Unsupervised Learning in General Connectionist Systems

J. A. Dente¹ and R. Vilela Mendes²

Abstract

As shown by Doyne Farmer, there is a common framework where many different connectionist systems (neural networks, classifier systems, immune networks, autocatalytic reactions) may be treated in a unified way. The general system in which they may all be mapped is a network where not only the connection strengths but also the node parameters have learning plasticity.

In this paper we discuss Hebbian-type and competitive learning algorithms for networks with node parameters. In particular we find improvements on the convergence rate for small eigenvalues in principal component analysis (PCA). Improved discrimination or robustness is also found for competitive learning and feature mapping.

For non-Gaussian signals, PCA algorithms or higher order generalisations provide incomplete or misleading information on the statistical properties of the data. Therefore we propose an algorithm for on-line learning of the characteristic function as an unsupervised learning scheme.

¹ Laboratório de Mecatrónica. Departamento de Engenharia Electrotécnica e de Computadores. Instituto Superior Técnico, Av. Rovisco Pais, 1096 Lisboa Codex. Portugal.

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

sw_9502

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 occurs 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 [1] has shown, there is a common mathematical framework where 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 (Fig. 1). The node parameter represents the possibility of changing, through learning, the nature of the linear or non-linear function $f_i(\sum_j w_{ij}x_j)$ at each node. 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 a variable global node parameter θ_i . In some cases, as in the B-cell immune network, the node parameter is the only means to control the relative influence of a node on others, the connection strengths being fixed chemical reaction rates.

One of the purposes of this paper is to generalise some of the learning algorithms used in neural networks to the more general network depicted in Fig. 1. In Sect. 2 we discuss unsupervised learning of Hebbian-type and in Sect. 3 competitive unsupervised learning. In all cases the learning algorithms act both on the connection strengths and the node parameters. In Sect. 4 we discuss how algorithms that perform principal component analysis (PCA) only provide incomplete information on the data. This information may even be misleading if the probability distribution has no moments. We then discuss an algorithm to extract the characteristic function of the data.

Fig. 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 will denote by x_i the output of node i . Hebbian learning [2] 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

$$\Delta w_{ji} = \eta x_j x_i \quad (2.1)$$

there is the problem that the weights may keep on growing without bound. Several schemes were proposed [3-5] to deal with this problem and keep the weights bounded.

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

$$\mathbf{Q}_{ij} = \langle x_i x_j \rangle \quad (2.2)$$

If the learning law is local, the lines of the connection matrix w_{ij} all tend to the eigenvector associated to the largest eigenvalue of the correlation matrix. To obtain the other eigenvector directions one needs non-local laws [6-8]. Sanger's [6] approach has the advantage of organising the connection matrix in such a way that the lines are the eigenvectors associated to the eigenvalues in decreasing order. It suffers however from slow convergence rates for the lowest eigenvalues (see below). The methods that have been proposed may, with small modifications, be used both for linear and non-linear 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 and this may already be obtained with linear units, we will discuss only this case.

The updating learning rules that we propose are a generalisation 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 will see, θ_i carries information independent from the w_{ij} 's and it plays a role in the convergence process, it is useful to deal with it as an independent parameter.

We consider a one-layer feedforward network (Fig.2) and the updating rules proposed for w_{ij} and θ_i are:

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

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

y_i is the output of node i

$$y_i = \theta_i \sum_j w_{ij} x_j \quad (2.4)$$

and γ_w and γ_θ are positive constants that control the learning rate.

Define the N -dimensional vectors

$$(\langle \mathbf{x} \rangle)_i = \langle x_i \rangle \quad (2.5a)$$

$$(\mathbf{w}_i)_j = w_{ij} \quad (2.5b)$$

We now prove the following statements:

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

a) the system (2.3) has a stable fixed point such that

$$\mathbf{Q} \mathbf{w}_i = \lambda_i \mathbf{w}_i \quad \lambda_1 \geq \lambda_2 \geq \dots \lambda_N \quad (2.6a)$$

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

and $\|\mathbf{w}_i\| = 1$. λ_i ($i=1, \dots, N$) are the eigenvalues of the correlation matrix. The fixed point is unique if all the eigenvalues are different.

