Proc. of the 1993 IEEE
Int.'l Conf. on Tools with AI,
Boston, Massachusetts, Nov. 1993

Transform Coding by Lateral Inhibited Neural Nets

Rüdiger W. Brause
J.W. Goethe-University, D-60054 Frankfurt (FRG)

Abstract

One of the most popular encoding technique for sensor data is transform coding. *This encoding schema is composed of two stages: a linear transformation stage with a non-zero kernel and a vector quantization stage. For the first stage, this paper describes a new implementation approach by artificial neural networks. The problem of determining the optimal transformation coefficients is solved by* learning *the coefficients by a* lateral inhibited neural network.

After a short introduction into the topic the paper focuses on this model and a local stability analysis of the fixpoints for the serial dynamics is provided. The resulting parameter regime is used in a network simulation example using picture statistics. Additionally, the simulations reveal that a biologically-like growing lateral inhibition influence leads to a speed-up of the learning convergence of that model.

1 Introduction

The encoding of sensor information is a very important subject. Results are used in picture and music encoding and compression (video and audio transmission and storage), in the preprocessing for speech recognition or in tactile and position sensoring for robot control.

One of the most popular encoding techniques is "transform coding" [22]. The classical transform encoding process consists of two stages: a linear transformation, which for instance in the JPEG and MPEG standard video encoding is implemented by a descrete cosinus transform, and a vector quantization stage. Both stages contain non-linear operations and reduce the data stream; the linear transformation has a non-zero kernel and the vector quantization maps all data of the neighbourhood to only a few code book vectors ("class prototypes"). The image encoding and decoding process is illustrated in figure 1.

In this paper we present a neural model for the first encoding stage, although the second stage, the vector

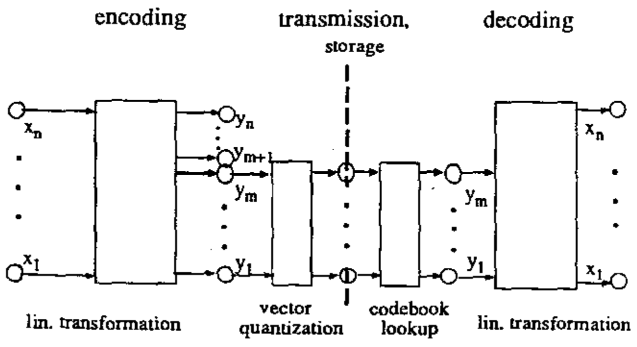

Fig. 1 The *transform coding* image encoding and decoding schema

quantization, can also be modelled by neural networks (e.g. the Kohonen map [11]).

Classically, the sensor signals are seen as time-varying features which are decomposed by a new feature set which is better usable, e.g. by the Fourier coefficients of a Fourier transformation. For instance, this approach is often used in picture processing [8] or in the preprocessing stages of speech-recognition systems, e.g. [11].

Nevertheless, this approach has some flaws: the coefficients of such a decomposition generally are correlated. Let us introduce another approach by the means of another decomposition or other sensor primitives.

1.1 The minimal mean squared reproduction error

Let us first consider the error which we make in the first stage. If we use the same number m of output lines as there are input lines, the $m{=}n$ linear output values $y_i = \Sigma_j w_{ij}x_j = \mathbf{x}^T\mathbf{w}$ are just the projection of the input $\mathbf{x}=(x_1,..,x_n)$ on the transformation coefficient vectors $\mathbf{w}_i=(w_{i1},..,w_{in})$

or the coordinates of x in a new base system $\{w_i\}$. When the w_i are linear independent and complete then we do not loose information and a complete reconstruction of the input by $y = (y_1,...,y_m)$ is possible.

However, if we use by $m<n$ less output than input lines we will get a reconstruction error. For linear systems, it is well known that the mean squared error is minimized by selecting those eigenvectors as neglected base vectors $w_{m+1},..,w_n$ which have the smallest eigenvalues [12]. To avoid the trivial solution of the zero vector **0**, the transformation should be neutral, i.e. the length of the base vectors should remain constant one.

Thus, the base vectors which span the subspace of the eigenvectors with the biggest eigenvalues can be considered as an *optimal* linear transformation and should be preferred to all other current linear transformations like the descrete Walsh-Hadamard transformation, the descrete Fourier transformation or the descrete Cosinus transformation [7]. If we additionally choose the base vectors to be the eigenvectors, the y_i are also decorrelated, see eq.(2.3b). The new base vectors point in the directions of maximal data variance $\langle y_i^2 \rangle$ and therefore called "transformation on principal axes".

1.2 Eigenvector decomposition of pictures

The decomposition of sensor signals (*transform coding*) into eigenvector components leads to the least mean square error for the reconstruction of the original signal by the components y_i and the eigenvectors e_i. If we treat a picture as a signal, all descrete NxM pixels of the picture can be arranged in one input vector. Thus, the eigenvectors of the corresponding autocorrelation matrix have NxM components and can be rearranged into a picture: they are a kind of *basis images* of the decomposition and called *eigen images* [9] .

The correlations in pictures decrease rapidly with increasing distance. Wintz [22] reports that for sufficient image reproduction it sufficies to consider correlations only 4-5 pels (=picture elements: here pixels) wide. Therefore, instead of including all correlations on NxM pixels we devide the picture into *m* subpictures, see figure 2, and describe the whole picture by *m* sets of eigenvectors with length n = NxM/m. By this approach we break the encoding process into parallel activities for *m* independent working processing units. The price we pay is a small error, depending on our subpicture size. This approach was first proposed by Habibi and Wintz [7].

