Neural Processing Letters **11:** 59–78, 2000.

Building Adaptive Basis Functions with a Continuous Self-Organizing Map

MARCOS M. CAMPOS and GAIL A. CARPENTER
Boston University, Center for Adaptive Systems and Department of Cognitive and Neural Systems, 677 Beacon Street, Boston, MA 02215, U.S.A., E-mail: gail@cns.bu.edu

Abstract. This paper introduces CSOM, a continuous version of the Self-Organizing Map (SOM). The CSOM network generates maps similar to those created with the original SOM algorithm but, due to the continuous nature of the mapping, CSOM outperforms the SOM on function approximation tasks. CSOM integrates self-organization and smooth prediction into a single process. This is a departure from previous work that required two training phases, one to self-organize a map using the SOM algorithm, and another to learn a smooth approximation of a function. System performance is illustrated with three examples.

Key words: basis functions, continuous function approximation, competitive learning, interpolation, neural networks, on-line learning, self-organizing map

1. Introduction

The traditional SOM algorithm (von der Malsburg [13]; Grossberg [9]; Kohonen [10]) is a competitive learning system that maps inputs to discrete points on a lattice, where lattice points correspond to coordinates of units that win the competititon. This scheme produces a limited representation for the inputs, since each input vector is represented by one node in a finite grid. The SOM discrete coding scheme often yields poor performance when used for function approximation: it can produce only a piecewise-constant approximation, with precision limited *a priori* by the number of coding nodes. To overcome this limitation, this paper proposes CSOM (*C*ontinuous *S*elf-Organizing *M*ap), a four-layer feedforward neural network (Figure 1). The main innovation of the model is the use of a distributed SOM to implement a continuous, topology-preserving coordinate transformation from the input space to a regular lattice (Figure 2). CSOM self-organizes maps in the same fashion as the traditional SOM. However, the distributed activity in the CSOM layer creates a more powerful coding scheme and allows improved function approximation.

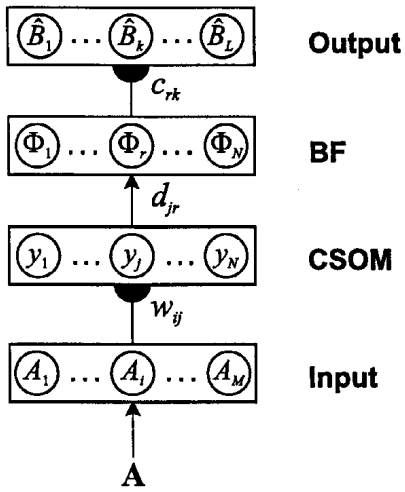

Figure 1. The CSOM network. A continuous learned coordinate transformation maps input **A** to an *S*-dimensional lattice of nodes at the CSOM layer, which is in turn mapped to a basis function (BF) layer. A second learned map transforms a combination of basis functions to the predicted output $\hat{\mathbf{B}}$.

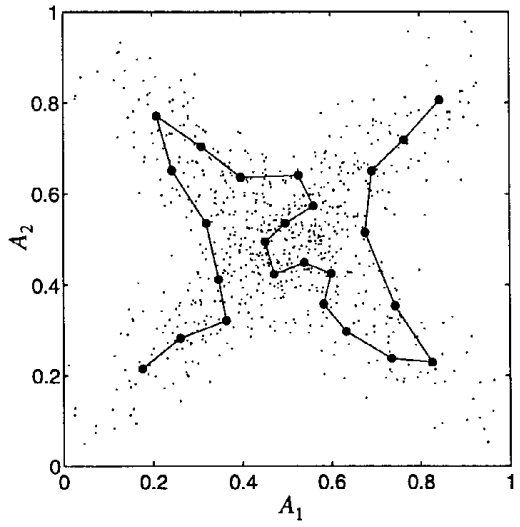

Figure 2. CSOM creates a one-dimensional map ($S = 1$) to represent a two-dimensional data distribution ($M = 2$) with a single feature. The map formation process can be interpreted as defining a new variable that represents the data more concisely. In the winner-takes-all case of the traditional SOM, the new variable takes only integer values along the grid (dark dots along the line: $N = 25$). In CSOM, the data points are mapped in a continuous fashion onto the line. As a result the new feature defined by CSOM is a continuous variable.

2. The CSOM Algorithm

In CSOM, an interpolation step maps inputs to a new coordinate system in a continuous fashion. A fixed $N \times N$ weight matrix $\mathbf{D} = (d_{jr})$ then maps the CSOM layer onto a basis function (BF) layer. The elements of the matrix $\mathbf{D}$ are non-adaptive: they encode the receptive field of each node in the BF layer. That is, in the network shown in Figure 1, a node in the BF layer performs a filtering of the distributed activity at the CSOM layer. This is equivalent to computing the discrete convolution of the CSOM layer's activity with a weight vector $\mathbf{d}_r$ for each unit in the BF layer. In the algorithm the convolution is calculated as follows. The distributed activation of the CSOM layer is represented by a point $\mathbf{X}^* = (X_1^* \dots X_s^* \dots X_S^*)$ in a continuous S-dimensional grid. The discrete receptive field of the r^{th} BF unit is approximated by a continuous function $\Psi(\mathbf{X}^*, \mathbf{X}^r, \delta)$, where $\mathbf{X}^r$ is the vector of CSOM grid coordinates of the center of the receptive field of the unit and δ is a vector of parameters that define the shape of the function Ψ. Using this approximation, the activitity at the BF layer is computed by a direct function evaluation.

Activation of the CSOM network during training is implemented by the following algorithm, with parameters and variables listed in Tables I and II.

