Learning in two-layered networks with correlated examples

J. Phys. A: Math. Gen. **30** (1997) 4983–4992. Printed in the UK PII: S0305-4470(97)82920-4

Learning in two-layered networks with correlated examples

Tom Heskes and Jeroen Coolen

RWCP[†] Novel Functions, SNN[‡] Laboratory, Department of Medical Physics and Biophysics, University of Nijmegen, Geert Grooteplein 21, NL 6525 EZ Nijmegen, the Netherlands

Received 26 March 1997

Abstract. On-line learning in layered perceptrons is often hampered by plateaus in the time dependence of the performance. Studies on backpropagation in networks with a small number of input units have revealed that correlations between subsequently presented patterns shorten the length of such plateaus. We show how to extend the statistical mechanics framework to quantitatively check the effect of correlations on learning in networks with a large number of input units. The surprisingly compact description we obtain makes it possible to derive properties of on-learning with correlations directly from studies on on-line learning without correlations.

1. Introduction

Recently, considerable progress has been made in the study of on-line learning [1–4]. The usual assumption is that presented examples are uncorrelated in time. This assumption is not only unnatural for biological learning systems, but also for artificial neural networks such as, for example, those that are applied in time-series analysis. In this paper, we study the effect of correlations on learning in two-layered perceptrons, networks that are commonly used in practical applications.

Theoretical studies on on-line learning can be roughly divided into two groups: those on 'small' networks using stochastic approximation methods and those on 'large' networks using tools from statistical mechanics. In the stochastic approach, on-line learning is described as an average process (the drift) with superimposed fluctuations (the diffusion). General properties can be derived by making expansions for small learning parameters [4–6]. In large networks the drift for each individual weight becomes an order of magnitude smaller than its diffusion. A description of the learning behaviour in terms of individual weights, as in the stochastic approach, no longer makes sense and is replaced by a description in terms of macroscopic order parameters. This is the statistical approach, which considers on-line learning in the thermodynamical limit, i.e. in the limit of an infinitely large number of input units [1–3].

A problem shared by both 'small' and 'large' two-layered networks, is the existence of plateaus: long time spans during which the performance of the learning machine hardly changes [4, 7]. Most of these plateaus are caused by the initial tendency of hidden units to represent the same features. The breaking of this initial symmetry, which is often necessary to make further progress, appears to be a painstaking process. In a previous study on learning in 'small' networks using stochastic approximation tools, it has been shown that

† Real World Computing Program.
‡ Foundation for Neural Networks.

time-dependent correlations between subsequently presented training examples help to speed up this symmetry breaking process [4]. In this paper, we will use statistical mechanics tools to study how correlations affect the learning dynamics of 'large' two-layered networks.

In section 2 we start to describe the student and teacher networks and the dynamics of the successive examples. We define the order parameters needed to compute the generalization error. Eventually we arrive at a general description of the learning dynamics valid for 'large' two-layered networks, irrespective of the number of hidden units of teacher and student, graded or non-graded, tree-structured or fully connected, and with or without adaptive hidden-to-output weights. In sections 3–5, we treat several special cases, simple perceptron learning (no hidden units), learning a soft-committee machine (hidden-to-output weights are fixed) and learning a two-layered network with adaptive hidden-to-output weights, respectively. We close the paper in section 6 with a summary and discussion of the results.

2. Theory

We consider a student network and a teacher network. Both student and teacher are two-layered networks with N input units and one linear output unit, but may have a different number of hidden units. Let g be a nonlinear, differentiable transfer function of the hidden units and H be the total number of the hidden units of student and teacher together. This system is characterized by a set of input-to-hidden weights $\boldsymbol{J}_i = (J_{i1}, \dots, J_{iN})$ with $i = 1, \dots, H$ and hidden-to-output weights $\boldsymbol{w} = (w_1, \dots, w_H)$. Some of these weights belong to the student, others to the teacher†. The idea is that the teacher, a network with fixed weights, gives feedback about errors made by the student, a network with adaptive weights.

Both student and teacher receive the same sequence of inputs $\{\boldsymbol{\xi}(n)\}$ with all $\boldsymbol{\xi} = (\xi_1, \dots, \xi_N) \in \mathbb{R}^N$ obeying a first-order Markov process such that

$$\langle \xi_k(n) \rangle_{\xi} = 0 \qquad \langle \xi_k(n)\xi_l(n) \rangle_{\xi} = \delta_{kl} \qquad \text{and} \qquad \langle \xi_k(n)\xi_l(n+1) \rangle_{\xi} = c\delta_{kl}.$$

In other words, the components of the input vector are independently and identically distributed with zero mean and unit variance, yet for $c \neq 0$ each new component is correlated with the same component at a previous time step. Note that the stationary distribution of input vectors is independent of the correlation c (for $|c| < 1$).