1.3 Neural implementations

Now, let us map each parallel processing unit to a subnet of *m* neurons. The parallel, distributed encoding of the whole picture is done by a neural network, where each subpicture of *n* pixels is coded by a local transformation process into *m* components by a subnetwork of *m*

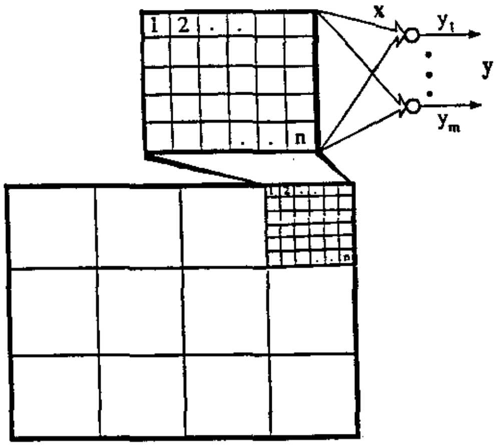

Fig. 2 The picture decomposition by parallel processing units

neurons. This concept has the advantages

- ♦ It can be implemented by integrating the neural network including the light-sensitive cells on a VLSI chip as analog circuits. This means much less circuitry than in the digital case. However, the output can also be made sequential like the usual video signal.
- ♦ The sensor device will then directly learn the eigenvectors by the input signal statistics, see e.g. [3], thus minimizing the error.
- ♦ On the receiver side, the reconstruction of the picture signal can be directly integrated in the LCD screen matrix.
- ♦ For noise-corrupted signals, non-linear neurons will enhance the encoding [16]. Here the coefficients are automatically adjusted, even if they do not represent any eigenvector space any more.

After an initial training phase, the optimal transformation coefficients (the weights) can be send from the sender device to the receiving device (e.g. the screen) once. Instead, we can also get the values for the weights by training a simulated system with pictures of the desired statistics or use directly the analytical solutions and implement the weights on the sender and on the receiver side of the system as pure ROM solutions without complicated learning mechanism. Here, the lateral inhibition model just serves a simulation tool for the training phase and is abondoned for the "ordinary" activity phase.

2 The symmetrical network model

There already exist several proposals for eigenvector decomposition networks. Since Oja's statement [14] that a linear, formal neuron using Hebb's learning rule and restricted weights will learn the eigenvector with the biggest eigenvalue several network architectures were proposed for a partial or complete eigenvector decomposition. Basically, they consist of two categories: networks which learns the eigenvectors sequentially in a certain order ("asymmetric networks"), based on the sequential Gram-Schmidt orthogonalization mechanism, and networks which learn them in parallel ("symmetric networks") and do not predetermine the mapping of the eigenvectors to the neurons. This has the advantage, that in difference to the assymmetric or conventional digital approach, the network allows the building of a dynamically defined, homogeneous encoder network which can easily be adápted to statistics, region-dependent resolutions and avoid the subpicture boarder fitting problem [20] by continous overlappings of the input space. The approaches use linear neurons, where each neural weight vector converges to one eigenvector.

Examples of the former architectures are the Sanger decomposition network [18] and the lateral inhibition network of Rubner and Tavan [17]. They use as a basic building block the linear correlation neuron which learns the input weights by a Hebb-rule, restricting the weights.

The learning rule for one neuron can be generalized, yielding a network where the input is inhibited simultaniously by the projections of the input to all weight vectors. This corresponds to the symmetric network approach. The Oja subspace network [15], the Williams subspace learning [21] and the lateral inhibition network of Földiák [5], which is a version of Kohonen and Ojas orthogonalizing filter [10], have the same problems:

- They provide the convergence of the weight vectors to the subspace of the eigenvectors, not necessary to the eigenvectors itselves. This can produce correlated output variables.
- Their heuristic backward-inhibition mechanism propagates whole weight vectors which is neither feasable for VLSI implementations (connection complexity) nor biologically plausible.

In fact, a fully symmetrical network for eigenvector decomposition, construced by an objective function and implemented by a biological plausible and easily realizable network mechanism is still missing. The task of this paper is to overcome the gap. Contrary to the opinion of Hornik and Kuan (1992), who are not in favour of symmetric PCA networks, we will introduce a new symmetric model in this section which is not covered by their general convergence analysis of the PCA models mentioned above and which interact by scalar signals ("lateral inhibition").

2.1 The model

Let us assume in a first step that we have *m* neurons which are laterally interconnected as shown in figure 3.

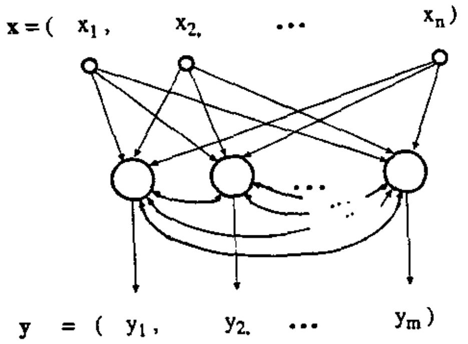

Fig. 3 The symmetric, lateral interconnected network model

Each neuron *i* has a randomly chosen weight vector $\mathbf{w}_i$. After we presented one input pattern $\mathbf{x}$ in parallel to each neuron of the linear system, the output of neuron *i* will result in

$$y_i = \mathbf{w}_i^T\mathbf{x} + T_i \qquad T_i = \Sigma_{j\neq i}\, u_{ij} y_j \qquad (2.1)$$

where T_i denotes the influence by the lateral connections which are weighted by the lateral weights u_{ij}. The input is assumed to be centered. If this is not the case, it can be made so by introducing a special threshold weight learned with an Anti-Hebb-rule.

Although the model is quite linear, we have reactions for random input and weights due to the feedback lines which are difficult to analyze. Nevertheless, for the prediction of the system behaviour the analysis of the expected equilibrium states of the system is sufficient.