CSOM training algorithm

0. Set $t = 1$, distribute weights w_{ij} randomly in $[w^-, w^+]$, and set all weights $c_{rk} = 0$
1. Decrease the w_{ij} learning rate β:
$$\beta = \begin{cases} \beta_0(\beta_1/\beta_0)^{\frac{t-1}{t_1-1}} & \text{if } 1 \le t < t_1 \\ \beta_1 & \text{if } t \ge t_1 \end{cases}$$
2. Decrease the CSOM neighborhood size σ:
$$\sigma = \begin{cases} \sigma_0(\sigma_1/\sigma_0)^{\frac{t-1}{t_1-1}} & \text{if } 1 \le t < t_1 \\ \sigma_1 & \text{if } t \ge t_1 \end{cases}$$
3. Get the t^{th} input vector $\mathbf{A}$ and output vector $\mathbf{B}$
4. Calculate the coordinates $\mathbf{X}^*$ to which the input $\mathbf{A}$ is mapped in the CSOM grid (see interpolation step below)
5. Compute the activity of the CSOM layer:
$h_j = \exp(-||\mathbf{X}^j - \mathbf{X}^*||^2/2\sigma^2)$
6. Normalize the CSOM layer activity:
$y_j = h_j / \sum_{l=1}^{N} h_l$
7. Compute the activity of the BF layer:
$\Phi_r = \Psi(\mathbf{X}^*, \mathbf{X}^r, \delta) = \exp(-||\mathbf{X}^* - \mathbf{X}^r||^2/2\delta^2)$
8. Normalize BF layer activity: $\tilde{\Phi}_r = \Phi_r / \sum_{l=1}^{N} \Phi_l$
9. Compute the output: $\hat{B}_k = \sum_{r=1}^{N} c_{rk}\tilde{\Phi}_r$
10. Adjust c_{rk} according to: $\Delta c_{rk} = \alpha\tilde{\Phi}_r(B_k - \hat{B}_k)$
11. Adjust w_{ij} according to: $\Delta w_{ij} = \beta y_j(A_i - w_{ij})$
12. If $t = n$ then stop. Else add 1 to t and go to step 1

Table I. CSOM parameters.

Parameter	*Description*
α	learning rate for weights c_{jk}
β	learning rate for weights w_{ij}
σ	neighborhood size in the CSOM algorithm
δ	standard deviation used in Ψ
λ	regularization parameter
w^-	lower bound for initial weights w_{ij}
w^+	upper bound for initial weights w_{ij}
β_0	initial value for β
β_1	final value for β
σ_0	initial value for σ
σ_1	final value for σ
t_1	number of training set inputs needed for β and σ to decrease to β_1 and σ_1
n	total number of training or test set inputs
S	number of dimensions of the CSOM grid
N_s	number of nodes along dimension s of the CSOM grid, $N = \prod_{s=1}^{S} N_s$
$\mathbf{X}^-$	vector with the lower bound for each coordinate of the CSOM grid $(X_1^- \dots X_s^- \dots X_S^-)$
$\mathbf{X}^+$	vector with the upper bound for each coordinate of the CSOM grid $(X_1^+ \dots X_s^+ \dots X_S^+)$
$\mathbf{X}^j$	position, in the CSOM integer grid coordinate system, of the j^{th} unit in the CSOM layer
$\mathbf{X}^r$	position, with respect to the CSOM integer grid coordinate system, of the center of the receptive field of the r^{th} unit in the BF layer

Table II. CSOM variables.

Variable	*Description*
$\mathbf{A}$	input vector $(A_1 \dots A_i \dots A_M)$
$\mathbf{X}^*$	position to which the input vector $\mathbf{A}$ is mapped in the CSOM grid coordinate system
$\mathbf{h}$	vector of activities of the CSOM layer $(h_1 \dots h_j \dots h_N)$
$\mathbf{y}$	normalized vector of activities of the CSOM layer $(y_1 \dots y_j \dots y_N)$
Φ	vector of activities of BF layer $(\Phi_1 \dots \Phi_r \dots \Phi_N)$
$\tilde{\Phi}$	vector of normalized activities of BF layer $(\tilde{\Phi}_1 \dots \tilde{\Phi}_r \dots \tilde{\Phi}_N)$
$\mathbf{B}$	target output vector $(B_1 \dots B_k \dots B_L)$
$\hat{\mathbf{B}}$	actual output $(\hat{B}_1 \dots \hat{B}_k \dots \hat{B}_L)$
$\mathbf{w}_j$	weight vector from input layer to the j^{th} unit of the CSOM layer $(w_{1j} \dots w_{ij} \dots w_{Mj})$
$\mathbf{c}_k$	weight vector from BF layer to the k^{th} unit of the output layer $(c_{1k} \dots c_{rk} \dots c_{Nk})$

Besides how $\mathbf{X}^*$ is computed (step 4), the key difference between the CSOM algorithm and the traditional SOM algorithm is the use of a normalized activity vector (**y**) to drive the map adaptation (step 11). Normalization of CSOM activation allows the activity in the CSOM layer to be interpreted as probability. It can also lead to slow initial adaptation, but small values of y_j are balanced by a large initial learning rate ($1 < \beta_0$).

During testing the same algorithm is applied, with learning rates $\alpha = \beta = 0$ and output equal to $\hat{\mathbf{B}}$ for each input **A**. If the task is categorial, the maximum $\hat{B}_k$ value chooses the output class. Setting $\alpha = \beta = 0$ makes steps 1, 2, 5, 6, 10, and 11 unnecessary during testing, and thus these steps can be ommitted.

Step 4 in the CSOM algorithm implements a continuous transformation from the input space to the CSOM layer grid coordinates. In the traditional SOM algorithm the components of $\mathbf{X}^*$ take on integer values and represent the winning unit in the CSOM layer. This is equivalent to a piecewise-constant coordinate transformation. In CSOM the coordinate transformation is implemented in a piecewise-linear fashion. This is accomplished using the following local linear coordinate transformation, which is based on an algorithm developed by Göppert and Rosenstiel [7].

2.1. STEP 4: CSOM INTERPOLATION

In this step, the neighbors of the winning node define a local linear system **L** that is used to decompose the input vector **A**. The coordinates thus obtained in turn define coordinates in another local linear system (**P**), now in the grid space, that specify the position $\mathbf{X}^*$ to which **A** is mapped in the CSOM grid.

Let $s = 1 \ldots S$ be the index of the dimensions on the CSOM layer grid.

