Proc. ICANN-93,
pp. 486-489
Springer 1993

A Symmetrical Lateral Inhibition Network for PCA and Feature Decorrelation

Rüdiger W. Brause,
J.W. Goethe-University, FB20 NIPS,
D - 60 054 Frankfurt, Germany

Abstract

This paper introduces a new network model for data decorrelation and principal component analysis which relays on the biological plausible lateral inhibition. Different to already existing approaches, the assignment of the eigenvectors to the neuronal weights are not predefined but in the lateral inhibited network the weights evolve by the network dynamics alone to the eigenvectors of the input data crosscorrelation matrix.

The paper introduces the model and an objective function, presents the learning equations and computes the conditions for the parameters to assure the convergence of the weight vectors to different eigenvectors.

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.

The encoding processe should consist of two stages: a linear transformation and a quantization of the output signals. In this paper we consider mainly the linear transformation. If we use the same number m of output channels as there are input lines, the $m=n$ output values y_i are just the projection of the input $\mathbf{x}$ on the vectors $\mathbf{w}_i$ or the coordinates of $\mathbf{x}$ in a new base system $\{\mathbf{w}_i\}$. When the $\mathbf{w}_i$ are linear independent and complete then we do not loose information and a complete reconstruction of the input by $\mathbf{y} = (y_1,...,y_m)$ is possible.

However, if we use with $m<n$ less output lines than input lines we will make a reconstruction error. For linear systems, it is well known that the mean square error is minimized by selecting only those base vectors (eigenvectors) with the biggest eigenvalues [12]. Thus, the eigenvector decomposition (*descrete Karhunen-Loéve transformation, principal component analysis (PCA)*) can be considered as an *optimal* transformation and should be preferred to all other current linear transformations as the descrete Walsh-Hadamard transformation, the descrete Fourier transformation or the descrete Cosinus transformation [6].

Since Oja's statement [7] that a linear, formal neuron using Hebb's learning rule and restricted weights will learn the eigenvector with the biggest eigenvalue several neural network architectures were proposed for a partial or complete eigenvector decomposition. Basically, they consist of two categories: networks which learns the eigenvectors sequentially ("asymmetric networks") which are based on the sequential Gram-Schmidt orthogonalization mechanism, and networks which learn them in parallel ("symmetric networks") and do not predetermine an order of the eigenvectors. The approaches use linear neurons, where each neural weight vector converges to one eigenvector.

Examples of the former architectures are the Sanger decomposition network [10], the lateral inhibition network of Rubner and Tavan [9] and the asymmetrical version of the lateral inhibition network of Földiák [4]. They use as a basic building block the linear correlation neuron which learns the input weights by a Hebb-rule, restricting the weights $\mathbf{w}_1,..,\mathbf{w}_n$. As Oja

showed [7], this learning rule let the weight vector of the neuron converge to the eigenvector of the expected autocorrelation matrix **C** of the input patterns **x** with the biggest eigenvalue λ_{max}. 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 symmetrical Oja subspace network [8], the Williams subspace learning [13] and the symmetrical decorrelation network of Silva and Almeida [11] have the same property: They assume the propagation of weight values in the network which is not desirable neither from the biological point of view nor from the pathway restrictions of VLSI-implementations.

In fact, a fully symmetrical, stable network for eigenvector decomposition, construced by an objective function and implemented by a biological plausible and easily realizable network mechanism is still missing. Contrary to the opinion of Hornik and Kuan [5], who are not in favour of an symmetric PCA network due to convergence problems (e.g. the model of Földiák [4]), we will introduce a new symmetrical, stable model in this section which is not covered by their general convergence analysis of the PCA models mentioned above.

2 The symmetric network for eigenvector decomposition

Let us assume in a first step that we have *m* neurons which are laterally interconnected as shown in figure 1.

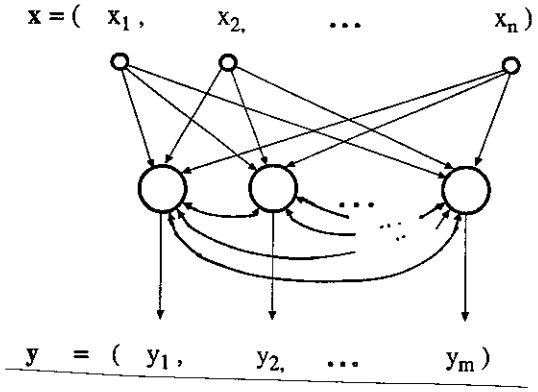

Fig. 1 The symmetric, lateral interconnected network model

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

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

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 by introducing a special threshold weight learned with an Anti-Hebb-rule, see [2].

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. 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 [5].
Then the output for neuron *i* becomes with Eq. (2.1)

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

and the output vector becomes $2\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{U}\mathbf{y}$ or $(2\mathbf{I}-\mathbf{U})\mathbf{y} = \mathbf{W}\mathbf{x}$ with the identity matrix **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.

