Learning-parameter adjustment in neural networks

Tom M. Heskes and Bert Kappen
Department of Medical Physics and Biophysics, University of Nijmegen, Geert Grooteplein Noord 21, 6525 EZ Nijmegen, The Netherlands
(Received 30 December 1991)

We present a learning-parameter adjustment algorithm, valid for a large class of learning rules in neural-network literature. The algorithm follows directly from a consideration of the statistics of the weights in the network. The characteristic behavior of the algorithm is calculated, both in a fixed and a changing environment. A simple example, Widrow-Hoff learning for statistical classification, serves as an illustration.

PACS number(s): 87.10.+e

I. INTRODUCTION

Recently, some progress has been made in understanding learning processes in neural networks [1]. Instead of considering the dynamics of the fast variables, such as the spins in Hopfield-type neural networks, one focuses on the dynamics of the slow variables, the synapses and the thresholds. We will refer to these adaptive elements as weights and will denote them by an N-dimensional vector $\mathbf{w}$. These weights are adapted according to a learning rule, which is typically of the form

$$\Delta\mathbf{w}\equiv\mathbf{w}(t+1)-\mathbf{w}(t)=\eta\mathbf{f}(\mathbf{w},\vec{x}) . \tag{1}$$

That is, the change in the network representation at time t is fully determined by the current network representation $\mathbf{w}$ and the presented training pattern, denoted by an n-dimensional vector $\vec{x}$. This training pattern $\vec{x}$ is drawn at random from the set Ω of all possible training patterns according to some probability function $\rho(\vec{x},t)$. The learning parameter η sets the typical magnitude of the change. Most learning rules for neural networks obey Eq. (1). Examples are backpropagation [2] for multilayered perceptrons, Kohonen-type learning [3,4] for topological maps, and Hebbian learning [5,6] for attractor neural networks.

In an abstract sense, learning is the way the network builds up an internal representation of its environment Ω. This representation is encoded in the weights $\mathbf{w}$ of the network. Consider for example a multilayered perceptron that is trained with backpropagation to perform a classification task. The environment Ω in this case is the set of all input-output relations to be learned. Learning takes place by presenting the examples $\vec{x}\in\Omega$, where each $\vec{x}$ represents an input-output pair. For every training pattern $\vec{x}$, the backpropagation learning rule is applied. If learning is successful, the relationship between input and output is encoded in the network state $\mathbf{w}$ and the network is ready for its classification task. In general, the function of a neural network, e.g., classification, recognition, feature extraction, or memory, depends on the internal representation that is formed, and thus on the network architecture and learning procedure.

In most neural networks, learning takes place only during a so-called learning phase. During this phase, the set of training patterns is presented to the network a (large) number of times. It is customary to choose the learning parameter η either as a constant or as a slowly decreasing function of the learning iteration. The learning phase usually stops as soon as a preset criterion is met. For instance, for a multilayered perceptron with backpropagation this can be the average quadratic error over the training set, or for a Kohonen network this may be some measure of topological order. After this, the network enters the operation mode, during which, usually, no learning takes place.

Both from a biological and from an applications point of view, this is an undesirable situation. Biological systems "learn" and "operate" throughout their existence. An example is the adaptation of motor programs as a result of limb growth. For industrial applications it is desirable to have systems that autonomously decide whether different operations are necessary and thus adapt their internal representations to meet the new requirements. Examples are easily found in many application areas, such as robotics, speech recognition ("cocktail party" effect), and financial modeling (the relation between economic quantities changes over time). We are therefore interested in neural networks that are capable of learning in fixed as well as in changing environments.

In a previous paper [1], we studied the behavior of learning rules obeying Eq. (1) for small constant learning parameters, both in a fixed environment, i.e., for $\rho(\vec{x},t)=\rho(\vec{x})$, independent of time t, and in a gradually changing environment. Since the training patterns are drawn at random, learning becomes a stochastic process. It is possible to write a master equation for the probability that the network will have a certain state $\mathbf{w}$ at time t; and from this master equation, evolution equations for the average network state and the fluctuations around this average can be derived. For large times, small learning parameters, and a slowly changing or nonchanging environment, the network has a very high probability of being in the neighborhood of an attractive fixed point $\mathbf{w}^*$ of the differential equation

$$\frac{d\mathbf{w}(t)}{dt}=\eta\int d^n x\,\rho(\vec{x},t)\mathbf{f}(\mathbf{w}(t),\vec{x})\equiv\eta\langle\,\mathbf{f}(\mathbf{w}(t),\vec{x})\rangle_\Omega\ . \tag{2}$$

This equation is the differential form of Eq. (1), but with the right-hand side averaged over the set of training patterns Ω. By linearizing around this fixed point or "locally optimal solution" $\mathbf{w}^*$, it is found that in a fixed environment the final average network state is almost equal to the desired state $\mathbf{w}^*$, but that there are remaining fluctuations in the network's representation, which are proportional to the learning parameter η.