(a) Find the winning CSOM unit: $J = \arg\min_j ||\mathbf{A} - \mathbf{w}_j||$

(b) Compute the local bases (**L** and **P**) for J:

b.1. Set $s = 1$

b.2. Let Ω_s be the set of indices of the nearest neighbors of J along the map dimension s

b.3. Compute the projections η_{js}:

$$\eta_{js} = \frac{(\mathbf{w}_j - \mathbf{w}_J) \cdot (\mathbf{A} - \mathbf{w}_J)}{(\mathbf{w}_j - \mathbf{w}_J) \cdot (\mathbf{w}_j - \mathbf{w}_J)},\ j \in \Omega_s$$

b.4. Set $K(s) = \arg\max_j \{\eta_{js}\}$

b.5. Compute the M-dimensional local basis vector $\mathbf{l}_s$ in feature space:

$$\mathbf{l}_s = \begin{cases} \mathbf{0} \text{ if } J \text{ is an interior grid point; and } \eta_{js} \leq 0,\ \forall_j \in \Omega_s \\ \mathbf{w}_{K(s)} - \mathbf{w}_j \text{ otherwise} \end{cases}$$

b.6. Compute the S-dimensional local basis vector $\mathbf{p}_s$ in grid space:

$$\mathbf{p}_s = \mathbf{X}^{K(s)} - \mathbf{X}^J$$

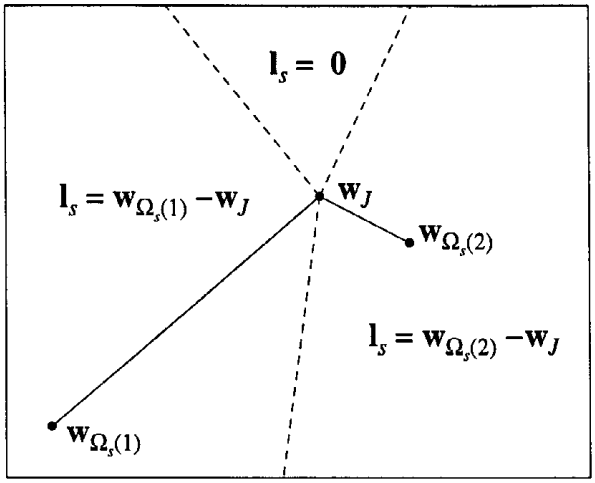

Figure 3. Selecting a basis vector for dimension s of a map. $\mathbf{w}_J$ is the weight vector for the winning unit J, and $\mathbf{w}_{\Omega_s(1)}$ and $\mathbf{w}_{\Omega_s(2)}$ are the weight vectors for the two neighbors of the unit J along dimension s in the grid. Depending in which of the three regions in the figure the input $\mathbf{A}$ falls, a different basis vector $\mathbf{l}_s$ is used for computing the affine coordinates $\mathbf{u}$.

b.7. If $s < S$ add 1 to s and go to step b.2

b.8. Define the local systems $\mathbf{L} = [\mathbf{l}_1, \mathbf{l}_2, \ldots \mathbf{l}_S]$ and $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_S]$

(c) Interpolate:

c.1. Compute the S-dimensional local affine coordinates $\mathbf{u}$: $\mathbf{u} = (\mathbf{L}^L\mathbf{L} + \lambda\mathbf{I})^{-1}\mathbf{L}^T(\mathbf{A} - \mathbf{w}_J)$

c.2. Set $v_s = X_s^J + \sum_{\mu=1}^{S} P_{s\mu}u_\mu$

c.3. Compute grid coordinates X_s^* for the input $\mathbf{A}$:

$$X_s^* = \begin{cases} X_s^- & \text{if } v_s < X_s^- \\ v_s & \text{if } X_s^- \le v_s \le X_s^+ \\ X_s^+ & \text{if } v_s > X_s^+ \end{cases}$$

The CSOM interpolation step first finds the grid unit J whose weight vector $\mathbf{w}_J$ is closest to the input (step 4(a)). The local system is then defined in terms of the nodes adjacent to the J^{th} unit in the S-dimensional CSOM layer grid. For each grid dimension s the neighbor with the largest projection η_{js} is selected (steps b.2–b.4). If both projections η_{js} are negative then $u_s = 0$ and the grid coordinate along dimension s remains equal that of the winner $J(\mathbf{X}_s^* = \mathbf{X}_s^J)$. This is equivalent to setting the local basis vector for that grid dimension to $\mathbf{0}$ (step b.5).

Figure 3 illustrates how selection of a local basis vector $\mathbf{l}_s$ depends on the location of input $\mathbf{A}$. Once a neighbor for each map dimension has been selected, an S-dimensional local coordinate system $\mathbf{L}$ is defined for the input space and another,

P, for the grid space (step b.8) (Figure 4). Using the local system **L** it is possible to compute affine coordinates for the vector $(\mathbf{A} - \mathbf{w}_J)$ (step c.1). The coordinates **u** can then be used with the local system **P**, defined in grid coordinates, and the grid position of the winning node $(\mathbf{X}^J)$ to create an initial representation (**v**) of input **A** in the CSOM grid (step c.2). Finally, the coordinates of **v** are truncated (step c.3) in order to keep them within specified bounds. This last step is necessary to guarantee that there will always be some CSOM activity in response to any input. This is crucial during the initial stages of the map formation, when the weight vectors $\mathbf{w}_j$ are typically packed together in a small area of the input space. In this situation input vectors can be far away from the weight vectors $\mathbf{w}_j$. Without bounds on CSOM grid coordinates, these input vectors would be mapped to coordinates far away from those of the nodes in the CSOM grid, which would cause the activity of the nodes in the grid to be zero.

Note that the matrix inversion operation in step c.1 is normally not too demanding because the number of dimensions used for the CSOM layer is usually small (maps with $S = 1$, 2, or 3 are typical). If a map created by CSOM has a topological defect, the matrix $\mathbf{L}^T\mathbf{L}$ may be singular. This problem can usually be avoided by an appropriate choice of grid topology and regularization parameter λ. Göppert and Rosenstiel (1995b, 1997) found that topologically well organized maps can have small values for λ while maps with topological defects require larger values for λ. In all simulations below, $\lambda = 0.001$.

3. Basis Functions

Step 5 of the CSOM algorithm specifies a gaussian radial basis function (RBF). The network may also be implemented with a variety of alternative basis functions (Figure 5). A key property of the radial basis functions created by CSOM is that they are adapted to the input distribution (Figure 6). In addition, the CSOM scheme allows for the specification of radial basis functions in a single training phase, in contrast to traditional RBF networks (Moody and Darken [11]), which uses two training phases. In two-phase training, the basis functions are first determined using an unsupervised algorithm (e.g., K-means). Standard deviations are then computed as average distances to the mean of each cluster. Alternative approaches such as the supervised growing cell structure (Fritzke [4]) use the variable topology of a map to select which nodes to include in computing the standard deviations of the basis functions, but this still requires a separate step. These approaches define the basis functions in feature space coordinates, with information on the location, in the feature space, of nearby nodes used to compute the standard deviations. This information changes constantly during training, unless the positions of the centers are kept fixed. In contrast, CSOM defines the basis function in grid coordinates. Because node positions do not change in this coordinate system, the standard deviation of the basis functions can be specified as a constant (δ) in the model. For gaussian basis functions and an integer CSOM grid, δ is set equal to $0.4247\sqrt{S}$.

