PERGAMON

Neural Networks 12 (1999) 43–58

Neural Networks

Contributed Article

Dynamics of multilayer networks in the vicinity of temporary minima

Nikolaos Ampazis[a], S.J. Perantonis[a], J.G. Taylor[b,*]

[a]*Institute of Informatics and Telecommunications, National Research Center 'Demokritos', Athens, Greece*
[b]*Department of Mathematics, King's College London, London, UK*

Received 27 November 1997; accepted 27 April 1998

Abstract

A dynamical system model is derived for a single-output, two-layer neural network, which learns according to the back-propagation algorithm. Particular emphasis is placed on the analysis of the occurrence of temporary minima. The Jacobian matrix of the system is derived, whose eigenvalues characterize the evolution of learning. Temporary minima correspond to critical points of the phase plane trajectories, and the bifurcation of the Jacobian matrix eigenvalues signifies their abandonment. Following this analysis, we show that the employment of constrained optimization methods can decrease the time spent in the vicinity of this type of minima. A number of numerical results illustrates the analytical conclusions.

Keywords: Feed-forward neural networks; Supervised learning; Back-propagation; Temporary minima; Dynamical systems; Jacobian matrix; Eigenvalues; Constrained optimization

1. Introduction

Most of the analysis of the back-propagation algorithm for the training of artificial neural networks, has examined the dynamical behavior of a single-layer, feed-forward neural network (Minsky and Papert, 1988; Sontag and Sussmann, 1991) and soft-committee machines (Biehl and Schwarze, 1995; Saad and Solla, 1995a; Saad and Solla, 1995b). There is, however, little analysis of the learning behaviour or dynamics during training of a multilayer neural network, with sigmoidal activation functions for all its nodes and with no restrictions imposed on any of its weights. Indeed, the convergence properties and the encounter of undesired minima of the back-propagation algorithm are generally derived by simulations, because mathematical analysis of the dynamics of non-linear systems such as a multilayer neural network is very complicated. To the best of our knowledge, in only one paper, a mathematical analysis of the dynamics of a feed-forward multilayer network has been published (Guo and Gelfand, 1991), in which a variation of the describing function method (Graham and McRuer, 1961; Gelb and Vander Velde, 1968) is applied in order to derive a simplified, non-linear, deterministic system. There has also been a number of publications that provide analytic solutions for the identification of local minima in specific problems (Lisboa and Perantonis, 1991; Sprinkhuizen-Kuyper and Boers, 1996), as well as general techniques for avoiding such minima (Gorse et al., 1993, 1997). Related work in the field includes the mathematical analysis of the phase transitions of learning, and of the encounter of temporary minima using geometrical approaches such as a vector decomposition method [Annema, J., Hoen, K., & Wallinga, H. (1994). Unpublished]. Still, however, a more detailed analysis of the dynamical behaviour of multilayer neural networks is needed, particularly one that can contribute to the understanding of the fundamental principles involved in learning, such as the occurrence of undesired minima.

It has already been shown [Annema, J., Hoen, K., & Wallinga, H. (1994). Unpublished] that temporary minima result from the development of internal symmetries and from the subsequent building of redundancy in the hidden layer. These types of minima are the most troublesome, because they correspond to almost flat plateaus of the cost function landscape. If the back-propagation system gets stuck during training on such a plateau, it usually takes a very long time to find its way down the cost function surface. Eventually the network may be able to escape, but performance improvement in these minima drops to a very low, but non-zero level, because of the very low gradient of the cost function. A temporary minimum can be recognized in the mean square error (MSE) versus epoch

* Corresponding author. E-mail: john.g.taylor@kcl.ac.uk

0893–6080/99/$ -see front matter
PII: S0893-6080(98)00103-8

curve, as an approximately flat part in which the MSE is virtually constant for a long training time after some initial learning. After a generally large number of epochs, this part in the energy landscape is abandoned, resulting in a significant and sudden drop in the MSE curve (Woods, 1988; Murray, 1991).

For the explanation of the dynamical behaviour of back-propagation, and in particular of the fundamental processes behind the occurrence of temporary minima, we propose that the derivation of a dynamical system model may provide a valuable insight into these important issues. In this paper, we are considering a two-layer network trained with the back-propagation algorithm. The network has an arbitrary number of input units, two units in the hidden layer and one output unit. Motivated by the connection between temporary minima and the build-up of redundancy, we introduce suitable state variables and linearization conditions, and we derive a linear dynamical system model which describes the dynamics of the back-propagation system in the vicinity of temporary minima. Using this model we study specific training tasks and find that the learning behaviour of the neural network can be explained as follows: temporary minima correspond to the phases during which the network remains in the vicinity of critical points of the phase plane trajectories, which are actually saddle points. These points, however, correspond to non-optimal solutions of the training problem. In these phases, the network is unable to move away from the critical points, because the largest eigenvalue of the Jacobian matrix of the linearized system is very small and, therefore, the system evolves very slowly. However, as training continues, small perturbations applied in the coefficients of the system are reflected in small perturbations in the eigenvalues, causing them eventually to bifurcate. At this point, the largest eigenvalue becomes large enough in order to allow the system to evolve at a much faster rate, so that the network rapidly abandons the minimum and the MSE curve suddenly drops to a significantly lower level.

This dynamical system model analysis allows us to speed up learning by minimizing the training time spent in the vicinity of temporary minima. To this end, we show that we can use constrained optimization methods that achieve simultaneous minimization of the cost function and maximization of the largest eigenvalue of the Jacobian matrix, in order to allow the system to evolve much faster. The learning speed is greatly improved, since the network avoids being trapped in a temporary minimum and, hence, total training time is significantly decreased.

Using our approach we study two classification problems, namely the XOR and the unit square problem. For the XOR problem in particular, we present a detailed analytical and experimental study of the network's behaviour in the vicinity of the temporary minimum.

The paper is organized as follows: in Section 2, we draw attention to temporary minima and provide an overview of the phase transitions during learning, which are responsible for their occurrence. Also, the fundamental processes behind the occurrence of temporary minima are explained mathematically in terms of the dynamical systems theory, and the Jacobian matrix of the system is derived. In Section 3, we introduce the constrained optimization method designed to facilitate learning using the constraints imposed on the eigenvalues of the Jacobian matrix. In Section 4, the theoretical results are applied to simple classification problems, and a comparative study of our constrained learning approach to a number of well-known training algorithms is presented. Finally, in Section 5, conclusions are drawn and future work is outlined.

2. The dynamical model

2.1. Motivation for the model

2.1.1. Derivation of the fundamental differential equations

Let us first consider the two-layer neural network shown in Fig. 1, which has N external input signals and one bias input. The bias signal is identical for all neurons in the network. The hidden layer consists of two neurons, and the output layer contains one neuron with sigmoid activation functions $f(s) = 1/[1 + \exp(-s)]$.

Given P training patterns indexed by p, the batch back-propagation weight update rule for the hidden-to-output connections gives:

$$\Delta w_i^{\mathrm{h}} = \eta \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) y_i^{\mathrm{h}(p)}, \quad i = 0, 1, 2, \qquad (1)$$

where η is the learning rate, w_i^{h} are the weight connections between each hidden node i and the output node, $y^{(\mathrm{p})}$ is the output of the network, $d^{(\mathrm{p})}$ is the desired response, and $y_i^{\mathrm{h}(p)}$ are the outputs of each hidden unit (with $y_0^{\mathrm{h}(p)}$ corresponding to the bias signal). When η is small, the difference equations can be approximated by differential equations in time, with

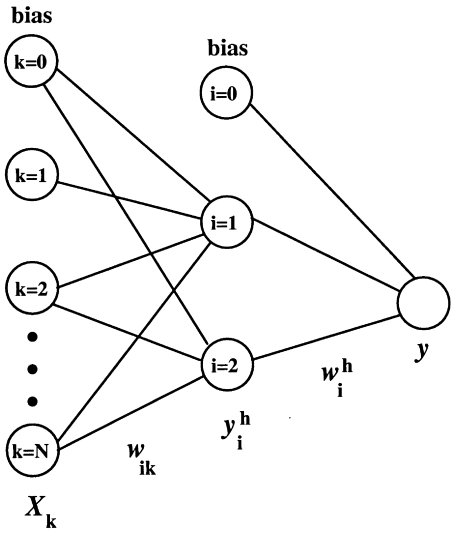

Fig. 1. A feed-forward network with two hidden nodes.

the quantities $\Delta w_i^{\mathrm{h}}/\eta$ representing the rate of change $\dot{w}_i^{\mathrm{h}} = \mathrm{d}w_i^{\mathrm{h}}/\mathrm{d}t$ so that

$$\dot{w}_i^h = \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) y_i^{h(p)}. \tag{2}$$

The customary practice with the back-propagation algorithm is to set all the free parameters of the network to random numbers that are uniformly distributed inside a small range of values. In this way, all units operate in their linear regions in the early stages of learning and premature saturation (Lee et al., 1991), if the sigmoid activation functions are avoided.

From Eq. (2) we can observe that only the last factor $y_i^{\mathrm{h}(p)}$ is different for the update of the different w_i^{h}. However, under the condition of very small initial weights, the responses $y_i^{\mathrm{h}(p)}$ of the neurons in the first layer, are approximately identical, and, in particular, $y_i^{\mathrm{h}(p)} \approx f(0)$. Therefore, the weights corresponding to the connections from the hidden to the output layer, will adapt identically after the presentation of all the training examples.

For the input–hidden connections, the back-propagation weight adaptations are:

$$\dot{w}_{ik} = \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) w_i^{\mathrm{h}} f'(x_i^{\mathrm{h}(p)}) x_k^{(p)}, \tag{3}$$

where w_{ik} represents the weight connection between hidden node i and the input node k, $x_i^{\mathrm{h}(p)}$ is the net input fed to the hidden node i, and $x_k^{(p)}$ is the signal from the input node k.

Written in vector notation, Eq. (3) becomes:

$$\dot{\boldsymbol{w}}_i = \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) w_i^{\mathrm{h}} f'(x_i^{\mathrm{h}(p)}) \boldsymbol{x}^{(p)}, \tag{4}$$

where

$$\boldsymbol{w}_i = (w_{i0} \ldots w_{\in})^T \tag{5}$$

is the weight vector of each hidden neuron i.

