi(\tau)}{d\tau} = f_1(\phi(\tau)) \tag{4.26}$$

and that the evolution of $\Pi(\zeta,\tau)$ is governed by the Fokker-Planck equation

$$\frac{\partial \Pi(\zeta,\tau)}{\partial \tau} = -f_1'(\phi(\tau))\frac{\partial}{\partial \zeta}\zeta\Pi(\zeta,\tau) + \frac{1}{2}f_2(\phi(\tau))\frac{\partial^2}{\partial \zeta^2}\Pi(\zeta,\tau)\,. \tag{4.27}$$

The solution of the Fokker-Planck equation (4.27) is a Gaussian, so it suffices to determine the first and the second moments of ζ:

$$\begin{aligned} \frac{d\langle\zeta\rangle_\Xi}{d\tau} &= f_1'(\phi(\tau))\langle\zeta\rangle_\Xi \\ \frac{d\langle\zeta^2\rangle_\Xi}{d\tau} &= 2f_1'(\phi(\tau))\left\langle\zeta^2\right\rangle_\Xi + f_2(\phi(\tau))\,. \end{aligned} \tag{4.28}$$

From these equations, we see that the fluctuations ζ are bounded if
$f_1'(\phi(\tau)) < 0$. If f satisfies this condition, the Ansatz (4.25) is *a posteriori* justified. On the other hand, in case of non-negative $f_1'(\phi(\tau))$, the fluctuations grow in time and the expansion need not to be valid. As in section 2.1, this condition on f does not depend on α. Note further that $\langle\zeta^2\rangle_\Xi = \sigma_1$, the variance defined in section 2.2: the equations (4.26) and (4.28) are exactly equal to the set (4.11) which we derived in the limits $\alpha \to 1$ and $\gamma = \eta/(1-\alpha)^2 \to 0$.

Direct application of Van Kampen's expansion to learning equations without momentum [31, 28] leads to the equations (4.26) and (4.27) with $\tilde{\eta} = \eta$ substituted. In the first place, this result is a verification of our analysis, since learning without momentum is learning with $\alpha = 0$ and $\tilde{\eta} = \eta$. In the second place, the result shows that for learning parameters $\eta \ll (1-\alpha)^2$, from the point of view of the Fokker-Planck approximation, learning with momentum term is equivalent to learning without momentum term with rescaled learning parameter $\tilde{\eta}$.

4.4 Simulations

To illustrate the analytical results of the previous sections, we simulate the process of on-line learning with a momentum term for the nonlinear learning rule of Oja [55] in two dimensions

$$\Delta w(n) \;=\; \eta\,(x^T w(n))\,[x \;-\; (x^T w(n))\,w(n)] \;+\; \alpha \Delta w(n-1)\,.$$

Oja's rule searches for the principal component of the input correlation matrix $\langle\, x\, x^T \rangle_\Omega$. Inputs x are drawn at random from a rectangle centered at the origin, with sides of length 2 and 1 along the x_1- and x_2-axis, respectively. Simulations are performed with an ensemble of 100 000 independently learning networks. The networks in the ensemble are asynchronously updated. This means that at each step only *one*, randomly chosen network in the ensemble is updated. Hence, for a single network in the ensemble, the time intervals between updates are binomially distributed. For a large ensemble this distribution approaches a Poisson distribution [15]. The time scale t is such that there is on the average one learning step per unit of time for each network in the ensemble. All networks are initialized at the weight configuration $w(0) = (0.3, 0.3)^T$. Since the principal component of the input correlation matrix lies along the longest side of the rectangle, the weights w_1 and w_2 tend to 1 and 0, respectively. In figures 4.1 and 4.2 we plot the evolutions of the average weights $\langle w \rangle_{\Xi(t)}$ and of the trace of the covariance matrix

$$\sigma^2(t) = \sum_{i=1}^{2} \left\langle (w_i - \langle w_i \rangle_{\Xi(t)})^2 \right\rangle_{\Xi(t)}$$

for various values of α and η.

In section 4.2 we derived that, for small learning parameters η and momentum parameters α close to 1, all combinations (η, α) with the same value of $\gamma = \eta/(1-\alpha)^2$ give rise to similar behavior. We verify this scaling property in figure 4.1. In each graph we keep γ constant ($\gamma = 0.1$, 1, and 5 for figure 4.1(a), (b), and (c), respectively) and present curves for different values of α ($\alpha = 0.9$, 0.8, and 0.6 for solid, dashed, and dash-dotted lines, respectively). Time and variance are rescaled with $\tilde{\eta}$, i.e., we plot

$$\phi(\tau) \;=\; \langle w \rangle_{\Xi(\tau/\tilde{\eta})} \quad \text{and} \quad \sigma_1(\tau) \;=\; \sigma^2(\tau/\tilde{\eta})/\tilde{\eta}$$

as functions of the rescaled time τ [compare with equation (4.10)]. Curves with equal γ are almost overlapping, except for the quite extreme values $\alpha = 0.6$ and $\eta = 0.8$ [dash-dotted lines in figure 4.1(c)]: the simulation results are in perfect agreement with the scaling properties derived in section 4.2.

In section 4.3 we considered the case $\eta \ll (1-\alpha)^2$, i.e., $\gamma \ll 1$, and showed that combinations (η, α) are equivalent to combinations $(\tilde{\eta}, 0)$. This claim is verified in figure 4.2. In figures 4.2(a) and (b) the rescaled learning parameter $\tilde{\eta}$ is kept constant ($\tilde{\eta} = 0.01$ and 0.1, respectively), thus there is no need to rescale time and variance. Each graph shows curves with different values of α ($\alpha = 0$, 0.5, and 0.9 for solid, dashed, and dash-dotted lines, respectively), i.e., different values of γ. Curves inside each graph are almost overlapping, even when the values of γ differ by a factor 10 in magnitude [solid and dash-dotted line in figure 4.2(a)]. The exception is the dashed-dotted line in figure 4.2(b) where $\gamma = 1$ is not small enough for our analysis to be valid. We conclude that these simulation results agree very well with the analytical results of section 4.3.

4.5 Discussion

In this paper we studied nonlinear on-line learning rules with momentum term for small learning parameters η. We considered two cases: momentum parameters α close to 1, and finite

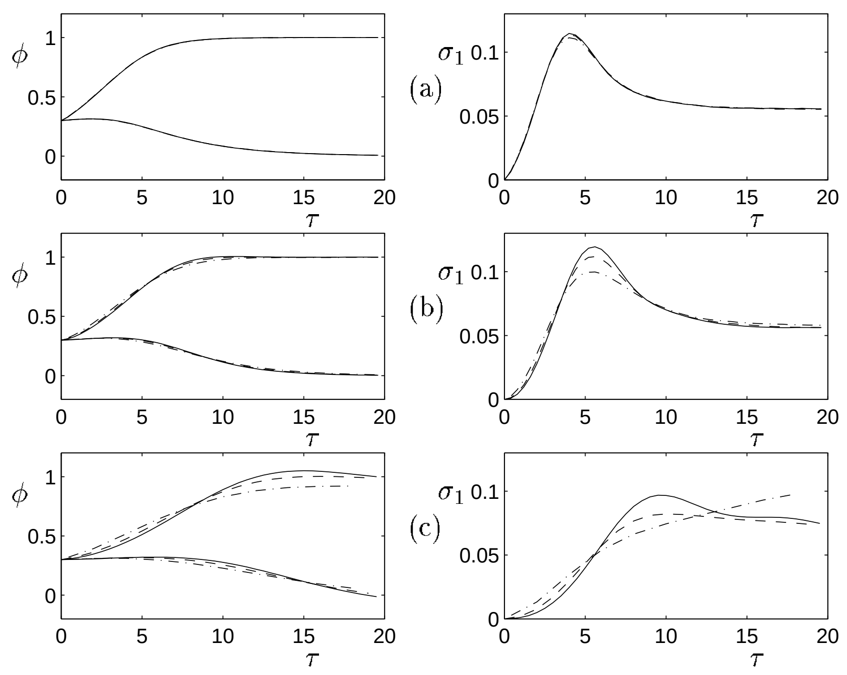

Figure 4.1 Oja learning with momentum updating. Means ϕ and rescaled sum of variances σ_1 as a function of rescaled time τ. All 100 000 networks start from $w = (0.3, 0.3)^T$. Momentum parameter $\alpha = 0.9$ for the solid li nes, $\alpha = 0.8$ for the dashed lines, and $\alpha = 0.6$ for the dash-dotted line s. (a) $\gamma = 0.1$; (b) $\gamma = 1$; (c) $\gamma = 5$.

momentum parameters α. In the first case we took the limits $\eta \to 0$ and $\alpha \to 1$, keeping $\gamma = \eta/(1-\alpha)^2$ constant. Using Van Kampen's expansion we arrived at evolution equations for the average weight vector and the fluctuations around this average. These evolution equations depend (after proper rescaling of time and size of fluctuations) only on this parameter γ. In the second case we kept α constant and again took the limit $\eta \to 0$. We arrived at the conclusion that learning with learning parameter η and momentum parameter α is equivalent to learning with rescaled learning parameter $\tilde{\eta} = \eta/(1-\alpha)$. Note that exactly the same conclusion follows from the first case in the limit $\gamma \to 0$. Thus, the analysis of the second case shows that the set of equations (4.9) resulting from the first case can be used not only for momentum parameters α close to 1, but also for more general values of α.

We tried to answer the question whether incorporation of the momentum parameter really improves the performance of general on-line learning rules. For learning parameters $\eta \ll (1-\alpha)^2$, we found that the effect of the momentum term is nothing but a rescaling of the learning parameter. For practical applications, this result is quite disappointing, but in agreement with the notion that the momentum term is hardly ever used in combination with on-line learning rules. The important exception is backpropagation. Indeed, it can be argued that incorporation of the momentum term is helpful for *batch-mode* backpropagation (see e.g. [71, 72, 60]), but neither these arguments, nor the results presented in this paper can explain the popularity of *on-line* backpropagation with momentum updating. There might be several reasons for this. Our analysis holds only for small learning parameters η and in regions of weight space in the vicinity of local minima, so, it could be that our analysis is too restricted. Another option is that the momentum term only helps if the momentum parameter α and learning parameter η are