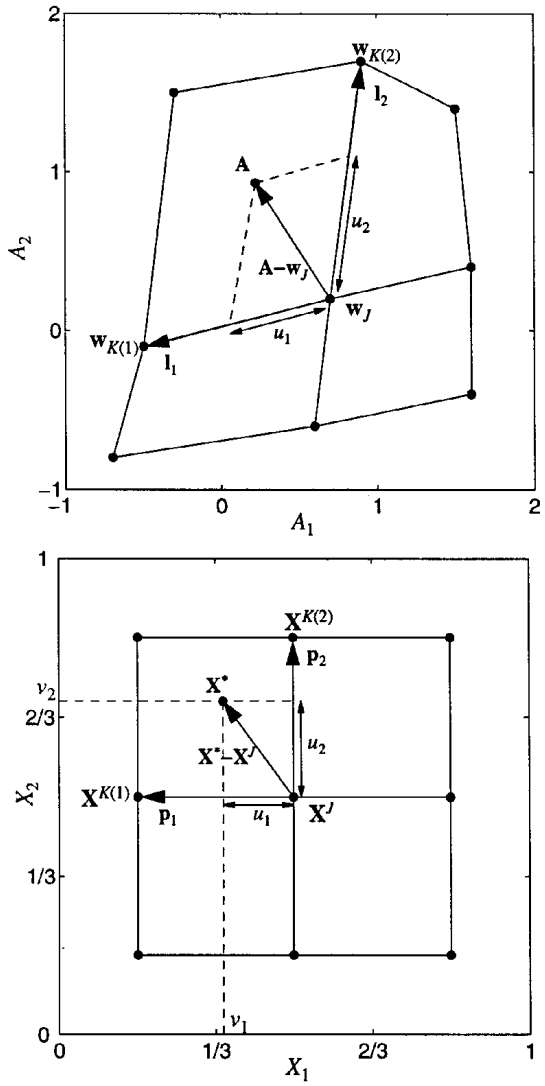

Figure 4. CSOM uses the local system $\mathbf{L} = [\mathbf{l}_1, \mathbf{l}_2]$ in the input space (above) to compute coordinates $\mathbf{u}$ for an input vector $\mathbf{A}$. These coordinates are then used with the local system $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2]$ in grid space (below) to construct $\mathbf{X}^*$, the position to which $\mathbf{A}$ is mapped in the grid space.

This value for δ causes the activity of each basis to equal 0.5 at a distance of $\sqrt{S}$ from the basis function center.

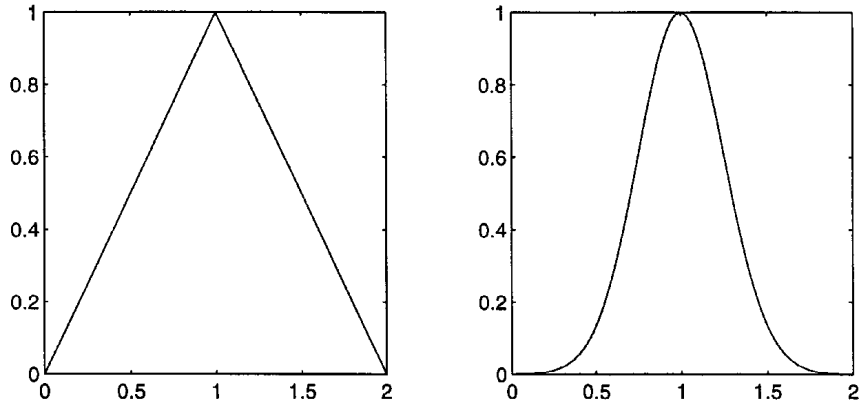

Figure 5. Examples of 1–D basis functions: hat function (left) and gaussian (right).

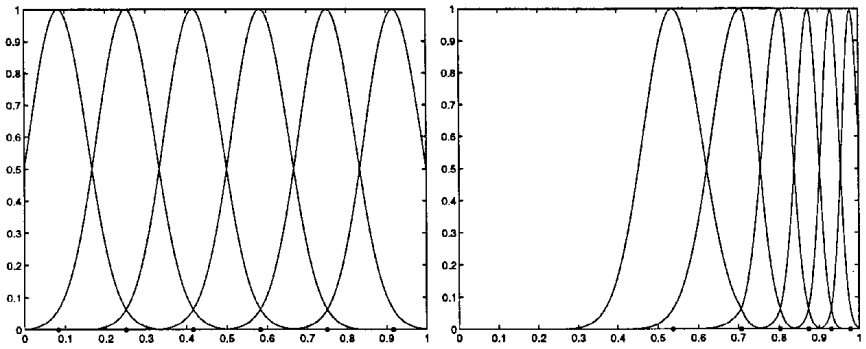

Figure 6. 1–D gaussian basis functions created by CSOM for two different input distributions: uniform distribution (left) and polynomial distribution (right). The dark dots mark the position in the input space of each CSOM grid node's weight vector $\mathbf{w}_j$. In the uniform case the nodes are evenly spaced, and the basis functions are symmetric. As a result the input space is evenly covered by the basis functions. In the polynomial case the basis functions are asymmetric and are concentrated in the region where most of the data points fall. This provides improved coverage of the input space where the input density is higher.

4. Examples

This section illustrates CSOM's capabilities with three function approximation tasks. In the first two tasks the CSOM grid has the same dimensionality as the input space ($S = M$). The third task illustrates how CSOM accomplishes dimensionality reduction. Performance is measured by the root mean squared error (RMSE), computed on a test set drawn from the same distribution as the training set, according to $\sqrt{\frac{1}{n}\sum_{t=1}^{n}||\mathbf{B}(t) - \hat{\mathbf{B}}(t)||^2}$. A measure of the *performance gain* for each model is computed as the percent reduction in RMSE compared to the traditional SOM algorithm. The parameters used in the simulations are listed in Table III. These

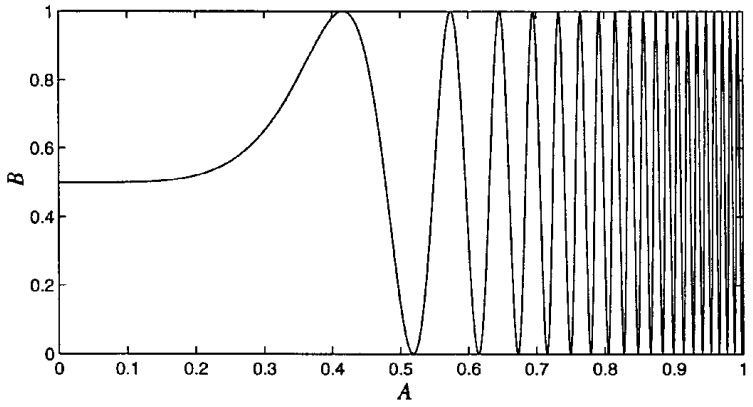

Figure 7. Fifth-order chirp signal used in the 1–D function approximation simulations: B = 0.5 + 0.5 sin $(\omega(A)A)$, where $\omega(A) = 40\pi A^4$.

values were determined, using only the training set data, by inspecting the quality of maps created by the CSOM and SOM algorithms.