Again, we can observe that the first three factors into the summation on the right-hand side of Eq. (4) are identical for all neurons in the first layer. In addition, as the weights w_i^{h} adapt identically after the presentation of all the training examples, the weights w_i^{h} are, by good approximation, identical. The factor $f'(x_i^{\mathrm{h}}) \approx f'(0)$ is also identical for all neurons, assuming very small initial weights. Finally, the input vector for the hidden layer $\boldsymbol{x}^{(\mathrm{p})}$ is fed to all neurons in this layer and is hence, identical for all neurons. Therefore, in the beginning of the training and under the condition of very small initial weights, the weight vectors of the hidden neurons also adapt approximately identically. Thus, the weight vectors of the units in the hidden layer move towards approximately identical positions in input space (note, however, that the weight vectors components will not neccessarily be identical).

Provided that training is successful, the weight vectors of the neurons in a feed-forward network eventually converge towards specific attractors in weight space (Parker, 1985; Guo and Gelfand, 1991). However, as has been shown analytically [Annema, J., Hoen, K., & Wallinga, H. (1994). Unpublished], the network surpasses several phases before it actually converges to a solution. In each of these different phases, the weight vector attractors in each phase will generally be different from the previous attractor positions. The hyperplane corresponding to the weight vector attractor will be denoted the attractor hyperplane.

From the analysis of Eq. (4), it follows that during the early stages of learning the attractor hyperplanes of the two neurons in the hidden layer are coinciding, and, therefore, these neurons make approximately identical classifications. In the beginning of training this common attractor, in which the two hyperplanes coincide, is the hyperplane that seems to provide the best possible linear discrimination between the different classes. The position of this attractor in weight space is independent of the order in which the training examples are presented, because the batch mode of training is used. However, the actual attractor depends on the specific problem and on the initial weights. It is a result of the coincidence of the attractor hyperplanes of the hidden units that the network builds up redundancy. This building of redundancy is an inherent property of back-propagation networks due to the nature of the back-propagation weight update rule. Therefore, after an initial error reduction that corresponds to the transition of the hidden weight vectors towards their common attractor hyperplane, the network is encountering a temporary minimum. This temporary minimum corresponds to the stage of the network where the two hidden weight vectors coincide completely with their attractor hyperplane. After this phase, the hidden layer is approximately reducible to one unit (Sussmann, 1992), and the weight vectors of the units in the hidden layer are approximately identical. As learning continues because of the continuous (but very small) change of all the weights, eventually the units in the hidden layer start to change to completely different attractors. In this phase the input space is subdivided, or partitioned (Liang, 1991), into two parts each classified by one neuron in the hidden layer. The vanishing of redundancy results in escaping from the temporary minimum (Murray, 1991), and the network finds its way down the cost function landscape. However, abolishing the redundancy is generally a slow process and neural networks usually ‘stick’ in a temporary minimum during training, for a relatively long time.

In terms of dynamical systems theory, the learning phases can be represented as points in the phase plane trajectories of the dynamics of the network. Therefore, it is essential to find the critical points of the differential equations governing the behaviour of the back-propagation system and to assess their type of stability. Now, for the case of a neural network trained with the back-propagation algorithm, it should be clear that it is virtually impossible to analyse exactly its dynamical behaviour, since it is a very complex dynamical system. Indeed, there is no general prescription

for finding tractable expressions for equilibrium sets and trajectories. However, if we succeed in introducing suitable state variables that map the temporary minimum to the origin of the phase plane, we can study the dynamical behaviour of the system in the vicinity of the temporary minima by linearizing the system around the origin. This is so because it is well known that non-linear systems of the form:

$$\dot{x}_i = F_i(\boldsymbol{x}),\ i = 1, \ldots, M, \tag{6}$$

can be linearized by considering orbits of the system close to the origin, assuming that $F_i(\mathbf{x})$ are twice differentiable (Coddington and Levinson, 1955; Boyce and DiPrima, 1986), then the behaviour of the system may, in general, be approximated locally by that of the linearized system:

$$\dot{\boldsymbol{x}} = \boldsymbol{J}\boldsymbol{x}, \tag{7}$$

where $\boldsymbol{J}$ is the Jacobian matrix whose elements are given by:

$$J_{ij} = \frac{\partial F_i(\boldsymbol{x})}{\partial x_j}. \tag{8}$$

The eigenvalues of the system are the roots of the polynomial equation

$$\det(\boldsymbol{J} - \lambda \boldsymbol{I}) = 0 \tag{9}$$

and assuming that the Jacobian matrix is non-singular, i.e. $det(\boldsymbol{J} - \lambda \boldsymbol{I}) \neq 0$, it follows that $\boldsymbol{x} = 0$ is the only solution of the equation $\boldsymbol{J}\boldsymbol{x} = 0$. Consequently, $\boldsymbol{x} = 0$ is the only critical point of the system of Eq. (7). The behaviour of the trajectories of the linear system is, therefore, completely determined by the nature of the eigenvalues of $\boldsymbol{J}$.

Our earlier analysis indicates that in the early stages of learning, temporary minima occur, because of the building of redundancy. Therefore, appropriate state variables that map the temporary minimum to the origin of the phase plane, are expected to be the difference between the weight vectors of the hidden units and the difference between the two non-bias weights of the hidden–output connections. Hence, with this selection of state variables for the back-propagation system, we should be able to derive a set of differential equations of the form of Eq. (6), and to classify the type of stability of the critical point by calculating the eigenvalues of its Jacobian matrix. This means that we should be able to determine whether the system encounters temporary minima, because the solutions approach or remain in the vicinity of the critical point and hence, to derive the conditions under which they would change their character, by moving away from it and eventually allowing the system to escape from the temporary minimum. In fact, when we look into specific examples, we will show that the escape from a temporary minimum corresponds to a bifurcation of the eigenvalues of the Jacobian matrix, as a result of the continuous perturbation of the coefficients of the system.

2.2. Derivation of the dynamical model

For the state variables mentioned in the previous subsection, the analysis can proceed as follows.

For the network of Fig. 1 let us define:

$$2\boldsymbol{\epsilon} = \boldsymbol{w}_1 - \boldsymbol{w}_2,\ 2\boldsymbol{\omega} = \boldsymbol{w}_1 + \boldsymbol{w}_2, \tag{10}$$

with w_i defined in Eq. (5). It follows that

$$2\dot{\boldsymbol{\epsilon}} = \dot{\boldsymbol{w}}_1 - \dot{\boldsymbol{w}}_2,\ 2\dot{\boldsymbol{\omega}} = \dot{\boldsymbol{w}}_1 + \dot{\boldsymbol{w}}_2. \tag{11}$$

Similarly we define

$$2\mu = w_1^{\mathrm{h}} - w_2^{\mathrm{h}},\ 2\nu = w_1^{\mathrm{h}} + w_2^{\mathrm{h}}. \tag{12}$$

It also follows that

$$2\dot{\mu} = \dot{w}_1^{\mathrm{h}} - \dot{w}_2^{\mathrm{h}},\ 2\dot{\nu} = \dot{w}_1^{\mathrm{h}} + \dot{w}_2^{\mathrm{h}}. \tag{13}$$

Hoevever, $\dot{w}_1^{\mathrm{h}}$ and $\dot{w}_2^{\mathrm{h}}$ are given by Eq. (2). Substituting these expressions into the first of Eq. (12) gives:

$$2\dot{\mu} = \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) [y_1^{\mathrm{h}(p)} - y_2^{\mathrm{h}(p)}]. \tag{14}$$

Now, $y_1^{\mathrm{h}(p)}$ and $y_2^{\mathrm{h}(p)}$ are given by:

$$y_1^{\mathrm{h}(p)} = f(x_1^{\mathrm{h}(p)}) = f\Big(\sum_k w_{1k} x_k^{(p)}\Big) = f(\boldsymbol{w}_1 \cdot \boldsymbol{x}^{(p)}) \tag{15}$$

and

$$y_2^{\mathrm{h}(p)} = f(x_2^{\mathrm{h}(p)}) = f\Big(\sum_k w_{2k} x_k^{(p)}\Big) = f(\boldsymbol{w}_2 \cdot \boldsymbol{x}^{(p)}). \tag{16}$$

Thus:

$$2\dot{\mu} = \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) [f(\boldsymbol{w}_1 \cdot \boldsymbol{x}^{(p)}) - f(\boldsymbol{w}_2 \cdot \boldsymbol{x}^{(p)})]. \tag{17}$$

However, from Eq. (10) it follows that

$$\boldsymbol{w}_1 = \boldsymbol{\omega} + \boldsymbol{\epsilon},\ \boldsymbol{w}_2 = \boldsymbol{\omega} - \boldsymbol{\epsilon}. \tag{18}$$

Substituting the above equations into Eq. (17) gives:

$$2\dot{\mu} = \sum p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) \{f[(\boldsymbol{\omega} + \boldsymbol{\epsilon}) \cdot \boldsymbol{x}^{(p)}] - f[(\boldsymbol{\omega} - \boldsymbol{\epsilon}) \cdot \boldsymbol{x}^{(p)}]\}. \tag{19}$$

Similarly, for $\dot{\epsilon}$ from the first of Eqs. (11) and (4), we have:

$$2\dot{\epsilon} = \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) \boldsymbol{x}^{(p)} [w_1^{\mathrm{h}} y_1^{\mathrm{h}(p)} (1 - y_1^{\mathrm{h}(p)}) - w_2^{\mathrm{h}} y_2^{\mathrm{h}(p)} (1 - y_2^{\mathrm{h}(p)})]. \tag{20}$$

However, from Eq. (12) it follows that:

$$w_1^{\mathrm{h}} = \nu + \mu,\ w_2^{\mathrm{h}} = \nu - \mu. \tag{21}$$

Substituting Eqs. (21) and (18) into Eq. (20), we obtain:

$$2\dot{\epsilon} = \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) \boldsymbol{x}^{(p)} \{(\nu + \mu) f\left[(\boldsymbol{\omega} + \boldsymbol{\epsilon}) \cdot \boldsymbol{x}^{(p)}\right] \times \left[1 - f\left[(\boldsymbol{\omega} + \boldsymbol{\epsilon}) \cdot \boldsymbol{x}^{(p)}\right]\right] (\nu - \mu) f\left[(\boldsymbol{\omega} - \boldsymbol{\epsilon}) \cdot \boldsymbol{x}^{(p)}\right] \times \left[1 - f\left[(\boldsymbol{\omega} - \boldsymbol{\epsilon}) \cdot \boldsymbol{x}^{(p)}\right]\right]\}. \tag{22}$$

Eqs. (22) and (19) are of the form:

$$\dot{\epsilon} = \boldsymbol{F}(\epsilon, \mu), \;\; \dot{\mu} = G(\epsilon, \mu), \tag{23}$$

and it is clear that if we set $\epsilon = 0$ and $\mu = 0$, then $\boldsymbol{F}(\epsilon,\mu) = 0$ and $G(\epsilon,\mu) = 0$, and, therefore, the origin is a critical point of the system. Moreover, $\boldsymbol{F}(\epsilon,\mu)$ and $G(\epsilon,\mu)$ are twice-differentiable and we can, therefore, proceed with the linearization of the system.