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

Proof:

The hypothesis about the time scale of the learning equations (2.3) means that all functions of the input signal $\{x_i\}$ may be replaced by their expectation values. Taking average values in Eq. (2.3a), one obtains, for $i=1$, the fixed point condition

$$\mathbf{Q}\mathbf{w}_1 = (\mathbf{w}_1, \mathbf{Q}\mathbf{w}_1)\mathbf{w}_1 \quad (2.7)$$

$$\text{hence } (\mathbf{w}_1, \mathbf{Q}\mathbf{w}_1) = \lambda_1 \quad |\mathbf{w}_1|^2 = 1$$

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

$$\mathbf{Q}\mathbf{w}_k = \lambda_k \mathbf{w}_k \quad k = 1, \dots, i-1 \quad (2.8)$$

with $|\mathbf{w}_k| = 1$.

Then for $k=i$ the fixed point condition is

$$\mathbf{Q}\mathbf{w}_i = \sum_{k=1}^i (\mathbf{w}_i, \mathbf{Q}\mathbf{w}_k) \mathbf{w}_k \quad (2.9)$$

Writing

$$\mathbf{w}_i = \sum_{k=1}^{i-1} \alpha_k \mathbf{w}_k + \mathbf{w}_\perp \quad (2.10)$$

with $\mathbf{w}_\perp$ orthogonal to all $\mathbf{w}_k$'s. ($k=1, \dots, i-1$) one obtains from (2.9)

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

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

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

that is, $\mathbf{Q}\mathbf{w}_i = \lambda_i \mathbf{w}_i$ and $|\mathbf{w}_i| = 1$. The induction is complete and one proves that the set of N vectors $\mathbf{w}_i$ $i=1, \dots, N$ such that $\mathbf{Q}\mathbf{w}_i = \lambda_i \mathbf{w}_i$ $\lambda_1 \neq \lambda_2 \neq \dots \neq \lambda_N$ is a fixed point of the dynamics defined by

Eq. (2.3a). Using this solution in Eq. (2.3b), averaged over the time scale of the input data, one obtains the fixed point in Eq. (2.6b).

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

$$\mathbf{w}'_i = \mathbf{w}_i + \delta_i$$

$$\theta'_i = \theta_i + \varepsilon_i$$

Then

$$\begin{aligned} \delta_i(t+1) - \delta_i(t) = \gamma_w \theta_i \left\{ (\mathbf{Q} - \lambda_i) \delta_i - \sum_{k=1}^i \lambda_k (\delta_i, \mathbf{w}_k) \mathbf{w}_k \right. \\ \left. - \sum_{k=1}^i \lambda_i (\mathbf{w}_i, \delta_k) \mathbf{w}_k \right\} + O(\delta^2) \end{aligned} \quad (2.11)$$

Expanding δ_i on the basis of the fixed point vectors

$$\begin{aligned} \delta_i = \sum_{l \neq i} c_{il} \mathbf{w}_l \\ \sum_{l \neq i} \{c_{il}(t+1) - c_{il}(t)\} \mathbf{w}_l = \gamma_w \theta_i \left\{ \sum_{l > i} c_{il} (\lambda_l - \lambda_i) \mathbf{w}_l \right. \\ \left. - \lambda_i \sum_{l < i} c_{il} \mathbf{w}_l - \sum_{l < i} \lambda_i c_{il} \mathbf{w}_l \right\} + o(\delta^2) \end{aligned} \quad (2.12)$$

Now, if all $\mathbf{w}_k$'s 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

$$\lambda_1 \geq \lambda_2 \geq \dots \lambda_N$$

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

For the stability of the θ_i fixed point we have

$$\begin{aligned} \varepsilon_i(t+1) - \varepsilon_i(t) &= \gamma_\theta \sum_i \{(\mathbf{w}_i, \langle \mathbf{x} \rangle) - 2\theta_i \lambda_i\} + o(\varepsilon^2) = \\ &= -\varepsilon_i \gamma_\theta \theta_i \lambda_i + o(\varepsilon^2) \end{aligned}$$

and the solution is stable if $\gamma\theta_i\lambda_i > 0$. With $\gamma > 0$ this uniquely fixes the direction of $\mathbf{w}_i$ to insure the positivity of $(\mathbf{w}_i \cdot \langle \mathbf{x} \rangle)$. 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 therefore unique without any sign ambiguity in $\mathbf{w}_i$. 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 $\mathbf{w}_i$ required the assumption that all $\mathbf{w}_k$'s 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 $\mathbf{w}$ of the connection strengths extracts the principal components of the correlation matrix $\mathbf{Q}$. The node parameters θ_i at their fixed point (2.6b) extract additional information on the mean value of the data vector $\langle \mathbf{x} \rangle$ and the eigenvalues of $\mathbf{Q}$. To deal with data with zero mean it is convenient to change the θ_i updating law to

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

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

The stable fixed point is now

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

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

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

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

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

Choosing $\alpha > 2$ accelerates convergence for the small eigenvalues.