Let us assume that after an input pattern has been presented the system activity stabilizes (see e.g. [10]). This is the case, when the feed-back does not induce additional oscillations, i.e. when all the eigenvalues of the feed-back matrix U are allways smaller than one [6] and there is no significant feedback delay [13].
Then the output for neuron *i* becomes with Eq. (2.1)

$$y_i = \mathbf{w}_i^T\mathbf{x} + \Sigma_{j\neq i}\, u_{ij}y_j = \mathbf{w}_i^T\mathbf{x} + \mathbf{u}_i^T\mathbf{y} - y_i \qquad u_{ii} = 1$$

and the output vector becomes

$$2\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{U}\mathbf{y}$$

or

$$(2\mathbf{I}-\mathbf{U})\mathbf{y} = \mathbf{W}\mathbf{x} \qquad \text{with the identity matrix } \mathbf{I}.$$

Thus, the system output

$$\mathbf{y} = (2\mathbf{I}-\mathbf{U})^{-1}\mathbf{W}\,\mathbf{x} = \mathbf{A}\,\mathbf{x} \qquad \mathbf{A} = (2\mathbf{I}-\mathbf{U})^{-1}\mathbf{W} \qquad (2.2)$$

depends again linearly on the input.

2.2 The learning of the weights

The learning rule for the weights $\mathbf{a}_i$ is determined by the following three conditions, introduced in the previous section 1:

- The new features should be decorrelated
  $$\langle y_i y_j \rangle = \langle y_i \rangle \langle y_j \rangle = 0 \qquad (2.3)$$
- The variance of the features should be maximal
  $$\Sigma_i \langle y_i^2 \rangle = \max \qquad (2.4)$$
- The decomposition should be neutral
  $$|\mathbf{a}_i| = 1, \text{ i.e. } \det(\mathbf{A}) = 1 \qquad \textit{no scaling} \qquad (2.5)$$

These conditions can be modelled by the minimum of deterministic objective function

$$\begin{aligned} R(\mathbf{a}_1,..,\mathbf{a}_m) &= 1/4\, \beta \Sigma_i \Sigma_{j\neq i} (\langle y_i y_j \rangle)^2 - 1/2\, \Sigma_i \langle y_i^2 \rangle \\ &= R^1 + R^2 \end{aligned} \qquad (2.6)$$

The first term R^1 ensures that the cross-correlation (2.3) is always counted positive. This results in a minimum of R(.) where the cross-correlation term R^1 becomes zero and $-R^2$, the sum of all variances, become maximal. Since the minimum of the objective function scaled by an arbitrary factor do not change, the parameter β denotes only the relative influence of the crosscorrelation with respect to the autocorrelation influence.

The third condition (2.5) have to be additionally assured during the learning process. This condition could also be integrated into the objective function. It was shown for one neuron [4] that this yields also the eigenvectors as solutions and can be compared to the approach using (2.5) to compute the unique maximum and minimum of the objective function [1,2].

In appendix A it is shown that the objective function R(a) takes its extrema when the $\mathbf{a}_i$, the lines of the matrix $\mathbf{A}$, are a subset of the eigenvectors of the autocorrelation matrix $\mathbf{C}=\langle \mathbf{x}\mathbf{x}^T \rangle$. Since $\mathbf{C}$ is symmetric and real, the eigenvalues λ_i are real and the eigenvectors form an orthogonal base system.
Here, the cross-correlations

$$\langle y_i y_j \rangle = \mathbf{a}_i^T \langle \mathbf{x}\mathbf{x}^T \rangle \mathbf{a}_j = \mathbf{a}_i^T \mathbf{C} \mathbf{a}_j = \mathbf{a}_i^T \lambda_j \mathbf{a}_j = 0 \quad \forall\, i \neq j \quad (2.3b)$$

become zero, and by definition (2.10), we have $\mathbf{U} = \mathbf{I}$ and

$$\mathbf{A} = (2\mathbf{I}-\mathbf{U})^{-1}\mathbf{W} = (2\mathbf{I}-\mathbf{I})^{-1}\mathbf{W} = \mathbf{W} \qquad (2.7)$$

Thus, the minimum of the objective function (2.6) is reached when the weight vectors become the eigenvectors of the autocorrelation matrix $\mathbf{C}$; the lateral inhibition weights become zero.

To learn the weight vectors $\mathbf{a}_i$, a gradient descend may be used. Nevertheless, with (2.2) this leads to complicated expressions for $\mathbf{w}_i$ and u_{ij}. Instead, by (2.7) we can conclude that a simplified learning rule for $\mathbf{w}_i$ which ensures the convergence to the eigenvectors of $\mathbf{C}$ will also do the job.

At this point, we change our neural modelling. We replace our lateral inhibition activity model, which leads to a linear network by (2.2), by the much more simple activity model of a non-coupled, linear network, which by (2.7) in the limit will have the same performance. Thus, for the learning phase we can split the influence of the two different weights $\mathbf{U}$ and $\mathbf{W}$ and use seperate learning rules for each weight type.

For the objective function (2.6) which now depends only on the $\mathbf{w}_i$ the minimum can be approximated by a gradient search for the weight vectors $\mathbf{w}_i$ directly, assuring conditions (2.3), (2.4), (2.5) by the usage of the objective function (2.6) for a=w. The learning rule for the lateral weights is split apart and will be treated seperately afterwords.
The (t+1)-th iteration step for the input weights is

$$\begin{aligned} &\tilde{\mathbf{w}}_i(t+1) = \mathbf{w}_i(t) - \gamma(t)\, \nabla_w R(\mathbf{w}_i) \\ \text{or} \quad &\Delta\mathbf{w}_i = \tilde{\mathbf{w}}_i(t+1) - \mathbf{w}_i(t) = -\gamma(t)\, \nabla_w R(\mathbf{w}_k) \end{aligned} \qquad (2.8)$$

$$\mathbf{w}_i(t+1) = \tilde{\mathbf{w}}_i(t+1)/|\tilde{\mathbf{w}}_i(t+1)| \qquad \textit{normalization} \qquad (2.9)$$

denoting the gradient by the Nabla-operator ∇_w. This is calculated in appendix A, eq.(A.2). With the definition

$$u_{ij} = -\langle y_i y_j \rangle \qquad \textit{lateral inhibition weights} \quad (2.10)$$

the deterministic learning rule (A.2) becomes

$$\begin{aligned} \Delta\mathbf{w}_i &= -\gamma(t)(\beta\Sigma_{j\neq i} -u_{ij}\langle \mathbf{x} y_j \rangle - \langle \mathbf{x} y_i \rangle) \\ &= +\gamma(t)\, \langle\, \mathbf{x}\, (y_i + \beta\Sigma_{j\neq i} u_{ij} y_j)\, \rangle \end{aligned} \qquad (2.11)$$