4.1. 1–D FUNCTION APPROXIMATION

The 1–D function approximation task seeks to estimate a fifth-order chirp function (Figure 7). Simulations consider two input distributions, a uniform distribution and a polynomial distribution of degree four, which matches the chirp function frequency. The training set for each distribution consists of 7,000 input points A in the interval [0,1], with output B = 0.5 + 0.5 sin$(\omega(A)A)$, where $\omega(A) = 40\pi A^4$. The test set contains $n = 3,000$ observations drawn from the same distribution as the training set.

Four models were compared: the traditional SOM, a gaussian radial basis function network (GRBF), CSOM with gaussian basis functions (CSOM-G), and CSOM with hat basis functions (CSOM-L). Each model had 150 nodes in the hidden layer (or the CSOM layer), and each was tested in both a fixed grid mode and an adaptive grid mode. In the fixed grid mode the weights w_{ij} were initialized according to the distribution of the inputs and were not adapted, the same distribution of weights w_{ij} being used for all models. In the adaptive grid mode, CSOM–G and CSOM–L had the same final map because the adaptation of the CSOM layer is independent of the choice of basis function. The GRBF model used the same grid learned by CSOM after the training procedure. The standard deviation of the gaussian of each node was set equal to the mean distance to the node's neighbors in the grid. This procedure is the same as in the supervised version of the growing cell structure model (Fritzke [4]). The hat basis functions for CSOM–L were defined to drop to zero at the coordinates of neighboring nodes.

Table III. Simulation parameters.

All simulations		
Parameter	*SOM*	*CSOM*
α	0.1	0.1
w^-	–0.001	–0.001
w^+	0.001	0.001
β_1	0.01	0.01
σ_1	0.01	0.05
λ	–	0.001
δ	–	$0.4247\sqrt{S}$
X_s^-	–	–1
X_s^+	–	$N_s + 2$
1–D function approximation		
β_0	0.5	2
σ_0	30	50
t_1	2,000	4,000
N_1	150	150

2–D function approximation				
	SOM		*CSOM*	
Parameter	1D	2D	1D	2D
β_0	0.5	0.5	5	5
σ_0	4	4	5	5
t_1	5,000	5,000	5,000	5,000
N_1	64	8	64	8
N_2	–	8	–	8
Inverse kinematics				
β_0	0.5	0.5	5	5
σ_0	30	7	20	4
t_1	8,000	8,000	8,000	8,000
N_1	100	10	100	10
N_2	–	10	–	10

Table IV. 1–D function approximation: RMSE and gain compared to SOM for the chirp task, with 20 training epochs. Boldface type indicates the best performance in each column.

Fixed grid	*Input Distribution*			
	Uniform		*Polynomial*	
SOM	0.1340	(0.0%)	0.0876	(0.0%)
GRBF	**0.1099**	**(18.0%)**	**0.0112**	**(87.2%)**
CSOM–G	**0.1099**	**(18.0%)**	**0.0112**	**(87.2%)**
CSOM–L	0.1113	(16.9%)	0.0142	(83.8%)

Adaptive grid	*Input Distribution*			
	Uniform		*Polynomial*	
SOM	0.1260	(0.0%)	0.1017	(0.0%)
GRBF	0.1008	(20.0%)	0.0292	(71.3%)
CSOM–G	**0.1005**	**(20.0%)**	**0.0289**	**(71.6%)**
CSOM–L	0.1030	(18.3%)	0.0332	(67.4%)

Table IV summarizes performance of the four models with uniform and polynomial input distributions and with fixed and adaptive grids. Each system was trained for 20 epochs, where one epoch corresponds to one pass through the training set. The result for CSOM–G and CSOM–L are significantly better than for the traditional SOM. CSOM–G also performs slightly better than GRBF.

The polynomial distribution of nodes in fixed mode approximates an optimal distribution. The larger RMS errors for the polynomial input distribution in the adaptive mode (Table IV) reflect imperfect learning at the CSOM layer, whereas nodes in fixed mode are chosen *a priori* to perfectly reflect the input distribution. As shown by Ritter and Schulten [12], the SOM algorithm under-represents high stimulation regions in favor of low stimulation ones. This shortcoming seems to carry over to the continuous version implemented by CSOM. This limitation might be ameliorated by modifying the CSOM algorithm along the lines proposed by Bauer, Der, and Herrmann [2] for the SOM algorithm.

4.2. 2–D FUNCTION APPROXIMATION

The 2–D function approximation task simulated here is similar to the one used by Göppert and Rosenstiel [6]. The goal is to approximate the inverse function of a polynomial. The components (A_1, A_2) are calculated as polynomial expressions of the output components (B_1, B_2):

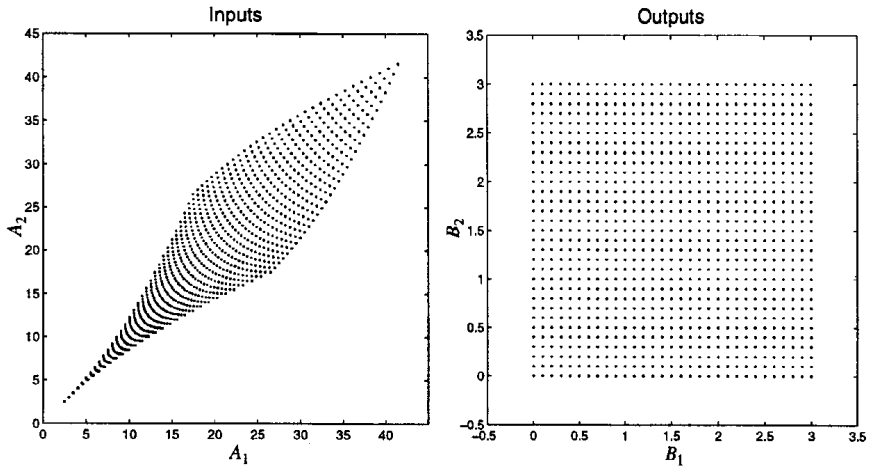

Figure 8. Input and output data distributions for the inverse function of polynomials task. A single data set consisting of 961 observations was used for training and testing. The output data were obtained from a 31×31 uniform grid and the respective inputs computed according to $A_1 = B_1^2 + 5(B_1 + B_2) + 2.5$ and $A_2 = B_2^2 + 5(B_1 + B_2) + 2.5$.

$$A_1 = B_1^2 + 5(B_1 + B_2) + 2.5$$

$$A_2 = B_2^2 + 5(B_1 + B_2) + 2.5.$$

A single data set consisting of 961 observations (Figure 8) was used for training and testing.

