Proc. of the 1992 IEEE
Int. Conf. on Tools with AI
Arlington, VA, Nov. 1992

The Minimum Entropy Network

Rüdiger W. Brause

J.W. Goethe-University,
FB20, NIPS, Postbox 111932,
D- 6000 Frankfurt 11, FRG

Abstract

Some researchers in pattern recognition theory claim that for pattern recognition and classification clustering transformations are needed which reduce the intra-class entropy. This is implemented for Gaussian sources by a linear transformation using the eigenvectors with the smallest *eigenvalues.*

This paper shows that using as basic building block a linear neuron with an Anti-Hebb rule and restricted weights, an asymmetric network which computes the eigenvectors in the ascending order of their corresponding eigenvalues can be build. The conditions for their convergence are obtained and demonstrated by simulations.

1 Introduction

For many purposes the necessary processing of sensor input signals is realized by using a system which implements the maximization of the transinformation from the input to the output of the system. For deterministic systems, this corresponds to the maximization of the output entropy (maximum entropy principle). In pattern recognition theory, it is well known that for Gaussian distributed sources this corresponds to the minimization of the mean square error of the output. For linear systems, this is done by a linear transformation to base of the eigenvectors of the autocorrelation matrix [5].

Furthermore, we can compress (encode) the input information by using only the base vectors (eigenvectors) with the biggest eigenvalues. Neglecting the ones with the smallest eigenvalues results in the smallest reconstruction error of the encoded input [3]. Generally, this approach can be used for sensor signal coding such as picture encoding, see e.g. [4].

The neural network implementations of this approach use linear neurons, where each neural weight vector corresponds to one eigenvector. Examples of those architectures are the Oja subspace network [7] and the SEC algorithm of Williams [12] which transforms the input into the subspace of the eigenvectors with the biggest eigenvalues (the principle components), the Sanger decomposition network [10] and the lateral inhibition network of Rubner and Tavan [9] or Földiák [2]. In the weight vector learning phase, the last mentioned networks decompose sequentially the input vector **x**, see figure 1.

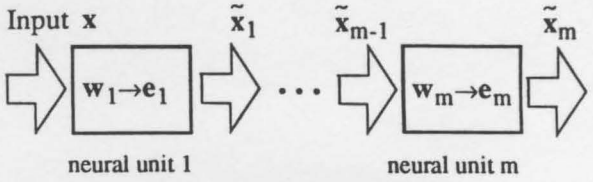

Fig.1 The sequential eigenvector decomposition by basic building blocks

They use as a basic building block the linear correlation neuron which learns the input weights by a Hebb-rule, restricting the length |w| of the weights $w=(w_1,..,w_n)$. As Oja showed [6], this learning rule let the weight vector **w** of the neuron converge to the eigenvector e_k of the expected autocorrelation matrix C of the input patterns **x** with the biggest eigenvalue λ_{max}:

$$C = \langle xx^T \rangle$$

$$w \rightarrow e_k \quad \text{with} \quad \lambda_k = \max_i \lambda_i \quad \text{and} \quad Ce_i = \lambda_i e_i$$

2 The minimum entropy principle

The maximum entropy principle maximizes the entropy, i.e. for Gaussian sources it minimizes the mean square error of the output coding. This aimes to minimize the reconstruction error for the input data from the encoded output.

In many applications, this is not the appropriate goal. If we want just to identify an object, we are not interested in the noisy representation of the object but in the code for the pure prototype of the object neglecting all variances. In the language of pattern recognition, all noisy instances of

0-8186-2905-3/92 $03.00 © 1992 IEEE

the object form a data point cloud (a cluster) around the prototype in the n-dimensional feature space. Here the goal of the transformation consists of projecting the cloud of data points onto the prototype. This is done by removing some uncertainty from the data points: the entropy of the cluster is reduced. It was shown by Tou and Gonzales [11], that for Gaussian distributed clusters with uniform variance the cluster entropy is maximally reduced by the linear transformation on the basis of the eigenvectors of the covariance matrix. Here the most reliable feature is given by the projection of the input to the eigenvector with the *smallest* eigenvalue. This necessity for clustering transformations motivates the question: Can we implement such a transformation also by a neural network ?

3.1 The minimum entropy neuron

The base of all three cited eigenvector decomposition networks consists of a neural unit learning the eigenvector of the input autocorrelation matrix with the biggest eigenvalue. In analyzing this approach we can derive the proper learning rule for the eigenvector with the *smallest* eigenvalue $\mathbf{e}_k$ and prove the stability of the solution.

Let us assume an input $\mathbf{x}=(x_1,..,x_n)$ for one neuron. Traditionally, the input is weighted by the weights $\mathbf{w}=(w_1,...,w_n)$ and summed up to the activation z of the neuron

$$z(t) = \Sigma_i w_i x_i = \mathbf{w}^T\mathbf{x} \qquad y(t)=S(z) \quad (3.1)$$

which is expressed as the scalar product of x and w which is denoted by a product of the transpose $\mathbf{w}^T$ and the vector in this paper. Since we assume linear neurons, with the linear output function S(z)=z the output y(t) becomes z(t).