In a changing environment, the fixed points $\mathbf{w}^*(t)$ of Eq. (2) are a function of time. Now, there is a conflict. On the average, the network state lags behind the desired state $\mathbf{w}^*(t)$. The difference between these two, which we will call the bias, can be shown to be inversely proportional to the learning parameter. On the other hand, the fluctuations are still proportional to the learning parameter, as in the static case. So, in order to obtain a great adaptability, one would like to choose a large learning parameter, but this leads to relatively large fluctuations and thus to inaccuracy. In order to quantify a trade-off between these two effects, we introduced an error $\mathcal{E}$ of the form

$$\mathcal{E}=\langle\,|\mathbf{w}-\mathbf{w}^*|^2\rangle_\Xi\ , \tag{3}$$

i.e., the squared distance between the network state and the desired state $\mathbf{w}^*$, averaged over a large ensemble Ξ of identical networks. This error has two components, the squared bias and the variance, so

$$\mathcal{E}=|\langle\mathbf{w}\rangle_\Xi-\mathbf{w}^*|^2+\langle\,|\mathbf{w}-\langle\mathbf{w}\rangle_\Xi|^2\rangle_\Xi\approx\alpha/\eta^2+\beta\eta\ .$$

The constants α and β depend on the rate of change in the environment, the fluctuations in the learning rule, and so on. Once the constants α and β are known, the "optimal" learning parameter that minimizes the error $\mathcal{E}$ can be calculated. In the simple examples discussed in [1] it was possible to compute these constants.

In general, however, all the information needed to calculate these constants is not directly available. In this paper, we will develop an algorithm that estimates the optimal learning parameter automatically from the statistics of the weights. We are faced with two practical problems. In the first place, the theory described in [1] is about an ensemble of networks. Yet it is very unprofitable to gather statistics from a large number of networks to update the learning parameter of the one that is trained. Therefore, we will replace the averages over an ensemble of networks by time averages for the network that is trained. Another problem is the huge amount of memory and computation needed to keep track of the statistics of all the weights in a large network. This problem is solved by considering the statistics of just one variable W. This W, a linear function of the weights in the network, is introduced in Sec. II. We derive a working hypothesis, which describes an approximate dynamics for this variable, as well as an error criterion similar to the one in Eq. (3).

The working hypothesis gives us the opportunity to estimate the dynamical properties of the variable W. These properties, which are worked out in the Appendix, can be used to obtain the final autonomous algorithm in Sec. III.

In Sec. IV an attempt is made to study the performance of the algorithm in three cases. In each of these cases a specific behavior is required. In a fixed environment, the learning parameter must converge slowly to zero, obeying the conditions derived by Ljung [7] and Kushner and Clark [8] for general stochastic processes. In case of a sudden change in the environment, the learning parameter must grow; in other words, the algorithm must act as a sort of "arousal detector." In a gradually changing environment, the algorithm must yield a learning parameter that gives a compromise between the adaptability and the accuracy of the network.

The performance of the algorithm is illustrated in Sec. V. A perceptron consisting of two weights and one threshold, trained with the Widrow-Hoff learning rule [9], has to find the decision boundary between two input classes. The algorithm is tested in the three different situations theoretically investigated in Sec. IV.

In Sec. VI the main results are summarized, a comparison is made with other adaptive algorithms, and the limitations of the algorithm are discussed.

II. ERROR CRITERION

Learning drives the network state $\mathbf{w}$ to "locally optimal solutions," which are attractive fixed points $\mathbf{w}^*$ of the differential equation (2). For an ensemble Ξ of networks, the dynamics of the average network state can be found by expanding around the fixed point $\mathbf{w}^*$ [1]

$$\begin{aligned}\Delta\langle w_i\rangle_\Xi\equiv<\langle\Delta w_i\rangle_\Omega>_\Xi&=\eta<\langle f_i(\mathbf{w},\vec{x})\rangle_\Omega>_\Xi\\&\approx-\eta\sum_j G_{ij}(\langle w_j\rangle_\Xi-w_j^*)\ ,\end{aligned} \tag{4}$$

where the positive-definite matrix G is the first derivative of the average learning rule

$$G_{ij}\equiv-\frac{\partial\langle f_i(\mathbf{w},\vec{x})\rangle_\Omega}{\partial w_j}\Big|_{\mathbf{w}=\mathbf{w}^*}\ .$$

We introduce the variable W as

$$W\equiv\sum_i a_i w_i,\quad W^*\equiv\sum_i a_i w_i^*\ ,$$

where $\mathbf{a}$ is some fixed N-dimensional vector, to be discussed below. Using Eq. (4), we find the average behavior of this variable in the neighborhood of the optimal W^*

$$\Delta\langle W\rangle_\Xi=-\eta\lambda(\mathbf{w})(\langle W\rangle_\Xi-W^*)\ , \tag{5}$$

with "spring constant"

$$\lambda(\mathbf{w})=\frac{\sum_{ij}a_i G_{ij}(w_j-w_j^*)}{\sum_{ij}a_i\delta_{ij}(w_j-w_j^*)}\ . \tag{6}$$

Note that $\lambda(\mathbf{w})$ does not depend on the network state $\mathbf{w}$ if and only if the vector $\mathbf{a}$ is a left eigenvector of the matrix G. We will avoid the difficult search for such an eigenvector and take a random vector $\mathbf{a}$, still assuming that we

may treat λ as a constant in our analysis.

Equation (5) describes the dynamics of the average behavior of the stochastic process $W(t)$. Our second assumption is that we can write this stochastic process as

$$\Delta W = W(t+1) - W(t) = -\eta[\lambda(W - W^*) + \xi] , \quad (7)$$

With ξ white noise obeying $\langle \xi \rangle_\Omega = 0$ and $\langle \xi^2 \rangle_\Omega = \chi^2$, independent of $\mathbf{w}$ and η. This noise is due to the fact that the training patterns are drawn at random from the environment Ω. Equation (7), together with $\lambda(\mathbf{w}) = \lambda$, is our working hypothesis. Basically it says that the variable W is attracted to its optimal value W^* like a noisy spring. Later on, we will give some arguments to support the rather crude approximations that led to this hypothesis.

