Network: Computation in Neural Systems **7** (1996) 123–139. Printed in the UK

Unsupervised learning in general connectionist systems

J A Dente† and R Vilela Mendes‡§

† Laboratório de Mecatrónica, Departamento de Engenharia Electrotécnica e de Computadores, Instituto Superior Técnico, Av. Rovisco Pais, 1096 Lisbon Codex, Portugal

‡ Grupo de Física–Matemática, Complexo II, Universidade de Lisboa, Av. Gama Pinto 2, 1699 Lisbon Codex, Portugal

Received 19 May 1995, in final form 16 October 1995

Abstract. There is a common framework in which different connectionist systems may be treated in a unified way. The general system in which they may all be mapped is a network which, in addition to the connection strengths, has an adaptive node parameter controlling the output intensity. In this paper we generalize two neural network learning schemes to networks with node parameters. In generalized Hebbian learning we find improvements to the convergence rate for small eigenvalues in principal component analysis. For competitive learning the use of node parameters also seems useful in that, by emphasizing or de-emphasizing the dominance of winning neurons, either improved robustness or discrimination is obtained.

1. Introduction

Connectionist models are dynamical systems composed of many elementary and similar units connected together to form a network. The dynamics of such a system occur on three different levels: that of the network states, that of the connection strengths and that of the architecture of the network itself.

The term connectionism is usually applied to neural network models but, as Doyne Farmer (1990) has shown, there is a common mathematical framework in which neural networks, classifier systems, immune networks and autocatalytic reaction networks may be treated in a unified way. The general model in which all these models may be mapped looks like a neural network where, in addition to the node state variables (x_i) and the connection strengths (w_{ij}), there is also a node parameter (θ_i) with learning capabilities (figure 1). The node parameter represents the possibility of changing the nature of the linear or nonlinear function $f_i(\sum_j w_{ij}x_j)$ at each node through learning. In the simplest case, θ_i will be simply an intensity parameter. Therefore, the degree to which the activity at node i influences the activity at other nodes depends not only on the connection strengths (w_{ki}) but also on an adaptive node parameter θ_i. In some cases, as in the B-cell immune network, the node parameter is the only means of controlling the relative influence of a node on others, the connection strengths being fixed chemical reaction rates.

The purpose of this paper is to generalize some of the learning algorithms used in neural networks to the more general network depicted in figure 1. In section 2 we discuss unsupervised learning of Hebbian type and in section 3 competitive unsupervised learning. In all cases, the learning algorithms act on both the connection strengths and the node parameters. We would emphasize, however, that the present work should not be considered

§ E-mail: vilela@alf4.cii.fc.ul.pt

0954-898X/96/010123+17$19.50

as the most general treatment of unsupervised learning in connectionist systems of the Farmer class. For example, it might also be interesting to generalize the B-cell immune network algorithm to a learning scheme incorporating adjustable connection strengths.

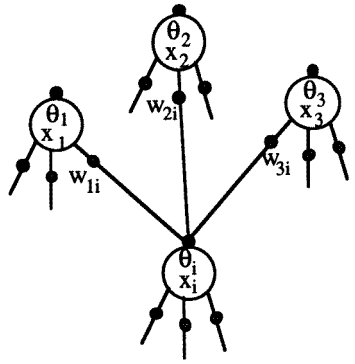

Figure 1. A general connectionist network with state variables (x_i), connection strengths (w_{ij}) and node parameters (θ_i).

2. Hebbian-type learning with a node parameter

We shall denote by x_i the output of node i. Hebbian learning (Hebb 1949) is a type of unsupervised learning where a connection strength w_{ij} is reinforced whenever the product $x_j x_i$ is large. In its simplest form,

$$\Delta w_{ji} = \eta x_j x_i \tag{2.1}$$

there is the problem that the weights may keep on growing without bound. Several schemes have been proposed (Oja 1982, Linsker 1988, Yuille *et al* 1989) to deal with this problem and keep the weights bounded.

As shown by several authors, Hebbian learning extracts the eigenvectors of the correlation matrix $\boldsymbol{Q}$ of the input data.

$$\boldsymbol{Q}_{ij} = \langle x_i x_j \rangle \tag{2.2}$$

where $\langle \cdots \rangle$ means the sample average. If the learning law is local, the rows of the connection matrix w_{ij} all tend to the eigenvector associated with the largest eigenvalue of the correlation matrix. To obtain the other eigenvector directions one needs non-local laws (Sanger 1989, Oja 1989, 1992). Sanger's approach has the advantage of organizing the connection matrix in such a way that the rows are the eigenvectors associated with the eigenvalues in decreasing order. However, it suffers from slow convergence rates for the lowest eigenvalues (see below). The methods that have been proposed may, with small modifications, be used for both linear and nonlinear networks. However, because the maximum information about a signal $\{x_i\}$, that may be coded directly in the connection matrix w_{ij}, is the principal components decomposition, which may already be obtained with linear units, we shall only discuss this case.

The learning rules we propose are a generalization of Sanger's scheme including a node parameter θ_i. The node parameter is, in this case, only a multiplicative intensity factor. Because the node units are linear there is an equivalent network with unit node parameters and connection strengths $w'_{ij} = \theta_i w_{ij}$. However, because, as we shall see, θ_i carries information independent from the w_{ij} and plays a role in the convergence process, it is useful to treat it as an independent parameter.

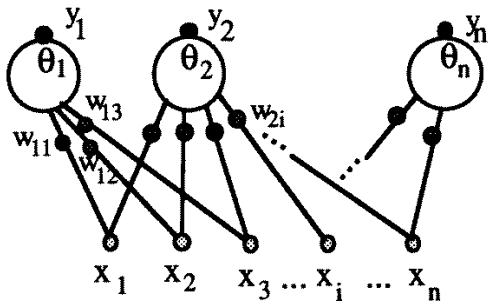

Figure 2. One-layer neural feedforward network for Hebbian learning with node parameter.

We consider a one-layer feedforward network with as many inputs as outputs (figure 2), where the updating rules proposed for w_{ij} and θ_i are:

$$w_{ij}(t+1) = w_{ij}(t) + \gamma_w y_i(t) \left\{ x_j(t) - \sum_{k=1}^{i} \theta_k^{-1} y_k(t) w_{kj}(t) \right\} \tag{2.3a}$$

$$\theta_i(t+1) = \theta_i(t) + \gamma_\theta y_i(t) \{1 - y_i(t)\} \tag{2.3b}$$

where y_i is the output of node i

$$y_i = \theta_i \sum_j w_{ij} x_j \tag{2.4}$$

and γ_w and γ_θ are positive constants that control the learning rate. As will be shown below, the learning dynamics of (2.3) are capable of accelerating the convergence rate for the small eigenvalues of $\boldsymbol{Q}$. To avoid undesirable fixed points of the dynamics, where this acceleration effect is not obtained, the learning rules (2.3) are supplemented by the following prescription: '*The starting θ_i's are all positive and are not allowed to decrease below a value $(\theta_i)_{\min}$. If, at a certain point of the learning process, θ_i hits the lower bound then one makes the replacement $w_{ij} \to -w_{ij}$ in row i of the connection matrix*' (see the proof below and remark (3)).