2 - In addition to its role in extracting additional information on the input signal, the node parameters θ_i also plays a role in accelerating the convergence to the stable fixed point. From Eq. (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 to the smallest eigenvalues. This is the reason for the convergence problems in Sanger's method and one also finds (see below) that sometimes the results for the small components are quite misleading. On the other hand to increase γ_w does not help because then the time scale of the w_{ij} learning law is no longer much smaller than the time scale of the data and one obtains large fluctuations in the principal components.

With a node parameter θ_i and the learning law (2.3b) 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.

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

$$x_1 = t_1$$

$$x_2 = t_1 + 0.08 * t_2$$

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

λ_j	w_{i1}	w_{i2}
2.0032	0.7060	0.7082
0.0032	0.7082	-0.7060

A one - layer network with node parameters as in Fig. 2 is used to perform PCA. Fig. 3 shows the data and the principal directions obtained by the algorithm. In Fig. 4 we display the convergence of the w 's to their final values in the learning process.

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

Fig. 3 - Signal distribution and its principal directions.

Fig. 4 - Evolution of w_1 and w_2 .

Fig. 5 shows the evolution of the eigenvalues for the final learning steps and Fig. 6 shows the variation of the node parameters.

Fig. 5 - Final evolution of the eigenvalues.

Fig. 6 - Evolution of θ_1 and θ_2 .

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 Fig. 7. The node parameters therefore act like a variable learning rate for the minor

components. Notice that the method does not induce oscillations in the major components as would happen with large γ_w values.

Fig. 7 - Evolution of w_1 and w_2 using Sanger's algorithm, without node parameters.

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

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 normalised inputs

$$\xi_i = \frac{x_i}{\sqrt{\sum_k x_k^2}} \quad (3.1)$$

The output y_i of unit i is

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

f being a non-decreasing function.

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

$$\theta_{i^*} f(\mathbf{w}_{i^*} \cdot \mathbf{x}) > \theta_j f(\mathbf{w}_j \cdot \mathbf{x}) \quad j \neq i^* \quad (3.3)$$

The learning laws for w_{ij} and θ_i are

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

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

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, measures the occurrence frequency of the pattern corresponding to the winning unit. Under successive application of the same pattern ξ , the connection strengths of winning unit tend to

$$w_{i^*j} = \xi_j \quad (3.6)$$

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 stabilises 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

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

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

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

$$\theta_i(t+1) = \theta_i(t) - \eta_\theta y_i(t) (1 - y_i(t)) + r_i \quad (3.8)$$

and the fixed point $\theta_{i^*}^{(2)} = r_i$ is stable if $\eta_\theta f(\mathbf{w}_{i^*} \cdot \mathbf{x}) < 2$

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 's and random w_{ij} 's, as soon as a unit wins for a vector $\mathbf{x}^*$, it becomes more sensitive and it tends to win again for all vectors closer to $\mathbf{x}^*$. It 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 that many units be assigned to the same cluster it maintains these units available to classify rare event categories.

For $\varepsilon=-1$ the effect is the opposite one. 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 [9]. It is called conscience because it is as if frequent winners were to feel guilty and reduce their winning chances.

For feature mappings [10] the laws (3.4) and (3.5) are generalised to

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

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

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

Fig. 8 - Training set and initial connection strengths of a competitive network.

The aim of competitive learning networks is to cluster the input data signals. Fig. 8 presents seven two-dimensional input vectors which are used to train a one layer competitive network of four neurons. In this figure small bullets show the initial values of the connection strengths.

Fig. 9 - Connection strength values after training.

Fig. 10 - Evolution of the connection strengths during training.

The training set is sampled according to a random uniform distribution and the w_{ij} 's 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 Fig. 9. The final value of three of the connection strengths corresponds 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 Fig. 10 that shows the connection strengths evolution during training.