Given an input vector $\boldsymbol{\xi}$, we define the 'local fields' $x_i = \boldsymbol{J}_i \cdot \boldsymbol{\xi}$, which, again, can belong to either the teacher or the student. For notational convenience we restrict ourselves to fully connected two-layered networks, but it is easy to extend the following to, for example, tree-based architectures. The outputs of student and teacher are written

$$\boldsymbol{\sigma}_{\text{student}}(\boldsymbol{x}) = \sum_{i \in \text{student}} w_i g(x_i) \qquad \text{and} \qquad \boldsymbol{\sigma}_{\text{teacher}}(\boldsymbol{x}) = \sum_{i \in \text{teacher}} w_i g(x_i)$$

with $\boldsymbol{x}$ being a vector containing all local fields. Unless otherwise stated, we choose the sigmoid $g(x) = \text{erf}(x/\sqrt{2})$ to make analytical calculations tractable.

After the presentation of the nth example $\boldsymbol{\xi}(n)$, the student weights are updated according to the gradient of the squared difference $\Delta \equiv \boldsymbol{\sigma}_{\text{teacher}} - \boldsymbol{\sigma}_{\text{student}}$ between teacher and student output. This yields for the input-to-hidden weights

$$\boldsymbol{J}_i(n+1) - \boldsymbol{J}_i(n) = -\frac{\eta_1}{N}\left(\frac{\partial}{\partial J_{i1}(n)}, \dots, \frac{\partial}{\partial J_{iN}(n)}\right)\frac{\Delta^2(\boldsymbol{x}(n))}{2} = \frac{\eta_1}{N}\delta_i(\boldsymbol{x}(n))\boldsymbol{\xi}(n) \tag{1}$$

† We use a notation which is slightly different from usual to stress the generality of our results.

with $\delta_i(\boldsymbol{x}(n)) \equiv w_i(n) g'(x_i(n)) \Delta(\boldsymbol{x}(n))$, and for the hidden-to-output weights

$$w_i(n+1) - w_i(n) = -\frac{\eta_2}{N} \frac{\partial}{\partial w_i(n)} \frac{\Delta^2(\boldsymbol{x}(n))}{2} = \frac{\eta_2}{N} g(x_i(n)) \Delta(\boldsymbol{x}(n)). \tag{2}$$

The learning parameters η_1 and η_2, which control the size of the steps, are explicitly scaled with the input size N. This description, if necessary with a slightly different choice for the functions $\delta_i(\boldsymbol{x})$, is also valid for simple perceptron learning without any hidden units.

In the thermodynamic limit $N \to \infty$, in which we will work from now on, most properties of the system depend on the order parameters $R_{ij} \equiv \boldsymbol{J}_i \cdot \boldsymbol{J}_j$ and the hidden-to-output weights w_i. For example, the generalization error $\varepsilon_{\mathrm{g}} = \langle \Delta^2/2 \rangle_{\boldsymbol{\xi}}$, follows from

$$\varepsilon_{\mathrm{g}} = \frac{1}{\pi} \sum_{i,j} \beta_{ij} w_i w_j \arcsin \frac{R_{ij}}{\sqrt{(1+R_{ii})(1+R_{jj})}} \tag{3}$$

where $\beta_{ij} = 1$ if both i, j belong to either the student or teacher and $\beta_{ij} = -1$ otherwise.

The dynamics of the weights w_i is given by (2), the dynamics of the order parameters R_{ij} can be written as usual [1, 2]:

$$R_{ij}(n+1) = R_{ij}(n) + \frac{\eta_1}{N} [x_i(n) \delta_j(\boldsymbol{x}(n)) + x_j(n) \delta_i(\boldsymbol{x}(n))] + \frac{\eta_1^2}{N} \delta_i(\boldsymbol{x}(n)) \delta_j(\boldsymbol{x}(n)) \tag{4}$$

where we have defined $\delta_i(\boldsymbol{x}) \equiv 0$ if i refers to a teacher weight.

Without any correlations, the machinery proceeds as follows. First one computes the distribution of the local fields which, because of the central limit theorem, comes out to be a Gaussian with covariance matrix $\mathcal{C} \equiv \langle \boldsymbol{x}\boldsymbol{x}^T \rangle_{\boldsymbol{x}}$ equal† to the order parameters $\mathcal{R}$. Next one turns the difference equations (4) and (2) into continuous-time differential equations, at the same step taking averages on the right-hand side. The resulting differential equations are of the form

$$\frac{\mathrm{d}R_{ij}(t)}{\mathrm{d}t} = F_{ij}(\mathcal{R}(t), \boldsymbol{w}(t)) \qquad \text{and} \qquad \frac{\mathrm{d}w_i(t)}{\mathrm{d}t} = f_i(\mathcal{R}(t), \boldsymbol{w}(t))$$

where $t = n/N$ is a rescaled 'time'. In many situations the averages

$$F_{ij} = \eta_1 \langle x_i \delta_j(\boldsymbol{x}) + x_j \delta_i(\boldsymbol{x}) \rangle_{\boldsymbol{x}} + \eta_1^2 \langle \delta_i(\boldsymbol{x}) \delta_j(\boldsymbol{x}) \rangle_{\boldsymbol{x}} \qquad \text{and} \qquad f_i = \eta_2 \langle g(x_i) \Delta(\boldsymbol{x}) \rangle_{\boldsymbol{x}} \tag{5}$$

can be calculated analytically [8].