Define the N-dimensional vectors

$$(\langle \boldsymbol{x} \rangle)_i = \langle x_i \rangle \tag{2.5a}$$

$$(\boldsymbol{w}_i)_j = w_{ij}. \tag{2.5b}$$

We now prove the following statements:

If the timescale of the system (2.3) is much slower than the averaging time for the input signal $\{x_i\}$ then the following hold:

(i) *The system (2.3) has a stable fixed point such that*

$$\boldsymbol{Q}\boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i \qquad \lambda_1 \geqslant \lambda_2 \geqslant \cdots \lambda_N \tag{2.6a}$$

$$\theta_i = \frac{1}{\lambda_i} (\boldsymbol{w}_i \cdot \langle \boldsymbol{x} \rangle) \tag{2.6b}$$

where $|\boldsymbol{w}_i| = 1$ and $\lambda_i > 0$ $(i = 1, \ldots, N)$ are the eigenvalues of the correlation matrix.

(ii) *Convergence to the fixed point is sequential in the sense that $\boldsymbol{w}_j$ is only attracted to the eigenvector described in (2.6a) if all vectors $\boldsymbol{w}_i$ for $i < j$ are already close to their corresponding eigenvector values.*

Proof. The hypothesis about the timescale of the learning equations (2.3) means that all functions of the input signal $\{x_i\}$ may be replaced by their expectation values. In practice, this means that the learning rates γ_w and γ_q must be sufficiently small to ensure this condition. This is the usual situation for on-line learning schemes (Biehl and Schwarze 1995). Taking average values in (2.3*a*), one obtains, for $i = 1$, the fixed-point condition

$$\boldsymbol{Q}\boldsymbol{w}_1 = (\boldsymbol{w}_1, \boldsymbol{Q}\boldsymbol{w}_1)\boldsymbol{w}_1 \tag{2.7}$$

hence $(\boldsymbol{w}_1, \boldsymbol{Q}\boldsymbol{w}_1) = \lambda_1$ with $|\boldsymbol{w}_1|^2 = 1$.

Assume now that for all $k < i$ the fixed-point condition $\boldsymbol{w}_k(t+1) = \boldsymbol{w}_k(t)$ is satisfied by

$$\boldsymbol{Q}\boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k \qquad k = 1, \ldots, i-1 \tag{2.8}$$

with $|\boldsymbol{w}_k| = 1$. Then for $k = i$ the fixed-point condition is

$$\boldsymbol{Q}\boldsymbol{w}_i = \sum_{k=1}^{i} (\boldsymbol{w}_i, \boldsymbol{Q}\boldsymbol{w}_k)\boldsymbol{w}_k. \tag{2.9}$$

Writing

$$\boldsymbol{w}_i = \sum_{k=1}^{i-1} \alpha_k \boldsymbol{w}_k + \boldsymbol{w}_\perp \tag{2.10}$$

with $\boldsymbol{w}_\perp$ orthogonal to all the $\boldsymbol{w}_k$ $(k = 1, \ldots, i-1)$ one obtains from (2.9)

$$\boldsymbol{Q}\boldsymbol{w}_\perp = (\boldsymbol{w}_i, \boldsymbol{Q}\boldsymbol{w}_i) \sum_{k=1}^{i-1} \alpha_k \boldsymbol{w}_k + (\boldsymbol{w}_i, \boldsymbol{Q}\boldsymbol{w}_i)\boldsymbol{w}_\perp.$$

The matrix $\boldsymbol{Q}$ must preserve its own eigenspaces, hence $\alpha_k = 0$, $k = 1, \ldots, i-1$. Then

$$\boldsymbol{Q}\boldsymbol{w}_\perp = (\boldsymbol{w}_i, \boldsymbol{Q}\boldsymbol{w}_i) w_\perp$$

that is, $\boldsymbol{Q}\boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i$ and $|\boldsymbol{w}_i| = 1$. The induction is complete and one proves that the set of N vectors $\boldsymbol{w}_i$, $i = 1, \ldots, N$ such that $\boldsymbol{Q}\boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i$ is a fixed point of the dynamics defined by (2.3*a*). Using this solution in (2.3*b*), averaged over the timescale of the input data, one obtains the fixed point in (2.6*b*).

The next step is to analyse the stability of the fixed point. We consider a perturbation of the solution around the fixed point

$$\boldsymbol{w}_i' = \boldsymbol{w}_i + \boldsymbol{\delta}_i \qquad \theta_i' = \theta_i + \varepsilon_i.$$

Then

$$\boldsymbol{\delta}_i(t+1) - \boldsymbol{\delta}_i(t) = \gamma_w \theta_i \Bigg\{ (\boldsymbol{Q} - \lambda_i)\boldsymbol{\delta}_i - \sum_{k=1}^{i} \lambda_k (\boldsymbol{\delta}_i, \boldsymbol{w}_k)\boldsymbol{w}_k - \sum_{k=1}^{i} \lambda_i (\boldsymbol{w}_i, \boldsymbol{\delta}_k)\boldsymbol{w}_k \Bigg\} + \mathrm{O}(\delta^2). \tag{2.11}$$

Expanding $\boldsymbol{\delta}_i$ on the basis of the fixed-point vectors, $\boldsymbol{\delta}_i = \sum_l c_{il}\boldsymbol{w}_l$

$$\sum_l \{c_{il}(t+1) - c_{il}(t)\}\,\boldsymbol{w}_l = \gamma_w \theta_i \Bigg\{ \sum_{l>i} c_{il}(\lambda_l - \lambda_i)\boldsymbol{w}_l - 2\lambda_i c_{ii}\boldsymbol{w}_i - \lambda_i \sum_{l<i} c_{il}\boldsymbol{w}_l - \sum_{l<i} \lambda_i c_{li}\boldsymbol{w}_l \Bigg\} + \mathrm{o}(\delta^2). \tag{2.12}$$

Now, if all $\boldsymbol{w}_k$ up to order $i-1$ have already converged to the fixed point, one has $c_{li}=0$ for $l<i$ and the last term in (2.12) vanishes. If, furthermore, the eigenvalues are ordered by

$$\lambda_1 \geqslant \lambda_2 \geqslant \ldots \lambda_N$$

(2.12) implies that $\boldsymbol{w}_i$ is stable. By induction one proves the stability of the eigenvalue-ordered fixed point if $\gamma_w \theta_i > 0$.