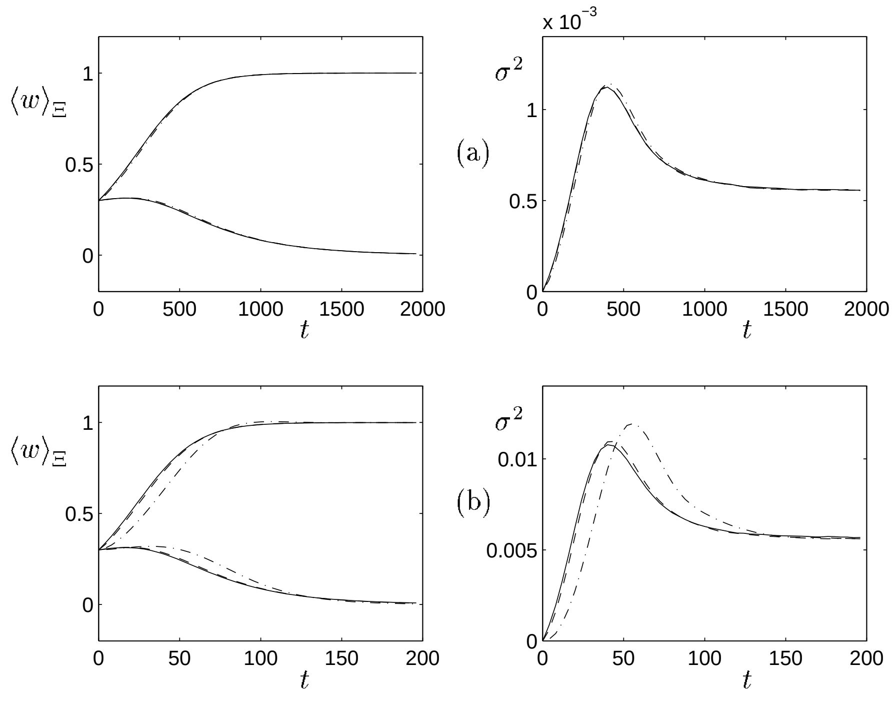

Figure 4.2 Oja learning with momentum updating. Means $\langle w \rangle_\Xi$ and sum of variances σ^2 as functions of time t. All 100 000 networks start from $w = (0.3, 0.3)^T$. Momentum parameter $\alpha = 0$ for the solid line s, $\alpha = 0.5$ for the dashed lines, and $\alpha = 0.9$ for the dash-dotted line s. (a) $\tilde{\eta} = 0.01$; (b) $\tilde{\eta} = 0.1$.

chosen such that $\gamma = \eta/(1-\alpha)^2 = \mathcal{O}(1)$. Analysis of the evolution equations for linear learning rules, that can be solved for any finite value of α and η and are valid in the whole weight space, do not show any significant improvement of on-line learning with momentum term if compared to learning without momentum term (unpublished results). For stronger evidence we will have to come up with a more general analysis of nonlinear learning with momentum updating and/or to work towards a better understanding of the nonlinear set of equations (4.9). At this point, we tend to the conclusion that the popularity of the momentum term in combination with on-line backpropagation cannot be explained in mathematical terms, but perhaps better in psychological terms: on-line backpropagators are afraid to choose a large learning parameter themselves.

Acknowledgments

This work was partly supported by a grant (P41RR05969) from the National Institutes of Health (TH). We thank Karin Krommenhoek and Bert Kappen for stimulating discussions.

4 Appendix

4.A Elimination of fast variables

In this appendix, we will show how to derive the evolution equation (4.24) for $P(w,t)$, starting from the evolution equation (4.14) for the moment vector $\vec{Q}(w,t)$. Since we are interested in

$P(w,t) = Q_0(w,t)$, we consider the zeroth component of

$$\frac{\partial}{\partial t}(\mathcal{P}\vec{Q})_0 = (\mathcal{H}\mathcal{P}\vec{Q})_0 \tag{4.A.1}$$

under the constraints

$$(\mathbf{1} - \mathcal{P})\vec{Q} = 0\,. \tag{4.A.2}$$

The operator $\mathcal{H}$ is defined in (4.15) and (4.16) and the projector $\mathcal{P}$ is written as a series expansion in (4.23). The unknown factors in (4.A.1) and (4.A.2) are not only the elements of the vector $\vec{Q}$, but also the components of the corrections $\mathcal{P}^{(n)}$ of the projector. The latter ones will be deduced from the relations $\mathcal{P}^2 = \mathcal{P}$ and $\mathcal{H}\mathcal{P} = \mathcal{P}\mathcal{H}$.

We expand both sides of (4.A.1) to the two lowest orders in $\tilde{\eta}$:

$$\begin{aligned}
&\frac{\partial}{\partial t}\sum_{j=0}^{\infty}\left\{\mathcal{P}^{(0)}_{0j} + \tilde{\eta}\mathcal{P}^{(1)}_{0j} + \mathcal{O}(\tilde{\eta}^2)\right\}Q_j \\
&= \sum_{j=0}^{\infty}\left\{\tilde{\eta}(\mathcal{H}^{(1)}\mathcal{P}^{(0)})_{0j} + \tilde{\eta}^2\{(\mathcal{H}^{(2)}\mathcal{P}^{(0)})_{0j} + (\mathcal{H}^{(1)}\mathcal{P}^{(1)})_{0j}\} + \mathcal{O}(\tilde{\eta}^3)\right\}Q_j\,,
\end{aligned} \tag{4.A.3}$$

where we used that $\mathcal{H}^{(0)}_{0j} = 0\ \forall_j$. Note that there is a global scale factor $\tilde{\eta}$ on the right hand side. This global scale factor will later be incorporated in a rescaled time. From (4.16) it follows that only the first $(n+1)$-th components of the zeroth row of $\mathcal{H}^{(n)}$ can be nonzero. Therefore we only need to calculate the first two rows $\mathcal{P}^{(1)}_{0j}$ and $\mathcal{P}^{(1)}_{1j}$, and the first three rows $\mathcal{P}^{(0)}_{0j}$, $\mathcal{P}^{(0)}_{1j}$ and $\mathcal{P}^{(0)}_{2j}$.

First, we calculate the components of the unperturbed projector $\mathcal{P}^{(0)}$. Using

$$\left[\mathcal{P}^{(0)}\right]^2 = \mathcal{P}^{(0)} \qquad \text{and} \qquad \mathcal{P}^{(0)}\mathcal{H}^{(0)} = \mathcal{H}^{(0)}\mathcal{P}^{(0)} = 0$$

we find that $\mathcal{P}^{(0)}$ has components

$$\mathcal{P}^{(0)}_{ij} = v_i\delta_{j0}$$

where $\vec{v}$ is the vector satisfying $\mathcal{H}^{(0)}\vec{v} = 0$, with the function-valued components

$$\begin{aligned}
v_0 &= 1\,, \\
v_1 &= f_1\,, \\
v_2 &= ((1-\alpha)f_2 + 2\alpha f_1^2)/(1+\alpha)\,, \\
v_3 &= \ldots\,.
\end{aligned}$$

Now we consider the first correction $\mathcal{P}^{(1)}$. From $\mathcal{P}^2 = \mathcal{P}$, the first correction $\mathcal{P}^{(1)}$ should satisfy

$$\mathcal{P}^{(1)} = \mathcal{P}^{(0)}\mathcal{P}^{(1)} + \mathcal{P}^{(1)}\mathcal{P}^{(0)}\,.$$

For the components of $\mathcal{P}^{(1)}$ this implies

$$\mathcal{P}^{(1)}_{ij} = \sum_{k=0}^{\infty}\mathcal{P}^{(1)}_{ik}v_k\delta_{j0} + v_i\mathcal{P}^{(1)}_{0j}\,,$$

which, after some rewriting, yields

$$\sum_{k=0}^{\infty} v_i\mathcal{P}^{(1)}_{0k}v_k = 0 \quad \text{and} \quad \mathcal{P}^{(1)}_{ij} = v_i\mathcal{P}^{(1)}_{0j} \quad \text{for } j \neq 0\,. \tag{4.A.4}$$

Since $\mathcal{P}$ commutes with $\mathcal{H}$, $\mathcal{P}^{(1)}$ should also obey

$$\mathcal{H}^{(1)}\mathcal{P}^{(0)} + \mathcal{H}^{(0)}\mathcal{P}^{(1)} = \mathcal{P}^{(0)}\mathcal{H}^{(1)} + \mathcal{P}^{(1)}\mathcal{H}^{(0)} \,. \tag{4.A.5}$$

Using the explicit forms of $\mathcal{H}^{(0)}$, $\mathcal{H}^{(1)}$ and $\mathcal{P}^{(0)}$, we deduce from (4.A.4) and (4.A.5) that

$$\begin{aligned}
\mathcal{P}^{(1)}_{00} &= \frac{\alpha}{1-\alpha}\frac{\partial}{\partial w} f_1 \,, \\
\mathcal{P}^{(1)}_{01} &= -\frac{\alpha}{1-\alpha}\frac{\partial}{\partial w} \,, \\
\mathcal{P}^{(1)}_{0k} &= 0 \qquad \text{for } k \geq 2 \,, \\
\mathcal{P}^{(1)}_{10} &= \frac{1+\alpha}{1-\alpha} f_1 \frac{\partial}{\partial w} f_1 - \frac{1}{(1-\alpha^2)}\frac{\partial}{\partial w}\{(1-\alpha) f_2 - 2\alpha f_1^2\} \,, \\
\mathcal{P}^{(1)}_{11} &= -\frac{\alpha}{(1-\alpha)} f_1 \frac{\partial}{\partial w} \,, \\
\mathcal{P}^{(1)}_{1k} &= 0 \qquad \text{for } k \geq 2 \,.
\end{aligned} \tag{4.A.6}$$

The constraint (4.A.2) gives the relation between the zeroth and the first component of $\vec{Q}$:

$$Q_1 = f_1 Q_0 + \mathcal{O}(\tilde{\eta}) \,. \tag{4.A.7}$$

Substitution of the expansions (4.16), (4.A.6) and (4.A.7) for the operator $\mathcal{H}$, the projector $\mathcal{P}$ and the moment vector $\vec{Q}$, respectively, into the evolution equation (4.A.1) finally leads to

$$\frac{\partial}{\partial t} Q_0 = \left\{ -\tilde{\eta}\frac{\partial}{\partial w} f_1 + \frac{\tilde{\eta}^2}{2}\frac{\partial^2}{\partial w^2} f_2 + \frac{\tilde{\eta}^2 \alpha}{1-\alpha}\left[\frac{\partial^2}{\partial w^2} f_1^2 - \frac{\partial}{\partial w} f_1 \frac{\partial}{\partial w} f_1\right]\right\} Q_0 + \mathcal{O}(\tilde{\eta}^3) \,,$$

which, after substitution of $Q_0(w,t) = P(w,t)$, is the desired result (4.24).

Chapter 5

The Connections of Large Perceptrons

Abstract

We derive analytical expressions for the connections of large perceptrons, by studying the fixed points of the perceptron learning rule. If the training set consists of *all* possible input vectors, we can calculate (for large systems) the connections as a series expansion in the system size. The leading term in this expansion turns out to be either the Hebb rule (for unbiased distributions) or the biased Hebb rule (for biased distributions). The performance of our asymptotic expressions (and finite size corrections) on small systems is studied numerically. For the more realistic case of having an extensive training set (patterns learned with training noise) we derive a self-consistent set of coupled non-linear equations for the connections. In the limit of zero training noise, the solution of these equations is shown to give the connections with maximal stability in the Gardner sense.

Adapted from: W. Wiegerinck and A. Coolen. The connections of large perceptrons, *Journal of Physics A*, 26: 2535–2548, 1993.

5.1 Introduction

One of the simplest (and oldest) models for the evolution in time of connections in neural systems is the perceptron [66, 51], equipped with the perceptron learning rule. Because of its simple architecture a perceptron can only perform a restricted set of operations, the so-called linearly separable functions. Nevertheless, perceptrons are a popular subject of study since the perceptron learning rule is one of the most transparent models for learning in neural systems for which a convergence theorem has been proved [51]. If a given task is linearly separable, then the perceptron learning rule converges in a finite number of iteration steps towards a connection vector that faithfully performs the task.