Three models were compared: SOM, CSOM–G, and GRBF. Each model had 64 nodes in the hidden layer or CSOM layer. For SOM and CSOM–G, the hidden nodes were arranged in an 8×8 grid. The GRBF network used the map learned by CSOM–G to compute the standard deviation of the gaussians. The standard deviations were taken to be the average distance to the neighboring nodes in the map.

Table V summarizes performance of the three models after 50 training epochs. The results for CSOM–G were considerably better than the traditional SOM and GRBF. Figure 9 show the maps learned by the SOM algorithm and CSOM–G, and the receptive fields of each basis function for GRBF and CSOM–G.

4.3. INVERSE KINEMATICS OF A TWO-JOINT ARM

The third example task is learning the inverse kinematics of a two-dimensional two-joint arm (Figure 10). For this task, given the end-effector position (x, y), a system is required to return joint angles (φ_1, φ_2) such that (x, y) and (φ_1, φ_2) are related through the forward kinematics equations:

Table V. 2–D function approximation: the inverse function of polynomials task, with 50 training epochs.

Model	*RMSE*	*Gain*
SOM	0.2639	0.0%
GRBF	0.0824	68.8%
CSOM–G	**0.0437**	**83.4%**

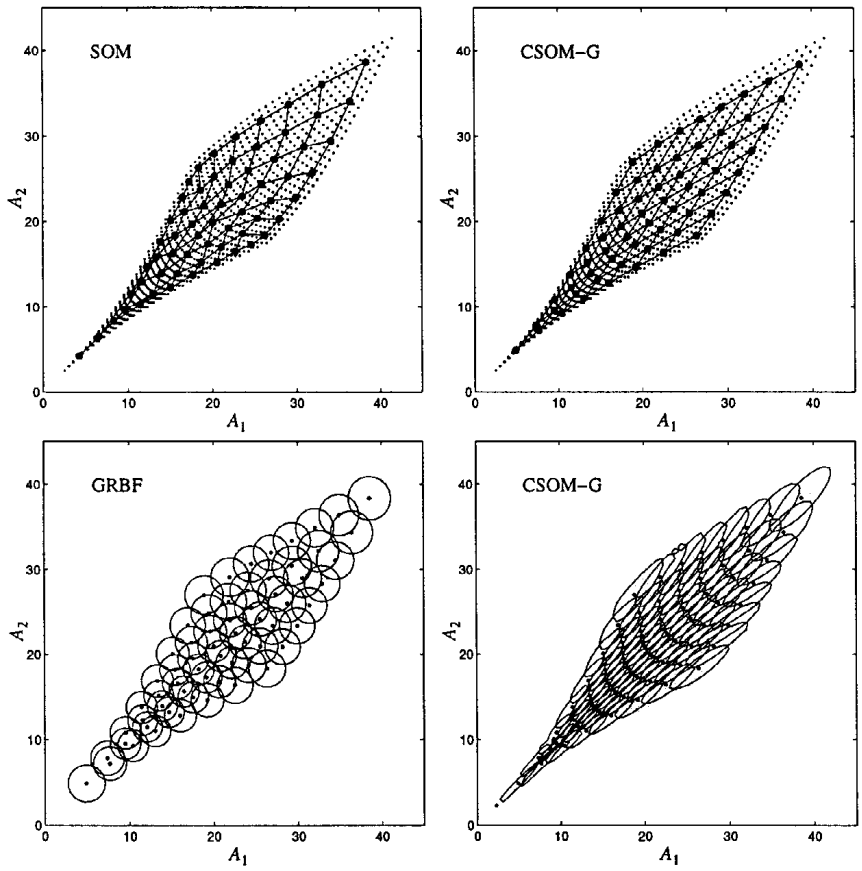

Figure 9. Final map configurations and receptive field shapes for the inverse function of polynomials task. Top row: maps learned by SOM and CSOM–G superimposed on the input distribution. CSOM–G self-organized a map similar to that of SOM. Bottom row: receptive field shapes of each basis function for GRBF and CSOM–G. The curves represent a non-normalized activity level (Φ_j) of 0.5. The graphics show how CSOM–G shapes the basis functions to match the distribution of the input data.

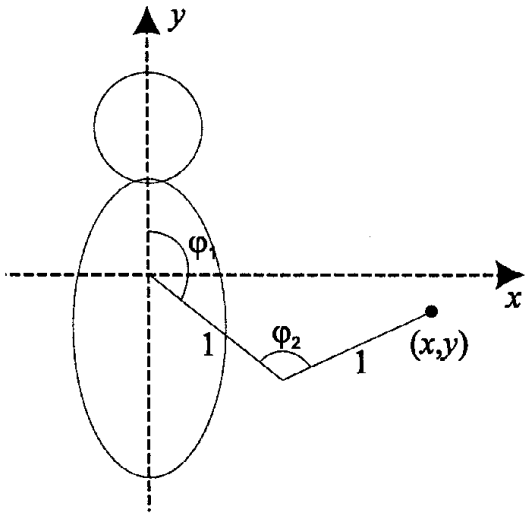

Figure 10. Coordinate definitions for the task of learning the inverse kinematics of a two-dimensional two-joint arm.

$$x = \sin\ \varphi_1 + \sin(\varphi_1 + \varphi_2 - \pi)$$

$$y = \cos\ \varphi_1 + \cos(\varphi_1 + \varphi_2 - \pi).$$

with $\pi/6 \leq \varphi_1 \leq 4\pi/3$ and $\pi/6 \leq \varphi_2 \leq \pi$. The training set consisted of 10,000 input and output points. The test set contained 3,000 observations. The data were obtained by randomly generating pairs of joint angles (φ_1, φ_2) and then computing the associated end-effector positions (x, y) using the forward kinematics equations. Components of the input and the output data were then scaled to fall in the interval [0, 1].