For the stability of the θ_i fixed point we have

$$\varepsilon_i(t+1)-\varepsilon_i(t)=\gamma_\theta \varepsilon_i(t)\sum_i\{(\boldsymbol{w}_i,\langle \boldsymbol{x}\rangle)-2\theta_i\lambda_i\}+\mathrm{o}(\varepsilon^2)=-\varepsilon_i\gamma_\theta\theta_i\lambda_i+\mathrm{o}(\varepsilon^2)$$

and the solution is stable if $\gamma_\theta\theta_i\lambda_i>0$. With $\gamma_\theta>0$ this requires θ_i to be positive, that is $\boldsymbol{w}_i\cdot\langle\boldsymbol{x}\rangle>0$, which uniquely fixes the direction of $\boldsymbol{w}_i$ that ensures the stability of the fixed point. If $\boldsymbol{w}_i\cdot\langle\boldsymbol{x}\rangle<0$ then $\theta_i<0$, but that fixed point would be unstable and θ_i moves to another fixed point at $\theta_i=0$. This is the reason for the prescription that changes the sign of $\boldsymbol{w}_i$ when θ_i reaches $(\theta_i)_{\min}$ (see remark (3)). Notice that a positive θ_i is also required for stability of the w_{ij} fixed-point solution. If all eigenvalues are different the stable solution is unique without sign ambiguity in $\boldsymbol{w}_i$. If some of the eigenvalues are degenerate the algorithm may choose any orthogonal set of eigenvectors in each degenerate subspace.

The induction proof of stability of each $\boldsymbol{w}_i$ required the assumption that all $\boldsymbol{w}_k$ for $k<i$ are already at their fixed-point values. Therefore, convergence to the fixed point will, in general, be sequential, as stated above.

Remarks (1) The matrix $\boldsymbol{w}$ of the connection strengths extracts the principal components of the correlation matrix $\boldsymbol{Q}$. The node parameters θ_i at their fixed point (2.6*b*) extract additional information on the mean value of the data vector $\langle\boldsymbol{x}\rangle$ and the eigenvalues of $\boldsymbol{Q}$. To deal with data with zero mean it is convenient to change the θ_i-updating law to

$$\theta_i(t+1)=\theta_i(t)+\gamma_\theta\left\{\theta_i(t)\sum_k w_{ik}\,(x_k+r_k)-\left(\theta_i(t)\sum_k w_{ik}x_k\right)^2\right\} \tag{2.13}$$

where $\boldsymbol{r}$ is a fixed vector.

The stable fixed point is now

$$\theta_i=\frac{1}{\lambda_i}(\boldsymbol{w}_i\cdot(\langle\boldsymbol{x}\rangle+\boldsymbol{r})) \tag{2.14}$$

and the stability analysis is the same as before. If one wishes to separate the eigenvalues from the information on the average data $\langle\boldsymbol{x}\rangle$ one may add another parameter μ_i to each node with a learning law

$$\mu_i(t+1)=\mu_i(t)+\gamma_\mu\left\{1-\mu_i^\alpha(t)\left(\sum_j w_{ij}x_j\right)^2\right\} \tag{2.15}$$

which, with the same assumptions about timescales as before, converges to the stable fixed point

$$\mu_i=\left(\frac{1}{\lambda_i}\right)^{1/\alpha}. \tag{2.16}$$

The convergence rate near the fixed point is $\gamma_\mu\alpha\lambda_i^{1/\alpha}$. Therefore, choosing a large α accelerates convergence for the small eigenvalues.

(2) In addition to its role in extracting additional information on the input signal, the node parameter θ_i also plays a role in accelerating the convergence to the stable fixed point. From equation (2.12) in the stability proof it is clear that, for fixed $\gamma_w\theta_i$, the rate of convergence to the fixed point is very slow for the components associated with the smallest eigenvalues. This is the reason for the convergence problems in Sanger's method and one also finds that, sometimes, the results for the small components are quite misleading. On the other hand, increasing γ_w does not help because then the timescale of the w_{ij} learning law is no longer much smaller than that of the data and one obtains large fluctuations in the principal components. With a node parameter θ_i and the learning law (2.3*b*) the situation is more favourable because for small eigenvalues the effective control parameter ($\gamma_w\theta_i$) is dynamically amplified. This accelerates the convergence of the minor components without inducing fluctuations on the principal components.

(3) We examine here the effect of the prescription to avoid the fixed points where some $\theta_i = 0$. Consider the average evolution of θ_i, assuming all the other variables fixed

$$\theta_i(t+1) = \theta_i(t)(1 + \gamma_\theta \boldsymbol{w}_i \cdot \langle \boldsymbol{x} \rangle) - \gamma_\theta \theta_i^2(t)(\boldsymbol{w}_i, \boldsymbol{Q}\boldsymbol{w}_i).$$

If $\boldsymbol{w}_i \cdot \langle \boldsymbol{x} \rangle > 0$ the stable fixed point is at $\boldsymbol{w}_i \cdot \langle \boldsymbol{x} \rangle / (\boldsymbol{w}_i, \boldsymbol{Q}\boldsymbol{w}_i)_i$ and if $\boldsymbol{w}_i \cdot \langle \boldsymbol{x} \rangle < 0$ it is at zero (see figure 3).

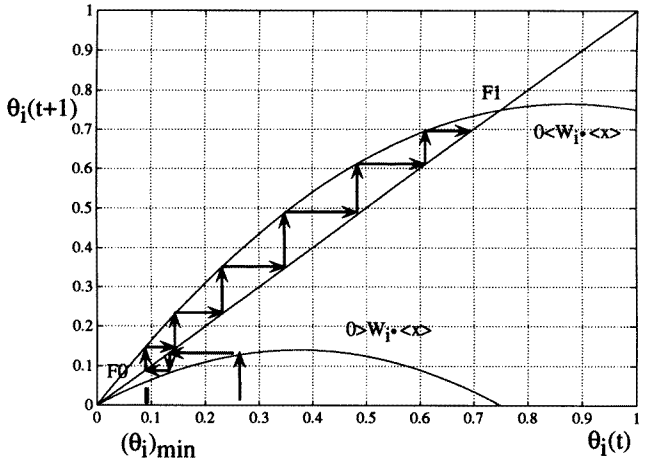

Figure 3. The effect of changing the sign of $\boldsymbol{w}_i$ in the approach to the stable fixed points.

If $\boldsymbol{w}_i \cdot \langle \boldsymbol{x} \rangle$ is < 0, θ_i moves towards the fixed point F0 at zero but, when it reaches $(\theta_i)_{\min}$, the change in the sign of the corresponding row $\boldsymbol{w}_i$ of connection matrix changes the dynamics and it is F1 that is now the attracting fixed point.

To illustrate the effect of node parameters in principal component analysis (PCA), consider the following two-dimensional x signal, where t_i are Gaussian distributed variables with zero mean and unit variance:

$$x_1 = t_1 \qquad x_2 = t_1 + 0.08t_2.$$

The principal components of the signal and the eigenvalues are given in table 1.

A one-layer network with node parameters as in figure 2 is used to perform PCA. Figure 4 shows the data and the principal directions that are obtained using the learning

Table 1.

λ_i	w_{i1}	w_{i2}
2.0032	0.7060	0.7082
0.0032	0.7082	−0.7060

laws (2.3*a*) and (2.13). The parameter values used are $\gamma_w = 0.015$, $\gamma_\theta = 0.005$ and $r = (0.002, 0.002)$. In figure 5 we display the convergence of the w to their final values in the learning process.