Statistical mechanical studies of perceptrons have resulted in a wealth of knowledge about properties like storage capacity, generalization [19, 40] and in a number of even more efficient perceptron-like learning rules [4, 12, 43] (with associated convergence theorems). For a more detailed overview of the literature on perceptrons and their properties we refer to textbooks like [51, 27, 53] or the recent review by Watkin et al [78]. In particular Opper [56, 57] seems to have been the first to study analytically the dynamics of Perceptron-like learning rules (he calculated learning times and the probability density of the embedding strengths of patterns in an optimally stabilized perceptron). What is still missing in the literature, however, is a method to *calculate analytically* the connections which are the outcome of such learning rules. More generally: given a linearly separable task T and given a training set Ω of input vectors one would like to calculate the connection vectors $\vec{J}$ that will faithfully perform the task T for all vectors in Ω.

In this paper we address this problem. We try to calculate the connection vectors $\vec{J}$ that perform a given task T on a given input set $\Omega \subseteq \{-1,1\}^N$, by using the fact that such connections are fixed points of the perceptron learning rule. If the training set consists of all possible input vectors, $\Omega = \{-1,1\}^N$, the fixed point equations enable us to calculate the connections as a series expansion in powers of $1/\sqrt{N}$ (N is the number of input units). If, on the other hand, the training set consists of an extensive number $p = \alpha N$ of prototype patterns $\vec{\xi}^\mu$ in combination with small regions Ω_μ around these patterns (i.e., training with noise), we find that the connections satisfy a self-consistent set of nonlinear equations. In the limit of zero training noise the solution of these equations gives exactly the interactions with maximal stability in the Gardner sense.

5.2 The Perceptron Fixed-Point Equation

A standard perceptron [66, 51] performs a mapping from $\{-1,1\}^N$ to $\{-1,1\}$ (N is the number of binary input units). The state u of the binary output unit depends on the states of the N binary input units $s_i \in \{-1,1\}^N$ in the following way:

$$u(\vec{s}) = \mathrm{sgn}(\vec{J}\cdot\vec{s}) \qquad \vec{J} \in \mathbb{R}^N . \tag{5.1}$$

Those mappings $T : \{-1,1\}^N \to \{-1,1\}$ that can be written in the form (5.1) are called linearly separable functions. Given a specific linearly separable function T and a set $\Omega \subseteq \{-1,1\}^N$ of input vectors, the perceptron problem is: find a vector $\vec{J} \in \mathbb{R}^N$ of connections such that $u(\vec{s}) = T(\vec{s})$ for all $\vec{s} \in \Omega$. The vectors $\vec{J}$ that solve the problem are the solutions of:

$$\mathrm{sgn}(\vec{J}\cdot\vec{s}) = T(\vec{s}) \qquad \forall \vec{s} \in \Omega . \tag{5.2}$$

This paper tries to calculate the solutions of (5.2) and to find an analytical expression for the connections $\vec{J}$ in terms of the task T on Ω.

Instead of trying to solve (5.2) directly, we will make use of a specific property of the perceptron learning rule [66, 51], of which we know that the fixed points are solutions of (5.2).

The perceptron learning rule is defined as the modification of connections via the following stochastic procedure

$$\begin{array}{ll} 1) & \text{draw at random an input vector } \vec{s} \in \Omega \\ & \text{according to the probability distribution } p(\vec{s}) \\ 2) & \Delta \vec{J} = \frac{1}{2}\epsilon\, \vec{s}\Big[T(\vec{s}) - \mathrm{sgn}(\vec{J}\cdot\vec{s})\Big] \\ 3) & \text{return to 1)} \end{array} \tag{5.3}$$

where $\epsilon > 0$ is the learning parameter. This procedure was shown [51] to converge in a finite number of iteration steps towards a solution of (5.2), provided that T is indeed linearly separable (which we assume to be the case). Calculating the fixed points of (5.3) is equivalent to solving the original problem (5.2). By writing the perceptron learning rule as a master equation, and expanding the master equation in powers of ϵ, we can separate the macroscopic part (of order ϵ^0) from the fluctuation part (of order $\sqrt{\epsilon}$) [77]. After a rescaling of time by a factor ϵ, the macroscopic part obeys the deterministic differential equation

$$\frac{d}{dt}\vec{J} = \frac{1}{2}\left\langle \vec{s}\Big[T(\vec{s}) - \mathrm{sgn}(\vec{J}\cdot\vec{s})\Big]\right\rangle_\Omega$$

where $\langle \cdots \rangle_\Omega$ indicates averaging over the distribution $p(\vec{s})$. Whatever the details of the fluctuation part, we know the perceptron rule will evolve towards a fixed-point. Fixed-points of the (stochastic) rule (5.3) are automatically fixed-points of the above *deterministic* equation. An important property of the perceptron learning rule is that the inverse of this statement is found to be true as well:

$$T(\vec{s}) = \mathrm{sgn}(\vec{J}\cdot\vec{s}) \quad \forall \vec{s}\in\Omega \qquad \Leftrightarrow \qquad \langle \vec{s}T(\vec{s})\rangle_\Omega = \left\langle \vec{s}\,\mathrm{sgn}(\vec{J}\cdot\vec{s})\right\rangle_\Omega . \tag{5.4}$$

It is trivial to prove that the right-hand side of (5.4) follows from the left-hand side. Here we will only prove the complementary statement. Since T is linearly separable on Ω (by definition), there exists a vector $\vec{B} \in \mathrm{I\!R}^N$, such that $T(\vec{s}) = \mathrm{sgn}(\vec{B}\cdot\vec{s})$ on Ω. This allows us to write:

$$\begin{aligned} 0 &= \vec{B}\cdot\langle \vec{s}T(\vec{s})\rangle_\Omega - \vec{B}\cdot\left\langle \vec{s}\,\mathrm{sgn}(\vec{J}\cdot\vec{s})\right\rangle_\Omega \\ &= \left\langle |\vec{B}\cdot\vec{s}|\left[1 - \mathrm{sgn}(\vec{B}\cdot\vec{s})\,\mathrm{sgn}(\vec{J}\cdot\vec{s})\right]\right\rangle_\Omega . \end{aligned}$$

Since $|\vec{B}\cdot\vec{s}| > 0$ ($\forall \vec{s}\in\Omega$), we must conclude that $\mathrm{sgn}(\vec{B}\cdot\vec{s}) = \mathrm{sgn}(\vec{J}\cdot\vec{s})$ ($\forall \vec{s}\in\Omega$) (which completes the proof of (5.4)). Note that the fixed point theorem (5.4) is exact for all N, all $\Omega \subseteq \{-1,1\}^N$, all non-zero distributions $p(\vec{s})$ on Ω and all linearly separable tasks T.

Theorem (5.4) provides a reduction of the original problem (5.2) of finding the solution of a set of $|\Omega|$ coupled inequalities to the problem of finding the solution of a set of N coupled non-linear equations. The rest of our paper aims at calculating the solutions of these equations:

$$\langle \vec{s}T(\vec{s})\rangle_\Omega = \left\langle \vec{s}\,\mathrm{sgn}(\vec{J}\cdot\vec{s})\right\rangle_\Omega \tag{5.5}$$

where there is still freedom in choosing any (non-zero) probability distribution on Ω. We interpret this distribution as defining the probabilities with which individual inputs $\vec{s}\in\Omega$ are drawn during the learning process.

5.3 Homogenous Distributions

In this section we use the fixed-point theorem (5.3) for calculating analytically (as a series expansion in inverse powers of the system size N) the connections that perform a given task T, for the simplest case in which the traing set consists of the set of all possible input vectors: $\Omega \equiv \{-1,1\}^N$. We study two choices with respect to the probability distribution on this training set: unbiased (uniform) probabilities and biased probabilities.

5.3.1 Unbiased Homogeneous Distribution

The first case we study is $\Omega \equiv \{-1,1\}^N$, $p(\vec{s}) = 2^{-N}$ (the uniform distribution). In this case the right-hand side of (5.5) can be calculated exactly (including all orders of N). First we rewrite:

$$\left\langle s_i \,\mathrm{sgn}(\vec{J}\cdot\vec{s})\right\rangle_\Omega = \int_{-\infty}^{\infty} dz P_i(z)\,\mathrm{sgn}[J_i+z] = 2\int_0^{J_i} dz P_i(z) \tag{5.6}$$

where

$$P_i(z) = \left\langle \delta\left[z - \sum_{j\neq i} J_j s_j\right]\right\rangle_\Omega .$$

Note that the inversion symmetry $p(\vec{s}) = p(-\vec{s})$ of the uniform distribution implies that $P_i(z) = P_i(-z)$. In Appendix 5.A we analyse probability distributions $P(z)$ of the above form. In terms of the variables at hand the result is

$$P_i(z) = \left[2\pi K_i^2\right]^{-\frac{1}{2}} \exp\left[-\frac{z^2}{2K_i^2}\right]\left[1 - \sum_{n\geq 2} D_{in}(\hat{J})(-1)^n 2^{-n} H_{2n}\left(\frac{z}{\sqrt{2}K_i}\right)\right] \tag{5.7}$$

where

$$K_i \equiv \sqrt{\sum_{j\neq i} J_j^2} = \|\vec{J}\|\sqrt{1-\hat{J}_i^2} \qquad (\hat{J}_i = J_i/\|\vec{J}\|)\,.$$

The functions $H(x)$ are the Hermite polynomials. The coefficients $D_{in}(\hat{J})$ are given by

$$\begin{aligned} D_{in}(\hat{J}) \;\equiv\; & \sum_{k\leq n/2-1} \frac{(-1)^k}{(k+1)!} \sum_{m_1=2}^{n} \cdots \sum_{m_{k+1}=2}^{n} \Big[\delta_{n,\sum m_i} \\ & \times\, C_{m_1}\ldots C_{m_{k+1}} Q_{im_1}(\hat{J})\ldots Q_{im_{k+1}}(\hat{J})\Big] \end{aligned}$$

where C_n and $Q_{in}(\hat{J})$ are defined as:

$$\begin{aligned} C_n \;&\equiv\; \frac{2^{2n-1}(2^{2n}-1)|B_{2n}|}{n(2n)!} \qquad (B_m \ \text{: Bernoulli numbers [22]}) \\ Q_{in} \;&=\; \sum_{j\neq i} \hat{J}_j^{2n}\left[1-\hat{J}_i^2\right]^{-n} \in [0,1]\,. \end{aligned}$$

Using (5.7) we can now perform the integral in (5.6). The equation from which the solution of the fundamental problem (5.5) must be calculated thereby becomes:

$$\begin{aligned} \langle s_i T(\vec{s})\rangle_\Omega \;=\; & \mathrm{erf}\left[\frac{\hat{J}_i}{\sqrt{2}\sqrt{1-\hat{J}_i^2}}\right] \\ & + \frac{2}{\sqrt{\pi}} \exp\left[-\frac{\hat{J}_i^2}{1-\hat{J}_i^2}\right] \sum_{n\geq 2} D_{in}(\hat{J}) 2^{-n}(-1)^n H_{2n-1}\left(\frac{\hat{J}_i}{\sqrt{2}\sqrt{1-\hat{J}_i^2}}\right) . \end{aligned} \tag{5.8}$$

So far no approximation has been made. Eq. (5.8) is completely equivalent to (5.5), including finite size effects. However, Eq.(5.8) is much more suitable for calculating the solution $\vec{J}$ as a series in powers of N than the original equation (5.5).