3 Learning the weights

The learning rule for the weights $\mathbf{a}_i$ is determined by the following three conditions

- The new features should be decorrelated $\langle y_i y_j \rangle = \langle y_i \rangle \langle y_j \rangle = 0$ (3.1)
- The variance of the features should be maximal $\Sigma_i \langle y_i^2 \rangle = \max$ (3.2)
- The decomposition should be neutral (no scaling) $|\mathbf{a}_i| = 1$, i.e. $\det(\mathbf{A}) = 1$ (3.3)

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

$$R(\mathbf{a}_1,..,\mathbf{a}_m) = 1/4 \, \Sigma_i \Sigma_{j \neq i} (\langle y_i y_j \rangle)^2 - 1/2 \, \alpha \Sigma_i \langle y_i^2 \rangle = R^1 + R^2 \qquad (3.4)$$

The first term R^1 ensures that the cross-correlation (3.1) is always counted positive. This results in a minimum of R(.) where the cross-correlation, becomes zero and $-R^2$, the sum of all variances, becomes maximal.

The third condition (3.3) have to be additionally ensured during the learning process. This condition could also be integrated into the objective function. It was shown for one neuron [3] that this yields also the eigenvectors as solutions and can be compared to the approach using (3.2) to compute the unique maximum and minimum of the objective function [2]. It can be shown that the objective function R(a) takes its extrema when the $\mathbf{a}_i$, the lines of the matrix **A**, are a subset of the eigenvectors of the autocorrelation matrix $\mathbf{C}_{xx} = \langle \mathbf{x}\mathbf{x}^T \rangle$. Since $\mathbf{C}_{xx}$ is symmetric and real, the eigenvalues λ_i are real and the eigenvectors form an orthogonal base system. Here, the cross-correlations

$$\langle y_i y_j \rangle = \mathbf{a}_i^T \langle \mathbf{x}\mathbf{x}^T \rangle \mathbf{a}_j = \mathbf{a}_i^T \mathbf{C}_{xx} \mathbf{a}_j = \mathbf{a}_i^T \lambda_j \mathbf{a}_j = 0 \qquad \forall \, i \neq j$$

become zero, and by (2.2), we have $\mathbf{U} = \mathbf{I}$ and $\mathbf{A} = (2\mathbf{I}-\mathbf{U})^{-1}\mathbf{W} = \mathbf{W}$. Thus, we can break up the weight vector **a** in two parts: the input weights $\mathbf{w}_i$ which should converge to the eigenvectors and the lateral inhibition weights u_{ij} which should become zero. The minimum of the objective function can therefore be approximated by a gradient search for the weight vectors $\mathbf{w}_i$ only where we assume the lateral inhibition to be a constant value at each learning step, learned seperately. The conditions (3.1), (3.2), (3.3) are ensured by using the objective function (3.4) for **a**=**w** which yields for $\mathbf{w}_i$ the eigenvectors as solution. The (t+1)-th iteration step is

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) - \gamma(t) \, \nabla_w R(\mathbf{w}_i) \qquad (3.5)$$

denoting the gradient by the Nabla-operator ∇_w. With the definition $u_{ij} = -\langle y_i y_j \rangle$ the learning rule is computed as

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \gamma(t) \langle \mathbf{x} \, (\alpha y_i + \Sigma_{j \neq i} u_{ij} y_j) \rangle \qquad (3.6)$$

The stochastic version is obtained by dropping the expectation brackets "⟨ ⟩".
The lateral weights should be updated by a rule which let them become the expected cross-correlation. It can be shown that for the average value of $-y_i(t)y_j(t)$ at step *t* can be obtained by

$$u_{ij}(t) = u_{ij}(t-1) - 1/t \, (u_{ij}(t-1) + y_i(t) y_j(t)) \qquad (3.7)$$

This learning rule gets the average of the random variable $v = y_i y_j$. But, this is not the quantity we are looking for, because v is not stationary for changing **w**. 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 of the learning rule (3.7) the temporal floating average of *N* observed data.

4 Stability conditions for the learning fixpoints

It is well known that the sequential gradient descend algorithm (3.5) confirms a monotonic decrease of the quadratic function (3.4), because we have

$$\frac{\partial R(t)}{\partial t} = \frac{\partial R(a)}{\partial a}\frac{\partial a}{\partial t} = -\frac{\partial R(a)}{\partial a}\,\gamma(t)\,\frac{\partial R(a)}{\partial a} = -\gamma\left(\frac{\partial R}{\partial a}\right)^2 \le 0 \qquad (4.1)$$

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

It can be 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 have *not* to be necessarily *different* ones. To ensure different eigenvectors, we can compute the condition for the case when all weight vectors but one have converged to different eigenvectors. This leads to the condition (see [1]) for the autocorrelation

$$\alpha < \lambda_k^2/(\lambda_k - \lambda_l) \qquad (4.2)$$

for all eigenvalues λ_k and λ_l. Additional conditions are obtained by a local fixpoint analysis for α and for the learning rate γ, see [1]. Thus, the convergence region is limited by (4.2), but to ensure convergence at fixed values of α and γ for all different eigenvectors we should choose

$$\alpha < 4\lambda_{min}, \qquad \gamma < 2/\lambda^2_{max} \qquad (4.3)$$

which is valid for deterministic iterations of (3.6).

References

[1] R.Brause: Transform Coding by Lateral Inhibited Neural Nets; Internal Report (1993)

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

[3] 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).

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

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

[6] 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

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

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

[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] F. Silva, L. Almeida: A Distributed Solution for Data Orthonormalization; Proc. ICANN-91, Artificial Neural Networks, Elsevier Sc.Publ., pp.943-948 (1991)

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

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