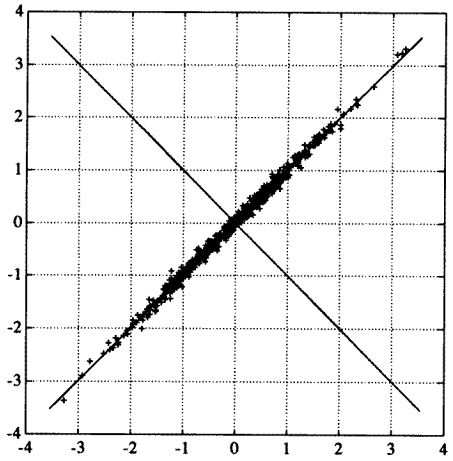

Figure 4. Signal distribution and its principal directions.

Figure 6 shows the variation of the node parameters.

Sanger's original algorithm is recovered by fixing the node parameters to unit values. In this case, the principal components are also extracted, but the convergence of the process is much slower, as shown in figure 7. With node parameters, improved convergence of

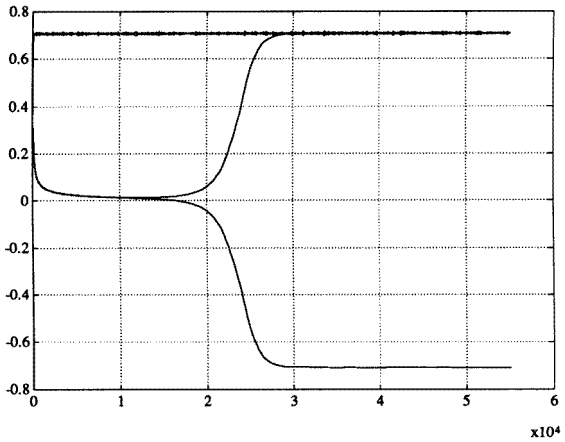

Figure 5. Evolution of w_1 and w_2.

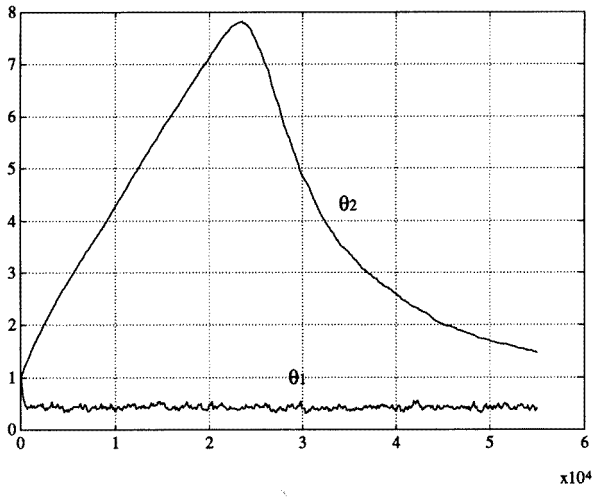

Figure 6. Evolution of θ_1 and θ_2.

the small eigenvalues is to be expected under generic conditions because the rates of convergence are controlled by $\gamma_w\theta_i$ (2.12) and θ_i is proportional to $1/\lambda_i$ ((2.6*b*) and (2.14)). In the example, the effect of θ_2 is further amplified by the fact that, before $\boldsymbol{w}_2$ starts converging, $\boldsymbol{w}_2 \cdot \boldsymbol{r}$ is large. Hence, in this case at least, the node parameter acts like a variable learning rate for the minor component. Notice that the method does not induce oscillations in the major components as would happen with large γ_w values.

Besides speeding up the learning process the node parameters also contain information about the first moments and the eigenvalues.

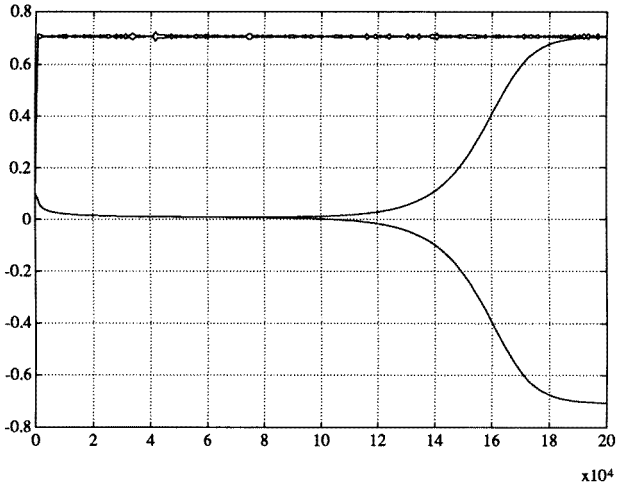

Figure 7. Evolution of $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ using Sanger's algorithm, without node parameters.

3. Competitive learning with a node parameter

In Hebbian learning the parameters of many units control the learning process at each step. By contrast, in competitive learning only one unit, or only one in each group, is in control each time. The units compete among themselves to be the one that is active. There are two basic types of competitive learning. In *simple competitive learning* there is no geometrical relationship between the winning units, whereas in *feature mapping* spatial correlations develop in the network.

We start by discussing simple competitive learning with a node parameter. Consider a single layer of units, each fully connected to a set x_i of input signals by strengths $w_{ij} > 0$. We denote by ξ the vector of normalized inputs

$$\xi_i = \frac{x_i}{\sqrt{\sum_k x_k^2}}. \tag{3.1}$$

The output y_i of unit i is

$$y_i = \theta_i f\left(\sum_k w_{ik} x_k\right) = \theta_i f\,(\boldsymbol{w}_i \cdot \boldsymbol{x}) \tag{3.2}$$

f being a positive non-decreasing function of its argument.

Each time an input signal is presented to the network, the unit i^* that wins the competition is the one with the largest output

$$\theta_{i^*} f\,(\boldsymbol{w}_{i^*} \cdot \boldsymbol{x}) > \theta_j f\left(\boldsymbol{w}_j \cdot \boldsymbol{x}\right) \qquad j \neq i^*. \tag{3.3}$$

The learning laws for w_{ij} and θ_i are

$$w_{ij}(t+1) = w_{ij}(t) + \eta_w \delta_{ii^*}(\xi_j - w_{ij}(t)) \tag{3.4}$$

$$\theta_i(t+1) = \theta_i(t) + \varepsilon \eta_\theta \delta_{ii^*} y_i(t)(1 - y_i(t)) \tag{3.5}$$

where ε is either $+1$ or -1.

The w_{ij} learning law (3.4) tends to align the winning unit with the input signals, as is usual in competitive learning. The θ_i parameter, on the other hand, is a function of the occurrence frequency of the pattern corresponding to the winning unit. Under successive application of the same pattern ξ, the connection strengths of the winning unit tend to

$$w_{i^*j} = \xi_j \tag{3.6}$$

which is a stable fixed point if $\eta_w > 0$. If the input patterns ξ for which i^* is the winning unit vary in some range, there is no fixed point, but the learning process stabilizes anyway if η_w is made to decrease slowly to zero during the learning period. For the θ_{i^*} of the winning unit, under repetitive application of the same pattern, the fixed points are

$$\theta_{i^*}^{(1)} = \frac{1}{f(\boldsymbol{w}_{i^*} \cdot \boldsymbol{x})} \quad \text{and} \quad \theta_{i^*}^{(2)} = 0. \tag{3.7}$$

In the case $\varepsilon = +1$, if $0 < \eta_\theta f(\boldsymbol{w}_{i^*} \cdot \boldsymbol{x}) < 2$, $\theta_{i^*}^{(1)}$ is stable and $\theta_{i^*}^{(2)}$ is unstable.