We will consider a network in a gradually changing environment. In a changing environment the locally optimal solution $\mathbf{w}^*$ and thus the system variable W^* are functions of time. We will define $\nu \equiv \Delta W^*$ as the rate of change of $W^*(t)$. An indication for the network performance is the average quadratic distance between W and W^* (compare with Eq. [3])

$$\mathcal{E} = \langle (W - W^*)^2 \rangle_\Xi = M^2 + \Sigma^2 .$$

This error has two components, the squared bias $(M \equiv \langle W \rangle_\Xi - W^*)$ and the variance $[\Sigma^2 \equiv \langle (W - \langle W \rangle_\Xi)^2 \rangle_\Xi]$. Note that Eq. (7) can be interpreted as a learning rule of the form (1). In [1] we derived for general learning rules of this type, operating in a gradually changing environment,

$$M \propto \frac{1}{\eta}, \quad \Sigma^2 \propto \eta . \quad (8)$$

There is a conflict: for a small bias one would like to choose a large learning parameter, for a small variance a small learning parameter. The compromise between these two conflicts is given by the minimum of the error $\mathcal{E}$ with respect to η. Suppose the system is learning with learning parameter η. As we will see, from the statistics of W we can estimate M and Σ^2. The new learning parameter η_{new} can be obtained by minimizing the error

$$\mathcal{E}(\eta_{\text{new}}) = \frac{\eta^2}{\eta_{\text{new}}^2} M^2 + \frac{\eta_{\text{new}}}{\eta} \Sigma^2 ,$$

yielding

$$\eta_{\text{new}} = \left[\frac{2M^2}{\Sigma^2} \right]^{1/3} \eta . \quad (9)$$

III. THE ALGORITHM

In order to calculate the new learning parameter η_{new} from Eq. (9), we must have estimates for the bias and the variance. In fact, the bias and variance in the preceding section are averages over an ensemble of networks. Of course, it is very expensive to gather statistics from a large ensemble of networks if one is only interested in an acceptable learning parameter for one of them. The solution to this problem is to replace the averages over the ensemble of networks by averages over a certain period of time T, denoted by $\langle \ \rangle_T$ instead of $\langle \ \rangle_\Xi$, just for the network that is trained. That is, we are searching for estimates M_{estimate} and $\Sigma^2_{\text{estimate}}$ for the bias and the variance in terms of the measurable quantities $\langle W \rangle_T$, $\langle \Delta W \rangle_T$, $\langle W^2 \rangle_T$, and $\langle (\Delta W)^2 \rangle_T$.

A way to find an estimate for the variance is to make a least-squares fit on the values of W during the time period T. The slope of the best straight line yields an estimate for the average change $\langle \Delta W \rangle_\Xi$, the remaining error an estimate for the variance Σ^2. We find

$$\Sigma^2_{\text{estimate}} = \langle W^2 \rangle_T - \langle W \rangle_T^2 - \frac{T^2 \langle \Delta W \rangle_T^2}{12} . \quad (10)$$

The last term in this equation is a correction for the average change of W.

To find an estimate for the bias M, we have to use the specific dynamics of W as given in Eq. (7). It is not possible to express the bias M directly in terms of the desired averages. A reasonable guess can be found by assuming that the variance is stationary. The details can be found in the Appendix. Equation (A4) yields, after replacing the ensemble averages by the time averages,

$$M_{\text{estimate}} = -\frac{2 \langle \Delta W \rangle_T \Sigma^2_{\text{estimate}}}{\langle (\Delta W)^2 \rangle_T - \langle \Delta W \rangle_T^2} . \quad (11)$$

Thus our algorithm for on-line learning-parameter adjustment reads as follows.

(i) Gather statistics from learning with learning parameter η during time T, yielding $\langle W \rangle_T$, $\langle W^2 \rangle_T$, $\langle \Delta W \rangle_T$, and $\langle (\Delta W)^2 \rangle_T$.

(ii) Calculate $\Sigma^2_{\text{estimate}}$ and M_{estimate} using Eqs. (10) and (11) and substitute these into Eq. (9) to obtain the new learning parameter η_{new}.

A basic assumption in the derivation of this algorithm is that Eq. (7) describes the evolution of the variable W with constant parameters λ, ν, and χ^2. Now, we can relax this requirement. We only require that these parameters be more or less constant during one time interval of length T. Furthermore, a learning-parameter adjustment algorithm has no need to be very precise since the learning of the weights is the primary issue and the adaptation of the learning parameter is secondary. The choice of the time window T affects the rate of change of the learning parameter. The time window must be large enough so that the network can get a rough impression of its environment. On the other hand, the time window must be small enough to justify the assumption that the parameters λ, ν, and χ^2 can be treated as constants. In most practical cases, a time window of a few hundred learning steps will do.

IV. PERFORMANCE OF THE ALGORITHM

A. Statement of the problem

In this section we will analyze the dynamics of M, Σ^2, and η, as a function of time under the following assumptions.

(i) The variable W obeys Eq. (7) with constant λ, ν, and χ^2.

(ii) The time averages, used to estimate the bias and the variance, are equal to the ensemble averages we would have found by simulating with a number of networks instead of just one.

Of course, in most practical situations these assumptions are violated. Therefore, the aim of this section is not to give rigorous quantitative results that can be checked in simulations, but is more a way to gain some insight to the tendency of the algorithm.