In Eq. (19), for the terms $f[(\omega \pm \epsilon)\cdot\boldsymbol{x}^{(p)}]$, we can use Taylor's theorem for the approximation of f close to the point $\omega\cdot\boldsymbol{x}^{(p)}$. Then, keeping only first-order terms, we obtain:

$$f[(\omega \pm \epsilon)\cdot\boldsymbol{x}^{(p)}] = f(\omega\cdot\boldsymbol{x}^{(p)}) \pm \epsilon\cdot\boldsymbol{x}^{(p)} f'(\omega\cdot\boldsymbol{x}^{(p)}). \tag{24}$$

Now, since the activation function is the sigmoidfunction, its derivative is given by:

$$f'(\omega\cdot\boldsymbol{x}^{(p)}) = f(\omega\cdot\boldsymbol{x}^{(p)})[1 - f(\omega\cdot\boldsymbol{x}^{(p)})]. \tag{25}$$

Thus:

$$f[(\omega \pm \epsilon)\cdot\boldsymbol{x}^{(p)}] = f(\omega\cdot\boldsymbol{x}^{(p)}) \pm \epsilon\cdot\boldsymbol{x}^{(p)} f(\omega\cdot\boldsymbol{x}^{(p)})[1 - f(\omega\cdot\boldsymbol{x}^{(p)})]. \tag{26}$$

Substituting this result into Eq. (19) finally gives the following expression for $\dot{\mu}$:

$$\dot{\mu} = \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) \epsilon\cdot\boldsymbol{x}^{(p)} f(\omega\cdot\boldsymbol{x}^{(p)})[1 - f(\omega\cdot\boldsymbol{x}^{(p)})]. \tag{27}$$

Similarly, for Eq. (22), using Taylor's approximation for $f[(\omega \pm \epsilon)\cdot\boldsymbol{x}^{(p)}]$, and ignoring all terms quadratic in ϵ, gives:

$$\begin{aligned} 2\dot{\epsilon} = \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) \boldsymbol{x}^{(p)} \cdot \{ & (\nu + \mu) \\ & \times \big[f(\omega\cdot\boldsymbol{x}^{(p)})[1 - f(\omega\cdot\boldsymbol{x}^{(p)})] \\ & + \epsilon\cdot\boldsymbol{x}^{(p)} f'(\omega\cdot\boldsymbol{x}^{(p)})[1 - 2f(\omega\cdot\boldsymbol{x}^{(p)})] \big] \\ & - (\nu - \mu) \big[f(\omega\cdot\boldsymbol{x}^{(p)})[1 - f(\omega\cdot\boldsymbol{x}^{(p)})] \\ & - \epsilon\cdot\boldsymbol{x}^{(p)} f'(\omega\cdot\boldsymbol{x}^{(p)})[1 - 2f(\omega\cdot\boldsymbol{x}^{(p)})] \big] \}. \end{aligned} \tag{28}$$

Collecting all common factors gives:

$$\begin{aligned} \dot{\epsilon} = \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) x^{(p)} & \\ \cdot \{ & f(\omega\cdot x^{(p)})[1 - f(\omega\cdot x^{(p)})]\mu \\ & + \epsilon\cdot x^{(p)} f'(\omega\cdot x^{(p)})[1 - 2f(\omega\cdot x^{(p)})]\nu \}. \end{aligned} \tag{29}$$

Finally, using Eq. (25) and collecting all common factors, the expression for $\dot{\epsilon}$ reduces to:

$$\begin{aligned} \dot{\epsilon} = \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) f(\omega\cdot x^{(p)}) & \\ \times [1 - f(\omega\cdot x^{(p)})] x^{(p)} \cdot \{ & \mu + \epsilon\cdot x^{(p)} [1 - 2f(\omega\cdot x^{(p)})]\nu \}. \end{aligned} \tag{30}$$

Eqs. (27) and (30) are the fundamental differential equations describing the dynamics of the back-propagation system in terms of the two state variables ϵ and μ.

From Eqs. (27) and (30) it follows that the Jacobian matrix of the system is given by:

$$\begin{aligned} J = \sum_p (d^{(p)} - y^{(p)}) y^{(p)} (1 - y^{(p)}) f(\omega\cdot x^{(p)})[1 - f(\omega\cdot x^{(p)})]. & \\ \times \Big([1 - 2f(\omega\cdot x^{(p)})] x^{(p)} x^{(p)T} \nu x^{(p)} x^{(p)T} 0 \Big). & \end{aligned} \tag{31}$$

Note that the Jacobian matrix depends on the input vectors, as well as on the corresponding desired responses for each one of them. Thus, the nature of the input–output mapping has a direct effect on the dynamics of the back-propagation system. This is, of course, an expected result, since the nature of the learning task determines the shape of the error surface over the weight space. We can also see that the Jacobian matrix is real and symmetric and, therefore, all its eigenvalues are real. In addition, all the corresponding eigenvectors are linearly independent, and if there are no eigenvalue multiplicities they will form an orthogonal set. Since the eigenvalues are real, this means that temporary minima do not correspond to spiral or centre points of the phase plane. As a consequence, small perturbations in the eigenvalues will not affect the stability or instability of the system. Owing to the exponential nature of the solutions, the followed trajectory will depend on the magnitude of the largest of these eigenvalues. If this eigenvalue is small then the trajectory will be in the vicinity of the critical point. However, if all the eigenvalues are small and do not differ too much, then small perturbations due to the continuous update of the weights at each epoch, can cause them eventually to bifurcate. Again, the followed trajectory will depend on the magnitude of the largest positive eigenvalue, but in this case the magnitude of this eigenvalue will be such as to allow the trajectory to move far away from the critical point. Hence, up to the bifurcation point, which does not appear until a sufficient number of these small perturbations, the solutions will not change much, and the network will remain in the vicinity of the temporary minimum for a relatively long time. As soon as this bifurcation occurs then the network is able to rapidly follow the trajectory with the large eigenvalue and, therefore, to abandon the minimum.

3. Constrained optimization method

Following the analysis of the previous section, it is evident that if the maximum eigenvalue of the Jacobian matrix of the system is relatively large, then the network is able to escape from the temporary minimum. Hence, instead of waiting for its growth, the objective of our new approach is to change each free parameter of the network (i.e. weights and thresholds) in order to reach a minimum of the cost function:

$$E = \frac{1}{2} \sum_{p=1}^{P} (d^{(p)} - y^{(p)})^2, \tag{32}$$

with respect to the variables w_{ij}, while simultaneously maximizing a quantity Φ, representing either an approximation or a lower bound of the maximum eigenvalue λ_{max} of the Jacobian matrix, expressed in terms of the network's free parameters.

For problems in which it is impossible to obtain an analytic expression for the maximum eigenvalue in a closed form, in terms of the weights, a strategy that we found to give good results is the following: it is well known from linear algebra that given a real and symmetric matrix $\boldsymbol{J} \in R^{N\times N}$ then:

$$x^T Jx \leq \lambda_{max}\, \boldsymbol{x}^T\boldsymbol{x}\ \forall \boldsymbol{x} \in R^N. \tag{33}$$

We have used the simplest choice for $\boldsymbol{x}$, namely $x=(11\ldots1)^T$, which means that the product on the left-hand side of Eq. (33) is simply the sum of the elements of the matrix. Therefore, for the constrained optimization method we use:

$$\Phi = \boldsymbol{x}^T\boldsymbol{J}\boldsymbol{x}, \tag{34}$$

thus obtaining an analytic expression by directly evaluating the sum of the elements of $\boldsymbol{J}$ as given by Eq. (31). With this selection of Φ, at each epoch of the learning process, we try to move the state of the network away from the origin at a fast rate, thus avoiding the building of redundancy.

Consider the weight vector $\boldsymbol{W}$, whose components are all the network weights and thresholds. We wish to reach a minimum of the cost function of Eq. (32) with respect to $\boldsymbol{W}$, and to simultaneously maximize Φ without compromising the need for a decrease of the cost function. The strategy which we will adopt for the solution of this problem follows the methodology for incorporating additional knowledge in the form of constraints in neural networks learning (Karras and Perantonis, 1995; Perantonis and Karras, 1995).