In the case $\varepsilon = -1$ it is more convenient to change the learning law to

$$\theta_i(t+1) = \theta_i(t) - \eta_\theta y_i(t)(1 - y_i(t)) + r_i \tag{3.8}$$

with $r_i < \eta_\theta/4$. The fixed point

$$\theta_{i^*}^{(2)} = \frac{1}{2f(\boldsymbol{w}_{i^*} \cdot \boldsymbol{x})}\left(1 - \sqrt{1 - \frac{4r_{i^*}}{\eta_\theta}}\right)$$

is stable when

$$\eta_\theta f(\boldsymbol{w}_{i^*} \cdot \boldsymbol{x})\left(1 - \sqrt{1 - \frac{4r_{i^*}}{\eta_\theta}}\right) > 0.$$

In both the $\varepsilon = +1$ and $\varepsilon = -1$ cases the dynamics of the node parameters keep track of how frequently each unit is the winner. The effect is, however, quite different in each case. When $\varepsilon = +1$, if one starts from small θ and random w_{ij}, as soon as a unit wins for a vector $\boldsymbol{x}^*$, it becomes more sensitive and tends to win again for all vectors closer to $\boldsymbol{x}^*$. This means that a single unit becomes assigned to each cluster or neighbourhood. It might imply a rough classification of vector categories but, on the other hand, by avoiding the assignment of many units to the same cluster, it maintains these units available to classify rare event categories.

For $\varepsilon = -1$ the effect is the opposite. If a unit wins it becomes less likely to win again. It is useful to resolve fine structures in a diffuse set of patterns. The dynamical effect of the node parameter in this case is analogous to raising the threshold as in the 'conscience' mechanism (Grossberg 1976). This is called conscience because it is as though frequent winners feel guilty and reduce their chances of winning.

For feature mappings (Kohonen 1982) the laws (3.4) and (3.5) are generalized to

$$w_{ij}(t+1) = w_{ij}(t) + \eta_w \Lambda(i, i^*)(\xi_j - w_{ij}(t)) \tag{3.9}$$

$$\theta_i(t+1) = \theta_i(t) + \varepsilon\eta_\theta \Lambda(i, i^*) y_i(t)(1 - y_i(t)) + r_i \tag{3.10}$$

where δ_{ii^*} is replaced by a neighbourhood function $\Lambda(ii^*)$, which is 1 for $i = i^*$ and falls off with the distance $|i - i^*|$.

The aim of competitive learning networks is to cluster the input data signals. Figure 8 presents seven two-dimensional input vectors which are used to train a one-layer competitive network of four neurons. The function f, in (3.2), that we use is the identity function. In the figure small bullets show the initial values of the connection strengths.

The training set is sampled according to a random uniform distribution and the w_{ij} are updated using the rules (3.3) and (3.4) with fixed node parameter values equal to one. After training the connections strengths have the final values shown in figure 9. The final values

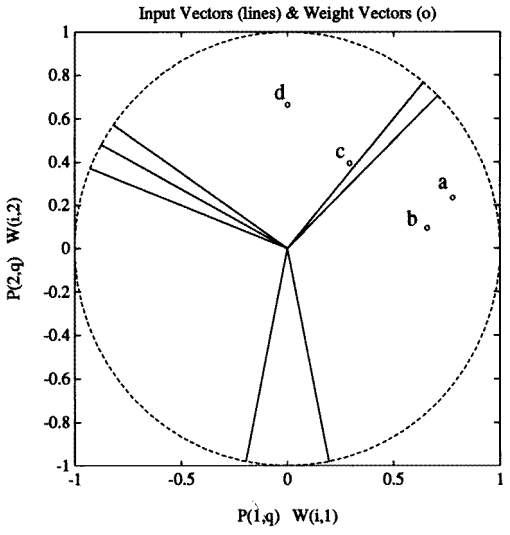

Figure 8. Training set and initial connection strengths for a competitive network.

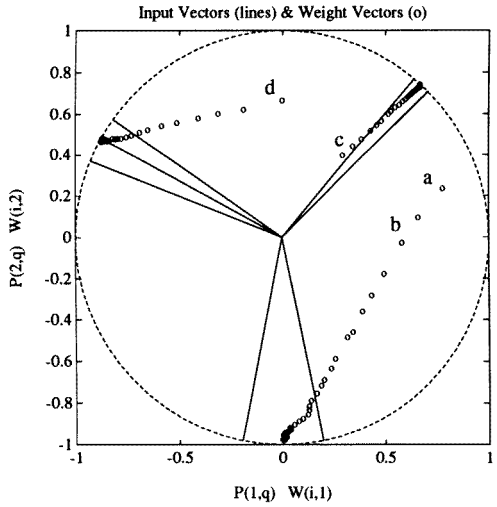

Figure 9. Connection strength values after training.

of three of the connection strengths correspond to the centres of gravity of the clusters in the training set and one of the strengths maintains its initial position. This result is also presented in figure 10 which shows the evolution of the connection strengths during training.

One of the neurons never wins during the training process as shown in figure 11 which plots the number of times each neuron wins in a typical trial run.

After training the two-dimensional space is clustered as shown in figure 12. The boundaries are determined by assigning random inputs inside the square to the neuron

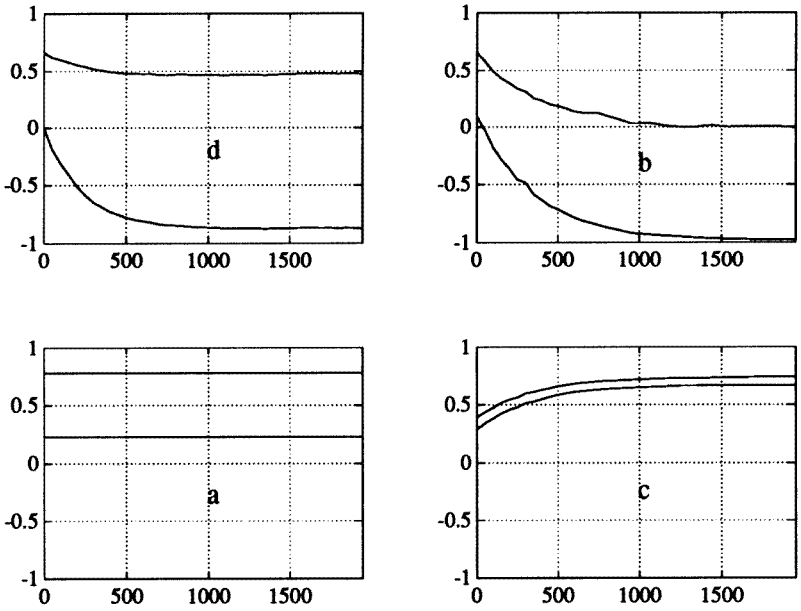

Figure 10. Evolution of the connection strengths during training.