For deterministic, linear systems formally we have no lateral inhibition influence for w, i.e. T=0

$$\begin{aligned} \text{and} \quad \Delta\mathbf{w}_i &= \gamma(t)\, (\langle \mathbf{x}\mathbf{x}^T \rangle \mathbf{w}_i - \beta\Sigma_{j\neq i} \langle \mathbf{x}\mathbf{x}^T \rangle \mathbf{w}_j (\mathbf{w}_j^T \langle \mathbf{x}\mathbf{x}^T \rangle \mathbf{w}_i)) \\ &= \gamma(t)\, (\mathbf{C}\mathbf{w}_i - \beta\mathbf{C}(\Sigma_{j\neq i} \mathbf{w}_j \mathbf{w}_j^T)\mathbf{C}\mathbf{w}_i) \end{aligned} \qquad (2.12)$$

The stochastic version of (2.11) is

$$\Delta\mathbf{w}_i = +\gamma(t)\, \mathbf{x}\, (y_i + \beta\Sigma_{j\neq i} u_{ij} y_j) \qquad (2.13)$$

The lateral weights should be updated by a rule which let them become the expected cross-correlation. There are classical learning rules for iterative averaging [19], but they are difficulties to apply them beause the random variable y_i are not stationary for changing $\mathbf{w}_i$. Therefore, random initial values of the weights can disturb the average for a long period of simulation time. To get rid of these random values and to accelerate the convergence, we might use instead the temporal floating average of *N* observed data

$$u_{ij}(t) = -\, [1/N]\, \Sigma_{k=t-N}^{t-1}\, y_i(k) y_j(k) \qquad (2.14)$$

which assumes a kind of weight decay process.

2.3 Stability conditions for the learning fixpoints

The fixpoints of the learning system are calculated in appendix A. Nevertheless, they do not indicate under what conditions (correlation parameter β and learning rate γ) the fixpoints are stable. It is well known that the sequential gradient descend algorithm (2.8) confirms a monotonic decrease of the quadratic objective function (2.6), because we have

$$\frac{\partial R(t)}{\partial t}=\frac{\partial R(\mathbf{a})}{\partial \mathbf{a}}\frac{\partial \mathbf{a}}{\partial t}=-\frac{\partial R(\mathbf{a})}{\partial \mathbf{a}}\gamma(t)\frac{\partial R(\mathbf{a})}{\partial \mathbf{a}} = -\gamma\left(\frac{\partial R}{\partial \mathbf{a}}\right)^2 \leq 0 \qquad (2.15)$$

Since the objective function R has a lower bound for m neurons of $\min(-1/2\Sigma_i\langle y_i^2\rangle) = -1/2\ m \max(\mathbf{a}_i^T\langle \mathbf{x}\mathbf{x}^T\rangle \mathbf{a}_i) = -1/2\ m \max(\mathbf{a}_i^T C \mathbf{a}_i) = -1/2\ m\ \lambda_{max}$ for linear systems, the objective function can be regarded as a Ljapunov function and the iteration will converge to the fixpoints.

Thus, for the sequential case principally we have shown the convergence of the system which is a general property of all gradient descend learning rules for limited objective functions.

Let us evaluate this more deeply to get some conditions for β and γ.

The crosscorrelation factor β *and the learning rate* γ

In appendix A it is proven that all the fixpoints of the system are at the eigenvectors of the autocorrelation matrix. Note that this means only that the fixpoints of the system are eigenvectors, they do not have to be different ones. It is evident that with decreasing γ the descrete step dynamics become the continuous time dynamics and the system is guaranteed to converge.

Nevertheless, for finite γ this is not true and we have to deal with it seperately. Unfortunately, the developement of the learning system due to initial and developing mutual correlations is quite complicated. To get some simple conditions for the learning rate which simulations showed to be relevant we limit our analysis to local stability considerations for the nearly converged system.

After step t+1, with Eq. (2.9) the new weight vector is

$$\mathbf{w}_i(t+1) = \frac{\mathbf{w}_i(t) + \Delta\mathbf{w}_i}{|\mathbf{w}_i(t) + \Delta\mathbf{w}_i|} \qquad (2.16)$$

In appendix B, this is evaluated for the k-th component of an eigenvector expansion.

Since the ratio of the components are independent of the length of the weight vector, it is interesting to observe the change of the ratio in different eigenvector directions. If the ratio $a_{ik}(t)/a_{ip}(t)$ of the k-th component $a_{ik}(t)=\mathbf{w}_i^T(t)\,\mathbf{e}_k$ increases at each time step for every other component p, we can conclude that weight vector $\mathbf{w}_i$ converges to eigenvector $\mathbf{e}_k$ of the autocorrelation matrix C. For the case of just one neuron, the ratio is with Eq.(B.1) and $b_{ij}=0$

$$\frac{|a_{ik}(t+1)|}{|a_{ip}(t+1)|} = \frac{|a_{ik}(t)(1+\gamma\lambda_k)|}{|a_{ip}(t)(1+\gamma\lambda_p)|} \qquad (2.17)$$

and will increase, if the condition $(1+\gamma\lambda_k) > (1+\gamma\lambda_p)$ or $\lambda_k > \lambda_p$ holds for all other components p. Thus, independently of the initial weights, the learning rate and the crosscorrelation factor, the weight vector will converge to the eigenvector with the biggest eigenvalue.