At each epoch of the learning process, the weight vector $\boldsymbol{W}$ will be incremented by $\mathrm{d}\boldsymbol{W}$, so that:

$$\|\mathrm{d}\boldsymbol{W}\|^2 = (\delta P)^2, \tag{35}$$

where δP is a constant. Thus, at each epoch, the search for an optimum new point in the weight space is restricted to a small hypersphere centred at the point defined by the current weight vector. If δP is small enough, the changes to E and Φ, induced by changes in the weights, can be approximated by the first differentials $\mathrm{d}E$ and $\mathrm{d}\Phi$. At each epoch, we seek to achieve the maximum possible change in $\mathrm{d}\Phi$, so that Eq. (35) is respected, and the change $\mathrm{d}E$ in E is equal to a predetermined quantity $\delta Q < 0$, i.e.

$$\mathrm{d}E = \delta Q. \tag{36}$$

This is a constrained optimization problem which can be solved analytically by introducing two Lagrange multipliers L_1 and L_2 to take account of Eqs. (36) and (35), respectively. We introduce the function ϕ, whose differential is defined as follows:

$$\mathrm{d}\phi = \mathrm{d}\Phi + L_1(\delta Q - \mathrm{d}E) + L_2[(\delta P)^2 - \|\mathrm{d}\boldsymbol{W}\|^2]. \tag{37}$$

On evaluating the differentials involved on the right-hand side, we readily obtain:

$$\mathrm{d}\phi = \boldsymbol{F}\cdot\mathrm{d}\boldsymbol{W} + L_1(\delta Q - \boldsymbol{G}\cdot\mathrm{d}\boldsymbol{W}) + L_2[(\delta P)^2 - \|\mathrm{d}\boldsymbol{W}\|^2], \tag{38}$$

where $\boldsymbol{G}$ and $\boldsymbol{F}$ are given by:

$$\boldsymbol{G} = \partial E/\partial \boldsymbol{W},\ \boldsymbol{F} = \partial\Phi/\partial\boldsymbol{W}. \tag{39}$$

To maximize $\mathrm{d}\phi$ at each epoch, we demand that:

$$\mathrm{d}^2\phi = \left(\boldsymbol{F} - L_1\boldsymbol{G} - 2L_2\mathrm{d}\boldsymbol{W}\right)\cdot\mathrm{d}^2\boldsymbol{W} = 0,$$

$$\mathrm{d}^3\phi = -2L_2\|\mathrm{d}^2\boldsymbol{W}\|^2 < 0. \tag{40}$$

Hence, the factor multiplying $\mathrm{d}^2\boldsymbol{W}$ in Eq. (40) should vanish, and, therefore, we obtain:

$$\mathrm{d}\boldsymbol{W} = -\frac{L_1}{2L_2}\boldsymbol{G} + \frac{1}{2L_2}\boldsymbol{F}. \tag{41}$$

Eq. (41) constitutes the weight update rule for the neural network, provided that L_1 and L_2 can be evaluated in terms of known quantities. This can be carried out as follows: from Eqs. (36) and (41) we obtain:

$$\delta Q = \frac{1}{2L_2}(I_{GF} - L_1 I_{GG}), \tag{42}$$

with I_{GG} and I_{GF} given by:

$$I_{GG} = \|\boldsymbol{G}\|^2,\ I_{GF} = \boldsymbol{G}\cdot\boldsymbol{F}. \tag{43}$$

Eq. (42) can be readily solved for L_1, giving:

$$L_1 = \frac{-2L_2\delta Q + I_{GF}}{I_{GG}}. \tag{44}$$

It remains to evaluate L_2. To this end, we substitute Eq. (41) into Eq. (35) to obtain:

$$4L_2^2(\delta P)^2 = I_{FF} + L_1^2 I_{GG} - 2L_1 I_{GF}, \tag{45}$$

where I_{FF} is given by:

$$I_{FF} = \|\boldsymbol{F}\|^2. \tag{46}$$

Finally, we substitute Eq. (44) into Eq. (45) and solve for L_2, to obtain:

$$L_2 = \frac{1}{2}\left[\frac{I_{GG}(\delta P)^2 - (\delta Q)^2}{I_{FF}I_{GG} - I_{GF}^2}\right]^{-1/2}, \tag{47}$$

where the positive square root value has been chosen for L_2 in order to satisfy the second part of Eq. (40). Note also the bound $|\delta Q| \leq \delta P\sqrt{(I_{GG})}$ set on the value of δQ by Eq. (47). We always use a value $\delta Q = -\xi\delta P\sqrt{(I_{GG})}$, where ξ is a constant between 0 and 1.

Thus, the final weight update rule has only two free parameters, namely δP and ξ. The value chosen for the free parameter ξ determines the contribution of the constraints to the weight update rule. A small value of ξ means that the weight update rule tries to satisfy the constraints without paying too much attention to the minimization of the cost function, while a large value of ξ has the opposite effect. In our simulations (Section 4.1.3 and Section 4.2), the values

recorded for δP and ξ are those giving the best performance. However, similar performances were recorded with $0.1 < \xi < 0.5$ and $0.3 < \delta P < 1.0$, indicating that results are not very sensitive to the exact values of the parameters.

4. Applications

In this section, we derive analytical and experimental results from the application of the dynamical systems analysis of back-propagation, proposed in Section 2, to specific tasks. Since our analysis is applicable only to networks with two hidden nodes and one output node, it is natural to study relatively small-scale problems which facilitate visualization of the main points behind the new approach. We study two learning tasks, namely the XOR and the unit square problem. For the XOR problem in particular, the study presented in the paper introduces new analytical results about the nature of temporary minima, which are then supported by the experimental evidence. We also include an experimental study of the dynamics of back-propagation (with and without momentum) for the specific benchmarks, in order to illustrate main conclusions drawn from the theoretical analysis.

Moreover, we apply the constrained optimization method proposed in Section 3 to the XOR and unit square problems, in order to confirm the validity of our method. In our experiments, we show that it is possible to take advantage of the new dynamical systems approach in order to help a small network escape from temporary minima. In addition, following common practice used in the literature to benchmark new learning algorithms, we include a comparison of the proposed constrained optimization method to back-propagation and more advanced training methods. However, the reader should keep in mind that at this stage of development our method can only be applied to small networks with two hidden nodes, which is, of course, a severe restriction. Consequently, generalization of the method to larger networks is needed before comparisons concerning large-scale benchmarks of more practical interest can become possible. An outline of the method by which this generalization may be achieved is given in Section 5.

4.1. The XOR problem

4.1.1. Evaluation of the Jacobian matrix

For the XOR problem, the bias augmented input vectors to the neural network are:

$$\boldsymbol{x}^{(1)} = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{x}^{(2)} \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}, \quad \boldsymbol{x}^{(3)} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix},$$

$$\boldsymbol{x}^{(4)} = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}, \tag{48}$$

and the desired responses for each input of the input vectors are $d^{(1)} = 0$, $d^{(2)} = 1$, $d^{(3)} = 1$ and $d^{(4)} = 0$. In this case, $\boldsymbol{\omega}$ is a three-component vector: $\boldsymbol{\omega} = (\omega_0\ \omega_1\ \omega_2)$. We assume that the network is initialized using weights in a range $[-q,q]$, where q is a small positive number. Initializing the network with weights of small magnitude, prevents it from approaching certain types of local minima, where some of the weights assume infinite values. On the other hand, when weights of small magnitudes are used, the network is initialized in the vicinity of a temporary minimum, characterized by the equality of outputs corresponding to all training patterns (Lisboa and Perantonis, 1991).

Using our dynamical systems analysis of Section 2, we can study the dynamics of the network while it remains in the vicinity of this stationary point. Until the network finally escapes, the synaptic weights remain small in magnitude. Therefore, it is useful to obtain Taylor expansions of quantities involved in the expression for the Jacobian matrix Eq. (31) in terms of the synaptic weights, and keep only first-order terms. Evidently, to first order, we can write:

$$f(\boldsymbol{\omega}\cdot\boldsymbol{x}^{(p)}) = \frac{1}{2} + \frac{1}{4}\boldsymbol{\omega}\cdot\boldsymbol{x}^{(p)} \tag{49}$$

$$y^{(p)} = f(\boldsymbol{w}^{\mathrm{h}}\cdot y^{\mathrm{h}(p)}) = \frac{1}{2} + \frac{1}{4}\boldsymbol{w}^{\mathrm{h}}\cdot y^{\mathrm{h}(p)} = \frac{1}{2} + \frac{1}{8}(w_1^{\mathrm{h}} + w_2^{\mathrm{h}} - 2w_0^{\mathrm{h}}), \tag{50}$$

where w^{h} is the weight vector of the output neuron and $\boldsymbol{y}^{\mathrm{h}}$ is the input vector to the output neuron (the bias-augmented vector with elements of the outputs $y_i^{\mathrm{h}(p)}$ of the hidden neurons).

Substituting Eqs. (50) and (49) into Eq. (31) and keeping only first-order terms for each element of the Jacobian, we arrive, after some algebra, at the following expression:

$$\boldsymbol{J} = \begin{pmatrix} 0 & \beta & \alpha & -2Z \\ \beta & -\beta & \gamma-\alpha-\beta & Z \\ \alpha & \gamma-\alpha-\beta & -\alpha & Z \\ -2Z & Z & Z & 0 \end{pmatrix}, \tag{51}$$

where

$$\alpha = -\frac{1}{64}\omega_1\nu, \beta = -\frac{1}{64}\omega_2\nu, \gamma = -\frac{1}{64}\omega_0\nu,$$

$$Z = -\frac{1}{64}(w_1^{\mathrm{h}} + w_2^{\mathrm{h}} - 2w_0^{\mathrm{h}}). \tag{52}$$

4.1.2. Evaluation of the Jacobian matrix eigenvalues

The eigenvalues of $\boldsymbol{J}$ can be determined from Eq. (9). The

evaluation of the determinant yields the equation:

$$\lambda^4 + k_1\lambda^3 + k_2\lambda^2 + k_3\lambda + k_4 = 0, \tag{53}$$

where

$$k_1 = \alpha + \beta,\ k_2 = 2\alpha\gamma + 2\beta\gamma - \alpha\beta - 2\alpha^2 - 2\beta^2 - \gamma^2 - 6Z^2,$$
$$k_3 = (\alpha + \beta - 2\gamma)(\alpha\beta + Z^2),\ k_4 = (\alpha + \beta - 2\gamma)^2 Z^2. \tag{54}$$

The left-hand side of Eq. (54) is the characteristic polynomial of the system and its solution gives the eigenvalues of the Jacobian matrix at each iteration of the back- propagation algorithm. We shall show that the Jacobian matrix has two positive and two negative eigenvalues. Using Vieta's relations for the roots λ_i of Eq. (53), we can write

$$k_2 = \frac{1}{2}\sum_{i\neq j}\lambda_i\lambda_j,\ \ k_4 = \prod_i \lambda_i. \tag{55}$$

From Eq. (54) and from the fact that the Jacobian matrix should be non-singular, it follows that $k_4 > 0$. Therefore, the Jacobian matrix may have zero, two, or four positive eigenvalues. However, k_2 can be written as a quadratic form in terms of α,β,γ and Z:

$$k_2 = [\alpha\ \beta\ \gamma\ Z]\,\boldsymbol{A}\begin{bmatrix}\alpha\\ \beta\\ \gamma\\ Z\end{bmatrix}$$
$$= [\alpha\ \beta\ \gamma\ Z]\begin{bmatrix}-2 & -0.5 & 1 & 0\\ -0.5 & -2 & 1 & 0\\ 1 & 1 & -1 & 0\\ 0 & 0 & 0 & -6\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\\ \gamma\\ Z\end{bmatrix}. \tag{56}$$

The matrix $\boldsymbol{A}$ is negative definite, since all its eigenvalues are negative and therefore, $k_2 < 0$. If follows from the first part of Eq. (55) that the Jacobian matrix can only have two positive and two negative eigenvalues. As a result, the critical point is a saddle point, so that the system will eventually be able to escape from it. This result is in agreement with the findings of Sprinkhuizen-Kuyper and Boers (1996) who have proved that all stationary points of the XOR problem with finite weights are, in fact, saddle points.