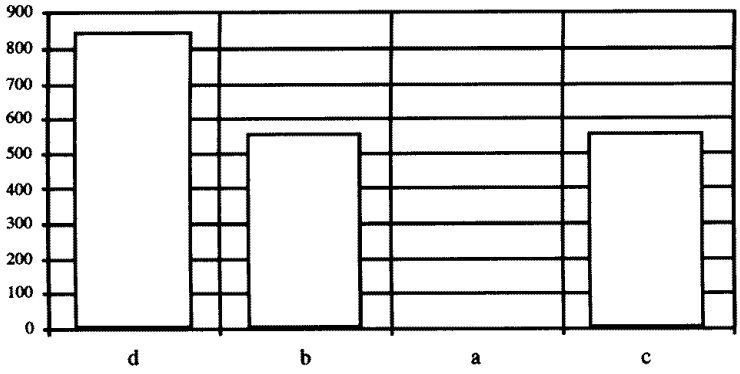

Figure 11. Winning frequency of each neuron.

that is most active under presentation of that input. Notice that there is one misleading cluster, which is not generated by the training set, but is due to the initial value of the $\mathbf{w}_i$ that is not modified during the training process.

Consider now the same training set, but with varying node parameters with updating laws (3.4) and (3.5) and $\varepsilon = 1$. The initial values of the connection strengths are the same as in the previous example and the initial θ_i values are made equal to a small value (0.05). The results are almost the same as in figures 8 to 11, but the two-dimensional input space partitioning is that of figure 13. That is, only three clusters are defined and they correspond to the training set. This results from the low value of the node parameter of the neuron that never wins (0.05). For the neurons that win many times the node parameter attains a final value near one.

If we use the updating rules (3.4) and (3.8) ($\varepsilon = -1$) the results are different. Figures 14, 15(*a*) and 16 show that all outputs are activated during training and the final connection strengths are determined by the input training set. The 'conscience mechanism' of the

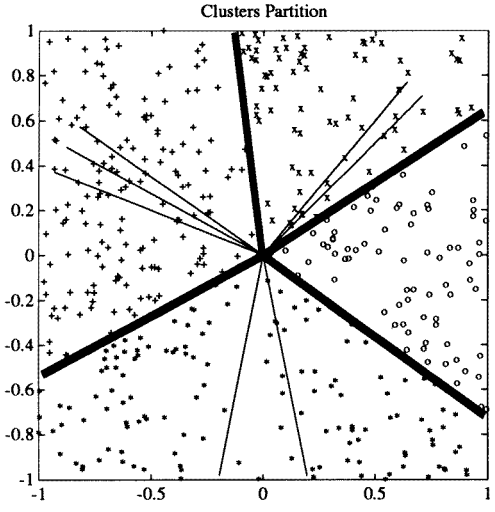

Figure 12. Two-dimensional input space partition after training.

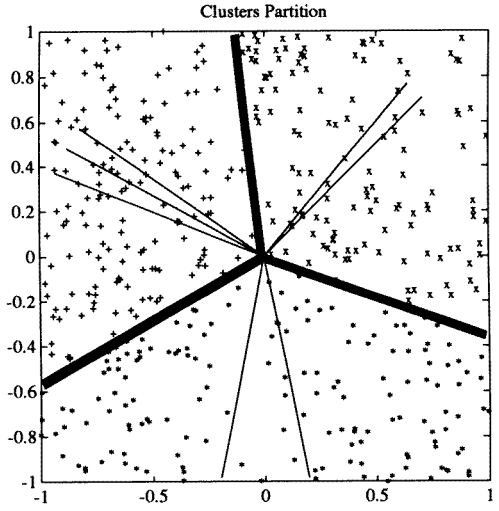

Figure 13. Two-dimensional input space partition after training using node parameters and $\varepsilon = +1$.

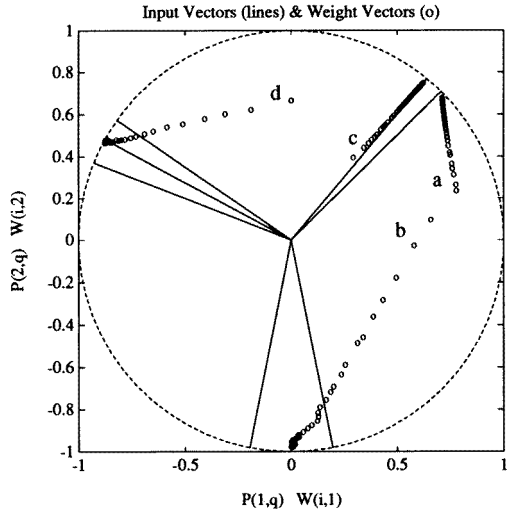

Figure 14. Connection strength values after training with node parameters and $\varepsilon = -1$.

($\varepsilon = -1$) learning rule enables all neurons to win part of the time and fine structures may be identified. The two-dimensional input space is clustered as shown in figure 17. This space is divided into four zones which are defined by the training set vectors. Notice that the initial values of θ_i must be greater than the final values they attain. The variation of the θ_i values in a ($\varepsilon = -1$) trial run is shown in figure 15(*b*).

For feature mapping, the use of node parameters, with learning laws (3.9) and (3.10), also seems to be useful. For $\varepsilon = -1$ a global participation of all neurons in the learning process is achieved, whereas for $\varepsilon = +1$ robust centres of gravity are created. At this moment we have no clear-cut preference concerning these two possibilities. However, we think that $\varepsilon = -1$ will be more adequate because it ensures the participation of all neurons, which is natural because feature mapping is a global process. Below, we present two examples of feature mapping with and without node parameters. In all cases, we use the same learning rate and neighbourhood functions. As usual, the learning rate and