Four models were trained to solve this problem: a one-dimensional SOM (SOM1), a one-dimensional CSOM with gaussian basis functions (CSOM–G1), a two-dimensional SOM (SOM2), and a two-dimensional CSOM with gaussian basis functions (CSOM–G2). Each model had 100 units in the map layer. For the two-dimensional models these units were arranged in a 10×10 grid.

Table VI summarizes the performance for the two models, and Figure 11 shows the final map configuration for SOM1 and CSOM–G1. In both the one-dimensional and the two-dimensional cases, CSOM–G significantly outperformed the traditional SOM. Most remarkably, while the two-dimensional SOM (SOM2) had worse performance than the one-dimensional SOM (SOM1), the reverse held for CSOM–G. Due to the piecewise constant approximation implemented by SOM1 and SOM2, the quality of the approximation is limited by the quantization created by the models. Because of the shape of input distribution, the 2–D grid learned by SOM2 does a poorer job at covering the input region with nodes than the 1–D grid used by SOM1, and thus SOM2 has a larger RMSE than SOM1. For CSOM–G, the quality

Table VI. Dimensionality reduction: the inverse kinematics task, with 5 training epochs.

Model	*RMSE*	*Gain*
SOM1	0.0764	0.0%
CSOM–G1	**0.0612**	**19.9%**
SOM2	0.0784	0.0%
CSOM–G2	**0.0482**	**38.5%**

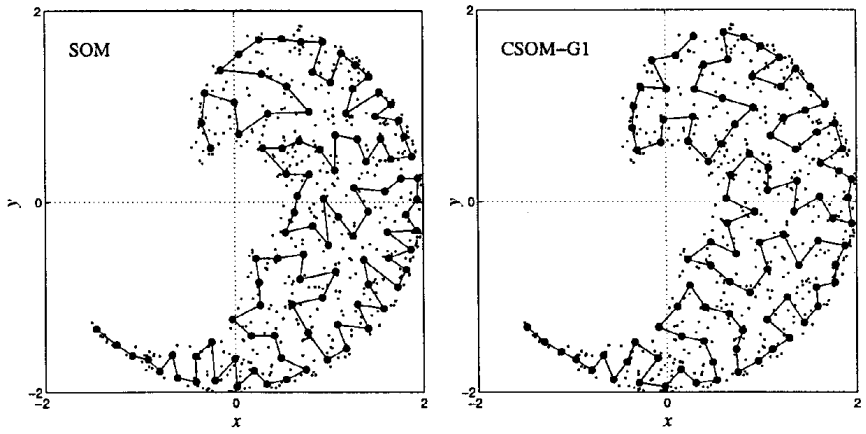

Figure 11. Final map configurations superimposed on the input distribution for the one-dimensional models used in the inverse kinematics of a two-dimensional two-joint arm task. Inputs correspond to "hand" positions (x, y), with angle constraints $\pi/6 \leq \varphi_1 \leq 4\pi/3$ and $\pi/6 \leq \varphi_2 \leq \pi$. CSOM–G1 self-organized a map similar to the one created by SOM, and each map does a good job of quantizing the input space.

of the approximation is a function of the basis functions used as well as the quantization. Even though CSOM–G2 does not quantize the inputs as well as CSOM–G1, the 2–D basis functions used by CSOM–G2 provide better interpolation than the ridge-like functions in CSOM–G1. As a result, CSOM–G2 has a smaller RMSE than CSOM–G1.

The one-dimensional CSOM–G1 shows how CSOM works when the dimensionality of the input space ($M = 2$) is greater than that of the map ($S = 1$). This is illustrated in more detail in Figures 12 and 13 for a map with 10 units trained with the same data as in Figure 11. The smaller number of nodes in the map makes it easier to visualize the CSOM–G1 basis functions.

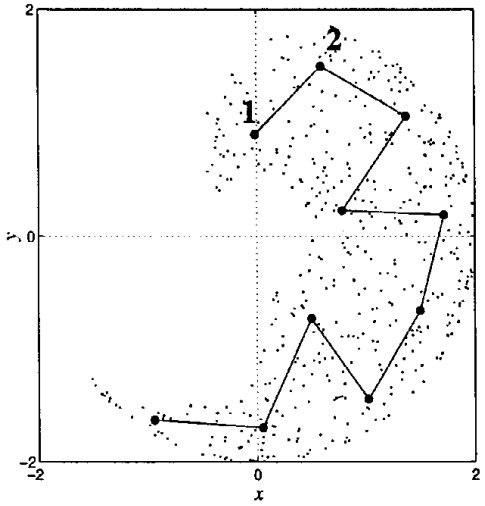

Figure 12. Input distrubtion and map topology for a 1–D map with 10 units trained with the data used in the inverse kinematics of a two-dimensional two-joint arm task. BF activity levels Φ_1 and Φ_2 are illustrated in Figure 13.

5. Related Work

The closest work to CSOM is the interpolation approach proposed by Göppert and Rosenstiel [6, 8] for the SOM algorithm. The similarities are accentuated because in the algorithm presented in this paper, CSOM, following that work, uses an affine transformation for finding intermediate positions in the CSOM grid. However the two approaches differ in their goals and overall implementation. Göppert and Rosenstiel propose a way to interpolate the predictions of a SOM *after* the map has been learned with the traditional SOM algorithm. In this way it is a two-stage process. CSOM completely integrates self-organization and smooth prediction. The two approaches also have different criteria for selecting neighbors and basis vectors for the local basis system. The differences between the algorithms derive from the goals of the two models. In particular, CSOM focuses on creating a continuous map that uses the affine transformation to define a continuous variable (feature), in the grid coordinates, that can then be used to define basis functions.

Another interpolation scheme for the SOM was proposed by Anguita, Passaggio, and Zunino [1], in the context of image compression. They proposed a simpler interpolation scheme that does not require the computation of the affine coordinates. Like Göppert and Rosenstiel, Anguita et al. introduce interpolation *after* the map has been learned with the traditional SOM. Although this scheme was not designed for function approximation tasks, it might be adapted to work with this class of problems. In this case, it could also be used to create an alternative implementation of CSOM by modifying the interpolation step.