With correlations, the distribution of the local fields does not only depend on the order parameters, but also has its 'own' dynamics. Using again the central limit theorem for large N, we derive for the dynamics of the local fields

$$x_i(n+1) = \boldsymbol{J}_i(n+1) \cdot \boldsymbol{\xi}(n+1) = c[x_i(n) + \eta_1 \delta_i(\boldsymbol{x}(n))] + u_i(n) \tag{6}$$

with $u_i(n) \equiv \boldsymbol{J}_i(n) \cdot [\boldsymbol{\xi}(n+1) - c\boldsymbol{\xi}(n)]$. In principle we have to study the combined dynamics of the weights and order parameters, as given by (2) and (4), respectively, and the local fields as given by (6). Luckily, however, the time scales of these two processes differ by a factor of order N: the local fields change much faster than the weights and order parameters. In the thermodynamic limit, we can 'adiabatically eliminate the fast variables' [6, 9], which basically means that we can act as if the local fields have reached their stationary distribution for fixed order parameters R_{ij} and weights w_i and use this distribution to compute the averages on the right-hand sides of (4). The correlations occur because this stationary distribution of the local fields depends on c. This may seem counterintuitive at first: Why can we not apply the central limit theorem to derive the distribution of $x_i(n) = \boldsymbol{J}_i(n) \cdot \boldsymbol{\xi}(n)$? The reason is that $\boldsymbol{\xi}(n)$ is no longer independent of $\boldsymbol{J}_i(n)$: the current

† Since $\langle x_i \rangle_{\boldsymbol{x}} = 0$ and $\langle x_i x_j \rangle_{\boldsymbol{x}} = R_{ij}$.

example is correlated with the recent examples to which $\boldsymbol{J}_i(n)$ has been adapted. This effect, which even in the thermodynamic limit is non-negligible for nonzero c, is captured in (6).

The remaining task is to compute the stationary distribution resulting from the dynamics (6). First, we observe that for fixed $\boldsymbol{J}_i(n)$, the variables $u_i(n)$ are normally distributed with average zero and covariance matrix

$$\begin{aligned}\langle u_i(n)u_j(m)\rangle_{\boldsymbol{\xi}} &= \langle(\boldsymbol{J}_i(n)\cdot[\boldsymbol{\xi}(n+1)-c\boldsymbol{\xi}(n)])(\boldsymbol{J}_j(m)\cdot[\boldsymbol{\xi}(m+1)-c\boldsymbol{\xi}(m)])\rangle_{\boldsymbol{\xi}}\\ &= (1-c^2)R_{ij}(n)\delta_{nm}.\end{aligned}\tag{7}$$

Next, because of the symmetry† $\delta_i(-\boldsymbol{x}) = -\delta_i(\boldsymbol{x})$, the distribution of local fields x_i must also be symmetric, i.e. $\langle x_{i_1}x_{i_2}\ldots x_{i_n}\rangle_{\boldsymbol{x}} = 0$ for any odd number of terms n. From the stationarity condition

$$\langle x_i(n+1)x_j(n+1)\rangle_{\boldsymbol{x}} = \langle x_i(n)x_j(n)\rangle_{\boldsymbol{x}}$$

and expressions (6) and (7) we obtain

$$C_{ij} = \langle x_i x_j\rangle_{\boldsymbol{x}} = R_{ij} + \frac{c^2}{1-c^2}[\eta_1\langle x_i\delta_j(\boldsymbol{x}) + x_j\delta_i(\boldsymbol{x})\rangle_{\boldsymbol{x}} + \eta_1^2\langle\delta_i(\boldsymbol{x})\delta_j(\boldsymbol{x})\rangle_{\boldsymbol{x}}].\tag{8}$$

Alas, the general solution of (8) is intractable. We can, however, make an excellent approximation by assuming that the stationary distribution of the local fields is a Gaussian‡. Then we can compute the averages on the right-hand side of (8) and obtain a self-consistent equation for the covariance matrix $\mathcal{C}$. In fact, it can be shown that the fourth-order cumulant (kurtosis) of the stationary distribution is of order c^4 and thus that, at least for small c, the assumption of normality is fair. In other words, all the following results are accurate up to order $\epsilon = c^2/(1-c^2)$.