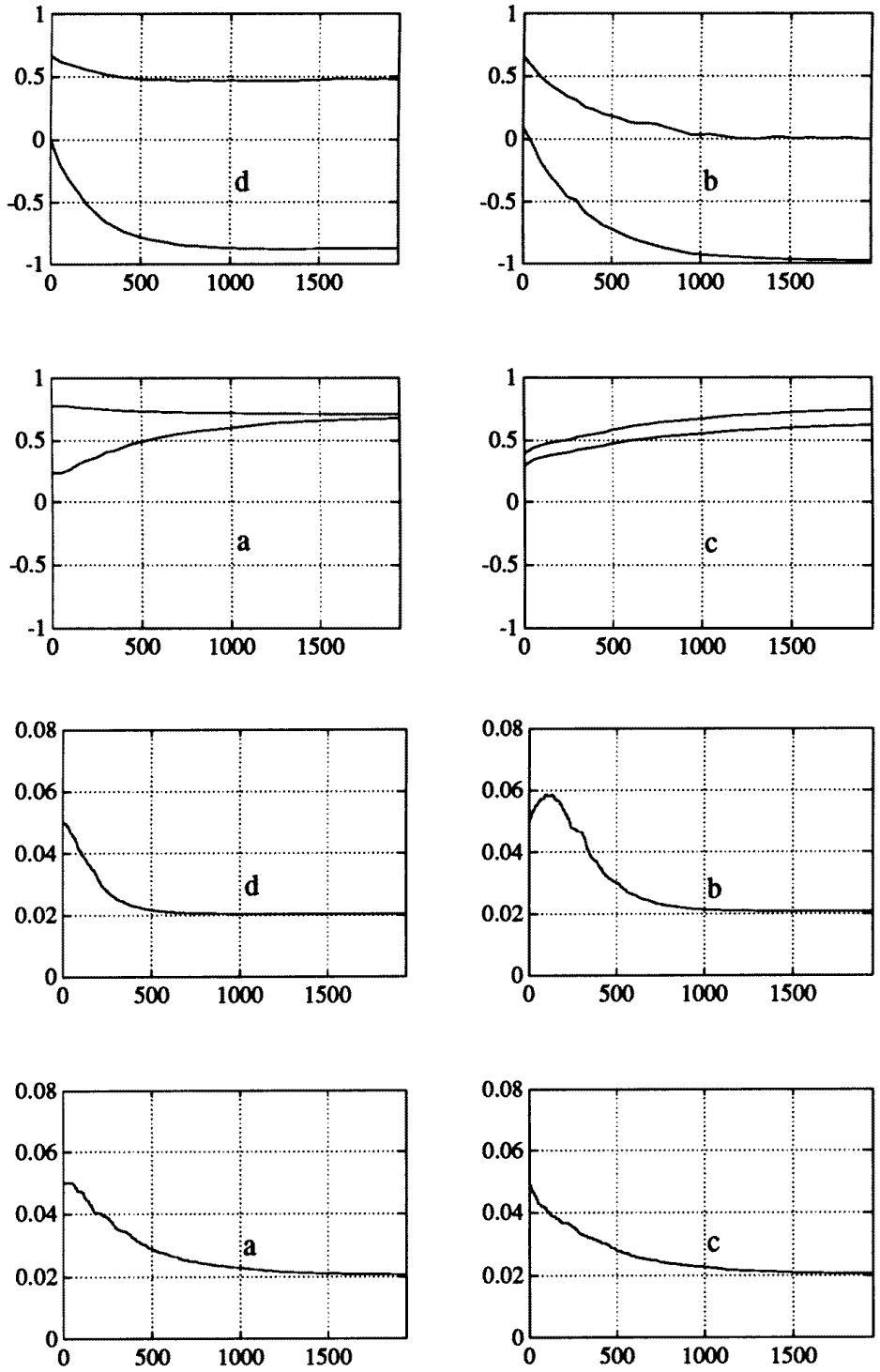

Figure 15. (*a*) Evolution of connection strengths during training with node parameters and $\varepsilon = -1$. (*b*) Evolution of the node parameters in a trial run $\varepsilon = -1$.

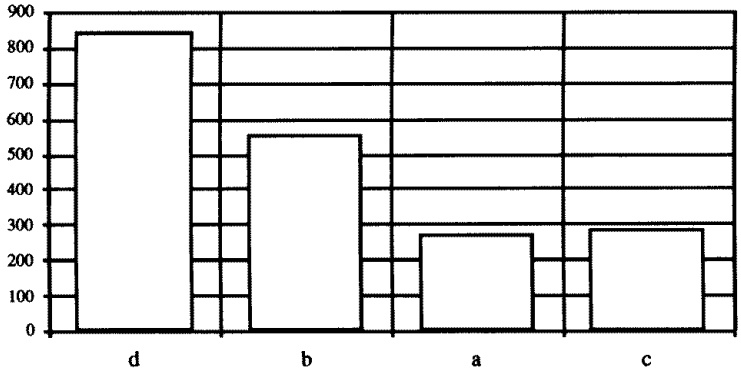

Figure 16. Winning frequency of each neuron with node parameters learning and $\varepsilon = -1$.

neighbourhood domain decrease linearly with time. We use the following expression for the neighbourhood function:

$$\Lambda_{(i,i^*)} = \exp\left(-\beta \frac{\|i - i^*\|^2}{d^2_{\max}} \right).$$

We use $\beta = 1.5$ and $d^2_{\max}$, in each trial run, varies from N_n to 1 following a linear law with saturation

$$d_{\max} = \begin{cases} N_n(1 - 4\nu) & \nu < (N_n - 1)/4N_n \\ 1 & \nu \geqslant (N_n - 1)/4N_n \end{cases}$$

for an $N_n \times N_n$ network. ν is the ratio m/M, m being the learning step number in a total of M learning steps. Figure 18 shows the results after $M = 20\,000$ steps of training a two-dimensional 8×8 neural network to map a triangle for three cases, namely without node parameters (*a*), $\varepsilon = -1$ (*b*) and $\varepsilon = 1$ (*c*). It is easy to verify that the node parameters

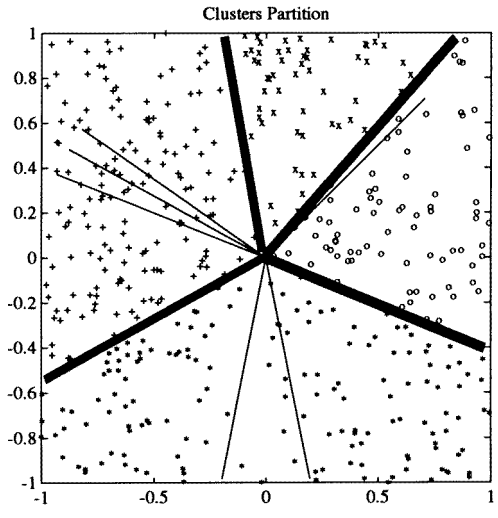

Figure 17. Two-dimensional input space partition after training using node parameters and $\varepsilon = -1$.