Now, assume that already m weight vectors have converged to the m eigenvectors with the biggest eigenvalues. Then the $m+1$-th weight vector $\mathbf{w}_i$ converges to eigenvector $\mathbf{e}_{m+1}=\mathbf{e}_k$ and not to one of the eigenvectors with a bigger eigenvalue λ_p, if with Eq. (B.1) the ratio

$$\frac{|a_{ik}(t+1)|}{|a_{ip}(t+1)|} = \frac{|a_{ik}(t)|\,|1+\gamma\lambda_k(1-\beta\Sigma_{j\neq i}\, a_{jk}(t)b_{ij}/a_{ik}(t))|}{|a_{ip}(t)|\,|1+\gamma\lambda_p(1-\beta\Sigma_{j\neq i}\, a_{jp}(t)b_{ij}/a_{ip}(t))|} > \frac{|a_{ik}(t)|}{|a_{ip}(t)|}$$

holds. With $a_{jk}=0$ and $b_{ij}=a_{ip}\lambda_p$ this relation becomes

$$|1+\gamma\lambda_k| \,/\, |1+\gamma\lambda_p(1-\beta\lambda_p)| \quad > 1$$

For $1-\beta\lambda_p > 0$, this means

$$1+\gamma\lambda_k > 1+\gamma\lambda_p(1-\beta\lambda_p) \quad \text{or} \quad \lambda_k > \lambda_p - \beta\lambda_p^2$$

which does not depend on the learning rate but is true if the relation

$$\beta > \beta_1 =: (\lambda_p-\lambda_k)/\lambda_p^2 \qquad \forall\, k,p \qquad (2.18)$$

is guaranteed. Let us assume that we have m different eigenvalues λ_i. What is the best choice for β to assure different weight vectors ? The function $f(x) = (x-a)x^{-2}$ which corresponds to Eq. (2.18) takes its maximum at $\partial f/\partial x = x^{-2}-2(x-a)x^{-3}=0$ or $x=2a$. The maximum is therefore reached at $f(2a) = 1/4a$. With $a=\lambda_p$ the function has its maximal value at $\lambda_p=\lambda_{min}$ and condition (2.18) becomes

$$\beta > (4\,\lambda_{min})^{-1} \qquad (2.19)$$

For $1-\beta\lambda_p < 0$, the absolute value of the whole term can become negative and the sign of the component can alternate after each iteration. Nevertheless, even the oscillating weight vector will converge to eigenvector $\mathbf{e}_k$ if

$$1+\gamma\lambda_k > -(1+\gamma\lambda_p(1-\beta\lambda_p))$$

or $\quad 2 > \gamma[\lambda_p(\beta\lambda_p-1)-\lambda_k]$

and thus (2.20)

$$\gamma \underset{>}{<} 2/[\beta\lambda_p^2-(\lambda_p+\lambda_k)] \quad \text{for} \quad [\beta\lambda_p^2-(\lambda_p+\lambda_k)] \underset{<}{>} 0$$

Since the condition always holds for $\gamma>0$, there is only one limit β_2 with $[\beta_2\lambda_p^2-(\lambda_p+\lambda_k)]=0$ or

$$\beta_2 = (\lambda_p+\lambda_k)/\lambda_p^2 \qquad (2.21)$$

If $\beta > \beta_2$ the condition (2.20) for γ must be satisfied, otherwise the convergence will not cover different eigen-

vectors.
Now, we can summarize the necessary parameter values for convergence to different eigenvectors:

$0 \le \beta \le \beta_1 = (\lambda_p - \lambda_k)/\lambda_p^2$	*no convergence to different eigenvectors possible*
$\beta_1 < \beta < \beta_2 = (\lambda_p + \lambda_k)/\lambda_p^2$	*convergence with no constraint on γ*
$\beta_2 < \beta$	*convergence, if $\gamma < 2/[\beta\lambda_p^2 - (\lambda_p + \lambda_k)]$*

Is there a set of parameters β and γ for all eigenvalues for which the convergence to different eigenvectors is guaranteed?

Unfortunately, we cannot rely on the parameter regime $\beta_2 > \beta > \beta_1$ because for certain eigenvalues $\lambda_k \sim 0$ the parameter β_2 can become maximally $\beta_2 \sim 1/\lambda_{max}$ which is not always bigger than $1/4\lambda_{min}$, implied by eq.(2.19). Thus, we have to consider the other possible interval $\beta > \beta_2$ with the weight vector component a_{ik} alternating in sign. For $\lambda_k \sim \lambda_p$, this transforms to $\beta > 2/\lambda_p$. Thus, to guarantee the different fixpoints for all pairs of eigenvalues λ_p, λ_k and all values of λ_p we have to choose

$$\beta > 2/\lambda_{min} \quad \text{and by (2.20)} \quad \gamma < 2/\lambda^2_{max} \qquad (2.22)$$

as sufficient conditions for a proper convergence to all eigenvectors with different eigenvalues.

3 The simulation of the learning

For demonstration puposes let us regard the simulation of a picture processing procedure. By equation (2.2), the network implements a linear transformation which is well understood.

Let us concentrate on the more interesting part of the system: the learning of the Karhunen-Loéve transformation. Since the highlight of this transformation is the adaption to the sensor signal statistics, let us first choose a representative signal statistic for picture processing.