We now wish to obtain a formula for the eigenvalues in closed form in terms of the weights. To this end, we make two further approximations.

The first approximation involves ignoring all dependence on Z in the above equations. To see why this is reasonable, let us consider how the value of Z is altered using the back-propagation rule in the early stages of learning. Using Eq. (2) we can evaluate the quantity:

$$\dot{Z} = -\frac{1}{64}(\dot{w}_1^{\mathrm{h}} + \dot{w}_2^{\mathrm{h}} - 2\dot{w}_0^{\mathrm{h}}).$$

Again, keeping only first-order terms in the weights, we find that $\dot{Z} = -1/4(Z)$. It follows that Z tends to 0 at an exponential rate in the initial stages of learning. Moreover, we notice from Eqs. (2) and (3) that the weight update rule for the input–hidden layer weights involves factors of the form:

$$y^{(p)}(1 - y^{(p)})w_i^{\mathrm{h}} \approx \frac{1}{4}w_i^{\mathrm{h}},$$

that are not present in the weight update rule for the hidden–output layer weights. As a result, since w_i^{h} are small, the input–hidden layer weights are updated at a much slower rate than the hidden–output layer weights, and do not change much at the beginning of training. As the hidden–output layer weights move faster, Z moves quickly towards 0, and becomes much smaller than any of the input–hidden layer weights just a few epochs after initialization of training. With this approximation in mind, we obtain a small 'factored' eigenvalue and three eigenvalues which can be found by solving the cubic equation:

$$\lambda^3 + K_1\lambda^2 + K_2\lambda + K_3 = 0, \tag{57}$$

where

$$K_1 = \alpha + \beta,\ K_2 = 2\alpha\gamma + 2\beta\gamma - \alpha\beta - 2\alpha^2 - 2\beta^2 - \gamma^2,$$
$$K_3 = \alpha\beta(\alpha + \beta - 2\gamma). \tag{58}$$

The second approximation involves ignoring the term K_3 in Eq. (57) and, thus, reducing the problem of finding the dominant eigenvalue to the solution of a quadratic equation. To see why this approximation is reasonable, let us examine the condition under which a root λ_c of Eq. (57) can be approximated by a root λ_q of the quadratic equation:

$$\lambda^2 + K_1\lambda + K_2 = 0. \tag{59}$$

For the approximation to be valid, the quantity $s = \lambda_c - \lambda_q$ must be small in magnitude compared with λ_q, i.e. $|s/\lambda_q| \ll 1$. Substituting $\lambda_c = \lambda_q + s$ for λ into Eq. (57), keeping terms linear in s, and solving for s, we obtain:

$$s = \frac{-K_3}{3\lambda_q^2 + 2K_1\lambda_q + K_2}. \tag{60}$$

Finally, using the fact that λ_q is a solution of the quadratic Eq. (59), we obtain:

$$|s/\lambda_q| = \frac{K_3}{K_1\lambda_q^2 + 2K_2\lambda_q},\ \ \lambda_q = \frac{1}{2}(-K_1 \pm \sqrt{K_1^2 - 4K_2}). \tag{61}$$

Taking into account Eqs. (52) and (58), we conclude that the right-hand side of Eq. (61) is effectively a function of $\boldsymbol{\omega}$ and ν. Assuming uniform distributions in $[-q,q]$ for these variables, it is easy to see from Eq. (61) that the distribution of $|s/\lambda_q|$ is independent of q. We have performed a Monte Carlo calculation in order to evaluate the distribution of $|s/\lambda_q|$, the result is shown in Fig. 2. It is evident that the distribution is strongly biased towards small values of $|s/\lambda_q|$. Indeed, the mean of the distribution was found to be 0.1,

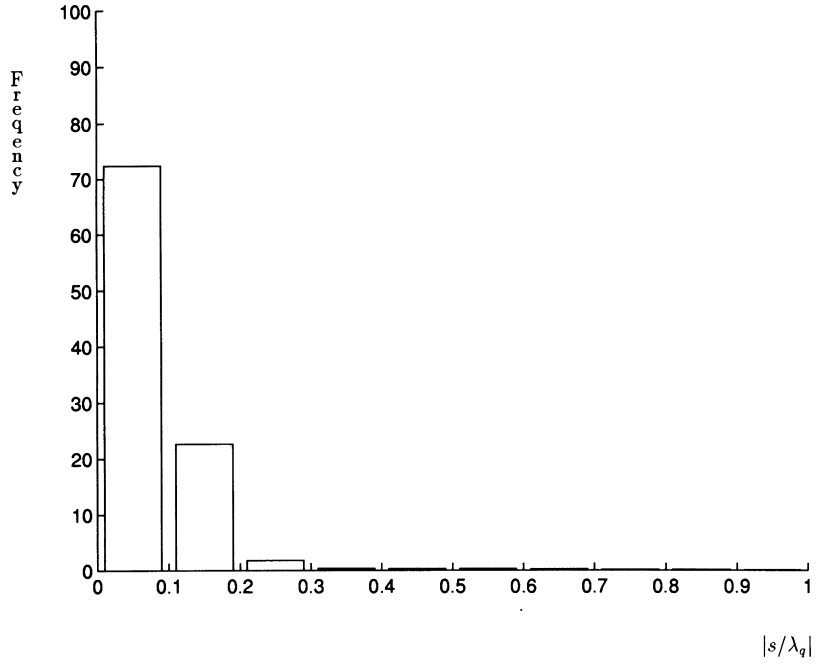

Fig. 2. Frequency histogram for the variable $|s/\lambda_q|$.

with 97% of the samples less or equal to 0.3. Hence, for 97% of different random weight initializations, $|s/\lambda_q|$ is indeed small compared with 1.

As a result of the above approximations, it is safe to assume that two eigenvalues of the Jacobian are given by the solutions of the quadratic Eq. (59):

$$\lambda_{q(1,2)} = \frac{-(\alpha+\beta) \pm \sqrt{9\alpha^2+9\beta^2+4\gamma^2+6\alpha\beta-8\alpha\gamma-8\beta\gamma}}{2}, \tag{62}$$

with the two other 'factored' eigenvalues much smaller in magnitude.

It is easy to see that the quadratic form:

$$Q_1 = 9\alpha^2+9\beta^2+4\gamma^2+6\alpha\beta-8\alpha\gamma-8\beta\gamma, \tag{63}$$

is positive definite, since Q_1 can be written as:

$$Q_1 = [\alpha\ \beta\ \gamma]\,\boldsymbol{B}\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = [\alpha\ \beta\ \gamma]\begin{bmatrix}9 & 3 & -4\\ 3 & 9 & -4\\ -4 & -4 & 4\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix}, \tag{64}$$

and all the eigenvalues of $\boldsymbol{B}$ are positive. Therefore, the eigenvalues of the back-propagation system remain real after the approximations leading to Eq. (62) have been made. Similarly, the quadratic form

$$Q_2 = 9\alpha^2+9\beta^2+4\gamma^2+6\alpha\beta-8\alpha\gamma-8\beta\gamma-(\alpha+\beta)^2, \tag{65}$$

is also positive definite, since Q_2 can be written as:

$$Q_2 = [\alpha\ \beta\ \gamma]\,\boldsymbol{C}\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix} = [\alpha\ \beta\ \gamma]\begin{bmatrix}8 & 2 & -4\\ 2 & 8 & -4\\ -4 & -4 & 4\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\\ \gamma\end{bmatrix}, \tag{66}$$

and the matrix $\boldsymbol{C}$ has only positive eigenvalues. Consequently, our approximations have revealed the existence of one positive and one negative eigenvalue, and this means that the classification of the critical point as a saddle point has been preserved. Owing to the exponential nature of the solutions, the time spent by the system in the vicinity of the origin, is determined by the magnitude of the eigenvalues and, in particular, by the magnitude of the positive eigenvalue. Since the other two eigenvalues are much smaller in magnitude, the trajectory followed by the system in the (ϵ,μ) space is approximately parallel to the plane spanned by the eigenvectors corresponding to λ_{q1} and λ_{q2}. Moreover, the trajectory will asymptotically approach the axis corresponding to the eigenvector of the largest eigenvalue. Finally, Eq. (62) gives the eigenvalues in closed form in terms of the synaptic weights, and can be used directly in the constrained learning algorithm of Section 3 in order to accelerate learning.

4.1.3. Simulation results

In our simulations, we studied the dynamics of the network shown in Fig. 1, trained to solve the XOR problem using back-propagation (BP) and back-propagation with momentum (BPM). In addition, we report performance results for the proposed constrained optimization method

Table 1
Experimental results for the XOR problem

Algorithms	Proposed	BP	BPM	QPROP	DBD	RPROP	ALECO-2
Parameters	$\delta P = 0.5$	$\eta = 6.5$	$\eta = 6.5$	$\eta = 2.0$	$\eta = 7.0$	$\eta^+ = 1.5$	$\delta P = 0.6$
	$\xi = 0.2$		$\alpha = 0.9$	$\mu = 1.2$	$\phi = 0.12$	$\eta^- = 0.8$	$\xi = 0.9$
	$\eta = 3.0$			$\omega = -10^{-4}$	$\theta = 0.70$	$\Delta_{max} = 1.0$	
	$\alpha = 0.8$			$\alpha = 0.0$	$\kappa = 0.25$	$\Delta_{min} = 10^{-66}$	
	$T_0 = 0.2$				$\alpha = 0.8$	$\Delta_0 = 0.1$	
No. of epochs	45	163	112	74	89	35	38
Successes (%)	99.1	95.9	91.7	70.2	90.8	74.0	77.0

and the following learning algorithms: QuickProp (QPROP) (Fahlman, 1988), Delta-Bar-Delta (DBD) (Jacobs, 1988), Resilient Propagation (RPROP) (Riedmiller and Braun, 1993) and ALECO-2 (Perantonis and Karras, 1995). In all cases, we performed 1000 trials with various initializations of the weights in the range -0.2–0.2. The maximum number of epochs per trial was set to 1000 and learning was considered successful when the '40–20–40' criterion of Fahlman (1988) was met. Learning parameters chosen to ensure the best possible performance are shown in Table 1.