When we use m<n neurons, the resulting mean square coding error is the mean output variance $\langle(y - \bar{y})^2\rangle$ (see[3]) which becomes the output intensity for centralized input $\bar{\mathbf{x}}:=\langle\mathbf{x}\rangle=\mathbf{0}$ and therefore $\bar{y}:=\langle y\rangle=0$.

$$f(\mathbf{w}) = \langle(y - \bar{y})^2\rangle = \langle y^2\rangle = \langle \mathbf{w}^T\mathbf{x}\mathbf{x}^T\mathbf{w}\rangle = \mathbf{w}^T\mathbf{C}\mathbf{w} \quad (3.2)$$

Since we are not interested in uniformly squeezing or expanding the pattern space, the space volume should be conserved by the linear transformation defined in (3.1). Thus, we assume for the matrix W with rows $\mathbf{W}_k = \mathbf{w}_k$ of all base vectors $\mathbf{w}_k$ the equation det(W)=1 which is confirmed by the demand $|\mathbf{w}_k|=1$. This restriction of the weights is often used in learning systems to prevent the Hebbian learning rule from "blowing up" the weights.

Let us now investigate the necessary conditions for the local extrema of the objective function (3.2) with respect to the constrain $|\mathbf{w}|^2-1=0$. It is well known that the necessary conditions for the local extrema of a function using the Lagrange multiplier μ

$$\begin{aligned} L(w_1,...,w_n,\mu) &:= f(\mathbf{w}) + \mu(|\mathbf{w}|^2 -1) \\ &= \mathbf{w}^T\mathbf{C}\mathbf{w} + \mu(\mathbf{w}^T\mathbf{w} -1) \end{aligned} \quad (3.3)$$

represent the desired multivariate extremum conditions for the corresponding restricted objective function f(w)

$$\nabla_{\mathbf{w}} L(\mathbf{w},\mu) = 2\mathbf{C}\mathbf{w} + 2\mu\mathbf{w} = 0 \quad (3.4a)$$

$$\partial L(\mathbf{w},\mu)/\partial\mu = \Sigma_i w_i^2 - 1 = 0 \quad (3.4b)$$

The necessary extremum conditions (3.4a) provide as solutions the *n* eigenvectors $\mathbf{e}_k$ of the expected autocorrelation matrix C

$$\mathbf{C}\mathbf{w}^* = -\mu\mathbf{w}^* \quad \text{with } \mathbf{w}^* = \mathbf{e}_k,\ \lambda_k=-\mu \quad (3.5)$$

with the corresponding eigenvalues λ_k. In this case we have as variance

$$f(\mathbf{w}^*) = \langle y^2\rangle = \mathbf{w}^{*T}\mathbf{C}\mathbf{w}^* = \lambda_k \mathbf{w}^{*2} = \lambda_k \quad (3.6)$$

Unfortunately, the approach with Lagrangian multipliers does not determine what kind of extrema we do have. In appendix A it is shown by a different, more detailed approach that the fixpoint of the eigenvector with the maximal eigenvalue λ_{max} provides a maximum, the eigenvector with the minimal eigenvalue λ_{min} a minimum of the objective function f(w). Beside these two unique extrema all other fixpoints are unstable saddle-points. Thus, to reach the minimum we can use a simple gradient descend algorithm

$$\hat{\mathbf{w}}(t+1) = \mathbf{w}(t) - \gamma\, \text{grad}\, f(\mathbf{w}) = \mathbf{w}(t) - \gamma\, \mathbf{C}\mathbf{w}(t) \quad (3.7)$$
$$\text{and } \mathbf{w}(t+1) = \hat{\mathbf{w}}(t+1) / |\hat{\mathbf{w}}(t+1)| \qquad \textit{normalization}$$

The stochastic version of this algorithm is with $\mathbf{C}\mathbf{w}=\langle \mathbf{x}\mathbf{x}^T\mathbf{w}\rangle=\langle \mathbf{x}y\rangle$

$$\hat{\mathbf{w}}(t+1) = \mathbf{w}(t) - \gamma(t)\, \mathbf{x}(t)\, y(t) \quad \textit{Anti-Hebb-rule} \quad (3.8)$$
$$\text{and } \mathbf{w}(t+1) = \hat{\mathbf{w}}(t+1) / |\hat{\mathbf{w}}(t+1)| \quad \textit{normalization}$$

If the learning rate $\gamma(t)$ satisfies all the convenient conditions for the stochastic approximation process (e.g. $\gamma(t):=1/t$), the convergence of the approximation process is guaranteed, see e.g. [6]. If we replace the negative sign by the positive sign at (3.7) and (3.8), the gradient uphill climbing will provide us with the familiar Hebb-Rule for the maximal eigenvalue.

3.2 Visualization of Fixpoints and saddle points

As an example, let us regard a system with at least one saddle point which can be visualized by a 3D-plot. This is best done by a three-dimensional system of n=3. For a visualization by needle graphics, see [1].

Relative to a base system of eigenvectors (see appendix

A), we have

$$f(w) = \Sigma_i a_i^2 \lambda_i \qquad (A.1)$$

with the components, see Eqs.(A.2, A.3)

$$a_1^2 = |\tilde{w}|^2 \cos^2\beta,\ a_2^2 = |\tilde{w}|^2 \sin^2\beta,\ a_3^2 = \cos^2\alpha\ .$$

Substitution in

$$|\tilde{w}|^2 = a_1^2 + a_2^2 = 1 - a_3^2$$

gives us

$$f(w) = \lambda_1(1-\cos^2\alpha)\cos^2\beta + \lambda_2(1-\cos^2\alpha)\sin^2\beta + \lambda_3\cos^2\beta.$$

For $\lambda_1=\lambda_2=\lambda_3=1.0$ the objective function f(w) is constant, because the variance in all directions of the space is equal; there is neither an unique maximum nor a minimum. If one eigenvalue becomes smaller then the situation slightly changes. In figure 2 the corresponding objective function is plotted.