3.1 The training data

There have been many attempts to model the statistics of natural pictures, see e.g. [7]. One of the most convincing ones is by the autocorrelation function between the picture element (pel) $\mathbf{x}^1$ and pel $\mathbf{x}^2$

$$C(\mathbf{x}^1,\mathbf{x}^2) = \exp(-a|x_1^2 - x_1^1| - b|x_2^2 - x_2^1|) \qquad a \sim 0.2,\ b \sim 0.1 \qquad (2.23)$$

For the picture material presented in [7] the two coefficients *a* for horizontal correlations and *b* for vertical correlations are different, reflecting the flat, ordered arrangements of artificial building or pictures containing an horizon. For homogenous natural pictures (e.g. trees) which have no horizontal or vertical preferences, we might assume that *a=b* which covers the correlations in all directions uniformly by the euclidean distance $|\mathbf{x}^1 - \mathbf{x}^2|$ of the two pels

$$C(\mathbf{x}^1,\mathbf{x}^2) = \exp(-a|\mathbf{x}^1 - \mathbf{x}^2|) \qquad (2.24)$$

The analytical solutions for this case are well known [7]; the 2-dim eigenfunctions are products of cosinus and sinus functions. In the descrete case, the set of samples of these eigenfunctions are the eigenvectors of the corresponding autocorrelation matrix and have to be numerically computed in advance to serve as a reference for the convergence error.

How can the necessary autocorrelation matrix be constructed ? The autocorrelation matrix of the 2-dim picture matrix will be a 4-dim tensor. To remain in our ordinary notation and to use our ordinary numerical procedure for calculating the eigenvectors, we will instead construct an ordinary 2-dim autocorrelation matrix. For this purpose we concatenate all the M rows of N pels x_{hk} to one vector c of length n=NxM

$$\mathbf{c} = (x_{11}, .., x_{1N}, x_{21}, .. ,x_{2N}, .. , .. , x_{M1}, .. ,x_{NN})^T$$

The expected autocorrelation $\langle \mathbf{c}\mathbf{c}^T \rangle$ of this vector forms the autocorrelation matrix $C=(C_{ij})$ where every entry is of the form

$$C_{ij} = \langle x_{kh} x_{st} \rangle = \exp(-a|(k,h)-(s,t)|) = \exp(-a((k-s)^2+(h-t)^2)^{1/2})$$

Thus, to construct a nxn correlation matrix, the euclidean distances between all the pels must be computed via the prior picture frame .

3.2 A convergence example of the network weights

For the case of m=4 neurons the learning of the network is demonstrated. For 3x3 pictures, we have a 9x9 autocorrelation matrix with 9 eigenvectors and 9 eigenvalues. For a=0.2, the maximal eigenvalue is $\lambda_{max} = \lambda_0 = 6.814$, $\lambda_1 = \lambda_2 = 0.6395$ and $\lambda_3 = 0.2273$. According to eq. (2.25) we choose $\beta = 9.1 > 2/\lambda_3$ and $\gamma = 0.043 < 2/\lambda_0^2$. The weight vectors are randomly initialized and normalized to length 1. For the deterministic iteration by eq.(2.12) we use the correlation matrix C obtained in the prevous section. Figure 4 shows the convergence of the four weight vectors by their normalized projections $\mathbf{w}_i^T \mathbf{e}_k / |\mathbf{w}_i||\mathbf{e}_k|$ on the corresponding eigenvector, i.e. the cosinus between the weight vector and the eigenvector for t=1..5000 iterations on a logarithmic scale.

Since the second and third eigenvalues are equal, all linear combinations of the eigenvectorsare also eigenvectors. Instead of only one direction the whole plane $\mathbf{p}_{12}$ spanned by the two eigenvectors is the convergence goal. Thus, the convergence must be measured by the projection length of the weight vector into this plane.

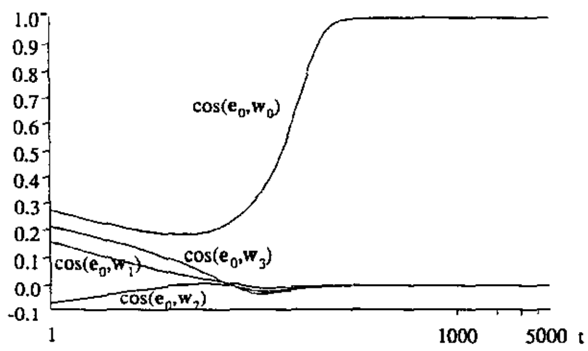

Fig. 4 The time course of the weight vector projections on eigenvector e_0

We see that the convergence is not straight and simple; there are quite complicated "movements" of the weight vectors in the input space. This is not surprising because the lateral inhibition can be compared to a repellent force in the many-body problem which yields such complicated time courses that is not yet analytically resolved.

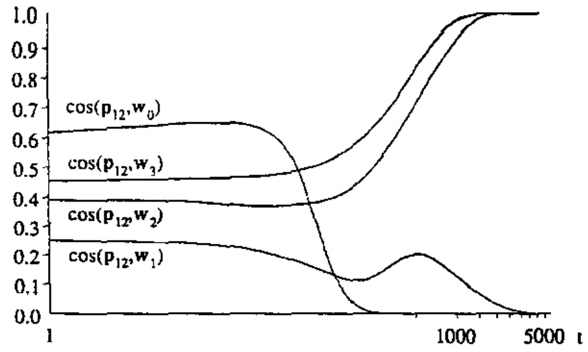

Fig. 5 The time course of the weight vector projections on eigenvector plane p_{12}

Now, is there a mean to speed up the convergence process ?

3.3 The developement of lateral inhibition

We know that we can not change the parameter regime very much to insure the convergence of the system. Nevertheless, for the biological counterpart we know that the main structure of the neurons are genetically preset and develops during the maturing of the organism. This is only true for the raw structure. The important feature in the system developement is the additional growth of the neural synapses and dendrites due to some data-specific, build-in pattern processing algorithm which we do not know yet. Nevertheless, we do know that in these systems the lateral network connections, and therefore also the inhibitions, grow by time to an important amount. What does this mean for our lateral inhibition network?