The dynamics of the network trained using BP in a representative trial is shown in Fig. 3. Fig. 3A shows the plot of the MSE versus epoch, while Fig. 3B and C show the plot of the corresponding eigenvalues (calculated from Eq. (62)) versus epoch. Fig. 3D and E show the plot of the maximum and minimum eigenvalues versus epoch, respectively, which are calculated directly from Eq. (31) using a numerical method, namely Householder's method for the diagonalization of matrices. From Fig. 3A, the temporary minimum can be characteristically recognized as the part of the MSE that is approximately flat. In addition to Fig. 3B and C, it is clear that as long as the network remains in the vicinity of the temporary minimum, the two eigenvalues are very small. In particular, the small magnitude of the positive eigenvalue corresponding to the positive sign in Eq. (62) shown in Fig. 3B, reveals that the network is unable to move away fast from the critical point. However, as learning continues, perturbations are applied to the coefficients of the system at each epoch, which eventually cause the bifurcation of the two eigenvalues. This bifurcation is clearly seen in Fig. 3B and C, and corresponds to the part of Fig. 3A where the network abandons the flat part of the MSE curve. Finally, a comparison of Fig. 3D and E with Fig. 3C and D, respectively, reveals that the differences are indeed very small and therefore the approximation of the eigenvalues with Eq. (62) is valid.

Since the other two eigenvalues (of the 4×4 Jacobian matrix) are much smaller in magnitude, the trajectory followed by the system in the (ϵ,μ) space is approximately parallel to the plane spanned by the two eigenvectors, ξ_1 and ξ_2, corresponding to λ_{q1} and λ_{q2}. The dotted line in Fig. 4 shows the projection of the trajectory of the system on this plane. Owing to the exponential nature of the solutions, the trajectory in the phase plane with increasing time is determined by the value of the positive eigenvalue and, therefore, as time progresses, the system moves away from the origin asymptotically parallel to the ξ_1 axis. At a certain epoch of the training phase where the coordinates on this plane are

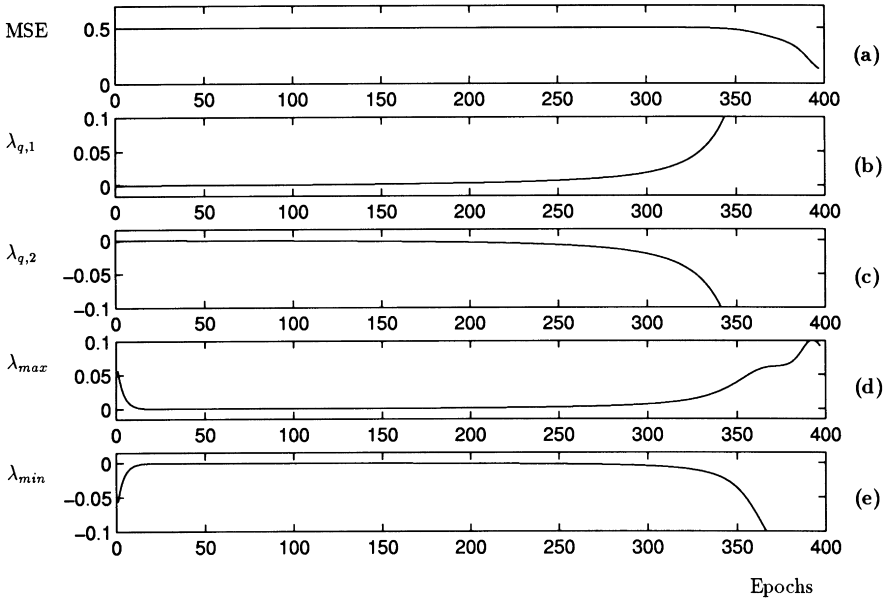

Fig. 3. Plots versus epoch for the XOR problem without momentum: (A) plot of the MSE; (B) plot of λ_{q1}; (C) plot of λ_{q2}; (D) plot of λ_{max}; and (E) plot of λ_{min}.

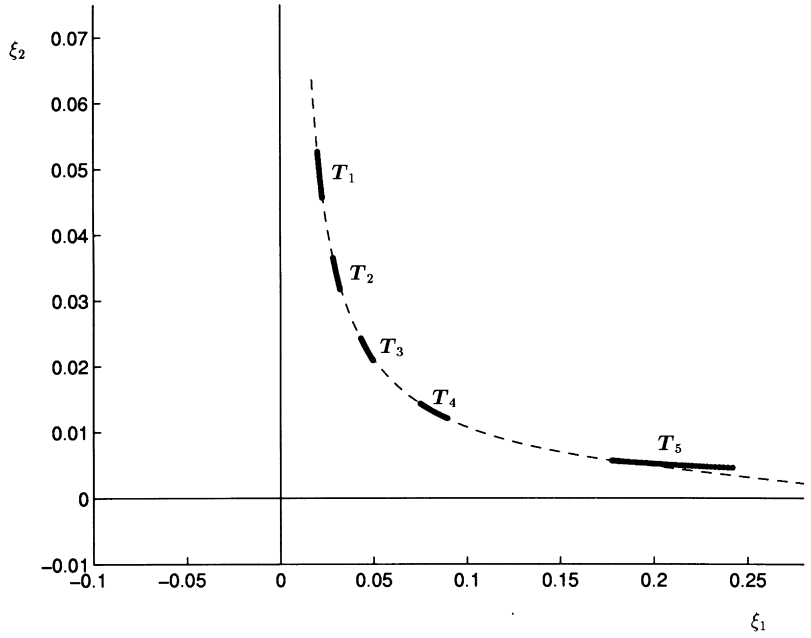

Fig. 4. Actual and predicted trajectories of the back-propagationsystem at the vicinity of the temporary minimum.

$\xi_{1,0}$ and $\xi_{2,0}$, the trajectory can be estimated instantaneously by the parametric equations $\xi_1 = \xi_{1,0}\exp(\lambda_{q1}t)$ and $\xi_2 = \xi_{2,0}\exp(\lambda_{q2}t)$, where λ_{q1} and λ_{q2} are the eigenvalues at the current epoch. The solid lines in Fig. 4 represent these estimates at five different points of the actual trajectory. Note that the actual and estimated trajectory are in very good agreement. At the earlier stages of learning, λ_{q1} is small and the system makes little progress along the ξ_1 axis. This is reflected in the small magnitude of the ξ_1 axis projections of the estimated trajectories T_1–T_3. However, as learning progresses and the eigenvalues bifurcate (T_4), the system moves rapidly away from the origin as reflected in trajectory T_5.

The dynamics of the network trained using BPM is illustrated in Fig. 5. Fig. 5A shows a plot of the MSE versus epoch for BPM, while Fig. 5B and C show the plot of the corresponding eigenvalues (again calculated from Eq. (62)) versus epoch. From these figures it is clear that even with the addition of momentum the dynamics of the network does not alter significantly. This is an expected result, since the only benefit of the inclusion of the momentum term is the slight decrease in the time spent at the temporary minimum, because of the effective increase of the learning rate achieved with momentum in flat plateaus. This, in turn, causes larger perturbations in the eigenvalues and hence, the bifurcation occurs earlier. This bifurcation of the eigenvalues can be clearly seen in Fig. 5B and C, and indicates the more rapid abandonment of the temporary minimum.

Next, we consider the application of the constrained optimization method introduced in Section 3. In this case, at each epoch we tried to maximize the quantity $\Phi = \lambda_{q1}$ with λ_{q1} given by Eq. (62). Clearly, Φ is given by an analytic expression which is a differentiable function of the weights and, therefore, all the derivatives required by the algorithm can be obtained. The constrained optimization algorithm was applied only when each component of ϵ was below a certain threshold T_0, indicating that the network was trapped at the temporary minimum. As soon as the threshold was exceeded, the training algorithm was switched to BPM. Values for all related parameters are shown in Table 1. The small number of required epochs for the solution of the problem, indicates a considerable improvement in the learning behaviour of the network with the proposed algorithm. In addition, since this significant improvement in the behaviour of the system was achieved by maximizing λ_{q1} as given by Eq. (62), the approximations made for the eigenvalues discussed in the previous subsection are once again justified. Fig. 6 shows a representative behaviour of the dynamics of the network with this learning scheme. Fig. 6A shows the MSE curve versus epoch, and Fig. 6B and C show the corresponding values of the eigenvalues versus epoch as given by Eq. (62). From the MSE curve we can see that the learning behaviour of the system is altered considerably, since the flat part of the curve is significantly reduced. Fig. 6B shows the maximization of the largest eigenvalue achieved with the proposed algorithm, which guarantees the fast evolution of the dynamics of the system towards a solution of the problem.