The first assumption means that the evolution equations of the bias and the variance, as given by Eq. (A3) in the Appendix, are exact within each time interval, i.e.,

$$\Delta M=-\eta\lambda M-\nu\ ,\quad \Delta\Sigma^2=-\eta\lambda(2-\eta\lambda)\Sigma^2+\eta^2\chi^2\ . \tag{12}$$

The second assumption implies that we can take $\Sigma^2_{\text{estimate}}=\Sigma^2$, but *not* that $M_{\text{estimate}}=M$, because in the derivation of this estimate, we had to make the assumption that the variance Σ^2 was stationary. Since the evolution of η is defined in terms of the estimated bias M_{estimate}, we must express M_{estimate} in terms of the real bias M and the variance Σ^2. This relation can be found using Eqs. (11) and (A2), so

$$M_{\text{estimate}}=\frac{2\lambda M\Sigma^2}{\eta(\lambda^2\Sigma^2+\chi^2)},\quad \Sigma^2_{\text{estimate}}=\Sigma^2\ .$$

These values are substituted in Eq. (9) to find the new learning parameter η_{new} in terms of the old one η, yielding

$$\eta_{\text{new}}=\left[\frac{8\lambda^2\eta M^2\Sigma^2}{(\lambda^2\Sigma^2+\chi^2)^2}\right]^{1/3}\ . \tag{13}$$

Equations (12) and (13) describe the learning system under the assumptions made above. However, they are still unsolvable. To proceed, we will make two more simplifications. Since we will work only in the limit $\eta\lambda\ll 1$, we can neglect the term $\eta\lambda$ if compared with 2 in Eq. (12) and the term $\lambda^2\Sigma^2$ if compared with χ^2, since Σ^2 is of order η, as can be seen from Eq. (8). And finally, we replace the difference equations (12) and (13) by the corresponding differential equations

$$\begin{aligned}
&\frac{dM(t)}{dt}=-\eta(t)\lambda M(t)-\nu\ ,\\
&\frac{d\Sigma^2(t)}{dt}=-2\eta(t)\lambda\Sigma^2(t)+\eta^2(t)\chi^2\ ,\\
&T\frac{d\eta(t)}{dt}=\left[\frac{8\lambda^2\eta(t)M^2(t)\Sigma^2(t)}{\chi^4}\right]^{1/3}-\eta(t)\ .
\end{aligned} \tag{14}$$

We will study this set of differential equations in three different cases: a fixed environment, a suddenly changing environment, and a gradually changing environment.

B. A fixed environment

First we will consider the static case $\nu=0$. Since for small learning parameters $T\ll 1/\eta\lambda$, it seems reasonable to assume that the learning parameter changes much more rapidly than the bias and the variance. Therefore we make the ansatz that we can treat the learning parameter as being instantaneous, yielding

$$\eta(t)=\frac{2\sqrt{2}\lambda|M(t)|\Sigma(t)}{\chi^2}\ .$$

A posteriori, we can check if this ansatz is correct. Now, we have two coupled differential equations for the bias and the standard deviation

$$\begin{aligned}
&\frac{dM(t)}{dt}=-\frac{2\sqrt{2}\lambda^2}{\chi^2}M(t)|M(t)|\Sigma(t)\ ,\\
&\frac{d\Sigma(t)}{dt}=-\frac{2\sqrt{2}\lambda^2}{\chi^2}\Sigma^2(t)|M(t)|+\frac{4\lambda^2}{\chi^2}M^2(t)\Sigma(t)\ .
\end{aligned} \tag{15}$$

Doing some rescaling and rewriting with definitions

$$r\equiv\frac{2\sqrt{2}\lambda^2}{\chi^2}t,\quad f\equiv\frac{1}{\sqrt{2}M^2}\ ,$$

the set of first-order differential equations (15) reduces to one second-order differential equation for $f(r)$

$$\frac{d^2f}{dr^2}=\frac{d(\ln f)}{dr}\ .$$

The long time, i.e., large r behavior, is characterized by

$$f(r)=r\ln r\left[1+\frac{\ln\ln r}{\ln r}+\cdots\right],$$

where integration constants are left out since they do not affect the long-time behavior. So, taking only the first term into account, we find

$$\begin{aligned}
&M(t)=\pm\left[\frac{\chi^2}{4\lambda^2 t\ln t}\right]^{1/2},\\
&\Sigma(t)=\left[\frac{\chi^2\ln t}{8\lambda^2 t}\right]^{1/2},\\
&\eta(t)=\frac{\chi}{2\lambda t}\ .
\end{aligned} \tag{16}$$

The sign of $M(t)$ depends on the initial conditions. *A posteriori,* we can easily check that it is justified to neglect the derivative of $\eta(t)$ in Eq. (14) in the long time, since it is of order $1/t^2$, whereas the other terms are of order $1/t$.

It is well known in the literature [7,8,10,4] that, in order to ensure convergence to a locally optimal solution $\mathbf{w}^*$, as given by Eq. (2), the learning parameter as a function of time or learning step must obey the requirements

$$\lim_{t\to\infty}\eta(t)=0\ ,\quad \sum_{t=1}^{\infty}\eta(t)=\infty\ .$$

The first requirement is necessary to prevent asymptotic fluctuations, the second one to be sure that the network can reach any state from any initial state. Algebraic decay $\eta(t)\propto 1/t$, as found in Eq.(16), is the fastest possible decay still satisfying these two requirements. So, although originally designed for learning in a changing environment, the algorithm works properly in a fixed environment as well.