We know for uncoupled neurons (β=0), that each neuronal weight vector will converge independently to the eigenvector with the biggest eigenvalue λ_{max}. Only by the crosscorrelation influence of the lateral inhibition the weight vectors are driven to different eigenvectors. If we start with a small lateral inhibition β the system should be oriented towards the eigenvector with the biggest eigenvalue. On the basis of this, augmenting β should cause a readjustment of the system on basis of an already found eigenvector and should speed up the convergence for the rest of the weight vectors.

This idea is implemented by a linear function $\beta_i(t)= a_i t+b_i$ for increasing β as shown in figure 6.

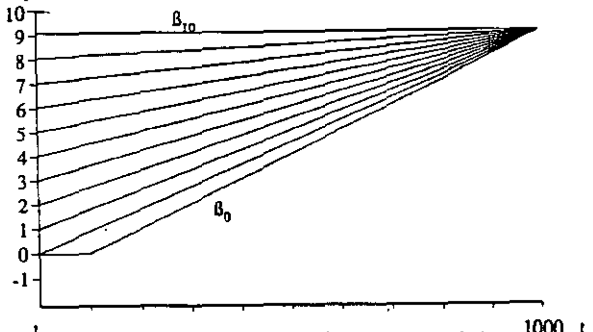

Fig. 6 The increase functions of the crosscorrelation

Here we have 11 different linear functions $\beta_i(t)$. All of them reach the necessary value β=9.1 .see above) for t=1000. This 11 different functions are used to iterate the same system of m=4 neurons of the previous section. The result is shown in figure 7 where the objective function $R(\beta_i,t)$ is plotted for different functions $\beta_i(t)$.

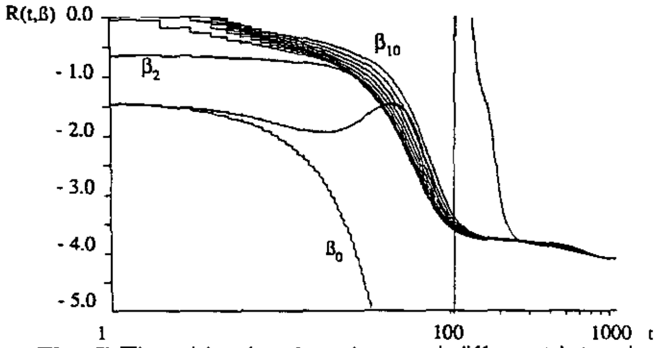

Fig. 7 The objective function and different lateral inhibition increase

As we can see, the increase of the lateral inhibition supports the convergence for a certain degree and yields better results as the constant inhibition $\beta_{10}(t)$=const=β. Nevertheless, if we prohibit the inhibition too long (e.g.by β_0 for 100 time steps), the network converges too much in the wrong direction and the new orientation slows the convergence down. It is interesting that this is only valid for the beginning of the convergence process until t=200. After this, the convergence speed does not depend any more of the differences in the inhibition factor β.

References

[1] R. Brause: The Minimum Entropy Neuron- A New Building Block for Clustering Transformations; in: Art. Neural Networks, I. Aleksander, J.Taylor (eds.), Elsevier Publ. Comp. (1992), pp.1095-1098

[2] R. Brause: The Minimum Entropy Network; Proc. IEEE Tools for Art. Intell. TAI-92, Arlington (1992)

[3] R. Brause: A VLSI-Design of the Minimum Entropy Neuron; in: J.Delgado-Fria, W. Moore(esds.): VLSI for Artificial Intelligence and Neural Networks, Plenum Publ. Corp., 1993

[4] Y.Chauvin: Principal Component Analysis by Gradient Descent on a Constrained Linear Hebbian Cell; IEEE Proc. Int. Conf. Neural Networks, pp. I/373-380 (1989).

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

[6] K. Hornik, C.-M. Kuan: Convergence Analysis of Local Feature Extraction Algorithms, Neural Networks, Vol.5, pp.229-240, 1992

[7] A. Habibi, P. Wintz: Image Coding by Linear Transformation and Block Quantization; IEEE Trans. on Comm. Techn., Vol. COM-19, No 1, pp.50-62, 1971

[8] A. K. Jain: Fundamentals of digital image processing, Prentice-Hall 1989

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

[10] T.Kohonen, E.Oja: Fast Adaptive Formation of Orthogonalizing Filters and Associative Memory in Recurrent Networks of Neuron-Like Elements; Biol. Cyb.21, pp.85-95 (1976)

[11] T. Kohonen, The "neural" Phonetic Typewriter of Helsinki Univ. of Technology, IEEE Computer, March 1988

[12] 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).

[13] C.M. Marcus, F.R. Waugh, R.M. Westervelt: Connection Topology and Dynamics in Lateral Inhibition Networks; in: R.Lippmann, J.Moody, D.Touretzky: Advances in Neural Information Processing Systems 3, Morgan Kaufmann Publ., San Mateo 1991

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

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

[16] E. Oja: Learning in non-linear Constrained Hebbian Networks; Proc. ICANN91, T.Kohonen et al. (Eds.), Artif. Neural Netw., Elsevier Sc. Publ. 1991, pp. 385-390

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

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

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

[20] L. Watson, R. Haralick, O. Zuniga: Constrained Transform Coding and surface fitting; IEEE Transactions on Communications, Vol. COM-31, No.5, pp.717-726, (1983)

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

[22] P. Wintz: Transform Picture Coding; Proc. IEEE, Vol. 60, No. 7, pp. 809-820 (1972)

Appendix A: The extrema of the objective function

Theorem:

The objective function (2.6) with condition $|\mathbf{a}_k| = 1$ has as necessary condition for an extremum $R(\mathbf{a}_1^*,..,\mathbf{a}_m^*)$ that the $\mathbf{a}_1^*,..,\mathbf{a}_m^*$ are eigenvectors of the autocorrelation matrix C.