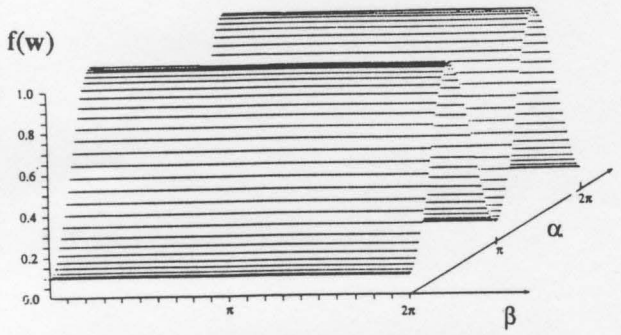

Fig. 2 The objective function with one small eigenvalue ($\lambda_1=\lambda_2=1.0$, $\lambda_3=0.1$)

The objective function becomes minimal at $\alpha=0,\pi$, i.e. at the eigenvector with the smallest eigenvalue. Here, the other angle β has no meaning. For the maximimum, the situation changes and with $\alpha=\pi/2$, $3\pi/2$ all possible values of β are solutions. This is quite instructive: With the variance in two directions beeing equal, the input space data clouds forms a disk where the direction of the smallest diameter can be determined, but not the biggest one.

If we choose all three eigenvalues different, figure 2 becomes figure 3. Here, the two fixpoints for a maximum (parallel and antiparallel to the eigenvectors) and the two for a minimum (each for β and α) mark the four fixpoints of the two directions of maximum and minimum variance. Additionally, between the "hills" at $\beta=\pi/2$ and $\alpha=\pi/2$ we have the third, unstable fixpoint: in the direction of β it is a minimum but in the direction α it is a maximum. To reach this fixpoint, all algorithms which are uphill gradient ascends (maximum search) have to balance the input

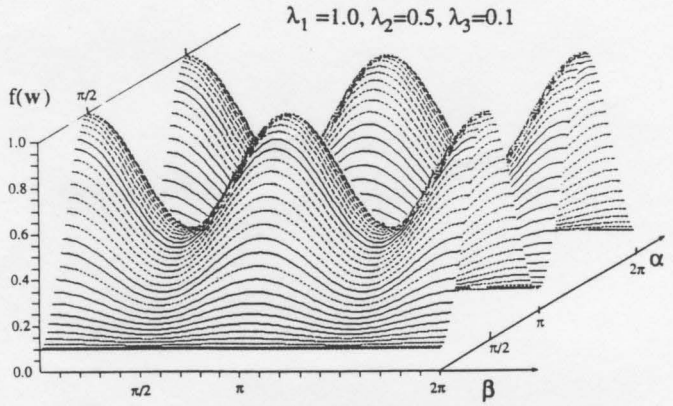

Fig.3 The objective function for three different eigenvalues

patterns such that thecomponents in β-direction have an expected value of $\beta^*=\pi/2$. On the other hand, all downhill algorithm (minimum search) have to maintain $\alpha=\pi/2$ to converge to the unstable point. This is the basis for all eigenvector decomposition networks.

3.3 Non-centered input

All the cited networks assume that the pattern statistics are centered, i.e. the expected input $\langle x\rangle$ is zero. Then the covariance matrix $\langle (x-\langle x\rangle)(x-\langle x\rangle)^T\rangle$ becomes the autocorrolation matrix $C_{xx}=\langle xx^T\rangle$. For the latter case, the normalized Hebbian (or Anti-Hebbian) rule let the weight vector converge to an eigenvector of the autocorrelation matrix. In the former case, we are in trouble - how can we learn the eigenvectors of the covariance matrix ?

This can be overcome by the following approach. Let us redefine the input $x^T=(x_1,..,x_n) \rightarrow \tilde{x}^T:=(x_1,..,x_n,1)$ by an additional, constant line. Then the corresponding input weight w_{n+1} of $\tilde{w}^T = (w^T, w_{n+1}) = (w_1,..,w_n,w_{n+1})$ is learned by the Anti-Hebbian rule (3.8)

$$w_{n+1}(t+1) = w_{n+1}(t) - \gamma(t+1)\, x_{n+1}(t+1)\, y(t+1) \qquad (3.11)$$

For the decreasing learning rate $\gamma(t):=1/t$ and the output $y(t+1)= \tilde{w}^T(t)\ \tilde{x}(t+1) = w^T(t)\ x(t+1) + x_{n+1} w_{n+1}$ this becomes with the linear activity (3.1)

$$\begin{aligned} w_{n+1}(t+1) &= w_{n+1}(t) - 1/(t+1)\ (z(t+1)+w_{n+1}(t)) \\ &= w_{n+1}(t)(1-1/(t+1)) - z(t+1)/(t+1) \end{aligned} \qquad (3.12)$$