There is a striking resemblance between equations (5) and (8). A combination of these expressions provides a simple and elegant summary of the effect of correlations on the learning dynamics of two-layered networks (including simple perceptrons and soft-committee machines):

$$\frac{\mathrm{d}\mathcal{R}}{\mathrm{d}t} = \mathcal{F}(\mathcal{C},\boldsymbol{w}) \qquad \text{and} \qquad \frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}t} = \boldsymbol{f}(\mathcal{C},\boldsymbol{w})\tag{9}$$

$$\mathcal{C} = \mathcal{R} + \epsilon\mathcal{F}(\mathcal{C},\boldsymbol{w})\tag{10}$$

with $\mathcal{R}$ being the set of order parameters, $\boldsymbol{w}$ being the set of hidden to output weights, $\mathcal{C}$ being the stationary covariance matrix of the local fields and $\mathcal{F}$ and $\boldsymbol{f}$ being functions that can be found in papers describing the (uncorrelated) learning dynamics of specific architectures and problems [8, 10]. The remarkable thing here is that in order to study learning with correlated examples we do not need to compute new difficult integrals. Provided that the normality assumption holds, which can easily be checked if there is any doubt, the dynamical equations for uncorrelated patterns are all we need to know to compute the dynamics for correlated patterns. In the next sections we will give specific examples.

3. Learning a simple perceptron

In this section we will consider simple perceptron learning, i.e. learning a network without hidden units. Simple perceptron learning fits into the general description of the previous

† Because of this condition, the article excludes networks with thresholds.

‡ If i belongs to the teacher, the distribution of x_i is indeed a Gaussian.

section if we take $\eta_2 = 0$ and $\forall_i w_i = 1$ with $i = 1, 2$. We set $i = 1$ aside to the student and $i = 2$ to the teacher. Initial conditions are set to $R_{11} = 1$, $R_{12} = 0$ with a normalized teacher vector, i.e. $R_{22} = 1$. For convenience, we choose $\eta_1 = 1$. In section 3.1 we study Hebbian learning for which we can compare the calculations of section 2 with exact calculations. In section 3.2 we study a graded-response perceptron learning by on-line gradient descent. Here we compare our (approximate) analytical results with computer simulations.

3.1. Hebbian learning

Hebbian learning is slightly different from the gradient descent learning rule (1) treated in the rest of the paper. We have $g(x) = \mathrm{sign}(x)$ and $\delta_1(\boldsymbol{x}) = \boldsymbol{\sigma}_{\mathrm{teacher}}(\boldsymbol{x}) = \mathrm{sign}(x_2)$, i.e.

$$\boldsymbol{J}_1(n+1) = \boldsymbol{J}_1(n) + \frac{1}{N}\,\mathrm{sign}[x_2(n)]\boldsymbol{\xi}(n).$$

A perceptron with binary output can serve as a tool for classification instead of regression. The generalization error is then defined as the probability that a new randomly drawn input is misclassified and reads (see, for example [11]):

$$\varepsilon_g(t) = \frac{1}{\pi}\arccos\left[\frac{R_{12}(t)}{\sqrt{R_{11}(t)}}\right]. \tag{11}$$

Let us first compute the evolution of the order parameters following the scheme of section 2. The averages (5) become

$$F_{11} = \langle x_1\,\mathrm{sign}(x_2)\rangle_{\boldsymbol{x}} + 1 = 2\sqrt{\frac{2}{\pi}}C_{12} + 1 \qquad \text{and} \qquad F_{12} = \langle |x_2|\rangle_{\boldsymbol{x}} = \sqrt{\frac{2}{\pi}}$$

where we used the Gaussian assumption only in the first equation. Using (9) and (10), we arrive at the set of differential equations

$$\frac{\mathrm{d}R_{11}(t)}{\mathrm{d}t} = 2\sqrt{\frac{2}{\pi}}R_{12}(t) + 1 + \frac{4\epsilon}{\pi} \qquad \text{and} \qquad \frac{\mathrm{d}R_{12}(t)}{\mathrm{d}t} = \sqrt{\frac{2}{\pi}}$$

with solution

$$R_{11}(t) = 1 + \left(1 + \frac{4\epsilon}{\pi}\right)t + \frac{2}{\pi}t^2 \qquad \text{and} \qquad R_{12}(t) = \sqrt{\frac{2}{\pi}}\,t.$$

Alternatively, the evolution of the norm $R_{11}(t)$ can be calculated exactly using the evolution of the student vector

$$\boldsymbol{J}_1(t) = \boldsymbol{J}_1(0) + \frac{1}{N}\sum_{n=0}^{Nt}\mathrm{sign}[x_2(n)]\boldsymbol{\xi}(n).$$

Straightforward calculations then give

$$\begin{aligned} R_{11}(t) = \boldsymbol{J}_1(t)\cdot\boldsymbol{J}_1(t) &= 1 + \left(1 + \frac{4}{\pi}\sum_{n=1}^{\infty} c^n \arcsin c^n\right)t + \frac{2}{\pi}t^2 \\ &= 1 + \left(1 + \frac{4\epsilon}{\pi} + \frac{2\epsilon^2}{3\pi} + \cdots\right)t + \frac{2}{\pi}t^2 \end{aligned}$$

and the same expression as before for the overlap $R_{12}(t)$.