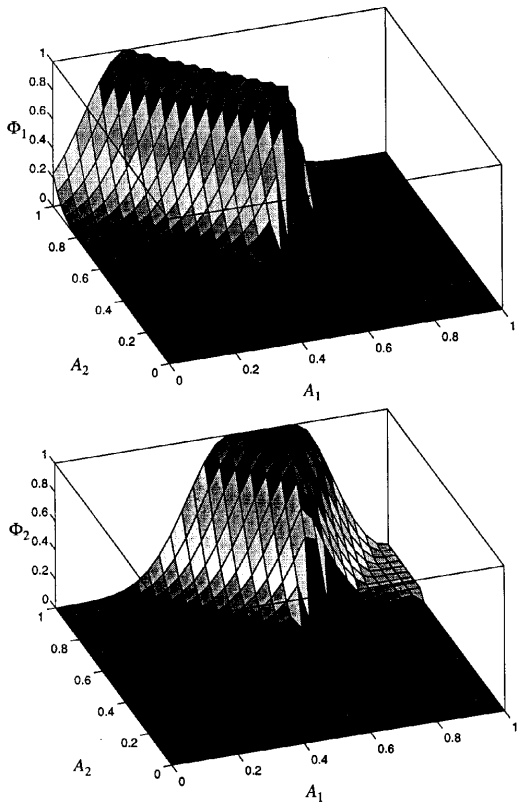

Figure 13. Non-normalized BF activity levels Φ_1 and Φ_2 for the gaussian basis functions for nodes 1 and 2 in Figure 12. Because of the dimensionality reduction, points along a direction orthogonal to a map segment are mapped to the same coordinate in the grid system. As a result, the basis functions are ridge-like functions oriented along the axes defined by the map connectivity, having a constant value in the direction orthogonal to a map segment.

Fritzke [4] has used maps with variable topology to design gaussian radial basis function networks. The topological information in the map is used to compute the standard deviations of the gaussians, which are defined in terms of input space coordinates. This approach requires constant updating of the standard deviation during training, and it generates only symmetric radial basis functions. In contrast, CSOM defines the basis functions in grid space (Section 3) and thus can generate basis functions that conform to input data distributions. The difference between the types of basis functions generated by the two approaches is illustrated in Figure 9, where the standard deviations of the basis functions for GRBF were computed using the approach suggested by Fritzke [4].

6. Conclusions

This paper introduces CSOM, a distributed version of the SOM algorithm capable of generating maps similar to those created with the original algorithm. Due to the continuous nature of the mapping, CSOM outperforms the SOM in function approximation tasks. CSOM also implements a new approach for building basis functions adapted to the distribution of the input data. The main idea proposed here is the mapping of the inputs onto a grid space in a continuous fashion and the specification of the basis functions in the grid space coordinates instead of the input space coordinates. Unlike related approaches, CSOM self-organizes the basis functions in a single training phase.

Two characteristic features of the CSOM model are its mapping or interpolating step, and its regular topology. Limitations of the current implementation are connected with the choices made for these two elements. The interpolation scheme has discontinuities in the first derivative at transition points (shifting from one winner to another and from choosing one neighbor over another) and is relatively expensive computationally. The investigation of alternative interpolation schemes is an area for future research. The use of a fixed topology for the CSOM layer can introduce topological defects in the map and negatively affect CSOM's performance. This can be improved by adapting the concepts introduced here to work with other approaches to topological map construction that use a variable topology, such as the growing cell structure (Fritzke [4]), growing grid (Fritzke [5]), or incremental grid growing (Blackmore and Miikkulainen [3]). A key aspect of these models is that the dimensionality of the grid is fixed, although the grid itself is incrementally built. The fixed dimensionality allows the system to define a coordinate system that can be exploited by CSOM. The incremental approach to grid construction allows the creation of grids that are adapted to the distribution of the inputs, thus minimizing the occurrence of topological defects and incorrect representation of the input probability distribution. The modification of CSOM to work with these incremental models is another area for future development.

Acknowledgments

This research was supported in part by the Office of Naval Research (ONR N00014–95–1–0409 and ONR N00014–95–1–0657).

References

1. Anguita, D., Passaggio, F. and Zunino, R.: SOM-based interpolation for image compression, In: *Proceedings of the World Congress on Neural Networks (WCNN'95)* **I**, Lawrence Erlbaum Associates, Mahwah, NJ, 1995, pp. 739–742.
2. Bauer, H.U., Der, R. and Herrmann, M.: Controlling the magnification factor of self-organizing feature maps, *Neural Computation* **8** (1996), 757–775.
3. Blackmore, J. and Miikkulainen, R.: Visualizing high-dimensional structure with the incremental grid growing neural network, In: A. Prieditis and S. Russell (eds.), *Machine Learning:*

Proceedings of the 12th International Conference (ICML'95), Kaufmann, San Francisco, 1995, pp. 55–63.

4. Fritzke, B.: Growing cell structures – a self-organizing network for unsupervised and supervised learning, *Neural Networks* **7**(9) (1994), 1441–1460.
5. Fritzke, B.: Growing grid – a self-organizing network with constant neighborhood range and adaptation strength, *Neural Processing Letters* **2**(5) (1995), 9–13.
6. Göppert, J. and Rosenstiel, W.: Interpolation in SOM: Improved generalization by iterative methods, In: F. Fogelman-Soulie and P. Gallinari (eds.), *Proceedings of the International Conference on Artificial Neural Networks (ICANN'95)*, EC2 & Cie, Paris, France, 1995a, pp. 69–74.
7. Göppert, J. and Rosenstiel, W.: Topology interpolation in SOM by affine transformations, In: M. Verleysen (ed.), *Proceedings of the 3rd European Symposium on Artificial Neural Networks (ESANN'95)*, D facto, Brussels, Belgium, 1995b, pp. 15–20.
8. Göppert, J. and Rosenstiel, W: The continuous interpolating self-organizing map, *Neural Processing Letters* **5** (1997), 185–192.
9. Grossberg, S.: Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors, *Biological Cybernetics* **23** (1976), 121–134.
10. Kohonen, T.: *Self-Organization and Associative Memory* (Second edition), Springer-Verlag, New York, 1988.
11. Moody, J. and Darken, C. J.: Fast learning in networks of locally-tuned processing units, *Neural Computation* **1**(2) (1989), 281–284.
12. Ritter, H. and Schulten, K.: On the stationary state of Kohonen's self-organizing sensory mapping, *Biological Cybernetics* **54** (1986), 99–106.
13. von der Malsburg, C.: Self-organization of orientation sensitive cells in the striate cortex, *Kybernetics* **14** (1973), 85–100.