C. A sudden change

In this subsection, we study the characteristic behavior of the algorithm in case of a sudden change in the environment. We will model this by assuming that the environment is fixed for $t<0$ and that the network is in a stationary state with a small learning parameter $\eta(0)$ and corresponding variance $\Sigma^2(0)=\eta(0)\chi^2/2\lambda$. At $t=0$, there is a sudden discontinuous change in the environment and thus a large bias $M(0)$. For short times t, Eq. (14) gives

$$M(t)=M(0)-\lambda\eta(0)M(0)t+\mathcal{O}(t^2)\ ,$$

$$\Sigma^2(t)=\Sigma^2(0)+\left[\frac{2M^2(0)}{\Sigma^2(0)}\right]^{1/3}\frac{\lambda}{T}\Sigma^2(0)t^2+\mathcal{O}(t^3)\ ,$$

$$\eta(t)=\eta(0)+\left[\frac{2M^2(0)}{\Sigma^2(0)}\right]^{1/3}\frac{1}{T}\eta(0)t+\mathcal{O}(t^2)\ ,$$

where we assumed a large initial bias, i.e.,

$$\left[\frac{2M^2(0)}{\Sigma^2(0)}\right]^{\frac{1}{3}}\gg 1\ .$$

Indeed, the algorithm acts as a sort of arousal detector: the change in the environment is noticed and leads to a larger learning parameter. However, the change of the learning parameter, and thus the speed of adaptation to this new situation, depends not only on the ratio between the initial squared bias $M^2(0)$ and the variance $\Sigma^2(0)$ and the choice of the time window T, but is also proportional to the initial learning parameter $\eta(0)$. Therefore it seems a good idea to keep the learning parameter always above a certain minimal value, say, $\eta\geq 0.001$, instead of letting it decrease to zero.

D. A gradually changing environment

In a gradually changing environment, there is a constant, nonzero velocity v, which we choose to be positive. The set of equations (14) has a nontrivial stationary solution

$$M(\infty)=-\left[\frac{\chi^2 v}{4\lambda^2}\right]^{1/3}\ ,$$

$$\Sigma^2(\infty)=\left[\frac{\chi^4 v^2}{2\lambda^4}\right]^{1/3}\ , \tag{17}$$

$$\eta(\infty)=\left[\frac{4v^2}{\chi^2\lambda}\right]^{1/3}\ .$$

Making a linear expansion around this stationary point, we will try to show that it is an attractive fixed point and to calculate the typical convergence time. We write

$$\frac{d}{dt}\begin{bmatrix}M(t)-M(\infty)\\ \Sigma^2(t)-\Sigma^2(\infty)\\ \eta(t)-\eta(\infty)\end{bmatrix}=\underline{A}\begin{bmatrix}M(t)-M(\infty)\\ \Sigma^2(t)-\Sigma^2(\infty)\\ \eta(t)-\eta(\infty)\end{bmatrix}\ ,$$

with

$$\underline{A}=\begin{bmatrix}-\lambda\eta(\infty) & 0 & -\lambda M(\infty)\\ 0 & -2\lambda\eta(\infty) & 2\lambda\Sigma^2(\infty)\\ \frac{2}{3T}\frac{\eta(\infty)}{M(\infty)} & \frac{1}{3T}\frac{\eta(\infty)}{\Sigma^2(\infty)} & -\frac{2}{3T}\end{bmatrix}\ .$$

The smallest eigenvalue of this matrix yields the typical decay time. Up to lowest order in $\lambda\eta(\infty)$, the eigenvalues are

$$\Lambda_{1,2}=\left[-\tfrac{3}{2}\pm\tfrac{1}{2}\sqrt{3}i\right]\lambda\eta(\infty),\quad \Lambda_3=-\frac{2}{3T}\ . \tag{18}$$

Since the real parts of all eigenvalues are negative definite, we can conclude that in a changing environment there is an exponential decay (unlike the algebraic decay in a fixed environment) to the stationary solutions given in Eq. (17), with typical decay time

$$\tau=\frac{2}{3\lambda\eta(\infty)}=\frac{2}{3}\left[\frac{\chi}{2\lambda v}\right]^{2/3}\ .$$

The eigenvector corresponding to the eigenvalue Λ_3 is approximately (0,0,1), i.e., in the direction of the learning parameter. The other two eigenvectors lie in the space spanned by the bias and the variance. So, first the learning parameter decays rapidly to (almost) the correct value, followed by the combined convergence of the bias and the variance. Of course, during this final convergence, the learning parameter undergoes slight corrections.

Considering the algebraic decay in a fixed environment, the arousal detection in case of a sudden change in the environment, and the exponential decay in a changing environment, we can conclude that it is a good strategy to try and minimize $\mathcal{E}$ during the learning procedure.

V. AN EXAMPLE

We consider a perceptron with two input units, one output unit, two weights (w_1 and w_2), and a threshold (w_0). The output of the network is given by

$$y(\mathbf{w},\vec{x})=\tanh\left[\sum_{i=0}^{2}w_i x_i\right]\ ,$$

where x_1 and x_2 are the two input values and $x_0\equiv -1$. The learning rule is the Widrow-Hoff rule [9]

$$\Delta w_i=\eta[y_{\text{desired}}-y(\mathbf{w},\vec{x})][1-y^2(\mathbf{w},\vec{x})]x_i\ .$$

The desired output y_{desired} depends on the particular input. There are two classes of inputs, one corresponding to $y_{\text{desired}}=0.9$, the other one to $y_{\text{desired}}=-0.9$. Inputs belonging to the first class are Gaussian distributed around the center point $(\sqrt{2}\sin\phi,\sqrt{2}\cos\phi)$ and those belonging to the second class around $(-\sqrt{2}\sin\phi,-\sqrt{2}\cos\phi)$

$$\rho(x_1,x_2,y_{\text{desired}})=\tfrac{1}{2}\sum_{+,-}\frac{1}{2\pi\sigma^2}\exp\left\{\frac{(x_1\pm\sqrt{2}\sin\phi)^2+(x_2\pm\sqrt{2}\cos\phi)^2}{2\sigma^2}\right\}\delta(y_{\text{desired}}\pm 0.9)\ .$$

ϕ is the angle between the line joining the two center points and the x_1 axis.