At the 2-th iteration with $w_{n+1}(1):=0$ this becomes

$$w_{n+1}(2) = w_{n+1}(1) - 1/2\, z(2) \ = -\ 1/2\, \Sigma_{i=1}^{2} z(i)$$

Thus, by induction we have for (3.12) at the t+1-th iteration step with $(1-1/(t+1)) = t/(t+1)$

$$w_{n+1}(t+1) = t/(t+1)\, w_{n+1}(t) - z(t+1)/(t+1)$$

$$= -1/(t+1)\,\Sigma_{i=1}^{t+1} z(i) = -\langle z(t+1)\rangle \quad (3.13)$$

By the additional weight the output is now

$$y = z - \langle z\rangle \qquad \text{with } \langle y\rangle = \langle z - \langle z\rangle\rangle = 0$$

The output becomes centered as if the input was centered; the objective function (3.2) remains valid and the weight vector converges to the eigenvector of the augmented input correlation matrix

$$C_{\tilde{x}\tilde{x}}\tilde{w}^* = \langle \tilde{x}\tilde{x}^T\rangle \tilde{w}^* = \begin{bmatrix} \langle xx^T\rangle, \langle x\rangle \\ \langle x^T\rangle, 1 \end{bmatrix} \begin{pmatrix} w^* \\ -\langle z\rangle \end{pmatrix}$$

$$= \begin{pmatrix} \langle xx^T\rangle w^* - \langle x\rangle\langle z\rangle \\ \langle x^T w^*\rangle - \langle z\rangle \end{pmatrix}$$

$$= \begin{bmatrix} \langle xx^T\rangle - \langle x\rangle\langle x^T\rangle, & 0 \\ 0, & 0 \end{bmatrix} \begin{pmatrix} w^* \\ w_{n+1}^* \end{pmatrix} = \lambda \tilde{w}^*$$

Since the expression

$$\langle xx^T\rangle - \langle x\rangle\langle x^T\rangle = \langle xx^T\rangle - 2\langle x\rangle\langle x^T\rangle + \langle x\rangle\langle x^T\rangle$$
$$= \langle (x-\langle x\rangle)(x-\langle x\rangle)^T\rangle$$

is the covariance matrix, the part w^* of the eigenvector $\tilde{w}^*$ of $C_{\tilde{x}\tilde{x}}$ is the eigenvector of the covariance matrix which we looked for. Thus, the weights (except the threshold) will converge to the eigenvectors of the covariance matrix.

Nevertheless, since the additional constant input has no variance, the additional eigenvalue is zero: it is the most stable feature.

4 The minimum entropy network

The minimum entropy neuron can be used as base unit in several ways. The direct approach replaces the Oja-unit of the cited eigenvector decomposition networks. Thus, the network of Sanger [10] will first find the eigenvector with the minimal eigenvalue and subtract all its components from the input space, cf. figure 1. In the remaining space the second neuron will find the eigenvector with the smallest eigenvalue again which is the next one of the eigenvectors in ascending order of their eigenvalues. The same mechanism can work in the lateral inhibition network of Ruben and Tavan [9]. In both networks, basically the Gram-Schmidt orthogonalization procedure is involved which depends only on the first base vector, the input statistics and the convergence goal (objective function) of each additional neuron.

The idea above sounds reasonable, but it does not work: The maximum entropy and minimum entropy objectives are not symmetrical! The following section analyze this more deeply and shows, how the equations must be changed to reflect the proper objective.

Let us consider a simple, one-layer network of minimum entropy neurons as it is shown in figure 4. The *activity* of the network at time step t is determined by the linear equation

$$y(t) = W\,x(t) \qquad \text{with } W := (w_1,\ldots,w_m)^T \qquad (4.1)$$

as it is associated with a classical, linear feed-forward network.

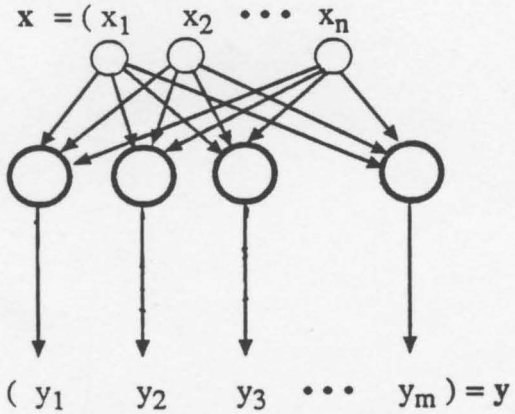

Fig. 4 The General-Anti-Hebbian activity network GAH

Nevertheless, for the procedure of *learning* the eigenvectors involves a completely different network. Starting with the first neuron each neuron gets its activity y_i, passes the reduced input to the next one, and correct and normalizes its weights (see fig.1). Since the idea is related to the General-Hebbian-network of Sanger [10], the network is called "General-Anti-Hebbian-network" (GAH).