Both methods are indeed equal up to order ϵ. The evolution of the generalization error (11) for $c = 0.0$, 0.6 and 0.9 is shown in figure 1(a). We compare the theory (full curves) with the alternative method (symbols). Even for large correlations, the symbols hardly leave the full curves, i.e. the Gaussian assumption seems to be a good

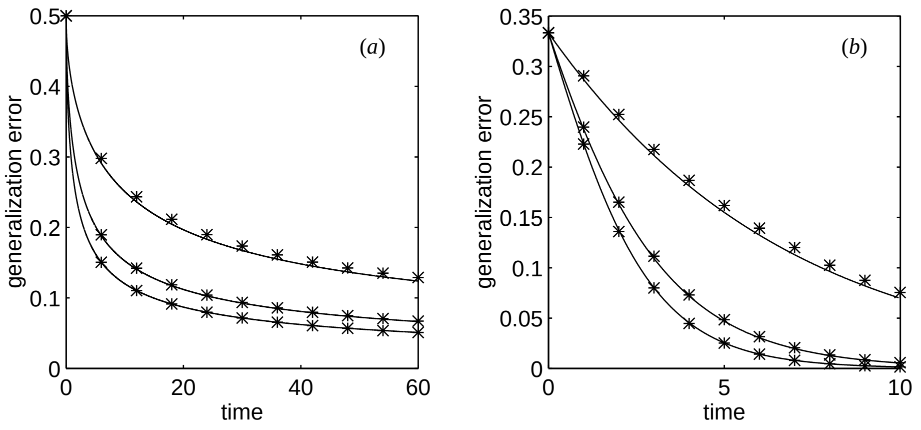

Figure 1. Correlations slow down the learning process. The generalization error, i.e. expression (11) in (*a*) and expression (3) in (*b*), is plotted as a function of time t for correlations $c = 0.0$, 0.6 and 0.9, (from bottom to top). (*a*) Hebbian learning. The theory (full curves) is checked by the alternative method (symbols). (*b*) Learning by on-line gradient descent. The theory is checked by simulations (symbols) for a network with N=100 input units, averaged over 100 independent runs.

approximation. Note also that the performance of the student perceptron at any time t for $c = 0.9$ is much worse than for $c = 0.6$. In other words, learning a simple perceptron with dependencies between successive examples decreases the learning performance. Intuitively we can imagine this since correlations slow down the process of covering the entire sample space, for example, for high correlations, examples are drawn for a relatively long time from a small region of the whole sample space.

3.2. Learning by on-line gradient descent

As a simple example of on-line gradient descent we consider in this section a graded response perceptron [2] whose output equals the sigmoid $g(x) = \mathrm{erf}(x/\sqrt{2})$. For this specific choice of the transfer function the averages (5) can be performed analytically (see [2]). Computation of the evolution of the covariances C_{11} and C_{12} and the order parameters R_{11} and R_{12} then becomes a straightforward exercise in numerical analysis, for example, by using the fourth-order Runge–Kutta formula [12].

In figure 1(*b*) we show the generalization error (3) as a function of time t for $c = 0.0$, 0.6 and 0.9. The theory (full curves) is now checked by simulations (symbols) for a network with $N = 100$ input units averaged over 100 independent runs. Simulations are done on the level of (1). For $c = 0.9$ we see that the theoretical curve deviates only slightly from its simulation. As in the case of Hebbian learning, correlations slow down the learning process.

4. Learning in soft-committee machines

The effect of correlations on the learning performance in the examples presented in the previous section is negative. In this section, we investigate how correlations affect learning in soft-committee machines.

The generalization performance of a soft-committee machine as a function of time is often dominated by a long time span in which this performance hardly improves [7, 8]. This so-called 'plateau' is in fact caused by a saddle point in the dynamics of the order parameters. The delayed repulsion from this saddle point is due to the fact that the corresponding positive eigenvalue of the linearized system, around this saddle point, is very small in comparison with the absolute values of the negative eigenvalues. The equivalence between the two sets of functions $\mathcal{F}$ in (9) and (10) makes it surprisingly straightforward to analyse the correlated learning dynamics in the neighbourhood of such a saddle point or, in general, of any fixed point.