If, for instance, we assume that $\hat{J}_i = \mathcal{O}\left(N^{-1/2}\right)$ (motivated by $\sum \hat{J}_i^2 = 1$), we can expand Eq.(5.8) in powers of $N^{-1/2}$ up to any desired order:

$$\langle s_i T(\vec{s})\rangle_\Omega = \sqrt{\frac{2}{\pi}}\hat{J}_i + \mathcal{O}\left(N^{-3/2}\right) = \sqrt{\frac{2}{\pi}}\left[\hat{J}_i + \frac{1}{3}\hat{J}_i^3 - \frac{1}{4}\hat{J}_i\sum_j \hat{J}_j^4\right] + \mathcal{O}\left(N^{-5/2}\right) \qquad (5.9)$$

where we have used $D_{in} = \mathcal{O}\left(N^{1-n}\right)$. Since $\hat{J}_i = \mathcal{O}\left(N^{-1/2}\right)$ for all i, we expand the solution $\vec{J}$ of (5.9) in powers of $N^{-1/2}$. Substitution of this expansion into Eq. (5.9) yields:

$$\begin{aligned}\hat{J}_i &= \sqrt{\frac{\pi}{2}}\langle s_i T(\vec{s})\rangle_\Omega + \mathcal{O}\left(N^{-3/2}\right) &(5.10)\\ &= \sqrt{\frac{\pi}{2}}\langle s_i T(\vec{s})\rangle_\Omega\left[1 - \frac{\pi}{6}\langle s_i T(\vec{s})\rangle_\Omega^2 + \frac{\pi^2}{16}\sum_j \langle s_j T(\vec{s})\rangle_\Omega^4\right] + \mathcal{O}\left(N^{-5/2}\right) &(5.11)\end{aligned}$$

where $\langle s_i T(\vec{s})\rangle_\Omega$ is found to be of order $N^{-1/2}$. Equations (5.10,5.11) show that, if for the training set we choose $\Omega = \{-1,1\}^N$ with uniform probabilities and if the task vector $\vec{B}$ and thus the vector $\vec{J}$ have components such that $\hat{J}_i = \mathcal{O}\left(N^{-1/2}\right)$, then (in first order in $N^{-1/2}$) we find the connections $\vec{J}$ to be proportional to the ones obtained by applying the Hopfield [34] version of Hebb's [26] rule to the full set $\{-1,1\}^N$ of input vectors. This result agrees with the findings of Vallet [73] who showed that for large systems ($N \to \infty$) and for a specific type of task vectors $\vec{B}$ (which satisfy our condition) Hebb's rule learns and generalizes well if the number p of examples (drawn from a uniform distribution) diverges sufficiently fast ($p/N \to \infty$ as $N \to \infty$). The second order ($N^{-3/2}$) in Eq. (5.11) can be interpreted as finite size corrections to Hebb's rule.

5.3.2 Biased Homogeneous Distribution: The Gaussian Approach

The next case we will study is the biased homogeneous distribution of input vectors: $\Omega = \{-1,1\}^N$, $p(\vec{s}) \equiv p_1(s_1)\dots p_N(s_N)$ (the variables $\{s_i\}$ are still independent). The individual probabilities are written as:

$$p_i(s) \equiv \frac{1}{2}(1+a_i)\delta_{s,1} + \frac{1}{2}(1-a_i)\delta_{s,-1} \qquad -1 < a_i < 1\,.$$

We will also allow for a threshold, both in the definition of the task

$$T(\vec{s}) \equiv \operatorname{sgn}(\vec{B}\cdot\vec{s} + B_0)$$

and in the perceptron itself (e.g. by adding a dummy input variable $s_0 \equiv -1$). According to the fixed point theorem (5.4) the solutions of the perceptron problem are the solutions $(\vec{J}; J_0)$ of

$$\begin{aligned}\langle \vec{s}T(\vec{s})\rangle_\Omega &= \left\langle \vec{s}\operatorname{sgn}(\vec{J}\cdot\vec{s} + J_0)\right\rangle_\Omega\\ \langle T(\vec{s})\rangle_\Omega &= \left\langle \operatorname{sgn}(\vec{J}\cdot\vec{s} + J_0)\right\rangle_\Omega\,.\end{aligned}$$

To simplify algebra we will now assume that for large N the terms $\sum_j J_j s_j$ have a Gaussian probability distribution (in Appendix 5.B we analyse the conditions to be imposed on $\vec{B}$ and

$\vec{J}$ for this assumption to be justified). In doing so we will no longer be able to calculate finite size corrections to the $N \to \infty$ result (in contrast to the approach followed in the previous subsection). After some algebra we now obtain:

$$\begin{aligned}
\langle s_i T(\vec{s})\rangle_\Omega &= \frac{1}{2}(1+a_i)\,\mathrm{erf}\left[\frac{J_i(1-a_i)+J_0+\vec{J}\cdot\vec{a}}{\sqrt{2}\sigma_i}\right] \\
&+ \frac{1}{2}(1-a_i)\,\mathrm{erf}\left[\frac{J_i(1+a_i)-J_0-\vec{J}\cdot\vec{a}}{\sqrt{2}\sigma_i}\right] \qquad (5.12) \\
\langle T(\vec{s})\rangle_\Omega &= \mathrm{erf}\left[\frac{J_0+\vec{J}\cdot\vec{a}}{\sqrt{2}\sigma_0}\right] \qquad (5.13)
\end{aligned}$$

in which $\sigma_0 = \sum_j J_j^2(1-a_j^2)$ and $\sigma_i^2 = \sigma_0^2 - J_i^2(1-a_i^2)$. Since $\sum_j J_j s_j$ has a Gaussian distribution we may use the fact that $\hat{J}_i \ll 1$ to expand (5.12,5.13)

$$\begin{aligned}
\langle s_i T(\vec{s})\rangle_\Omega &= a_i\,\mathrm{erf}\left[\frac{(\hat{J}_0+\hat{J}\cdot\vec{a})}{\sqrt{Z}}\right] \\
&+ \hat{J}_i(1-a_i^2)\frac{2}{\sqrt{\pi Z}}\exp\left[-\frac{(\hat{J}_0+\hat{J}\cdot\vec{a})^2}{Z}\right] + \mathcal{O}\left(\hat{J}_i^2\right) \\
\langle T(\vec{s})\rangle_\Omega &= \mathrm{erf}\left[\frac{(\hat{J}_0+\hat{J}\cdot\vec{a})}{\sqrt{Z}}\right]
\end{aligned}$$

where $\hat{J}_0 \equiv J_0/J$ and $Z \equiv 2J^{-2}\sigma_0^2$. We can now invert these relations and find for $N \to \infty$ in leading order:

$$\begin{aligned}
\hat{J}_i &= \frac{\sqrt{\pi Z}}{2(1-a_i^2)}\langle (s_i-a_i)T(\vec{s})\rangle_\Omega \exp\left[\mathrm{erf}^{-1}(\langle T(\vec{s})\rangle_\Omega)\right]^2 \qquad (5.14) \\
\hat{J}_0 &= \sqrt{Z}\left[\mathrm{erf}^{-1}(\langle T(\vec{s})\rangle_\Omega)\right. \\
&\left. - \frac{\sqrt{\pi}}{2}\sum_j \frac{a_j}{1-a_j^2}\langle (s_j-a_j)T(\vec{s})\rangle_\Omega \exp\left[\mathrm{erf}^{-1}(\langle T(\vec{s})\rangle_\Omega)\right]^2\right]. \qquad (5.15)
\end{aligned}$$

Since a rescaling of both $\vec{J}$ and J_0 does not affect the mapping performed by the perceptron $(\vec{J}; J_0)$, there is in principle no need to calculate the factor Z explicitly. If one puts $a_i \equiv a$ and if the thresholds B_0 and J_0 are choosen to be zero, then (5.15) reduces to

$$\hat{J}_i = \frac{1}{Z'}\langle (s_i - a)T(\vec{s})\rangle_\Omega \qquad (5.16)$$

where Z' is a proper normalization factor. Finally one can verify that for $a = 0$ one recovers the first order of (5.10).

The final result of this section, Eqns.(5.14,5.15,5.16) shows that (in leading order in the system size N) the connections $\vec{J}$, expressed in terms of biased statistics of the binary input variables s_i, are found to be proportional to the ones obtained by applying the biased Hebbian rule of [3] to the full set $\{-1,1\}^N$ of input vectors. The biased Hebbian rule of [3] seems to make use of *global* information; since the Perceptron learning rule is *nonlocal* (because of the appearance of the crucial *global* term $T(\vec{s}) - \mathrm{sgn}(\vec{J}\cdot\vec{s})$) the resulting interactions are indeed allowed to depend on nonlocal quantities. The condition on the task vectors $\vec{B}$ for our analysis to apply is that for large N the inner product $\vec{B}\cdot\vec{s}$ must have a Gaussian probability distribution. In addition we have found an expression for the threshold J_0. In Appendix 5.B we show that the assumption of a Gaussian distribution is justified with probability one if the task vectors $\vec{B}$ are drawn at random from, for instance, a spherically symmetric distribution or a hypercube in $\mathbb{R}^N$.

5.3.3 Numerical Results

The performance in finite systems of our asymptotic ($N \to \infty$) expressions for the connections is studied numerically. We calculate over a given ensemble $\mathcal{P}$ of linearly separable tasks the average overlap $\mathcal{Q}_N$ between the task function $T(\vec{s}) \equiv \text{sgn}(\vec{B}\cdot\vec{s})$ and the perceptron mapping upon choosing for the connections $\vec{J}(\vec{B})$ either the truncated expansions (5.10,5.11) or the result (5.16) obtained with the Gaussian approach:

$$\mathcal{Q}_N \equiv \int d\vec{B}\;\mathcal{P}(\vec{B})\,\left\langle \text{sgn}\left[\vec{B}\cdot\vec{s}\right]\,\text{sgn}\left[\vec{J}(\vec{B})\cdot\vec{s}\right]\right\rangle_{\vec{s}} \tag{5.17}$$

with

$$\langle\ldots\rangle_{\vec{s}} \equiv 2^{-N}\sum_{\vec{s}\in\{-1,1\}^N}\cdots$$

If $\mathcal{Q}_N = 1$ then, with probability one, the mappings performed by $\vec{B}$ and $\vec{J}(\vec{B})$ will be identical for tasks drawn from $\mathcal{P}$. For the ensemble $\mathcal{P}(\vec{B})$ of tasks we took the uniform probability distribution over the N-dimensional hypercube. The integral in (5.17) is estimated numerically from 100 samples of randomly drawn task vectors vectors $\vec{B}$. The average over the input vector distribution is calculated exactly; since this average involves 2^N input vectors $\vec{s}$, we have restricted the range of our experiments to $N < 20$.