In the optimal situation, the weights and threshold of the network correspond to a decision boundary going through the origin perpendicular to the line joining the two center points. In other words, the attractive fixed-point solution $\mathbf{w}^*$ of Eq. (2) corresponds to a decision boundary that is described by the line

$$x_2=-x_1\tan\phi\ .$$

By changing the center points, i.e., $\phi(t)$, as a function of time, learning in a changing environment can be modeled. During the learning process, the algorithm described in Sec. III takes care of the recalibration of the learning parameter. Figures 1–3 show snapshots of the learning system in the three situations studied in the previous section. The bold line in each of these figures is the decision boundary at that time. Inputs belonging to positive outputs are indicated by diamonds, those belonging to negative outputs by crosses. The last 100 training patterns are shown. In the graphs on the right, the learning parameter, the squared bias, and the variance are plotted as a function of time, all calculated from the statistics of the variable W. Unless stated otherwise, simulations start with random weights, an initial learning parameter $\eta=0.1$, $\sigma=1$, $\phi(0)=\pi/4$, and a constant time window $T=500$ learning steps.

In the first simulation (Fig. 1), the environment is fixed, that is, the probability of drawing a combination $(x_1,x_2,y_{\text{desired}})$ is time independent. Since we start with a relatively large learning parameter, the decision boundary approaches its optimal position quickly. The algorithm decreases the learning parameter to reduce the fluctuations. The final result is a correct, constant position of the decision boundary. The asymptotic learning parameter is very small.

The second simulation (Fig. 2) serves to illustrate the arousal mechanism. We start with the network and the learning parameter in the situation of the first simulation at time $t=5000$, after ten updates of the learning parameter. Now the center points of the two distributions are suddenly displaced from (1,1) and (−1,−1) to (1,−1) and (−1,1), and thus ϕ changes from $\pi/4$ to $-\pi/4$. The algorithm reacts to this change by raising the learning parameter. As explained in Sec. IV, it takes some time before the learning parameter reaches its maximum. Of

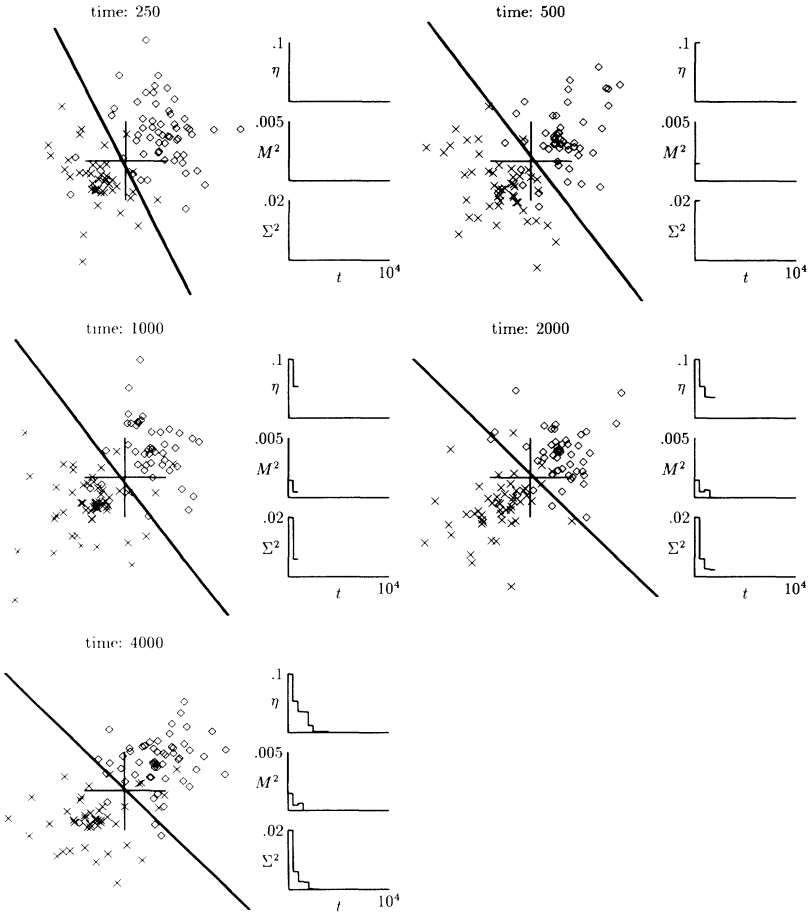

FIG. 1. A fixed environment: $\phi(t)=\phi(0)=\pi/4$. The bold line is the decision boundary found by the network. The last 100 training patterns are shown; diamonds represent positive outputs, crosses represent negative outputs. Graphs on the right give the learning parameter, squared bias, and variance as a function of time, all calculated from the statistics of the weights. Variance of input examples $\sigma=1$ and time window $T=500$.

course, after a while the learning parameter is turned down again.

To study the behavior of the algorithm in a continuously changing environment (Fig. 3), we rotate the center points $\phi(t)=\omega t$ with $\omega=2\pi/1000$. Because of the large fluctuations, little can be said about the convergence to a stable asymptotic solution. Nevertheless, as can be seen from the pictures, the overall performance is acceptable.

Besides the presence of the fluctuations, which were neglected in the preceding section, there seems to be a good qualitative correspondence between the calculated behavior of the algorithm and the features seen in the three simulations.

VI. DISCUSSION

In this paper we have derived an algorithm for on-line adaptation of the learning parameter. The algorithm optimizes an error criterion with respect to the learning parameter. The elegance and usefulness of this algorithm can be summarized in the following notions

(i) The algorithm is completely general. It can be used for all learning rules that can be written in the form (1). In addition, the algorithm operates well in any arbitrary environment, i.e., fixed or changing.