The output unit of soft-committee machines is linear and the couplings from all the hidden units to the output unit are positive and of unit strength, i.e. $\forall_i w_i = 1$ and $\eta_2 = 0$. In other words, we only have to consider the dynamics of the order parameters. Linearization of the learning dynamics near a fixed point, where $\mathcal{F}(\mathcal{C}_{\mathrm{fp}}) = 0$, yields

$$\frac{\mathrm{d}\mathcal{R}}{\mathrm{d}t} = \mathcal{H}(\mathcal{R} - \mathcal{R}_{\mathrm{fp}}) \qquad \text{with } \mathcal{H} = \frac{\partial \mathcal{F}(\mathcal{C})}{\partial \mathcal{R}} = \frac{\partial \mathcal{F}(\mathcal{C})}{\partial \mathcal{C}} \frac{\partial \mathcal{C}}{\partial \mathcal{R}} = \frac{\partial \mathcal{F}(\mathcal{C})}{\partial \mathcal{C}} \left[\mathbb{1}_{|\mathcal{R}|} - \epsilon \frac{\partial \mathcal{F}(\mathcal{C})}{\partial \mathcal{C}} \right]^{-1}$$

with all derivatives evaluated at $\mathcal{C} = \mathcal{C}_{\mathrm{fp}}$ and where the last step follows from differentiation of (10) with respect to $\mathcal{R}$. In this symbolic notation $\mathcal{R}$ is best read as a vector consisting of all, say $|\mathcal{R}|$, order parameters which makes $\mathcal{H}$ an $|\mathcal{R}| \times |\mathcal{R}|$ matrix; $\mathbb{1}_{|\mathcal{R}|}$ stands for the $|\mathcal{R}|$-dimensional identity matrix. The eigenvalues λ of this matrix determine the stability of a fixed point. From (10) we deduce that $\mathcal{R}_{\mathrm{fp}} = \mathcal{C}_{\mathrm{fp}}$. In other words, a fixed point for learning without correlations is also a fixed point for learning with correlations. But then

$$\left. \frac{\partial \mathcal{F}(\mathcal{C})}{\partial \mathcal{C}} \right|_{\mathcal{C}_{\mathrm{fp}}} = \left. \frac{\partial \mathcal{F}(\mathcal{R})}{\partial \mathcal{R}} \right|_{\mathcal{R}_{\mathrm{fp}}} \equiv \mathcal{H}_0 \qquad \text{and thus} \qquad \mathcal{H} = \mathcal{H}_0 [\mathbb{1}_{|\mathcal{R}|} - \epsilon \mathcal{H}_0]^{-1}$$

where $\mathcal{H}_0$ refers to the matrix for $c = 0$. Correlations do not change the eigenvectors of the matrix $\mathcal{H}$, but do transform an eigenvalue λ_0 of the matrix $\mathcal{H}_0$ into

$$\lambda = \frac{\lambda_0}{1 - \epsilon \lambda_0} \qquad \text{and thus} \qquad \mathrm{Re}(\lambda) = \mathrm{Re}(\lambda_0) + \epsilon [\mathrm{Re}(\lambda_0)^2 - \mathrm{Im}(\lambda_0)^2] + \cdots.$$

The eigenvalue with the largest real part is the most interesting one. In case of a stable fixed point, the least negative eigenvalue governs the speed of convergence. With correlations this eigenvalue then becomes less negative, and thus increases the time to convergence. This is consistent with our previous results in section 3. The most positive eigenvalue rules the repulsion from a saddle point and is thus a key factor for determining the length of a plateau. Following the general setting of this paper, we should say that the effect of correlations depends on whether the real part of the largest eigenvalue dominates the imaginary part. In the situations we have encountered, both in our own experience and in the literature, the imaginary part is either completely absent or smaller (in absolute sense) than the real part. Calling this the typical situation, we conclude that correlations make unstable points more unstable and thus lead to shorter plateaus. The length of the plateau is inversely proportional to the most positive eigenvalue λ; the shortening, as a result of correlations, is therefore proportional to ϵ:

$$\text{shortening} \propto \frac{1}{\lambda_0} - \frac{1}{\lambda} = \epsilon. \tag{12}$$

As an example, let us consider a soft-committee machine with two hidden units ($i = 1, 2$) trained to implement a simple task defined by a single-layer teacher ($i = 3$) as in [2]. The averages F_{11}, F_{12}, F_{22}, F_{13} and F_{23} have been computed analytically and are used in (9) and (10) to compute the evolution of the order parameters. In order to show the effect of the correlations on the length of the plateau, we choose zero correlations initially, and