For the first neuron Eqs. (3.8) are valid and, as we know by section 3.1, the neuron will converge to the eigenvector with the smallest eigenvalue. Thus, convergence is shown for the first step of the induction proof. Now, to show the general step from neuron s to $s+1$ in the network, we have to show that each neuron will converge to the eigenvector with the smallest eigenvalue of the n-s ones which remain, provided that all other s neurons have already converged to the eigenvectors with the s smallest eigenvalues.Now the neuron $s+1$ will see as input

$$\tilde{x}_s = x + a\Sigma_i^s \tilde{y}_i w_i \qquad (4.2)$$

and gives as output

$$\tilde{y}_{s+1}(t) = w_{s+1}^T \tilde{x}_s \qquad (4.3)$$

Thus, the objective function (3.2) of the neuron becomes

$$f(w_{s+1}) = w_{s+1}^T C_{\tilde{x}\tilde{x}} w_{s+1} \qquad (4.4)$$

and the weights w_s will converge to the eigenvector of $C_{\tilde{x}\tilde{x}}$ with the smallest eigenvalue by the gradient descend of Eqs. (3.8)

$$\hat{w}_i(t+1) = w_i(t) - \gamma(t)\, \tilde{x}_{i-1}(t)\, y_i(t) \quad 1 \le i \le m \qquad (4.5a)$$

Anti-Hebb-Rule

$$\tilde{x}_0 := x, \quad \tilde{x}_i := \tilde{x}_{i-1} + a\, \tilde{y}_i w_i \quad \textit{space correction} \qquad (4.5b)$$

and $$w_i(t+1) = \hat{w}_i(t+1) / |\hat{w}_i(t+1)| \quad \textit{normalization} \qquad (4.5c)$$

The eigenvector equation (3.5) becomes

$$\begin{aligned} C_{\tilde{x}\tilde{x}} \tilde{w}^* &= \langle \tilde{x}\tilde{x}^T \rangle_s \tilde{w}^* \\ &= \langle (x + a\textstyle\sum_i \tilde{y}_i w_i)\,(x + a\sum_j^s \tilde{y}_j w_j)^T \rangle\, w^* \\ &= \langle x x^T \rangle\, w^* + a^2 \textstyle\sum_i^s \sum_j^s \langle \tilde{y}_i \tilde{y}_j \rangle\, w_i w_j^T\, w^* \\ &\quad + 2a \textstyle\sum_i^s \langle y_i x w_i^T \rangle w^* \\ &= \lambda w^* \end{aligned} \qquad (4.6)$$

Since we assume that the s weights have already converged to the eigenvectors e_i, we know that $w_i^T w_j = 1$ only for i=j, otherwise it is zero. Therefore, we have for all i,j <s+1

$$\begin{aligned} \tilde{y}_i &= w_i^T \tilde{x}_{i-1} = w_i^T (x + a\textstyle\sum_k^{i-1} \tilde{y}_k w_k) \\ &= y_i + a\textstyle\sum_k^{i-1} \tilde{y}_k w_i^T w_k = y_i \end{aligned}$$

and therefore (4.6) becomes with (3.6)

$$\begin{aligned} C_{\tilde{x}\tilde{x}} \tilde{w}^* &= C w^* + a^2 \textstyle\sum_i^s \sum_j^s \langle y_i y_j \rangle\, e_i e_j^T\, w^* \\ &\quad + 2a \textstyle\sum_i^s \langle y_i x e_i^T \rangle\, w^* = \lambda w^* \end{aligned} \qquad (4.7)$$

As solution for w^* all n eigenvectors e_k of C are valid: if w^* is one of the s already obtained, the equation (4.7) will become

$$\begin{aligned} C_{\tilde{x}\tilde{x}} e_k &= \lambda_k e_k + a^2 \lambda_k e_k + 2a \langle y_k x \rangle \\ &= \lambda_k e_k + a^2 \lambda_k e_k + 2a C e_k = \lambda_k (1 + a^2 + 2a) e_k \\ &= \lambda e_k \end{aligned}$$

with the eigenvalue

$$\lambda = \lambda_k (1 + a^2 + 2a) \qquad (4.8)$$

If the eigenvector is new, the last two terms of (4.7) will become zero and the eigenvalue becomes

$$\lambda = \lambda_k \qquad (4.9)$$

Therefore, if we would like to obtain a descending order of eigenvalues, we just have to choose a=−1. Then all old eigenvectors have an eigenvalue of zero and a gradient ascend (Eq. (4.5a) with positive sign) will find of the remaining ones the eigenvector with the biggest eigenvalue. This is basically the General Hebbian decomposition network [10].

Nevertheless, the problem to find the eigenvectors with the *minimal* eigenvalues is *not* symmetric. If we would use the gradient descend by Eqs. (4.5), the choice of a negative a will make us find one of the eigenvectors already found which have here the eigenvalue of zero: there is no other smaller eigenvalue! The eigenvalue in (4.8) of every eigenvector already found is only bigger than λ_{s+1}, the next one in the ascending order, if

$$\lambda_{s+1} < \lambda_k (1 + a^2 + 2a) \qquad \text{for all } k < s+1$$

This must be true, even for the eigenvalues $\lambda_{s+1} = \lambda_{max}$ and $\lambda_k = \lambda_{min}$ of C. This condition means

$$\lambda_{max} / \lambda_{min} < (1 + a^2 + 2a) = (a+1)^2$$

and for positive a

$$a > (\lambda_{max} / \lambda_{min})^{1/2} - 1 \qquad (4.10)$$

The following figure 5 shows the convergence of all the weights to the appropriate eigenvectors. As an error measure the absolute cosinus $g(e_k, w_i) := |e_k^T w_i| / |w_i|$ is plotted against the number t of iterations for the example of a cyclic three pattern input $x^{(i)}$. As you can observe, the error decreases very fast for the first neuron, whereas the convergence of the other two weights depend on the first one.