In figure 5.1 we show the values of $\mathcal{Q}_N$ thus obtained upon choosing for $\vec{J}(\vec{B})$ the leading order in N for unbiased distributions (the Hebb rule) as given by (5.10) ('+'), the first two leading orders in N for unbiased distributions (the Hebb rule corrected for finite size effects) as given by (5.11) ('$*$') and the leading order in N for biased distributions (the biased Hebb rule), for $a = 0.5$, as given by (5.16) ('o'). For unbiased distributions we find that the asymptotic expressions (5.10,5.11) are such that the corresponding perceptron mappings are almost identical to the task to be learned, even for relatively small system sizes. Including the second order in N ('$*$') indeed improves performance by correcting for finite size effects. The fact that even the leading term ('+') performs perfectly for $N < 4$ can be proved analytically. For biased distributions ('o') we find that finite size effects play a considerably more important role.

5.4 Inhomogeneous Distributions: Patterns

In this section we use the fixed-point theorem (5.3) for calculating analytically the connections that perform a given task T for the more realistic case in which the training set consists of a union of clusters around $p = \alpha N$ patterns $\vec{\xi}^\mu$ (training noise): $\Omega \equiv \bigcup_\mu \Omega_\mu \neq \{-1,1\}^N$. We show that the asymptotic ($N \to \infty$) connections are given by the solutions of a set of coupled non-linear equations.

5.4.1 Training with Noise

We consider the case of having to classify a given set of $p \equiv \alpha N$ input patterns $\vec{\xi}^\mu \in \{-1,1\}^N$ (for $N \to \infty$) using a perceptron without a threshold. To obtain a classification which is stable

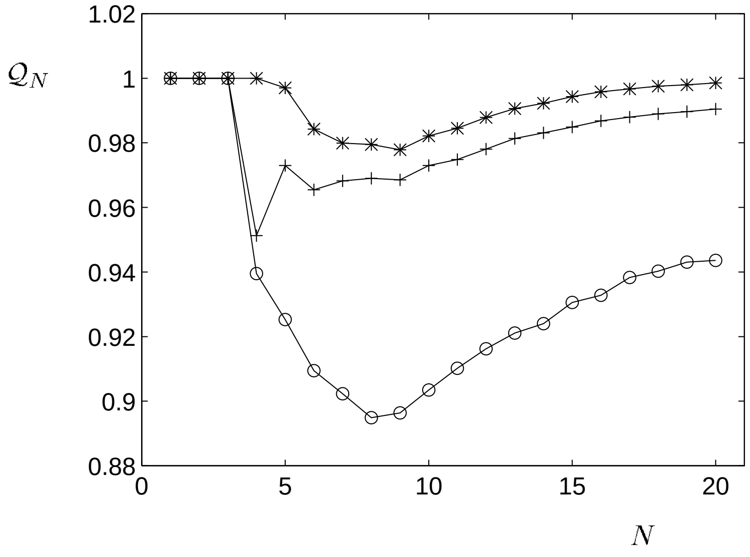

Figure 5.1 The average performance Q_N as a function of the system size N. The connections are defined by the leading order in Eq. (5.10) (the Hebb rule for unbiased input distributions) ('+'), the two leading orders in of Eq. (5.11) (the Hebb rule plus finite size corrections) ('$*$') and the leading order in Eq. (5.16) (the biased Hebb rule for biased input distributions) with bias $a = 0.5$ ('o').

against input noise, the task is defined on small, equally large, disjunct neighborhoods Ω_μ around the pattern $\vec{\xi}^\mu$:

$$T(\vec{s}) = T(\vec{\xi}^\mu) \qquad \forall \vec{s} \in \Omega_\mu \, .$$

The set Ω is the union of the p subsets: $\Omega \equiv \bigcup_\mu \Omega_\mu$. According to our fixed-point theorem (5.4), applied to the present situation, the connections performing the task T on Ω are the solutions of:

$$\frac{1}{p}\sum_{\mu=1}^{p} T(\vec{\xi}^\mu)\,\langle \vec{s}\rangle_{\Omega_\mu} = \frac{1}{p}\sum_{\mu=1}^{p}\left\langle \vec{s}\,\mathrm{sgn}(\hat{J}\cdot\vec{s})\right\rangle_{\Omega_\mu} \, . \tag{5.18}$$

To simplify analysis we replace the *hard* constraint (restricting the training vectors to the union of the p discrete subsets Ω_μ) by a *soft* constraint, in which training vectors have a probability of occurrence which is strongly peaked near the patterns $\vec{\xi}^\mu$:

$$\left\{\Omega = \bigcup_\mu \Omega_\mu,\;\; p(\vec{s}) = \ldots\right\} \;\;\rightarrow\;\; \left\{\Omega = \{-1,1\}^N,\;\; p(\vec{s}) = \frac{1}{p}\sum_\mu \tilde{p}_\mu(\vec{s})\right\}$$

with

$$\tilde{p}_\mu(\vec{s}) \equiv \prod_i \left[\frac{1}{2}(1+a)\delta_{s_i,\xi_i^\mu} + \frac{1}{2}(1-a)\delta_{s_i,-\xi_i^\mu}\right]$$

in which a is chosen close to one. Formally the task corresponding to the soft constraint is not necessarily linearly separable and hence solutions of the correponding fixed point equations need not exist. However, for $a \rightarrow 1$ and $N \rightarrow \infty$, the overlap between the individual distributions $\tilde{p}_\mu$ becomes arbitrarily small, so that one can expect solutions to exist. These solutions then correspond to the connections of a perceptron which is trained with noise, as studied by Wong and Sherrington [84]. Replacing the hard constraint by the soft one in the way described above, we obtain instead of (5.18) the problem (5.19):

$$\frac{a}{p}\sum_{\mu=1}^{p} T(\vec{\xi}^\mu)\vec{\xi}^\mu = \frac{1}{p}\sum_{\mu=1}^{p}\left\langle \vec{s}\,\mathrm{sgn}(\hat{J}\cdot\vec{s})\right\rangle_{\tilde{p}_\mu} . \tag{5.19}$$

To simplify notation we introduce the vectors $\vec{\zeta}^\mu \equiv T(\vec{\xi}^\mu)\vec{\xi}^\mu$, in terms of which (due to the absence of a threshold) the task T can be written as: $T(\vec{\zeta}^\mu) = 1$ $(\forall\mu)$. We can readily perform the remaining averages in (5.19), with the result:

$$\frac{a}{p}\sum_{\mu=1}^{p}\zeta_i^\mu = \frac{1}{p}\sum_{\mu=1}^{p}\left[a\zeta_i^\mu\,\mathrm{erf}\left(\frac{a\hat{J}\cdot\vec{\zeta}^\mu}{\sqrt{2(1-a^2)}}\right)\right.$$
$$\left. + \;\hat{J}_i\sqrt{\frac{2(1-a^2)}{\pi}}\exp\left(-\frac{(a\hat{J}\cdot\vec{\zeta}^\mu)^2}{2(1-a^2)}\right)\right] + \mathcal{O}(\hat{J}_i^2) .$$

In leading order we may therefore write:

$$\hat{J} = \sqrt{\frac{\pi a^2}{2(1-a^2)}}\left\{\sum_\mu \vec{\zeta}^\mu\left[1 - \mathrm{erf}\left(\frac{a\hat{J}\cdot\vec{\zeta}^\mu}{\sqrt{2(1-a^2)}}\right)\right]\right\}$$
$$\bigg/\left\{\sum_\rho \exp\left(-\frac{(a\hat{J}\cdot\vec{\zeta}^\rho)^2}{2(1-a^2)}\right)\right\} . \tag{5.20}$$

For convenience we introduce the parameter $\Lambda \equiv \frac{1}{2}a^2/(1-a^2)$ (so $\Lambda \to \infty$ as $a \to 1$) and the stability parameters $\gamma_\mu \equiv \hat{J}\cdot\vec{\zeta}^\mu$. Assuming that a solution $\hat{J}$ of (5.20) exists with $\gamma_\mu > 0$ for all μ, we can use the asymptotic expansion of $\mathrm{erf}(x)$,

$$\mathrm{erf}(x) = 1 - \frac{1}{\sqrt{\pi}x}\exp(-x^2) + \ldots$$

and obtain

$$\hat{J} = \frac{\sum_\mu \vec{\zeta}^\mu\gamma_\mu^{-1}\exp(-\Lambda\gamma_\mu^2)}{\sum_\rho \exp\left(-\Lambda\gamma_\rho^2\right)} . \tag{5.21}$$

Note that (5.21) guarantees that any solution $\hat{J}$ will indeed be properly normalized (providing a nice self-consistency test, since normalization has not explicitly been put in):

$$\hat{J}^2 = \frac{\sum_\mu \hat{J}\cdot\vec{\zeta}^\mu\gamma_\mu^{-1}\exp(-\Lambda\gamma_\mu^2)}{\sum_\rho \exp\left(-\Lambda\gamma_\rho^2\right)} = 1 .$$

By taking in (5.21) the inner products with the vectors $\vec{\zeta}^\mu$, we obtain equations in terms of stability parameters only:

$$\gamma_\mu = \frac{1}{\alpha}\frac{\sum_\nu C_{\mu\nu}\gamma_\nu^{-1}\exp(-\Lambda\gamma_\nu^2)}{\frac{1}{p}\sum_\rho \exp\left(-\Lambda\gamma_\rho^2\right)} , \qquad \gamma_\mu > 0 \tag{5.22}$$

in which the correlation matrices $C_{\mu\nu}$ are defined by

$$C_{\mu\nu} \equiv \frac{1}{N}\vec{\zeta}^\mu\cdot\vec{\zeta}^\nu .$$

Eqn. (5.22) is the main result of this section. The solution of (5.22), inserted into (5.21), yields the solution of our original problem: the connections $\hat{J}$.

Restoring the original variables according to $\vec{\zeta}^\mu \equiv \vec{\xi}^\mu T(\vec{\xi}^\mu)$, we find that the connections (5.21) are written in the form of a weighted Hebb rule with embedding strengths $\{w_\mu\}$:

$$\hat{J} = \frac{1}{N}\sum_\mu w_\mu T(\vec{\xi}^\mu)\vec{\xi}^\mu \quad \text{with} \quad w_\mu = \frac{1}{\alpha}\frac{\gamma_\nu^{-1}\exp(-\Lambda\gamma_\nu^2)}{\frac{1}{p}\sum_\rho \exp\left(-\Lambda\gamma_\rho^2\right)}. \tag{5.23}$$

These equations give interesting relations between stability parameters and embedding strengths. The relations (5.23), in combination with

$$\gamma_\mu = \sum_\nu C_{\mu\nu} w_\mu \tag{5.24}$$

are equivalent to the equations (5.21,5.22). A trivial example for which (5.23,5.24) is solvable, is the case of orthogonal patterns $C_{\mu\nu} = \delta_{\mu\nu}$, for which one finds $\gamma_\mu = \frac{1}{\sqrt{\alpha}}$ and the connections are given by a normalized Hebb-rule.