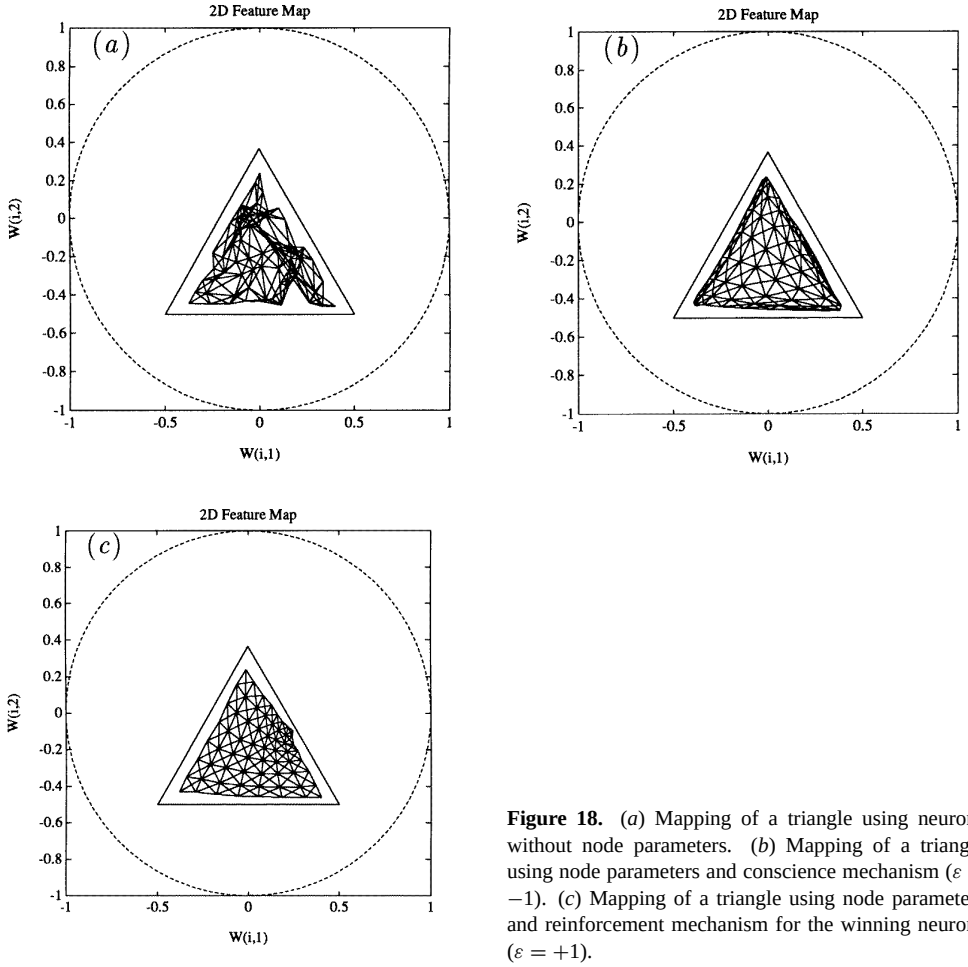

Figure 18. (*a*) Mapping of a triangle using neurons without node parameters. (*b*) Mapping of a triangle using node parameters and conscience mechanism ($\varepsilon = -1$). (*c*) Mapping of a triangle using node parameters and reinforcement mechanism for the winning neurons ($\varepsilon = +1$).

help the learning process. The conscience mechanism ($\varepsilon = -1$) creates a softer map than that created using the reinforcement algorithm ($\varepsilon = +1$). Notice that without using node parameters we might have obtained better performance than that shown in figures 18(*a*) and 19(*a*), by adjusting the learning rate carefully. What our results seem to show, however, is that the same effect is obtained in a more-or-less automatic way by the node parameter laws, while preserving a fast convergence rate.

Figure 19 shows similar results for the map of a non-convex region. The results are obtained after $M = 25\,000$ training steps for a two-dimensional 10×10 neuronal network.

In both the $\varepsilon = +1$ (conscience) and the $\varepsilon = -1$ (reinforcement) cases, our algorithm seems to perform satisfactorily. However, in the first case the weights cluster near the boundaries, whereas in the second they are more evenly distributed. The choice between these will depend on taste and the intended application.

References

Biehl M and Schwarze H 1995 Learning by on-line gradient descent *J. Phys. A: Math. Gen.* **28** 643
Doyne Farmer J 1990 A Rosetta stone for connectionism *Physica* **42D** 153

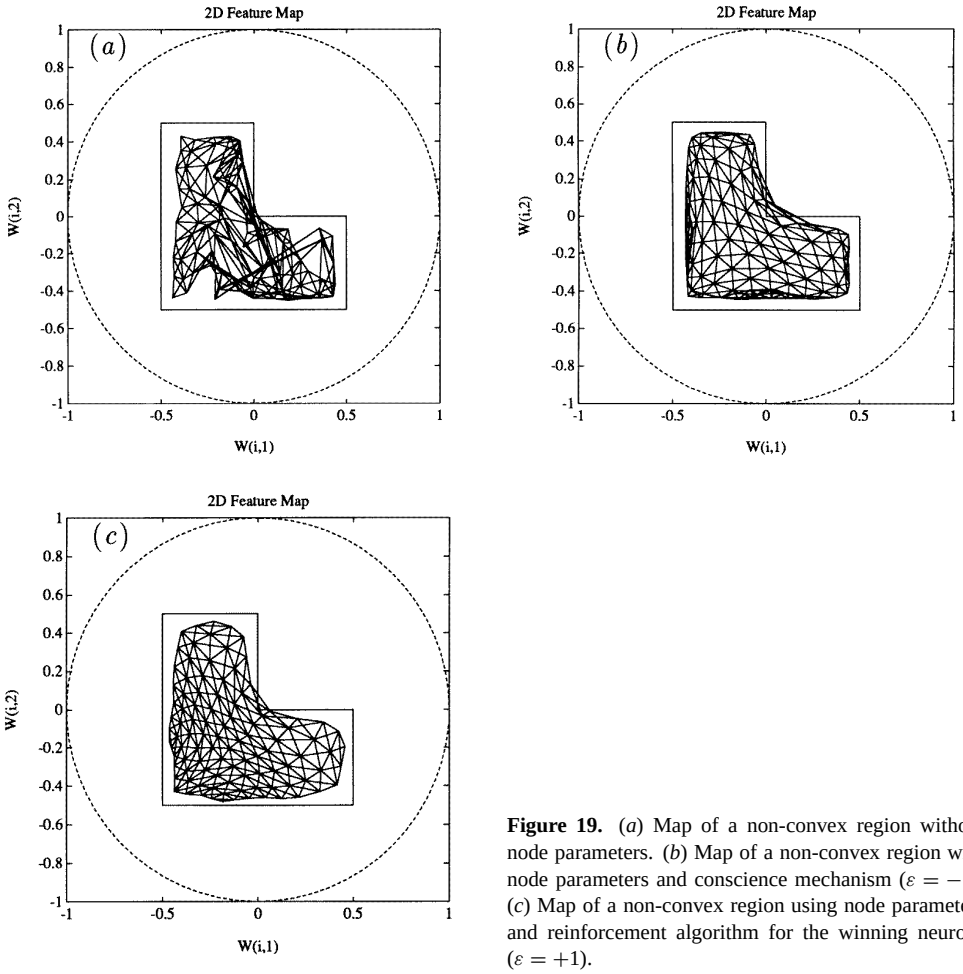

Figure 19. (*a*) Map of a non-convex region without node parameters. (*b*) Map of a non-convex region with node parameters and conscience mechanism ($\varepsilon = -1$). (*c*) Map of a non-convex region using node parameters and reinforcement algorithm for the winning neurons ($\varepsilon = +1$).

Grossberg S 1976 Adaptive pattern classification and universal recoding: II. Feedback, expectation, olfaction, illusions *Biol. Cybern.* **23** 187

Hebb D O 1949 *The Organization of Behaviour* (New York: Wiley)

Kohonen T 1982 Self-organised formation of topologically correct feature maps *Biol. Cybern.* **43** 59

Linsker R 1988 Self-organisation in a perceptual network *Computer* March 105

Oja E 1982 A simplified neuron model as a principal component analyser *J. Math. Biol.* **15** 267

Oja E 1989 Neural networks, principal components and subspaces *Int. J. Neural Sys.* **1** 61

Oja E 1992 Principal components, minor components and linear neural networks *Neural Networks* **5** 927

Sanger T D 1989 Optimal unsupervised learning in a single-layer linear feedforward neural network *Neural Networks* **2** 459

Yuille A L, Kammen D M and Cohen D S 1989 Quadrature and development of orientation selective cortical cells by Hebb rules *Biol. Cybern.* **61** 183