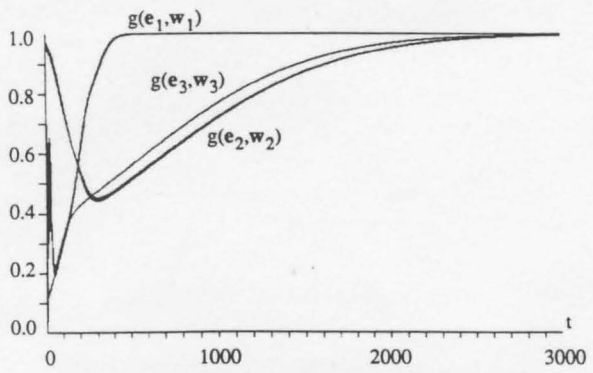

Fig.5 The convergence of the minimum entropy GAH network
$x^{(1)}=(1,0,0)$, $x^{(2)}=(1,1,0)$, $x^{(3)}=(1,1,1)$
$\lambda_1 = 0.1026$, $\lambda_2 = 0.2143$, $\lambda_3 = 1.6823$
$\gamma(t)=\gamma=0.01$, $a=5$, $w_i(0)=(1,1,1)$

In Figure 6 the case is shown when a=4 is too small. Then condition (4.8) is not met for the last eigenvalue $\lambda_3 = \lambda_{max}$ and the variance of the deterministic input disturb the convergence.

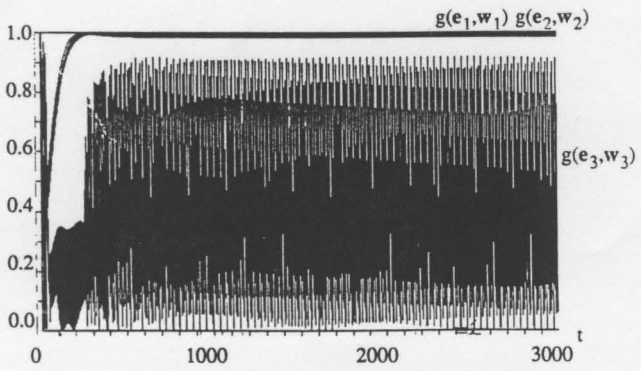

Fig.6 Partially non-converging GAH network γ=0.1, a=4

5 Discussion and conclusion

The paper showed how cluster transformation can be implemented by the base unit of a linear neuron where the weight vector converges to the eigenvector of the input pattern autocorrelation matrix with the smallest eigenvalue.

For the case where the number of input dimensions and neurons are the same, all the already known networks and the newer proposed ones decompose the input space into the complete set of eigenvectors. So, what is the difference of the proposed networks to the already existing ones? The main difference becomes evident for the case when not all, but only a few eigenvectors are selected as target base. The maximum entropy networks choose first the eigenvectors with the *biggest* eigenvalues i.e. the features containing most of the information, neglecting all the rest. In contrast, the proposed ones will first select the eigenvectors with the *smallest* eigenvalues, thus choosing those features which are the most stable ones.

It should be noted that the proposed mechanism involves only linear neurons. Additional non-linearities in the neural output function S(z) (squashing function) will lead to further reduction of the cluster entropy, but do not provide directly the eigenvector decomposition [8]. In the binary version it becomes the vector quantization which can directly be used for symbolic postprocessing in an object recognition system.

References

[1] R. Brause: The Minimum Entropy Neuron- a new building block for clustering transformations; Proc. Int. Conf. on Art. Neural Networks ICANN-92, Brighton 1992.

[2] P. Földiák: Adaptive Network for Optimal Linear Feature Extraction; IEEE Proc. Int. Conf. Neural Networks; pp. I/401-405 (1989).

[3] K. Fukunaga: Introduction to Statistical Pattern Recognition; Academic Press, New York 1972.

[4] N.S. Jayant, Peter Noll: Digital Coding of waveforms, Prentice Hall 1984.

[5] H.P. Kramer, M.V. Mathews: A Linear Coding for Transmitting a Set of Correlated Signals; Transact. 1956 Symp. on Inf. Theory; in IEE Trans. on Inf. Th. IT-2 (1956).

[6] Erkki Oja: A Simplified Neuron Model as a Principal Component Analyzer, J. Math. Biol. 13: 267-273 (1982)

[7] Erkki Oja: Neural Networks, Principal Components, and subspaces, Int. J. Neural Systems, Vol 1/1 pp. 61-68 (1989)

[8] E. Oja: Learning in non-linear Constrained Hebbian Networks; Proc. ICANN-91, T.Kohonen et al. (Eds.), Artif. Neural Netw., Elsevier Sc. Publ. 1991, pp. 385-390

[9] J. Rubner, P. Tavan: A Self-Organizing Network for Principal-Component, Analysis, Europhys.Lett., 10(7), pp. 693-698 (1989).