(ii) The algorithm has a theoretical basis that enables us to describe its qualitative behavior. Calculations predict algebraic decay of the learning parameter in a fixed environment, arousal detection in case of a sudden change, and exponential convergence in a gradually changing environment.

(iii) The algorithm requires only a small amount of extra memory and computation power. One has to keep track of four extra variables to calculate the averages $\langle W\rangle_T$, $\langle \Delta W\rangle_T$, $\langle W^2\rangle_T$, and $\langle(\Delta W)^2\rangle_T$. The new learning parameter follows directly from these averages.

Most of the algorithms for adaptation of the learning parameter are designed specifically for one learning rule (see [11] for a short review on learning-parameter adjustment for backpropagation), and only for operation in a fixed environment. Since the adaptation rules for the learning parameter are usually just posed instead of derived, their usefulness can only be judged from experimental evidence in these specific cases. The lack of a theoretical basis makes it difficult to understand why they work and how they can be generalized to other situations. The "learning of the learning rule" for adaptive pattern classifiers in a fixed environment [12] shows a behavior similar to the algorithm we derived. Far from the optimal solution the learning parameter is increased to accelerate the convergence; near the optimal solution the learning parameter is decreased to obtain a greater accuracy. However, since this algorithm is not derived from an error criterion like Eq. (3), it is not clear how this algorithm behaves in a changing environment.

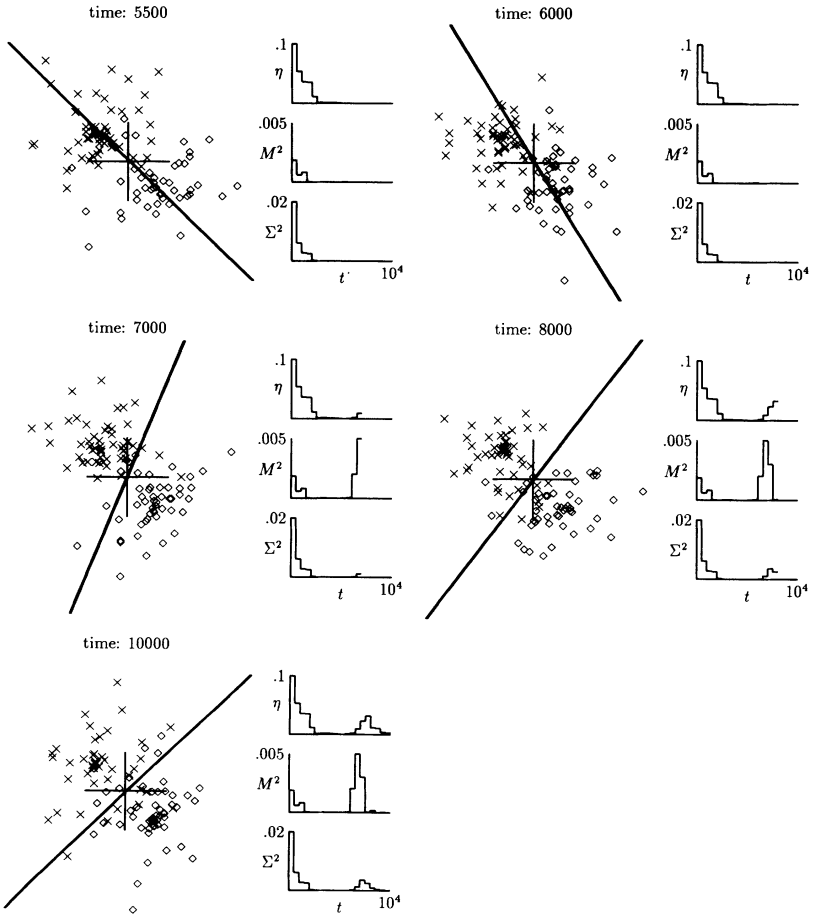

FIG. 2. A sudden change in the environment: $\phi(t)$ changes abruptly from $\pi/4$ to $-\pi/4$ at $t=5000$. See Fig. 1 for further explanation.