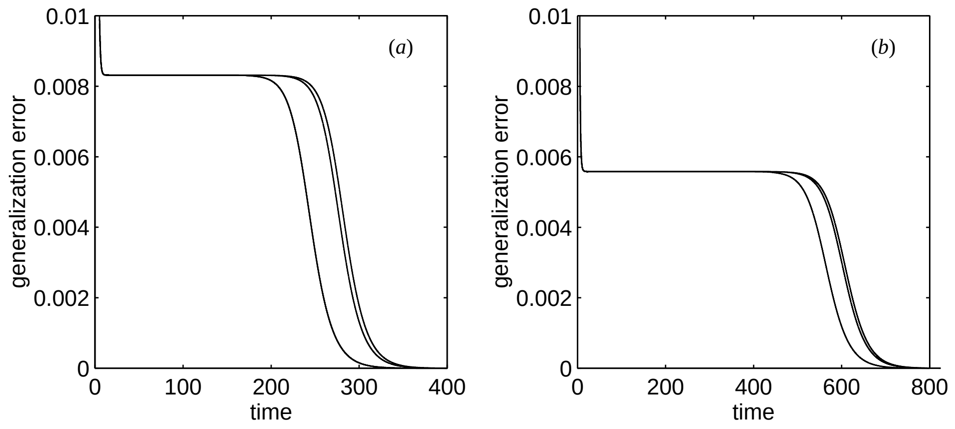

Figure 2. Correlations shorten the length of the plateau for soft-committee machines. The generalization error (3) is plotted as a function of time t for correlations $c = 0.0$, 0.6 and 0.9 (from right to left). A soft-committee machine with two hidden units, is trained, with learning parameter $\eta_1 = 1.0$ in (a) and $\eta_1 = 0.5$ in (b), to implement a simple task defined by a teacher perceptron. Initial conditions are set according to $R_{11} = R_{12} = R_{13} = R_{23} = 0.0$ and $R_{22} = 0.0001$.

add correlations when the system gets stuck at the saddle point. In figure 2 we show the evolution of the generalization error for $c = 0.0$, 0.6 and 0.9. The learning parameter in figure 2(a) is equal to 1 whereas in figure 2(b) the learning parameter is equal to 0.5. As predicted by (12), the shortening when going from correlations $c = 0.6$ to $c = 0.9$ is much more prominent than when going from $c = 0.0$ to 0.6. Furthermore, the shortening is about the same for both learning parameters.

5. Learning with adaptive hidden-to-output weights

No such general statements as in the previous section can be made with adaptive hidden-to-output weights. For any particular situation, however, the effect of correlations can be calculated using the set of expressions (9) and (10).

As an illustration we consider a student with two hidden units ($i = 1, 2$) trained by a teacher, also with two hidden units ($i = 3, 4$). The teacher is chosen to be symmetric, i.e. $R_{33} = R_{44} = 1$ and $R_{34} = 0$, with hidden-to-output weights $w_3 = w_4 = 1$. The student is initialized with small weights $\boldsymbol{J}_i$ and w_i and has learning parameters $\eta_1 = \eta_2 \equiv \eta$. Initialization with small weights is the standard procedure in practical applications of backpropagation for multilayer perceptrons, which supposedly reduces the chance to end up at a suboptimal local minimum. The origin, however, is a saddle point where all derivatives are zero. It is different from the saddle points usually studied where the problem is to break the symmetry between the student's hidden units. The escape from the origin saddle point, on the other hand, appears to require no symmetry breaking, yet a combined increase of the hidden-to-output weights $w_1 = w_2 \equiv w$ and an alignment of the student weights $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$ to the teacher weights $\boldsymbol{J}_3$ and $\boldsymbol{J}_4$, i.e. an increase of the inner products $R_{13} = R_{14} = R_{23} = R_{24} \equiv R$. The evolution of w and R follows from (9) and (10) where the functions $\mathcal{F}$ and $\boldsymbol{f}$ computed in [8, 10] are linearized around the origin:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} w \\ R \end{pmatrix} = \frac{2\eta}{\pi}\begin{pmatrix} 2\gamma & \sqrt{2} \\ \frac{1}{2}\sqrt{2} & 0 \end{pmatrix}\begin{pmatrix} w \\ R \end{pmatrix} \qquad \text{with } \gamma = \frac{\eta}{\pi}\frac{c^2}{1-c^2}. \tag{13}$$

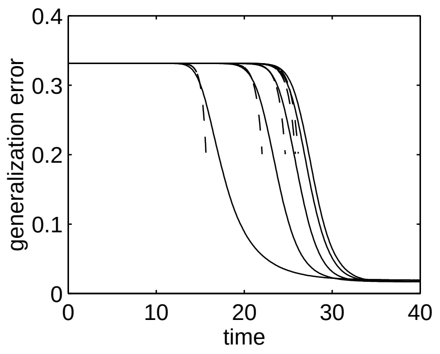

Figure 3. Correlations shorten the length of the plateau for two-layered networks. The generalization error (3) is plotted as a function of time t for correlations $c = 0.0$, 0.2, 0.4, 0.6, and 0.8 (from right to left). Both student and teacher are two-layered networks with two hidden units. The full curves are simulations, the broken curves correspond to the theoretical approximation (13). The simulations were done with a network of size $N = 1000$ averaged over 50 independent runs.