[10] Sanger: Optimal unsupervised Learning in a Single-Layer Linear Feedforward Neural Network; Neural Networks Vol 2, pp.459-473 (1989)

[11] J.T. Tou, R.C. Gonzales: Pattern Recognition Principles; Addison-Wesley Publ. Comp., 1974

[12] R.Williams: Feature Discovery through Error-Correction Learning; ICS Report 8501, University of Cal. and San Diego, 1985

Appendix A

The extrema of the objective function

The objective function is defined as

$$f(w) = \langle y^2 \rangle = \langle w^T x x^T w \rangle = w^T C w \qquad (3.2)$$

Suppose that the symmetric C is of full rank. Then there exist a base vector system of orthogonal (and orthonormal) eigenvectors e_k of C with $Ce_k = \lambda_k e_k$ such that each vector w can be represented in this base by

$$w = \Sigma_i a_i e_i \quad \text{and } a_i = w^T e_i = |w|\,|e_i| \cos(w,e_i)$$

with the projection a_i of w on the eigenvector e_i. Due to the orthonormality of e_i and the constrain of w the coefficients depend only on the angle α_i between w and the eigenvector and we have

$a_i = \cos(w,e_i) = \cos \alpha_i$

By this condition we change our coordinates from Cartesian to polar based description. Nevertheless, by the constrain the *n* coordinates remain implicitely dependend from each other. Therefore, to eliminate the dependence we choose the first two variables α_1 and α_2 and replace them by an independant variable β. For this purpose, let us regard the projection of w on the plane between e_1 and e_2, see figure A.1.

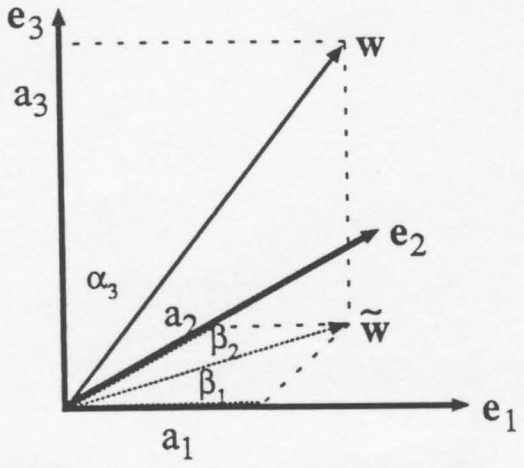

Figure A.1 The projection on a plane

The projection of w on one plane has the form $\tilde{w} = a_1 e_1 + a_2 e_2$ because the difference vector $(w - \tilde{w})$ is orthogonal on the plane: $(w - \tilde{w})^T e_1 = 0 = (w - \tilde{w})^T e_2$. For the projection $\tilde{w}$ we can replace the angle β_2 by its complemental counterpart β_1

$$\cos \beta_2 = \cos(\pi/2 - \beta_1) = \sin \beta_1$$

Thus, the objective function (3.2) becomes

$$f(w) = w^T C w = (\Sigma_i a_i e_i)^T C (\Sigma_j a_j e_i) = \Sigma_i \Sigma_j a_i a_j e_i^T C e_j = \Sigma_i a_i^2 \lambda_i \qquad (A.1)$$

with the components

$$a_1 = \tilde{w}^T e_1 = |\tilde{w}|\,|e_1| \cos\beta = |\tilde{w}| \cos\beta \qquad \beta := \beta_1 \qquad (A.2)$$
$$a_2 = \tilde{w}^T e_2 = |\tilde{w}|\,|e_2| \sin\beta = |\tilde{w}| \sin\beta \qquad (A.3)$$

The length of the projection $|\tilde{w}|^2 = a_1^2 + a_2^2$ depends on all the other angles α_i but not on β

$$|w|^2 = 1 = a_1^2 + a_2^2 + \Sigma_{k=3}^n a_k^2$$
$$\Rightarrow |\tilde{w}|^2 = 1 - \Sigma_{k=3}^n a_k^2 \qquad (A.3)$$

Now, the objective function depends only on the *n-1* independant variables $\beta, \alpha_3, .., \alpha_n$. The necessary conditions for the extrema are

$$\text{grad } f(w) = \text{grad } f(\beta, \alpha_3, .., \alpha_n) = \mathbf{0}$$

Let us evaluate this for all variables $\beta, \alpha_3, .., \alpha_n$.

$$\frac{\partial}{\partial\beta} f(w) = \frac{\partial}{\partial\beta} \Sigma_k a_k^2 \lambda_k = \frac{\partial}{\partial\beta} a_1^2 \lambda_1 + \frac{\partial}{\partial\beta} a_2^2 \lambda_2$$
$$= -\lambda_1 2|\tilde{w}|^2 \cos\beta \sin\beta + \lambda_2 2|\tilde{w}|^2 \sin\beta \cos\beta$$
$$= (\lambda_2 - \lambda_1) 2|\tilde{w}|^2 \cos\beta \sin\beta \qquad (A.4)$$

because the length $|\tilde{w}|$ of the projection does not change when the angle β is changed.
For $(\lambda_2 - \lambda_1) \neq 0$ and $|\tilde{w}| \neq 0$ the conditions (A.4) become zero when w^* or β^* is given by

$$\sin\beta^* = 0 \iff \beta^* = 0, \pi \qquad (A.5)$$
$$\cos\beta^* = 0 \iff \beta^* = \pi/2, 3\pi/2$$