One of the neurons never wins during the training process as shown in Fig. 11 that plots the winning frequency of each neuron.

Fig. 11 - Winning frequency of each neuron.

After training the two-dimensional space is clustered as shown in Fig. 12. Notice that there is one misleading cluster, because it is not generated by the training set, but is due to the initial value of the w_i that is not modified during the training process.

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

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 Figs 8 to 11, but the two-dimensional input space partitioning is the one in Fig. 13. That is, only three clusters are defined and they correspond to the training set. This result is explained by 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.

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

If we use the updating rules (3.4) and (3.8) ($\varepsilon=-1$) the results are different. Figs 14, 15 and 16 show that all outputs are activated during training and the final connection strengths values are determined by the input training set. The "conscience mechanism" of the ($\varepsilon=-1$) 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 Fig. 17. This space is divided in four zones which are defined by the training set vectors. Notice that the initial values of θ_i must be greater than the final value that they attain ($\approx r_i$).

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

Fig. 15 - Evolution of connection strengths during training with node parameters and $\epsilon = -1$.

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

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

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 centers of gravity are created. At this moment we have not a clear cut preference concerning these two possibilities. However, we think that $\varepsilon=-1$ will be more adequate because it insures the participation of all neurons and this is natural because feature mapping is a global process. Below we present two example 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 neighbourhood domain decrease linearly with time. We use for the neighbourhood function the following expression:

$$\Lambda_{(i,i^*)} = \exp(-\beta \frac{\|i - i^*\|^2}{d_{\max}^2})$$

Figure 18 shows the results after 20000 steps of training a two-dimensional 8x8 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 help the learning process. The conscience mechanism ($\varepsilon=-1$) creates a softer map than the one created using the reinforcement algorithm ($\varepsilon=+1$). Notice that without using node parameters we might have obtained better performance than that shown in Fig. 18a and 19a, by adjusting carefully the learning rate. 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.

Fig. 18. a - Mapping of a triangle using neurons without node parameters.

Fig. 18 b - Mapping of a triangle using node parameters and conscience mechanism ($\epsilon=-1$).

Fig. 18 c - Mapping of a triangle using node parameters and reinforcement mechanism for the winning neurons ($\epsilon=+1$).

Fig. 19 shows similar results for the map of a convex region. The results are obtained after 25000 training steps of a 2-dimensional 10x10 neuronal network.

Fig. 19 a - Map of a convex region without node parameters.

Fig. 19 b - Map of a convex region with node parameters and conscience mechanism ($\varepsilon=-1$).

Fig. 19 c - Map of a convex region using node parameters and reinforcement algorithm for the winning neurons ($\varepsilon=+1$).

4. - LEARNING HIGHER ORDER CORRELATIONS AND THE CHARACTERISTIC FUNCTION

The principal component analysis (PCA) algorithms discussed in Sect. 2 determine the characteristic directions of the correlation matrix $(Q)_{ij} = \langle x_i x_j \rangle$. If the data has zero mean ($\langle x_i \rangle = 0$) these are the orthogonal directions along which the data has maximum variance. If the data is Gaussian in each channel, it is distributed as a hyperellipsoid and the correlation matrix Q contains all possible information about the statistical properties. That is, higher order moments of the data may be obtained from the second order moments. However if the data is not Gaussian distributed, the PCA analysis is not a complete one and higher order correlations are needed to characterise the statistical properties.

This led some authors [11-12] to propose networks with higher order neurons to obtain the higher statistical correlations of the data. An higher order neuron is one that is capable of accepting, in each of its input lines, data from two or more channels at once. Therefore there is a set of adjustable strengths $w_{ij_1}, w_{ij_1 j_2} \dots w_{ij_1 \dots j_n}$, n being the order of the neuron.

Networks with higher order neurons have very interesting applications, for example in fitting data to a high-dimensional hypersurface. However there is a basic weakness in the characterisation of the statistical properties of unknown non-Gaussian data by higher order moments. Because the

existence of the moments of a distribution function depend critically in the behaviour of this function at infinity it happens frequently that a distribution has moments up to a certain order, but no higher ones. A well-behaved probability distribution might even have no moments of order higher than one (the mean). In addition a sequence of moments does not necessarily determine a probability distribution function uniquely. Two different distributions may have the same set of moments. Therefore for non-Gaussian data the PCA algorithms, or higher order generalisations, may just lead to misleading results.