5.4.2 The Limit of Zero Training Noise

In this subsection we show that the solution(s) of the self consistency equations (5.22), in the limit of zero training noise ($\Lambda \to \infty$), are identical with the optimal connections in the Gardner sense. It has been demonstrated previously [4, 56] that in characterizing the optimal connections one can distinguish two subsets of patterns. Patterns in the so-called active set have positive embedding strengths w_μ; the optimal connections are given by the pseudo-inverse rule, restricted to the patterns in the active set. Patterns which are *not* in the active set have zero embedding strengths; their stability parameters, however, are larger than the stability parameters of the patterns in the active set.

We assume that, for large Λ, the stability parameters depend analytically on Λ^{-1}:

$$\gamma_\mu = \sum_{n\geq 0} \gamma_{\mu n}\Lambda^{-n}\,.$$

Insertion into (5.22) gives the identity:

$$\gamma_{\mu 0} = \lim_{\Lambda\to\infty} \frac{1}{\alpha}\frac{\sum_\nu C_{\mu\nu}\left[\gamma_{\nu 0} + \mathcal{O}(\Lambda^{-1})\right]^{-1}\exp\left[-\Lambda(\gamma_{\nu 0}^2 + 2\gamma_{\nu 0}\gamma_{\nu 1}\Lambda^{-1} + \mathcal{O}(\Lambda^{-2}))\right]}{\frac{1}{p}\sum_\rho \exp\left[-\Lambda(\gamma_{\rho 0}^2 + 2\gamma_{\rho 0}\gamma_{\rho 1}\Lambda^{-1} + \mathcal{O}(\Lambda^{-2}))\right]}\,.$$

We now introduce $\gamma_{\min} \equiv \min_\mu \gamma_{\mu 0}$, with which we can write

$$\gamma_{\mu 0} = \lim_{\Lambda\to\infty} \frac{1}{\alpha}\frac{\sum_\nu C_{\mu\nu}\left[\gamma_{\nu 0} + \mathcal{O}(\Lambda^{-1})\right]^{-1}\exp\left[-\Lambda(\gamma_{\nu 0}^2 - \gamma_{\min}^2) - 2\gamma_{\nu 0}\gamma_{\nu 1} + \mathcal{O}(\Lambda^{-1})\right]}{\frac{1}{p}\sum_\rho \exp\left[-\Lambda(\gamma_{\rho 0}^2 - \gamma_{\min}^2) - 2\gamma_{\rho 0}\gamma_{\rho 1} + \mathcal{O}(\Lambda^{-1})\right]}\,. \tag{5.25}$$

By taking the limit $\Lambda \to \infty$, those exponents vanish for which $\gamma_{\mu 0} > \gamma_{\min}$ (by construction there is at least one index μ with $\gamma_{\mu 0} = \gamma_{\min}$). We define the index set

$$\mathcal{K} = \{\mu|\ \gamma_{\mu 0} = \gamma_{\min}\}\,.$$

For $\Lambda \to \infty$ we obtain from (5.25):

$$\gamma_{\mu 0} = \frac{1}{\alpha} \frac{\sum_{\nu \in \mathcal{K}} C_{\mu\nu} \gamma_{\min}^{-1} \exp\left[-2\gamma_{\min}\gamma_{\nu 1}\right]}{\frac{1}{p}\sum_{\rho \in \mathcal{K}} \exp\left[-2\gamma_{\min}\gamma_{\rho 1}\right]} \, .$$

This means that the $\Lambda \to \infty$ embedding strengths w_μ, defined in (5.23), will obey

$$\begin{aligned} w_\mu &= \frac{1}{\alpha} \frac{\gamma_{\min}^{-1} \exp[-2\gamma_{\min}\gamma_{\mu 1}]}{\frac{1}{p}\sum_{\rho \in \mathcal{K}} \exp[-2\gamma_{\min}\gamma_{\rho}]} \qquad && \forall \mu \in \mathcal{K} \\ w_\mu &= 0 && \forall \mu \notin \mathcal{K} \end{aligned}$$

Apparently the embedding strengths corresponding to indices in the index set $\mathcal{K}$ satisfy:

$$\forall \mu \in \mathcal{K} \qquad \gamma_{\min} = \sum_{\nu \in \mathcal{K}} C_{\mu\nu} w_\nu$$

hence, if we denote by $C(\mathcal{K})$ the correlation matrix C, restricted to the indices in the set $\mathcal{K}$, the embedding strengths w_μ are given by

$$\begin{aligned} w_\mu &= \gamma_{\min} \sum_{\nu \in \mathcal{K}} C(\mathcal{K})^{-1}_{\mu\nu} \qquad && \mu \in \mathcal{K} \\ w_\mu &= 0 && \mu \notin \mathcal{K} \, . \end{aligned}$$

These are exactly the embedding strengths corresponding to the optimal perceptron, described in the introduction of this subsection. One immediately recognises the structure of a pseudo-inverse applied to a active pattern set (our index set $\mathcal{K}$). A pattern outside the active set has zero embedding strength. On the other hand, its stability parameter is larger than the minimal stability parameter ($\gamma_{\mu 0} > \gamma_{\min}$ for $\mu \notin \mathcal{K}$).

This a posteriori justifies our assumption that, for large Λ (small amount of training noise), the hard constraint on the training set could be replaced by a soft one. Our solution is also in agreement with the work by Wong and Sherrington [84], who studied the learning of noisy patterns and found that for infinitesimally small amounts of noise one obtains the maximally stable connections.

5.4.3 Numerical Results

Apart from proving that in the limit of zero training noise $\Lambda \to \infty$ the solutions of the set of equations (5.22) become identical to the optimal interactions in the Gardner sense, one can of course also simply solve the set (5.22) numerically. The result, presented in the form of the familiar distribution $\rho(\gamma)$ of stabilities, shows how for finite Λ one approaches the analytical expression for $\rho(\gamma)$ as found by Abbott and Kepler [39]. Figure 5.2 shows such a result, obtained by solving (5.22) numerically for $p = 600$ randomly drawn patterns in an $N = 400$ network with a level of training noise given by $\Lambda = 16$. The distribution $\rho(\gamma)$ [39] of Gardner's optimal interactions [19] for $\alpha = 1.5$ is plotted as a reference.

5.5 Discussion

The aim of this paper was to find analytical expressions for the connections of large perceptrons. We tried to calculate the connection vectors $\vec{J}$ that perform a given task T on a given input set $\Omega \subseteq \{-1, 1\}^N$, by using the fact that such connections are fixed points of the perceptron learning rule. For small values of the learning parameter this rule can be split into a macroscopic

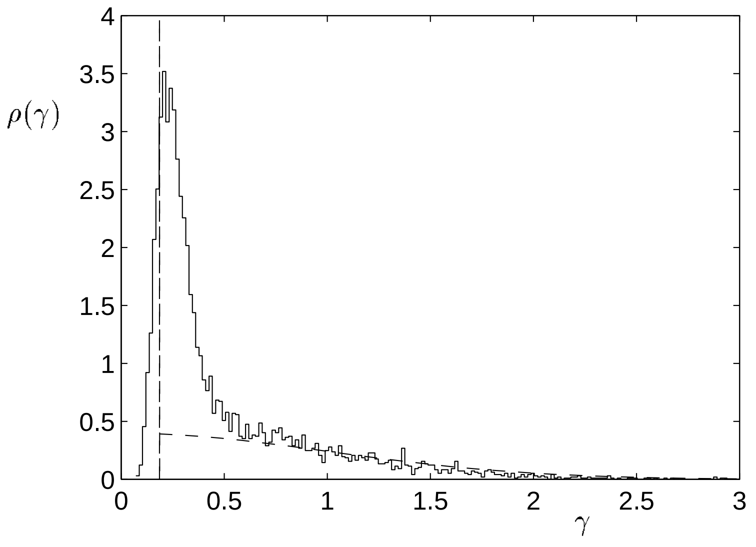

Figure 5.2 Distribution $\rho(\gamma)$ of stabilities $\{\gamma_\mu\}$ obtained by solving numerically the set of equations (5.22) for randomly drawn patterns ($N = 400$, $p = 600$, $\Lambda = 16$). The result is averaged over 10 pattern realizations. Dashed line: distribution of stability parameters for Gardner's optimal connections (according to [39]) if $\alpha = 1.5$.

differential equation describing deterministic evolution and a part describing fluctuations. By proving that the fixed-points of the deterministic equation are identical to the fixed-points of the full stochastic rule, we obtain a reduction of the original problem (finding the solution of a set of $|\Omega|$ coupled inequalities) to the problem of finding the solution of a set of N coupled non-linear equations.

For the simplest case in which the training set consists of all possible input vectors, $\Omega = \{-1, 1\}^N$, the fixed point equations enable us to calculate the connections as a series expansion in powers of $1/\sqrt{N}$. The leading term in this expansion turns out to be either the Hebb rule (for unbiased distributions) or the biased Hebb rule (for biased distributions). The performance of our asymptotic expressions (and finite size corrections) on small systems is studied numerically. If, on the other hand, the training set consists of an extensive number $p = \alpha N$ of prototype patterns $\vec{\xi}^\mu$ in combination with small regions Ω_μ around these patterns (i.e., training with noise), we find that the connections satisfy a self-consistent, physically transparent set of non-linear equations. In the limit of zero training noise the solution of these equations is shown to correspond exactly to the interactions with maximal stability in the Gardner sense.

Most statistical mechanical studies of (maximally stable) perceptrons concentrate on studying *properties* of trained systems [78] (storage capacity, average training error, average generalization error, nature of phase transitions, etc.). In order to obtain these results one has to average the quantities of interest (or, equivalently, the free energy from which such quantities can be obtained by differentiation) over the distribution from which the training set is choosen. We believe that our approach may be complementary to such studies, in that we focus on the *explicit construction* of the connections of trained perceptrons. Furthermore, in the case of having an extensive ($p = \alpha N$) training set, the embedding strengths are formulated, through Eqns. (5.22,5.23), directly in terms of the pattern correlation matrix; no averaging over the distribution of input vectors is involved.

Acknowledgements

WW would like to thank the British Council for financial support and the University of Oxford for hospitality.