For all other variables α_i, i=3..n we have

$$\frac{\partial}{\partial\alpha_i} f(w) = \frac{\partial}{\partial\alpha_i} \Sigma_k a_k^2 \lambda_k = \frac{\partial}{\partial\alpha_i} a_1^2 \lambda_1 + \frac{\partial}{\partial\alpha_i} a_2^2 \lambda_2 + \frac{\partial}{\partial\alpha_i} a_i^2 \lambda_i$$
$$= (\lambda_1 \cos^2\beta + \lambda_2 \sin^2\beta - \lambda_i) 2\cos\alpha_i \sin\alpha_i \qquad (A.6)$$

$$\text{with } \frac{\partial}{\partial\alpha_i} |\tilde{w}|^2 = \frac{\partial}{\partial\alpha_i} (1 - \Sigma^n_{k=3} a_k^2) = -\frac{\partial}{\partial\alpha_i} a_i^2$$
$$= 2\cos\alpha_i \sin\alpha_i \qquad (A.6b)$$

and therefore with $(\lambda_1 \cos^2\beta + \lambda_2 \sin^2\beta - \lambda_i) \neq 0$ we have

$$\sin\alpha_i^* = 0 \iff \alpha_i^* = 0, \pi \quad \forall\, i=3, ..., n \qquad (A.7)$$
$$\cos\alpha_i^* = 0 \iff \alpha_i^* = \pi/2, 3\pi/2$$

The solutions (A.5) and (A.7) correspond to the solutions obtained earlier in (3.5): the extrema occur for all w parallel ($\beta^* = 0$, $\alpha_i^* = 0$) or antiparallel ($\beta^* = \pi$, $\alpha_i^* = \pi$) to the eigenvector e_1 or e_i which is orthogonal ($\beta^* = \pi/2, 3\pi/2$ and $\alpha_i^* = \pi/2, 3\pi/2$) to the other eigenvectors.

In the formulation with *n-1* independent angles we can discuss the nature of the extrema (and thus the nature of the fixpoints of the corresponding gradient algorithm) by the use of the second derivatives in the Hesse matrix $\mathbf{A} = (f_{ij}) = (\partial^2 f(a) / \partial\alpha_i \partial a_j)$ at the extrema

$$w^* = e_1 \iff \beta^* = 0, \pi\,, \qquad \alpha_i^* = \pi/2, 3\pi/2 \qquad (A.8)$$
$$w^* = e_2 \iff \beta^* = \pi/2, 3\pi/2\,, \quad \alpha_i^* = \pi/2, 3\pi/2$$
$$w^* = e_i \iff \beta^* = \pi/2, 3\pi/2\,, \quad \alpha_i^* = 0, \pi$$

The mixed terms with $i \neq 1$ of A are by (A.4) and (A.6b)

$$\frac{\partial^2 f(w)}{\partial\beta \partial\alpha_j} = (\lambda_2 - \lambda_1) 2 \frac{\partial}{\partial\alpha_j} |\tilde{w}|^2 \cos\beta \sin\beta$$
$$= (\lambda_2 - \lambda_1) 4 \cos\alpha_i \sin\alpha_i \cos\beta \sin\beta \qquad (A.9)$$

which is identical to $\partial^2 f(w)/\partial\alpha_j\partial\beta$. At all extrema of (A.8) the mixed terms (A.9) become zero.
The other terms for all i,j=3, ..., *n* are with (A.6)

$$\frac{\partial^2 f(w)}{\partial\alpha_i\partial\alpha_j} = (\lambda_1\cos^2\beta + \lambda_2\sin^2\beta - \lambda_i)\, 2\frac{\partial}{\partial\alpha_j}\cos\alpha_i\sin\alpha_i$$

which becomes zero for all $i\neq j$, otherwise by $\cos^2\alpha + \sin^2\alpha = 1$ we get

$$\frac{\partial^2 f(w)}{\partial^2\alpha} = (\lambda_1\cos^2\beta + \lambda_2\sin^2\beta - \lambda_i)2(1-2\sin^2\alpha_i) \qquad (A.10)$$

Finally, for i=j=1 we have

$$\frac{\partial^2 f(\beta)}{\partial^2\beta} = \frac{\partial}{\partial\beta}(\lambda_2-\lambda_1)2|\tilde{w}|^2\cos\beta\sin\beta$$
$$= (\lambda_2-\lambda_1)2|\tilde{w}|^2(1-2\sin^2\beta) \qquad (A.11)$$

Since all mixed terms are zero, the *n-1* dimensional Hesse matrix becomes a diagonal matrix; its eigenvalues are just the components (A.10) and (A.11).

For a minimum of the objective function at $\mathbf{e}_1$, all the second derivatives must be greater than zero. This is the case at ($\beta^*=0,\pi$, $\alpha_i^*=\pi/2, 3\pi/2$) when by (A.10) $(\lambda_i-\lambda_1)>0$ and by (A.11) $(\lambda_2-\lambda_1)>0$, i.e. the eigenvalue λ_1 is smaller than λ_2 and any other λ_i, it must be the smallest eigenvalue. Since the choice of $\mathbf{e}_1$ was arbitrary, the same arguments hold for any other eigenvector $\mathbf{e}_i$ with the smallest eigenvalue: it is the unique minimum. This can be verified by the interested reader by the application of the other extrema at $\mathbf{e}_i$ in (A.8) to Eqs. (A.10) and (A.11).

The equivalent argumentation holds for the unique maximum: (A.10) and (A.11) are negative for an extremum (a maximum) only iff λ_i is the maximal eigenvalue.

Beside the two eigenvectors with the biggest and the smallest eigenvalues, by the arguments above, all other eigenvectors correspond to extrema which fulfill both maximum and minimum conditions. The nature of the extrema depend on the direction of approaching them: they are saddle points which correspond to unstable fixpoints.