The comparative performance results summarized in Table 1 (percentage of successful trials and mean number of epochs in successful trials) show that the proposed method does not only exhibit a much lower mean number of epochs than BP, but is also among the faster of the algorithms studied. Moreover, it has the additional advantage that it can overcome the presence of temporary minima in almost all trials, and, therefore, is the only method that

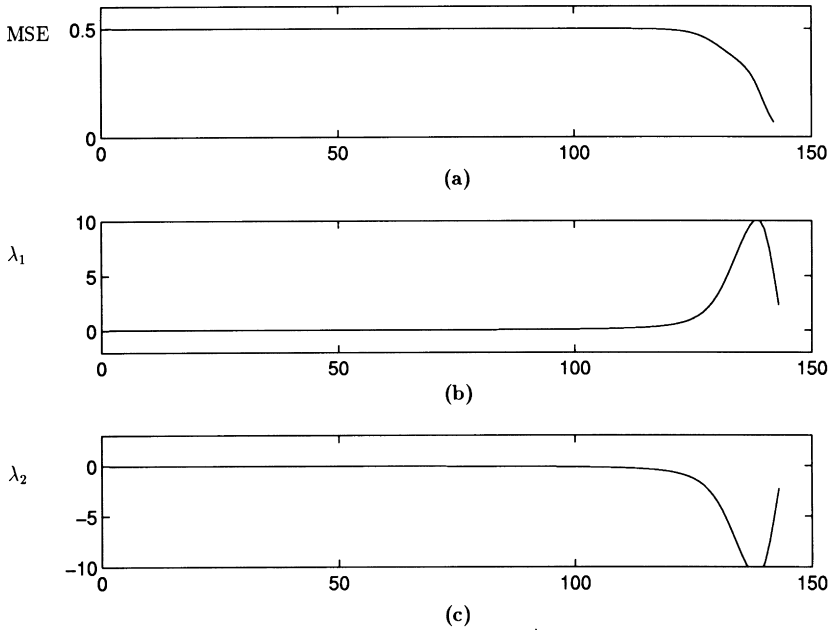

Fig. 5. Plots versus epoch for the XOR problem with momentum: (A) plot of the MSE; (B) plot of λ_{q1}; and (C) plot of λ_{q2}.

exhibits almost 100% success in converging to the desired global minimum.

4.2. The unit square problem

The objective of this training task is to distinguish between the two classes in the two-dimensional space shown in Fig. 7. Thirty samples of each class, picked at random, were used for the training of the network. It should be clear that for this particular problem it is practically impossible to obtain an analytical expression for the eigenvalues of the Jacobian matrix. Therefore, we can use Eq. (33) in order to apply the constrained optimization method.

As in the XOR problem, we studied the dynamics of the 2-2-1 network using back-propagation (BP) and back-propagation with momentum (BPM). We also report performance results for the proposed optimization method and the other algorithms mentioned in Section 4.1.3 with the same trial statistics, initialization and stopping criteria. Learning parameters for all algorithms are shown in Table 2.

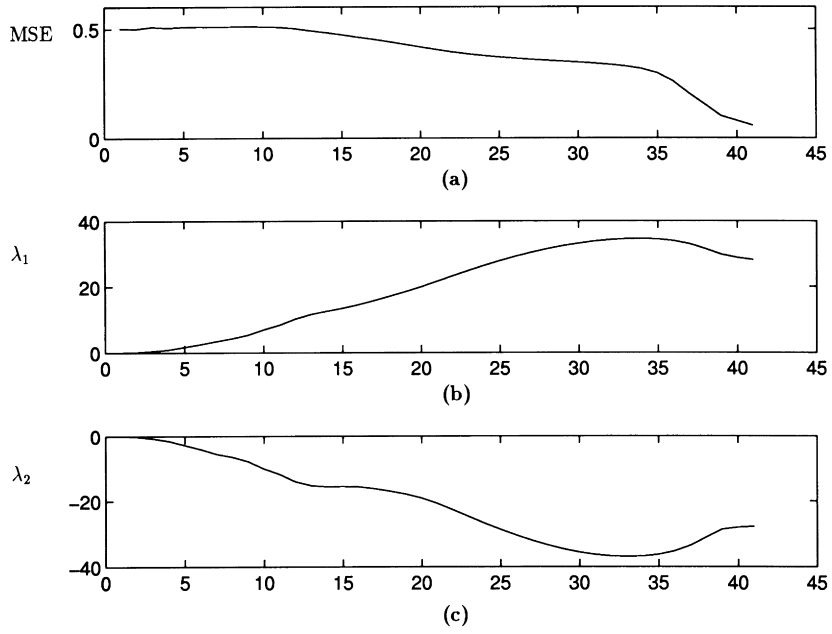

Fig. 6. Plots versus epoch for the XOR problem with constrained optimization: (A) plot of the MSE; (B) plot of λ_{q1}; and (C) plot of λ_{q2}.

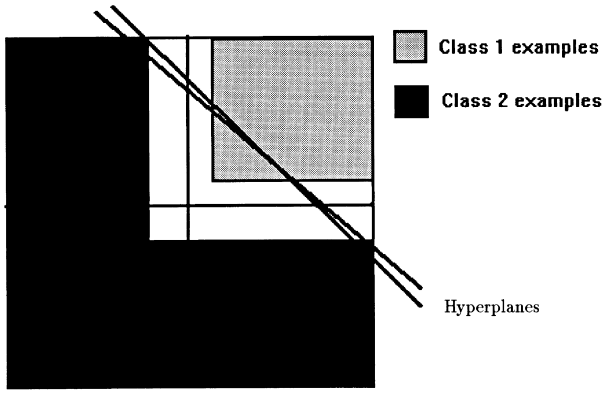

Fig. 7. The two classes of the unit square problem and the hyperplanes of the two hidden nodes at the temporary minimum.

Fig. 8 shows a representative behaviour of the dynamics of the network trained using BP. Fig. 8A shows the plot of the MSE versus epoch, while Fig. 8B and C show the plot of the corresponding eigenvalues (calculated directly from Eq. (31)) versus epoch. From the MSE curve we can see that after an initial error reduction corresponding to the first 200 iterations, the network encounters a temporary minimum, from which it seems unable to escape for a relatively long time. However, after a sufficient number of iterations, we can observe the sudden drop of the MSE curve which indicates the abandonment of the temporary minimum. The initial error reduction phase corresponds to the situation whereby the two hidden weight vectors move towards their common attractor hyperplane. When they coincide completely with it, the network enters the phase where it sticks to the temporary minimum. This coincidence of the hyperplanes is clearly seen in Fig. 7. Fig. 8B shows the small magnitude of the maximum eigenvalue in the vicinity of the temporary minimum, and its eventual increase that allows the network to escape. At this point it is interesting to make the following remark: suppose that we make the two classes linearly separable (by increasing the area of the white region in Fig. 7), then, when the weight vectors of the two hidden units coincide with their common attractor, the network solves the problem and thus, no temporary minimum is encountered. This is because the common attractor is the line that separates the two different classes and thus, no further learning is required. In this case, the MSE curve will fall almost linearly to zero level.

The dynamics of the network trained using BPM is illustrated in Fig. 9. Fig. 9A shows the plot of the MSE versus epoch, while Fig. 9B and C show the plot of the corresponding maximum and minimum eigenvalues, respectively, versus epoch. From these Fig. 9 we can again see that even with the addition of momentum, the dynamics of the network remains almost the same, the only difference being the slight decrease in the time spent at the temporary minimum due to the effect of momentum.

Next, we consider the application of the constrained optimization method introduced in Section 3. The large reduction in the required number of epochs for the solution of the problem, and the dramatic increase in the number of successful trials, once again indicate the considerable improvement in the learning behaviour of the network with the proposed method. Fig. 10 shows a representative behaviour of the dynamics of the network with this learning scheme. Fig. 10A shows the MSE curve versus epoch, and Fig. 10B and C show the corresponding values of the maximum and minimum eigenvalues, respectively, versus epoch. From the MSE curve, we can see the network is able to solve the problem in a very small number of epochs compared with the cases of BP or BPM. Fig. 10B shows the maximization of the largest eigenvalue achieved with the constrained optimization algorithm, which verifies the validity of the approximation made for Φ, whose maximization ensures the fast evolution of the dynamics of the system away from any temporary minima.

The comparative performance results summarized in Table 2, show that for the unit square problem the proposed constrained optimization method is among the faster of the algorithms studied. Once again, the method is not affected by the presence of temporary minima, and, therefore, exhibits the highest success rate among all algorithms.

5. Conclusions

The dynamical systems theory has been used for the mathematical analysis of the dynamics of a two-layer, feed-forward network trained according to the back-propagation algorithm. The utility of this novel analysis has been demonstrated by describing and explaining the occurrence of temporary minima. A greater understanding of the fundamental mechanisms behind this type of minima

Table 2
Experimental results for the unit square problem

Algorithms	Proposed	BP	BPM	QPROP	DBD	RPROP	ALECO-2
Parameters	$\delta P = 0.7$ $\xi = 0.2$ $\eta = 0.5$ $\alpha = 0.8$ $T_0 = 0.8$	$\eta = 1.2$	$\eta = 0.6$ $\alpha = 0.9$	$\eta = 1.0$ $\mu = 1.4$ $\omega = -10^{-4}$ $\alpha = 0.0$	$\eta = 0.4$ $\phi = 0.12$ $\theta = 0.70$ $\kappa = 0.25$ $\alpha = 0.8$	$\eta^+ = 1.4$ $\eta^- = 0.8$ $\Delta_{max} = 1.0$ $\Delta_{min} = 10^{-6}$ $\Delta_0 = 0.1$	$\delta P = 2.0$ $\xi = 0.5$
No. of epochs	40	257	196	40	79	79	34
Successes (%)	99.9	73.5	86.1	82.4	85.0	80.0	87.0

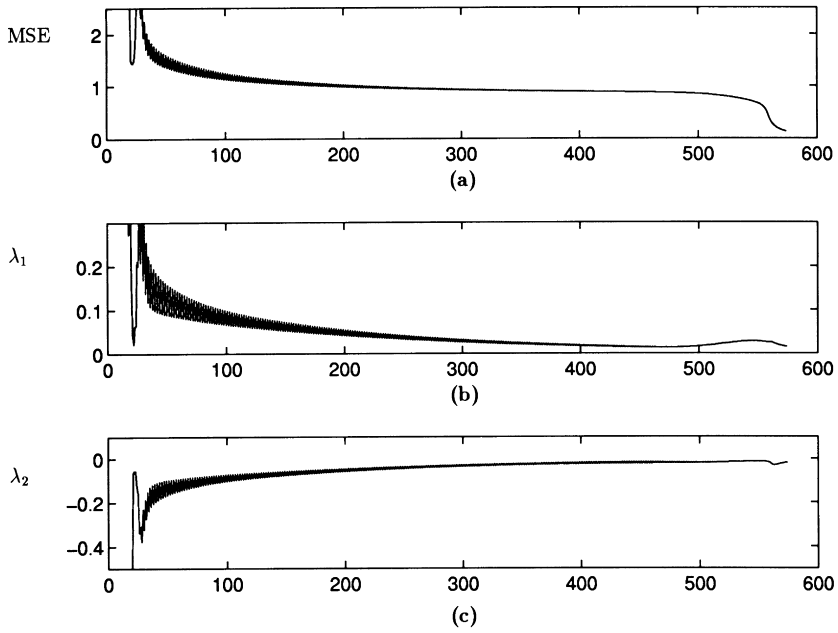

Fig. 8. Plots versus epoch for the unit square problem without momentum: (A) plot of the MSE; (B) plot of λ_{max}; and (C) plot of λ_{min}.

is important, because it usually takes a long time for the network to eventually escape from them, and hence, training is significantly impaired. To alleviate this problem, we introduced a constrained optimization method that achieves simultaneous minimization of the cost function, and maximization of the largest eigenvalue of the Jacobian matrix of the dynamical system model, so that the network avoids getting trapped at a temporary minimum and hence, total training time is significantly decreased.