5 Appendix

5.A The Distribution P(z)

In this appendix we calculate for any given vector $\vec{K} \in \mathbb{R}^N$ the probability distribution $P(z)$ (in the spirit of the Edgeworth series [16]), defined by

$$P(z) \equiv \left\langle \delta\left(z - \vec{K}\cdot\vec{s}\right)\right\rangle_{\vec{s}}$$

where $\vec{s} \in \{-1,1\}^N$ and $p(\vec{s}) \equiv 2^{-N}$. Using the integral representation of the δ-function we find:

$$P(z) = \frac{1}{2\pi K}\int dk \exp\left[\frac{ikz}{K} + \sum_j \log\cos(k\hat{K}_j)\right] \tag{5.A.1}$$

where $K \equiv \|\vec{K}\|$ and $\hat{K} \equiv K^{-1}\vec{K}$. We now expand $\log\cos(x)$ in a power series [22]:

$$\log\cos(x) = -\frac{1}{2}x^2 - \sum_{n\geq 2} C_n x^{2n} \quad \text{with} \quad C_n \equiv \frac{2^{2n-1}(2^{2n}-1)|B_{2n}|}{n(2n)!}\,.$$

The coefficients B_k are the Bernoulli numbers [22] ($B_0 = 1$, $B_1 = -1/2$, $B_2 = 1/6$, etc.). This expansion enables us to write (5.A.1) as:

$$P(z) = \frac{1}{2\pi K}\int dk \exp\left[-\frac{1}{2}k^2 + \frac{ikz}{K} - \sum_{n\geq 2} C_n Q_n(\hat{K})k^{2n}\right] \tag{5.A.2}$$

where $Q_n(\hat{K}) \equiv \sum_j \hat{K}_j^{2n} \in [0,1]$. If we also make the expansion

$$\exp\left[-\sum_{n\geq 2} C_n Q_n(\hat{K})k^{2n}\right] \equiv 1 - \sum_{n\geq 2} D_n(\hat{K})k^{2n}$$

we can perform the integration over the variable k in (5.A.2) and arrive at the final result:

$$P(z) = \frac{1}{\sqrt{2\pi K^2}}\exp\left[-\frac{z^2}{2K^2}\right]\left[1 - \sum_{n\geq 2} D_n(\hat{K})(-1)^n 2^{-n} H_{2n}\left(\frac{z}{K\sqrt{2}}\right)\right] \tag{5.A.3}$$

where the functions $H_m(x)$ are the Hermite polynomials [15]:

$$H_m(x) \equiv (-1)^m \exp(x^2)\frac{d^m}{dx^m}\exp(-x^2)\,.$$

The coefficients $D_n(\hat{K})$ are given by

$$\begin{aligned} D_n(\hat{K}) \;\equiv\; & \sum_{k\leq n/2-1} \frac{(-1)^k}{(k+1)!}\sum_{m_1=2}^{n}\cdots\sum_{m_{k+1}=2}^{n}\Big[\delta_{n,\sum m_i} \\ & \times\, C_{m_1}Q_{m_1}(\hat{K})\ldots C_{m_{k+1}}Q_{m_{k+1}}(\hat{K})\Big]\,. \end{aligned}$$

5.B Validity of the Gaussian Assumption

In this appendix we briefly discuss the validity of the assumption (often made in the literature) that the stochastic variable $z = \sum_{j=1}^{N} J_j s_j$ has a Gaussian probability distribution in the limit $N \to \infty$, where

$$p(\vec{s}) \equiv \prod_j p_j(s_j), p_j(s) \equiv \frac{1}{2}(1+a_j)\delta_{s,1} + \frac{1}{2}(1-a_j)\delta_{s,-1},$$

with

$$|a_j| \leq |a_{\max}| < 1 \ .$$

It is clear that $P(z)$ will not always be Gaussian, since one can easily construct counter-examples:

$$J_k \equiv k^{-1} \qquad a_k \equiv 0 \qquad (k = 1 \dots N) \ . \tag{5.B.4}$$

For this specific example one finds:

$$\langle z \rangle = \left\langle z^3 \right\rangle = 0 \qquad \lim_{N\to\infty} \left\langle z^2 \right\rangle = \frac{\pi^2}{6} \qquad \lim_{N\to\infty} \left\langle z^4 \right\rangle = \frac{11\pi^4}{180} \ .$$

So the distribution of z tends not to a Gaussian, since even in the limit $N \to \infty$ one finds $\left\langle z^4 \right\rangle \neq 3 \left\langle z^2 \right\rangle^2$. Starting from the central limit theorem [16], it is straightforward to show that the condition on the normalized vector $\hat{J}$ for arriving at a Gaussian distribution for $z = \sum_j J_j s_j$, is:

$$\lim_{N\to\infty} \sum_{j=1}^{N} \theta \left[J_j - \epsilon \sum_{k=1}^{N} J_k^2 \right] = 0 \qquad \forall \epsilon > 0 \tag{5.B.5}$$

in which $\theta[x]$ is the step function. One can check that if (5.B.5) holds, all the non-Gaussian contributions in the probability distribution (5.A.3) will vanish in the limit $N \to \infty$. The condition (5.B.5) is clearly violated by the counter-example (5.B.4). If the vector $\vec{J}$ is drawn from a spherically symmetric distribution, or from a hypercube with uniform distribution, then one can show that condition (5.B.5) is satisfied with probability one.

Bibliography

[1] I. Aleksander and J. Taylor, editors. *ICANN'92: Proceedings of the International Conference on Artificial Neural Networks,Brighton,UK*, Amsterdam, 1993. North-Holland.

[2] S. Amari. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, 16:299–307, 1967.

[3] D. Amit, H. Gutfreund, and H. Sompolinsky. Information storage in neural networks with low levels of activity. *Physical Review A*, 23:2293–2303, 1987.

[4] J. Anlauf and M. Biehl. AdaTron: and adaptive perceptron algorithm. *Europhysics Letters*, 10:687–692, 1989.

[5] E. Barakova and R. Venema. personal communication, 1994.

[6] E. Barnard. Optimization for training neural nets. *IEEE Transactions on Neural Networks*, 3:232–240, 1992.

[7] D. Bedeaux, K. Lakatos-Lindenberg, and K. Shuler. On the relation between master equations and random walks and their solutions. *Journal of Mathematical Physics*, 12:2116–2123, 1971.

[8] A. Benviste, M. Metivier, and P. Priouret. *Adaptive algorithms and stochastic approximations.* Springer-Verlag, Berlin, 1987.

[9] S. Brunak, J. Engelbrecht, and S. Knudsen. Cleaning up gene databases. *Nature*, 343:123, 1990.

[10] C. Cachin. Pedagogical pattern selection strategies. *Neural Networks*, 7:175–181, 1994.

[11] P. Churchland and J. Sejnowski. *The Computational Brain.* The MIT press, Cambridge, Massachusetts, 1992.

[12] S. Diederich and M. Opper. Learning of correlated patterns in spin-glass networks by local learning rules. *Physical Review Letters*, 58:949–952, 1987.

[13] Neural computing applications portfolio. DTI's Neural Computing Technology Transfer Programme, 1993.

[14] Best practice guidelines for developing neural computing applications. DTI's Neural Computing Technology Transfer Programme, 1994.

[15] W. Feller. *An Introduction to Probability and its Applications*, volume I. Wiley, New York, 1966.

[16] W. Feller. *An Introduction to Probability and its Applications.* Wiley, New York, 1966.

[17] W. Finnoff. Diffusion approximations for the constant learning rate backpropagation algorithm and resistance to local minima. *Neural Computation*, 6:285–295, 1994.

[18] C. Gardiner. *Handbook of Stochastic Methods.* Springer, Berlin, second edition, 1985.

[19] E. Gardner. The space of interactions in neural network models. *Journal of Physics A*, 21:257–270, 1988.

[20] S. Gasiorowicz. *Quantum Physics.* Wiley, New York, 1974.

[21] E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Reading, 1989.

[22] I. Gradsteyn and I. Ryzhik. *Table of Integrals, Series and Products.* Academic Press, San Diego, 1980.

[23] H. Haken. *Synergetics, An Introduction.* Springer, New York, 1978.

[24] L. Hansen, R. Pathria, and P. Salamon. Stochastic dynamics of supervised learning. *Journal of Physics A*, 26:63–71, 1993.

[25] S. Haykin. *Neural Networks, A Comprehensive Foundation.* MacMillan, Hamilton, Ontario, 1994.

[26] D. Hebb. *The organization of behaviour.* Wiley, New York, 1949.

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

[28] T. Heskes. On Fokker-Planck approximations of on-line learning processes. *Journal of Physics A*, 27:5145–5160, 1994.

[29] T. Heskes and B. Kappen. Learning processes in neural networks. *Physical Review A*, 44:2718–2726, 1991.

[30] T. Heskes and B. Kappen. Learning-parameter adjustment in neural networks. *Physical Review A*, 45:8885–8893, 1992.

[31] T. Heskes and B. Kappen. On-line learning processes in artificial neural networks. In J. Taylor, editor, *Mathematical Foundations of Neural Networks*, pages 199–233. Elsevier, Amsterdam, 1993.

[32] T. Heskes and W. Wiegerinck. A theoretical comparison of batch-mode, on-line, cyclic, and almost cyclic learning. *IEEE Transactions on Neural Networks*, 1995. Accepted.

[33] T. Hondou and Y. Sawada. Analysis of learning processes of chaotic time series by neural networks. *Progress of Theoretical Physics*, 91:397–402, 1994.

[34] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79:2554–2558, 1982.

[35] R. Hoptroff. The principles and practice of time series forecasting and business modelling using neural nets. *Neural Computing and Applications*, 1:59–66, 1993.

[36] D. Hush, B. Horne, and J. Salas. Error surfaces for multilayer perceptrons. *IEEE Transactions on Systems, Man, and Cybernetics*, 22:1152–1161, 1992.

[37] B Kappen and S. Gielen, editors. *Neural Networks: Artificial Intelligence and Industrial Applications. Proceedings of the Third Annual SNN Symposium on Neural Networks, Nijmegen, The Netherlands, 14-15 September 1995*, London, 1995. Springer.

[38] H. Kappen and C. Gielen, editors. *ICANN'93: Proceedings of the International Conference on Artificial Neural Networks, Amsterdam*, London, 1993. Springer-Verlag.

[39] T. Kepler and L. Abbot. Domains of attraction in neural networks. *Journal de Physique*, 49:1657–1662, 1988.

[40] W. Kinzel. In L. Garrido, editor, *Statistical Mechanics of Neural Networks*, Berlin, 1990. Springer.

[41] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[42] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[43] W. Krauth and Mézard M. Learning algorithms with optimal stability in neural networks. *Journal of Physics A*, 20:L745–L752, 1987.

[44] C. Kuan and H. White. Artificial neural networks: an econometric perspective. *Econometric Reviews*, 1993. in press.

[45] S. Kuffler and J. Nicholls. *From neuron to brain.* Sinauer Associates, Sunderland, Massachusetts, 1977.

[46] A. Lapedes and R. Farber. How neural networks work. In D. Anderson, editor, *Neural Information Processing Systems*, pages 442–456, New York, 1988. American Institute of Physics.

[47] Y. Lee, S. Oh, and M. Kim. The effect of initial weights on premature saturation in backpropagation learning. In *International Joint Conference on Neural Networks*, pages 765–770. IEEE, 1991.

[48] T. Leen and J. Moody. Weight space probability densities in stochastic learning: I. Dynamics and equilibria. In S. Hanson, J. Cowan, and L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 451–458, San Mateo, 1992. Morgan Kaufmann.

[49] J. Ludik and I. Cloete. Incremental increased complexity training. In M. Verleysen, editor, *Proceedings of the European Symposium on Artificial Neural Networks '94*, pages 161–165, Brussels, 1994. D facto publications.

[50] M. Marinaro and G. Morasso, editors. *ICANN'94: Proceedings of the International Conference on Artificial Neural Networks,Sorrento,Italy*, London, 1994. Springer-Verlag.

[51] M. Minsky and S. Papert. *Perceptrons.* The MIT press, Cambridge, Massachusetts, 1969.

[52] G. Mpitsos and M. Burton. Convergence and divergence in neural networks: processing of chaos and biological analogy. *Neural Networks*, 5:605–625, 1992.

[53] B. Mueller and J. Reinhardt. *Neural Networks.* Springer, Berlin, 1990.