Proof:

The constrained extrema of the objective function can be obtained by the method of Lagrange parameters. Here, we construct the function

$$L(\mathbf{a}_1,..,\mathbf{a}_m, \mu_1,..,\mu_m)=R(\mathbf{a}_1,..,\mathbf{a}_m)+\mu_1(|\mathbf{a}_1|^2-1)+ ..+ \mu_m(|\mathbf{a}_m|^2-1)$$

The $2m$ necessary conditions characterize the multivariate extrema

$$\partial L/\partial \mathbf{a}_k = 0 \quad \partial L/\partial \mu_k = 0 \qquad \forall k=1..m$$

and give us beside our restriction $|\mathbf{a}_k| = 1$ the conditions

$$\nabla_a L(\mathbf{a}_k^*)=\nabla_a R(\mathbf{a}_k^*)+\mu_i \nabla_a(|\mathbf{a}_k^*|^2-1)=0 \quad \forall k=1..m \quad \text{(A.1)}$$

Let us evaluate the gradient $\nabla_a R(\mathbf{a}_k)$ first. With a different ordering of the double sum of eq.(2.6) and by Eq. (2.1), only the terms containing y_k remain non-zero in $\nabla_a R(\mathbf{a}_k)$ and we get

$$\nabla_a R(\mathbf{a}_k) = \beta\Sigma_{j\neq k} \langle y_k y_j\rangle \nabla_a(\langle y_k y_j\rangle) - \langle y_k \nabla_a y_k\rangle$$

$$= \beta\Sigma_{j\neq k} \langle y_k y_j\rangle \langle \mathbf{x} y_j\rangle - \langle \mathbf{x} y_k\rangle \qquad \text{(A.2)}$$

andby (2.2) $\quad =[\beta C(\Sigma_{j\neq k} \mathbf{a}_j \mathbf{a}_j^T)C - C]\, \mathbf{a}_k \qquad$ (A.3)

The condition (A.1) becomes with (A.3)

$$\nabla_a L(\mathbf{a}_k^*) = [\beta C(\Sigma_{j\neq k} \mathbf{a}_j^* \mathbf{a}_j^{*T})C - C]\, \mathbf{a}_k^* + 2\mu_i \mathbf{a}_k^* = 0 \qquad \forall k=1..m$$

or $\quad [C-\beta C(\Sigma_{j\neq k}\mathbf{a}_j^* \mathbf{a}_j^{*T})C]\, \mathbf{a}_k^* = \theta_k \mathbf{a}_k^* \qquad \theta_k=2\mu_i \qquad$ (A.4)

This is an eigenvector equation for the matrix [.]. It is easy to see that this has as solution the eigenvectors of C: Suppose the $\mathbf{e}_k$ are all eigenvectors of C and we have s_i weight vectors $\mathbf{a}_j$ converged to eigenvector $\mathbf{e}_i$. Then (A.4) becomes

$$[C - \beta C(\Sigma_{j\neq k} \mathbf{a}_j^* \mathbf{a}_j^{*T})C]\, \mathbf{e}_k = (\lambda_k - \beta\lambda_k^2 s_k)\mathbf{e}_k = \theta_k \mathbf{e}_k$$

The eigenvectors of C are also the eigenvectors of the matrix [.] and fulfill condition (A.4). Therefore, they are also solutions for the extrema of the objective function, Q.E.D.

Appendix B: The iteration of the weight vector

One iteration step of the learning rule is by Eqs. (2.9) and (2.12) with $g = |\mathbf{w}_i(t) + \Delta\mathbf{w}_i|$

$$\mathbf{w}_i(t+1) = g^{-1} (\mathbf{w}_i(t) + \Delta\mathbf{w}_i)$$

$$= g^{-1} [\mathbf{w}_i(t)+\gamma(t)(C\mathbf{w}_i-\beta C(\Sigma_{j\neq i}\mathbf{w}_j \mathbf{w}_j^T)C\mathbf{w}_i)]$$

Let us write the weights $\mathbf{w}_i(t) = \Sigma_l\, a_{il}(t)\, \mathbf{e}_l$ in the base of the orthonormal eigenvectors $\mathbf{e}_1,..,\mathbf{e}_n$ of C. Then, the k-th component a_{ik}, denoted also by $[.]_k$, evolves to

$$a_{ik}(t+1) = g^{-1}\{a_{ik}(t) + \gamma ([C (\Sigma_l\, a_{il}(t)\, \mathbf{e}_l)]_k - [\beta C(\Sigma_{j\neq i}\mathbf{w}_j \mathbf{w}_j^T)C (\Sigma_l\, a_{il}(t)\, \mathbf{e}_l)]_k) \}$$

$$= g^{-1}\{a_{ik}(t)+\gamma\lambda_k a_{ik}(t) -\gamma[\beta C\, \Sigma_{j\neq i}(\Sigma_l\, a_{jl}(t)\mathbf{e}_l)(\Sigma_l\, a_{jl}(t)a_{il}(t)\, \lambda_l)]_k\}$$

With the abbreviation $b_{ij} := \Sigma_l\, a_{jl}(t)\, a_{il}(t)\, \lambda_l$ we finally get

$$a_{ik}(t+1)= g^{-1}\{a_{ik}(t) + \gamma\, \lambda_k\, a_{ik}(t) -\gamma[\beta C\, \Sigma_{j\neq i}\Sigma_l\, a_{jl}(t) b_{ij}\, \mathbf{e}_l\,]_k \}$$

$$= g^{-1}\, a_{ik}(t)\, \{ 1+ \gamma\lambda_k\, (1 - \beta\Sigma_{j\neq i}\, a_{jk}(t) b_{ij}\, /a_{ik}(t)) \} \qquad \text{(B.1)}$$