The derivatives of the inner products R_{11}, R_{12} and R_{22}, concerning only student weights, are of higher order in R and w and do therefore not influence the escape from the origin. This escape is dominated by the eigenvalue

$$\lambda = \frac{2\eta}{\pi}\left(\gamma + \sqrt{1+\gamma^2}\right)$$

which, through γ, strongly depends on the amount of correlations c. An important side effect is that with adaptive hidden-to-output weights the eigenvectors and thus the directions of escape are affected by the amount of correlations. In our simulations, the student network, after leaving the first plateau due to the saddle point at the origin, gets stuck at another saddle point. The different escape directions determine that the values of the order parameters $\mathcal{R}$ and weights $\boldsymbol{w}$, and thus the height of this second plateau, depend on the correlation c.

In figure 3 we show simulations of the evolution of the generalization error (3) as a function of time t for different amounts of correlations c for fixed $\eta = 1$. The student weights are initialized such that $R_{12} = R_{13} = R_{14} = R_{23} = R_{24} = 0$ and $R_{11} = R_{22} = 10^{-12}$, and with $w_1 = w_2 = 0$. The simulations, indicated by the full curves, are on the level of the weights as given by (1) and (2). The broken curves result from set (13). These differential equations predict the time at which the networks escape from the saddle point quite accurately, but are, of course, no longer valid after this escape. Furthermore, it can be seen that correlations shorten the plateau and that this effect roughly scales with c^2.

6. Summary and discussion

We have studied an exactly solvable model of learning in ‘large’ two-layered networks with correlations between successive training examples. Correlations change the (stationary) distribution of the local fields. Assuming that this distribution is a Gaussian, we arrived at a set of expressions which can be solved numerically to yield the evolution of the order parameters and the hidden-to-output weights. We came to the remarkable conclusion that the expressions that govern the dynamics for uncorrelated patterns are all that is needed to compute the (approximate) dynamics for correlated patterns. These dynamical equations are accurate up to first order in $\epsilon = c^2/(1 - c^2)$, where c is the correlation parameter ($|c| < 1$), but in simulations appear to give a more than reasonable approximation for correlations as high as $c = 0.9$. For several networks, including simple perceptrons, soft-committee machines and two-layered networks with adaptive hidden-to-output units, we investigated the evolution of the generalization performance for different amounts of correlations and compared it with our theoretical results. The effect of correlations on networks with thresholds is somewhat more complicated and deserves further attention.

In principle, we could argue that it is a good idea to add correlations when learning seems to be stuck at a saddle point, yet to prevent correlations during any other stage of the learning process. The beneficial effect on the plateau, however, is for the 'large' networks studied in this paper relatively small and might as well be achieved through a simple increase in the learning parameter. The effect of correlations reported in studies on backpropagation in 'small' multilayer perceptrons is much more dramatic: here the presence of correlations can make the difference between at some point leaving the plateau or being stuck forever [4]. The reason is that in those studies learning is trapped not at a saddle point, but on a 'real' plateau in weight space where the largest eigenvalue is zero instead of positive. An escape from this plateau requires both an increase of the weights and, at the same time, a breaking of the symmetry between the hidden units. In the 'large' networks studied in the statistical mechanics framework, we have not yet, to the best of our knowledge, encountered such a combination.

Acknowledgments

We would like to thank Peter Riegler for valuable discussions that motivated this research. We are grateful to Bert Kappen, Wim Wiegerinck, Stan Gielen and Piërre van de Laar for their suggestions and comments on an earlier version of this manuscript.

References

[1] Saad D and Solla S 1995 Exact solution for on-line learning in multilayer neural networks *Phys. Rev. Lett.* **74** 4337
[2] Biehl M and Schwarze H 1995 Learning by on-line gradient descent *J. Phys. A: Math. Gen.* **28** 643
[3] Barkai N, Seung H and Sompolinsky H 1995 On-line learning of dichomoties *Phys. Rev. Lett.* **75** 1415
[4] Wiegerinck W and Heskes T 1996 How dependencies between successive examples affect on-line learning *Neural Comput.* **8** 1743
[5] Heskes T and Kappen B 1991 Learning processes in neural networks *Phys. Rev.* A **44** 2718
[6] Wiegerinck W and Heskes T 1994 On-line learning with time-correlated patterns *Europhys. Lett.* **28** 451
[7] Biehl M, Riegler P and Wöhler C 1996 Transient dynamics of on-line learning in two-layered neural networks *J. Phys. A: Math. Gen.* **29** 4769
[8] Saad D and Solla S 1995 On-line learning in soft committee machines *Phys. Rev.* E **52** 4225
[9] Haken H 1978 *Synergetics, an Introduction* (New York: Springer)
[10] Riegler P and Biehl M 1995 On-line backpropagation in two-layered networks *J. Phys. A: Math. Gen.* **28** L507
[11] Watkin T, Rau A and Biehl M 1993 The statistical mechanics of learning a rule *Rev. Mod. Phys.* **65** 499
[12] Press W, Teukolsky S, Vetterling W and Flannery B 1992 *Numerical Recipes in C* (Cambridge: Cambridge University Press) 2nd edn