[54] P. Munro. Repeat until bored: A pattern selection strategy. In J. Moody, S. Hanson, and R. Lippman, editors, *Advances in Neural Information Processing Systems 4*, pages 1001–1008, San Mateo, 1992. Morgan Kaufmann.

[55] E. Oja. A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982.

[56] M. Opper. Learning times of neural networks: exact solution for a perceptron algorithm. *Physical Review A*, 38:3824–3826, 1988.

[57] M. Opper. Learning in neural networks: solvable dynamics. *Europhysics Letters*, 8:389–392, 1989.

[58] G. Orr and T. Leen. Weight space probability densities in stochastic learning: II. Transients and basin hopping times. In S. Hanson, J. Cowan, and L. Giles, editors, *Advances in Neural Information Processing Systems 5*, pages 507–514, San Mateo, 1992. Morgan Kaufmann.

[59] G. Orr and T. Leen. Momentum and optimal stochastic search. In M. Mozer, P. Somlensky, D. Touretzky, J. Elman, and A. Weigend, editors, *Proceedings of the 1993 Connectionist Models Summer School.* Erlbaum, 1993.

[60] B. Pearlmutter. Gradient descent: second-order momentum and saturating error. In J. Moody, S. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 887–894, San Mateo, 1991. Morgan Kaufmann.

[61] W. Penney, A. Coolen, and D. Sherrington. Coupled dynamics of fast spins and slow interactions in neural networks and spin systems. *Journal of Physics A*, 26:3681–3695, 1993.

[62] W. Press, B. Flannery, A. Teukolsky, and W. Vettering. *Numerical Recipes in C.* Cambridge University Press, 1989.

[63] G. Radons. On stochastic dynamics of supervised learning. *Journal of Physics A*, 26:3455–3461, 1993.

[64] H. Ritter and K. Schulten. Convergence properties of Kohonen's topology conserving maps: fluctuations, stability, and dimension selection. *Biological Cybernetics*, 60:59–71, 1988.

[65] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[66] F. Rosenblatt. *Principles of Neurodynamics.* Spartan, New York, 1960.

[67] D. Rumelhart, G. Hinton, and R. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[68] D. Rumelhart, J. McClelland, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* MIT Press, 1986.

[69] H. Schuster. *Deterministic Chaos.* VCH, Weinheim, second revised edition, 1989.

[70] H. Seung, H. Sompolinsky, and N. Tishby. Statistical mechanics of learning from examples. *Physical Review A*, 45:6056–6091, 1992.

[71] J. Shynk and S. Roy. The LMS algorithm with momentum updating. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 2651–2654, 1988.

[72] M. Tugay and Y. Tanik. Properties of the momentum LMS algorithm. *Signal Processing*, 18:117–127, 1989.

[73] F. Vallet. The Hebb rule for learning linearly separable Boolean functions: learning and generalization. *Europhysics Letters*, 8:747–751, 1989.

[74] F. Vallet and J.-G. Cailton. Recognition rates of the Hebb rule for learning Boolean functions. *Physical Review A*, 41:3059–3065, 1990.

[75] P. van der Smagt. Minimisation methods for training feed-forward networks. *Neural Networks*, 7(1):1–11, 1994.

[76] N. van Kampen. Elimination of fast variables. *Physics Reports*, 124:69–160, 1985.

[77] N. van Kampen. *Stochastic Processes in Physics and Chemistry.* North-Holland, Amsterdam, 1992.

[78] T. Watkin, A. Rau, and M. Biehl. The statistical mechanics of learning a rule. *Reviews of Modern Physics*, 65:499–556, 1993.

[79] A. Weigend and N. Gershenfeld, editors. *Predicting the Future and Understanding the Past: A Comparison of Approaches.* Addison-Wesley, 1993.

[80] A. Weigend, B. Huberman, and D. Rumelhart. Predicting the future: a connectionist approach. *International Journal of Neural Systems*, 1:193–209, 1990.

[81] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974.

[82] H. White. Some asymptotic results for learning in single hidden-layer feedforward network models. *Jour. Amer. Stat. Ass.*, 84:1003–1013, 1989.

[83] F. Wong. Time series forecasting using backpropagation networks. *Neurocomputing*, pages 147–159, 1991.

[84] K. Wong and D. Sherrington. Training noise adaptation in attractor neural networks. *Journal of Physics A*, 23:175–182, 1990.

Samenvatting

Stochastische Dynamica van On-Line Leren in Neurale Netwerken.

Neurale netwerken zijn informatieverwerkende systemen die voor een belangrijk deel geïnspireerd zijn op huidige inzichten in de werking van het brein. Een neuraal netwerk (de naam zegt het al) bestaat uit een netwerk van onderling verbonden neuronen. De neuronen zijn eenvoudige rekenelementjes. Door de onderlinge verbindingen kunnen de neuronen gegevens met elkaar uitwisselen. Tenslotte zorgen de input en output neuronen ervoor dat het neuraal netwerk met zijn omgeving kan communiceren. De sterktes van de verbindingen oftewel de gewichten reguleren de gegevensuitwisseling in het neuraal netwerk. Zodoende bepalen uiteindelijk de gewichten hoe het neuraal netwerk functioneert.

Neurale netwerken staan erom bekend dat zij kunnen leren. Hierbij is leren het proces waarbij een neuraal netwerk zijn gewichten volgens een bepaalde *leerregel* stap voor stap aanpast aan de hand van voorbeelden uit zijn omgeving. Dit kunnen bijvoorbeeld gewenste input-output relaties zijn. Leren heeft als groot voordeel dat de gewichten van een neuraal netwerk niet precies door een programmeur ingesteld hoeven te worden.

Een zeer natuurlijke vorm van leren is het zogenaamde *on-line* leren. In on-line leren worden er in de omgeving voortdurend – en in het algemeen met een zekere willekeur – voorbeelden gegenereerd en één voor één aan het neuraal netwerk aangeboden. Na ieder voorbeeld maakt het neuraal netwerk onmiddellijk een leerstap. (In het zogenaamde *batch-mode* leren moet er eerst een set voorbeelden verzameld worden. Vervolgens kan er worden geleerd. Hierbij wordt iedere leerstap gemaakt op basis van de totale set voorbeelden.) Vaak bestaat bij on-line leren het leerproces uit vele kleine leerstapjes. Dit is om ervoor te zorgen dat het neuraal netwerk een goede representatie van de omgeving krijgt en zich niet bij elke leerstap teveel instelt op het voorbeeld dat toevallig op dat moment wordt aangeboden.

On-line leren in neurale netwerken kan op een wiskundige manier worden gemodelleerd en bestudeerd. Het toeval in de gepresenteerde voorbeelden maakt het leren tot een stochastisch (= toevals)proces. Om dit proces te analyseren zijn er een groot aantal technieken uit de kansrekening en de theorie van stochastische processen beschikbaar.

Er is al veel onderzoek gedaan aan de theorie van on-line leren. In één van de onderzoeksrichtingen wordt er uitgegaan van een algemeen, niet nader beschreven neuraal netwerk met een algemene, niet nader beschreven leerregel die de leerstap definieert. Deze algemene aanpak heeft als voordeel dat de resultaten geldig zijn voor een groot aantal soorten neurale netwerken waaronder het *meerlaags perceptron* met de bekende *backpropagation* leerregel in on-line mode en de topologisch geordende kaart met de zelforganiserende leerregel van Kohonen. De belangrijkste aanname in deze theorie is dat de leerstappen schalen met een zogenaamde *leerparameter* die klein verondersteld mag worden. Uit deze theorie volgt dat het leerproces goed beschreven wordt door een trend – het gemiddelde van veel kleine leerstappen – en fluctuaties hierop. Deze fluctuaties worden veroorzaakt door de afwijkingen van individuele leerstappen ten opzichte van de trend.

Binnen dit algemene theoretische kader wordt er in het grootste gedeelte van dit proefschrift gekeken naar leerprocessen waarbij opeenvolgende leerstappen gecorreleerd zijn, m.a.w. meer (of wellicht juist minder) op elkaar lijken dan op grond van puur toeval verwacht zou mogen worden. De analyse van leren met gecorreleerde leerstappen heeft als extra complicatie dat als je de kans op het maken van een leerstap op een bepaald tijdstip t wilt berekenen, je het verleden vóór t niet buiten beschouwing kunt laten. Correlaties tussen de opeenvolgende leerstappen kunnen verschillende oorzaken hebben. Zij kunnen bijvoorbeeld simpelweg het directe gevolg zijn van correlaties tussen de aangeboden voorbeelden. Er zijn ook leeralgoritmes waarbij bewust correlaties tussen de leerstappen zijn aangebracht. De bekendste hiervan is het leren met een *momentum term*. Een momentum term zorgt ervoor dat opeenvolgende leerstappen op elkaar lijken met als bedoeling om de (ongewenste) fluctuaties die optreden in het leerproces te dempen. Dit proefschrift richt zich in het bijzonder op deze twee gevallen. Vragen die worden gesteld zijn bijvoorbeeld: Hebben correlaties tussen de voorbeelden invloed op de trend en de fluctuaties in het leerproces? In hoeverre helpt een momentum term om fluctuaties in het leerproces te dempen?

Het blijkt dat correlaties tussen voorbeelden geen invloed hebben op de trend in het leerproces. Hier is alleen de *gemiddelde* kans om een voorbeeld aan te treffen van belang. Op de fluctuaties hebben zij wel degelijk invloed. Dit is als volgt samen te vatten: hoe groter de kans dat twee opeenvolgende voorbeelden door de onderlinge correlaties op elkaar lijken, des te groter zijn de fluctuaties in het leerproces. Een ander resultaat in dit proefschrift is de quantitatieve verklaring waarom neurale netwerken sommige taken wel kunnen leren met gecorreleerde voorbeelden die ze met ongecorreleerde voorbeelden niet of veel moeilijker kunnen leren. Het blijkt dat er in zo'n situatie sprake is van een 'plateau in het foutenlandschap', waardoor het leerproces vrijwel stil komt te liggen. De correlaties tussen de voorbeelden zorgen voor een klein additioneel effect dat in reguliere situaties niet van belang is, maar in geval van een plateau net het leerproces weer aan de gang kan helpen. Tenslotte blijkt het toevoegen van een momentum term in on-line leren niet of nauwelijks zin te hebben. Het toevoegen van een momentum term komt in het algemeen effectief slechts neer op een herschaling van de leerparameter.

Tenslotte is er in één hoofdstuk wel naar een specifiek neuraal netwerk met een specifieke leerregel gekeken. Voor grote een-laags perceptrons met de bekende perceptron leerregel is onderzocht wat het resultaat van het leerproces is. Als de trainingset uit alle mogelijke inputvectoren bestaat blijkt dit resultaat in laagste orde door de regel van Hebb beschreven te worden. Als de trainingset bestaat uit clusters van op elkaar lijkende voorbeelden, waarbij het aantal clusters schaalt met de grootte van het perceptron, dan kan er een zelf-consistent stelsel vergelijkingen voor de gewichten worden afgeleid. In de limiet waarin de groottes van de clusters naar nul gaat, worden de oplossingen van dit stelsel gegeven door de gewichten met maximale stabiliteit.