As an example considers the signal distribution shown in Fig. 20a. Fig. 20b shows the evolution of the connection strengths w_{11} and w_{12} when this signal is passed through Sanger's PCA algorithm. Large oscillations appear and finally the algorithm overflows. Smaller learning rates do not introduce qualitative modifications in this evolution. The values may at times appear to stabilize, but large spikes do occur.

The reason is that the seemingly harmless data in Fig. 20a is generated by a linear combination of a Gaussian with the following distribution

$$p(x) = K(2 + x^2)^{-\frac{3}{2}}$$

which has first moment, but no moments of higher order.

Fig. 20 a - Signal distribution.

Fig. 20 b - Evolution of connection strengths w_{11} and w_{12} .

Moments of an arbitrary probability distribution may not exist. However, because every bounded and measurable function is integrable with respect to any distribution, the existence of the characteristic function $f(t)$ is always assured.

$$f(t) = \int e^{it \cdot x} dF(x) = \langle e^{it \cdot x} \rangle \quad (4.1)$$

t and x are N-dimensional vectors, x is the data vector and $F(x)$ its distribution function.

The characteristic function providing a precise and compact description of the statistical properties of the data, it would be interesting to develop efficient algorithms for on-line calculation of this function. In Fig. 21 we sketch the algorithm we have been using.

Fig. 21 - Diagram of an algorithm to extract the characteristic function of a signal.

For each data vector x , $\cos(\alpha \cdot x) + i \sin(\alpha \cdot x)$ is formed for a finite set of different α vectors. For each x and all α 's the current average

$$g(\alpha) = \frac{1}{k} \sum_{i=1}^k \cos \alpha \cdot x_i + i \sin \alpha \cdot x_i = \frac{1}{k} S_k(\alpha) \quad (4.2)$$

is presented as a goal to a feed-forward network and the connection strengths updated by backpropagation. In the large k limit, the feed-forward network will provide an approximation of the characteristic function. In the algorithm, the calculation of the cosine and the sine and the moving average are performed computationally. All steps may however be implemented in hardware

Figure 22 shows the theoretical value and the value computed by our algorithm for the real and imaginary parts of the characteristic function corresponding to the signal in Fig. 20. For the real part the average error is 2% and for imaginary part 5%.

Fig. 22a - Computed (left) and theoretical (right) real part of the characteristic function.

Fig. 22b - Computed (left) and theoretical (right) imaginary part of the characteristic function.

REFERENCES

- [1] J. Doyne Farmer: "A Rosetta stone for connectionism", *Physica* D42,153(1990).
- [2] D. O. Hebb; "The organisation of behaviour", Wiley, New York,1949.
- [3] E. Oja; "A simplified neuron model as a principal component analyser" *J. of Math. Biology* 15, 267 (1982).
- [4] R. Linsker; "Self-organisation in a perceptual network", *Computer*, March,105 (1988).
- [5] A. L. Yuille, D. M. Kammen and D. S. Cohen; "Quadrature and development of orientation selective cortical cells by Hebb rules", *Biological Cybernetics* 61,183 (1989).
- [6] T. D. Sanger; "Optimal unsupervised learning in a single-layer linear feedforward neural network", *Neural Networks* 2, 459 (1989).
- [7] E. Oja; "Neural networks, principal components and subspaces", *Int. J. of Neural Systems* 1, 61 (1989).
- [8] E. Oja; "Principal components, minor components and linear neural networks", *Neural Networks* 5, 927 (1992).
- [9] S. Grossberg; "Adaptive pattern classification and universal recoding: II. Feedback, expectation, olfaction, illusions", *Biological Cybernetics* 23, 187 (1976).
- [10] T. Kohonen; "Self-organised formation of topologically correct feature maps", *Biological Cybernetics*, 43 59 (1982).
- [11] W. R. Softky and D. M. Kammen; "Correlations in high dimensional or asymmetric data sets: Hebbian neuronal processing", *Neural Networks* 4, 337 (1991).
- [12] J. G. Taylor and S. Coombes; "Learning higher order correlations", *Neural Networks* 6, 423 (1993).
- [13] E. Lukacs; "Characteristic functions", Griffin, London 1970.