There are several research issues pertaining to this novel approach to the dynamics of back-propagation networks. The important problem of extending the analysis to networks with an arbitrary number M of hidden units each connected to N inputs, is currently under investigation. For this generalization, motivated by the redundancy as an inherent property of multilayer networks, appropriate state variables that map the temporary minimum to the origin of the phase plane, are expected to be the differences of each hidden weight vector from the average of all the hidden weight vectors, and the differences of each output weight component from the average of all the output weight components. This gives a set of independent quantities that

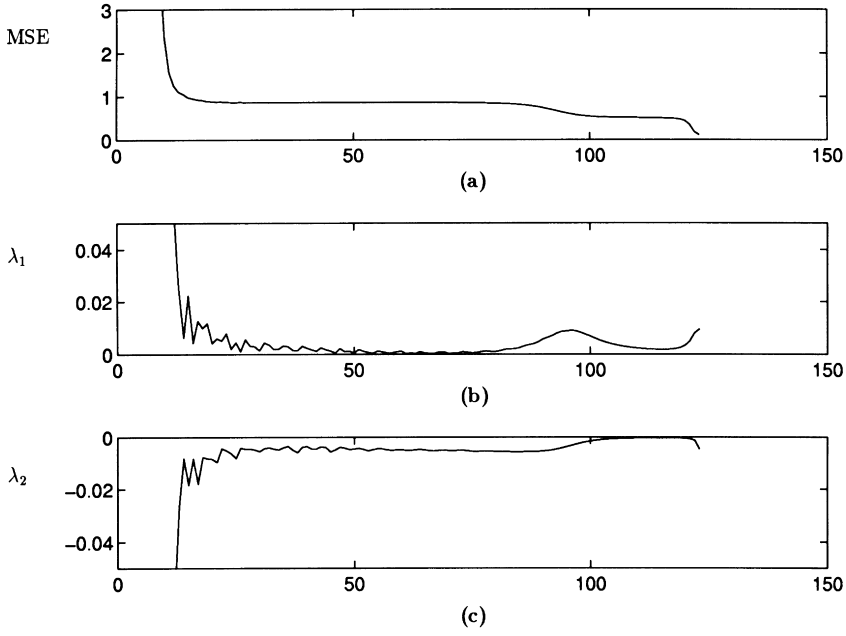

Fig. 9. Plots versus epoch for the unit square problem with momentum: (A) plot of the MSE; (B) plot of λ_{max}; and (C) plot of λ_{min}.

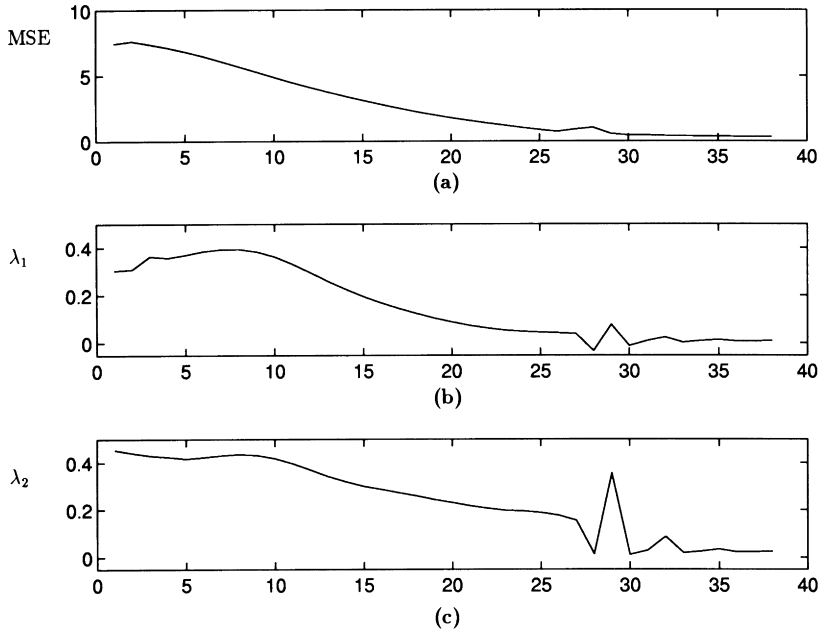

Fig. 10. Plots versus epoch for the unit square problem with constrained optimization: (A) plot of the MSE; (B) plot of λ_{max}; and (C) plot of λ_{min}.

extend the 2-hidden node case studied in this paper. For a given task, the initial bifurcation will involve the splitting of the difference sets into two clusters. The membership of those clusters will determine which further clusters can then be constructed by later bifurcations. Therefore, the resulting Jacobian matrix of such a general network can be analytically evaluated in the vicinity of the temporary minimum that appears after each bifurcation. The constrained optimization method is, of course, independent of the number of hidden nodes, and can be applied using lower bounds to the maximum eigenvalue of the Jacobian matrix. It would be interesting to investigate whether this approach can facilitate learning in large scale problems. It is our belief that the answers to these questions can provide a valuable insight into many aspects of learning, help develop new efficient tools for a much needed mathematical/analytical approach to the study of feed-forward networks, and lead to more efficient methods of minimizing the time spent in temporary minima.

References

Biehl, M., & Schwarze, H. (1995). Learning by on-line gradient descent. *Journal of Physics, A28*, 643–656.

Boyce, W.E., & DiPrima, R.C. (1986). *Elementary differential equations and boundary value problems*. New York: Wiley.

Coddington, E.A., & Levinson, N. (1955). *Theory of ordinary differential equations*. New York: McGraw-Hill.

Fahlman, S.E. (1988). Faster learning variations on back-propagation: an empirical study. In D. Touretzky, G. Hinton & T. Sejnowski (Eds.), *Proceedings of the Connectionist Models Summer School* (pp. 38–51). San Mateo, CA: Morgan Kaufmann.

Gelb, A., & Vander Velde, W.E. (1968). *Multiple-input describing functions and nonlinear system design*. New York: McGraw-Hill.

Gorse, D., Shepherd, A., & Taylor, J.G. (1993). Avoiding local minima using a range expansion algorithm. *Neural Network World, 5*, 503–510.

Gorse, D., Shepherd, A., & Taylor, J.G. (1997). The new ERA in supervised learning. *Neural Networks, 10*, 343–352.

Graham, D., & McRuer, D. (1961). *Analysis of nonlinear control systems*. New York: Wiley.

Guo, H., & Gelfand, S.B. (1991). Analysis of gradient descent learning algorithms for multilayer feedforward neural networks. *IEEE Transactions on Circuits and Systems, 38*, 883–894.

Jacobs, R.A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks, 1*, 295–307.

Karras, D.A., & Perantonis, S.J. (1995). An efficient constrained training algorithm for feedforward networks. *IEEE Transactions on Neural Networks, 6*, 1420–1434.

Lee, Y., Oh, S., & Kim, M. (1991). The effect of initial weights on premature saturation in back-propagation learning. In *Proceedings of International Joint Conference on Neural Networks* (Vol. 1, pp. 765–770) Seattle, WA.

Liang, P. (1991). Design artificial neural networks based on the principle of divide-and-conquer. In *Proceedings of International Conference on Circuits and Systems* (pp. 1319–1322).

Lisboa, P.J.G., & Perantonis, S.J. (1991). Complete solution of the local minima in the XOR problem. *Network, 2*, 119–124.

Minsky, M.L., & Papert, S.A. (1988). *Perceptrons*, expanded ed. Cambridge, MA: MIT Press.

Murray, A.F. (1991). Analog VLSI and multi-layer perceptrons — accuracy, noise and on-chip learning. In *Proceedings of Second International Conference on Microelectronics for Neural Networks* (pp. 27–34).

Parker, D.B. (1985). Learning-logic: casting the cortex of the human brain in silicon. (Tech. Rep. TR-47). Center for Computational Research in Economics and Management Science, Massachussets Institute of Technology.

Perantonis, S.J., & Karras, D.A. (1995). An efficient constrained learning algorithm with momentum acceleration. *Neural Networks, 8*, 237–249.

Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In *Proceedings of the International Conference on Neural Networks* (Vol.1, pp. 586–591). San Francisco, CA.

Saad, D., & Solla, S.A. (1995a). On-line learning in soft-committee machines. *Physical Review*, *E52*, 4225–4243.

Saad, D., & Solla, S.A. (1995b). Exact solution for on-line learning in multilayer neural networks. *Physical Review Letters*, *74*, 4337–4340.

Sontag, E.D., & Sussmann, H.J. (1991). Backpropagation separates where perceptrons do. *Neural Networks*, *4*, 243–249.

Sprinkhuizen-Kuyper, I.G., & Boers, E.J.W. (1996). The error surface of the simplest XOR network has only global minima. *Neural Computation*, *8*, 1301–1320.

Sussmann, H.J. (1992). Uniqueness of the weights for minimal feedforward nets with a given input–output map. *Neural Networks*, *5*, 589–593.

Woods, D. (1988). Back and counter propagation aberrations. In *Proceedings of International Joint Conference on Neural Networks* (Vol. 1, pp. 343–353).