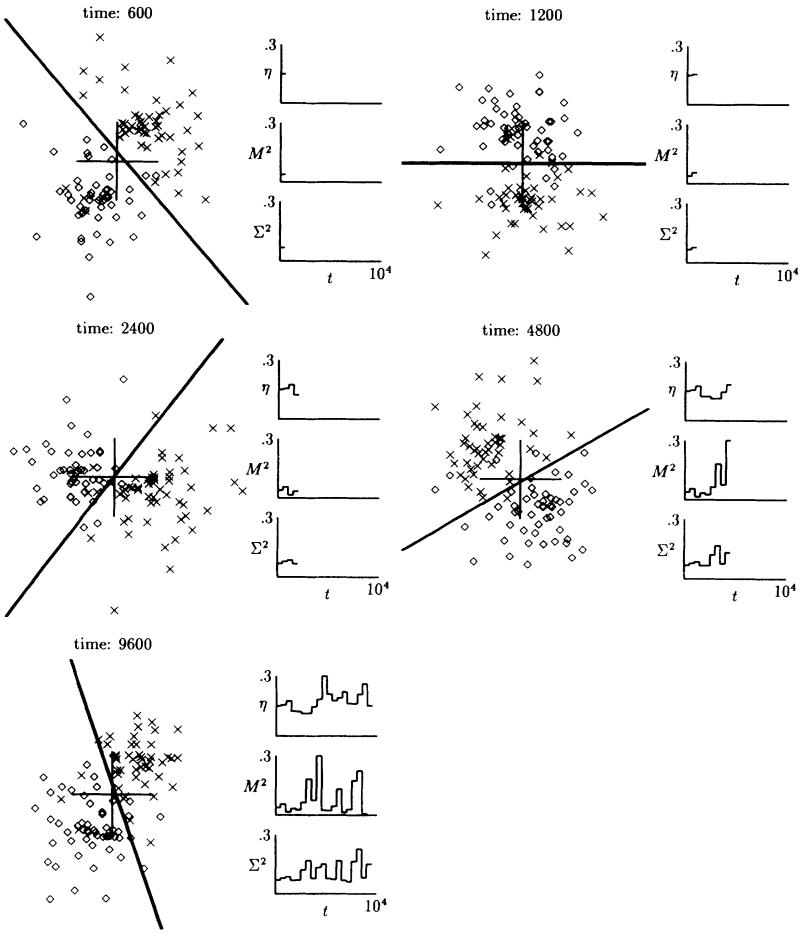

FIG. 3. A continuously changing environment: $\phi(t)=\pi/4+2\pi t/1000$. See Fig. 1 for further explanation.

We conclude with a discussion on a few limitations of our algorithm.

(i) The algorithm is not local. That is, the statistics of all the weights is needed. However, if one agrees upon having *one* learning parameter, this immediately implies that one needs global information, in this case the statistics of the global variable W, to update this learning parameter. It is possible to use this algorithm for each weight separately, each weight having its own learning parameter, but this requires much more memory (four extra variables for each weight) and far more computations.

(ii) The algorithm has no obvious physiological equivalent. We do not know an explicit physiologically plausible model or scheme for this algorithm. Nevertheless, the qualitative features of the algorithm, i.e., turning down the learning parameter if there is no new information, raising the learning parameter in case of a sudden change ("arousal detection"), and continuous learning in a constantly changing environment, seem very natural from a psychological and a biological point of view.

(iii) There is no guarantee that a learning algorithm yields the best solution in a global sense. This is the case with or without learning-parameter adaptation and is a general problem of learning rules. However, dynamic adaptation of the learning parameter might help to escape local minima. To derive a learning-parameter adaptation algorithm that can distinguish between global and local minima, a better understanding of the effect of the learning parameter on global optimization for learning rules of the form (1) is necessary.

ACKNOWLEDGMENT

This work was partly supported by the Dutch Foundation for Neural Networks.

APPENDIX

In this Appendix, we derive an expression for the bias M in terms of averages that can be calculated on-line. The change in the bias M and the variance Σ^2 follow directly from the working hypothesis (7)

$$\begin{aligned}\Delta M &= \langle \Delta W \rangle_\Xi - \Delta W^* = (\Delta W)_\Xi - \nu \ , \\ \Delta\Sigma^2 &= \langle (\Delta W)^2 \rangle_\Xi - \langle \Delta W \rangle_\Xi^2 + 2 < (W - \langle W \rangle_\Xi)\Delta W >_\Xi \ .\end{aligned} \tag{A1}$$

The various expectation values can be calculated using Eq. (7):

$$\begin{aligned}\langle \Delta W \rangle_\Xi &= -\eta\lambda M \ , \\ \langle (\Delta W)^2 \rangle_\Xi - \langle \Delta W \rangle_\Xi^2 &= \eta^2\lambda\Sigma^2 + \eta^2\chi^2 \ , \\ < (W - \langle W \rangle_\Xi)\Delta W >_\Xi &= -\eta\lambda\Sigma^2 \ ,\end{aligned} \tag{A2}$$

yielding

$$\Delta M = -\eta\lambda M - \nu \ ,$$
$$\Delta\Sigma^2 = -\eta\lambda(2-\eta\lambda)\Sigma^2 + \eta^2\chi^2 \ . \tag{A3}$$

These are the evolution equations of the bias and the variance for constant η, λ, ν, and χ^2.

For a stationary process, i.e., $\Delta\Sigma^2=0$, we obtain from Eq. (A1) and (A2) the relation

$$\langle(\Delta W)^2\rangle_\Xi - \langle\Delta W\rangle_\Xi^2 = 2\eta\lambda\Sigma^2 \ .$$

This can be used to eliminate λ in Eq. (A2), yielding

$$M = -\frac{2\langle\Delta W\rangle_\Xi\Sigma^2}{\langle(\Delta W)^2\rangle_\Xi - \langle\Delta W\rangle_\Xi^2} \ . \tag{A4}$$

[1] T. Heskes and B. Kappen, Phys. Rev. A **44**, 2718 (1991).

[2] D. Rumelhart, G. Hinton, and R. Williams, Nature **323**, 533 (1986).

[3] T. Kohonen, Biol. Cybernetics **43**, 59 (1982).

[4] H. Ritter and K. Schulten, Bio. Cybernetics **60**, 59 (1988).

[5] D. Hebb, *The Organization of Behaviour* (Wiley, New York, 1949).

[6] J. Hopfield, Proc. Nat. Acad. **79**, 2554 (1982).

[7] L. Ljung, IEEE Trans. Autom. Control **AC-22**, 551 (1977).

[8] H. Kushner and D. Clark, *Stochastic Approximation Methods for Constrained and Unconstrained Systems* (Springer, New York, 1978).

[9] B. Widrow and M. Hoff, in *Institute of Radio Engineers Western Electronic Show and Convention,* Convention Record (IRE, New York, 1960), Part 4, pp. 96–104.

[10] D. Clark and K. Ravishankar, Neural Networks **3**, 87 (1990).

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

[12] S. Amari, IEEE Trans. Electron. Comput. **16**, 299 (1967).