Building Adaptive Basis Functions with a Continuous SOM

Marcos M. Campos
Gail A. Carpenter

Boston University
Center for Adaptive Systems and
Department of Cognitive and Neural Systems
677 Beacon Street
Boston, MA 02215
Fax Number: (617) 353-7755
Email: gail@cns.bu.edu

June, 1998

Submitted to *Proceedings of the ICCI&N'98 Conference*

Abstract

This paper introduces CSOM, a distributed version of the Self-Organizing Map network capable of generating maps similar to those created with the original algorithm. Due to the continuous nature of the mapping, CSOM outperforms the traditional SOM algorithm in function approximation tasks. System performance is illustrated with three examples.

Introduction

In the traditional SOM algorithm [4,5,7] inputs are mapped to discrete points on a lattice, corresponding to coordinates of the winning units. This results in a very limited representation for the inputs (each input is represented by an integer between 1 and the number of nodes in the map). This discrete coding scheme generally yields poor performance when using the SOM network for function approximation: the best it can do is a piecewise constant approximation of the function. To overcome this limitation, this paper proposes CSOM (*C*ontinuous *S*elf-*O*rganizing *M*ap). CSOM is a four-layer feedforward neural network (Figure 1). The main idea of the model is the use of a distributed SOM to implement a continuous coordinate transformation from the input space to a regular lattice (Figure 2). CSOM can self-organize 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.

The CSOM Algorithm

In CSOM, an interpolation step maps the inputs to the new coordinate system in a continuous fashion. The $N \times N$ weight matrix $\mathbf{D} = (d_{jr})$ then maps the CSOM layer onto a basis functions (BF) layer. The elements of this matrix are non-adaptive: they encode the receptive field of each node in the BF layer. In the network shown in Figure 1, each 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 the shape of the receptive field of each unit in the BF layer. In the algorithm this computation is represented by the following approximation: the distributed activation of the CSOM layer is indexed by a single point $\mathbf{X}^*$, and the discrete receptive field of the BF units is approximated by a continuous function $\Psi(\mathbf{X}^*, \mathbf{X}^r, \delta)$, where $\mathbf{X}^r$ is the CSOM grid coordinates of the center of the receptive field of the r^{th} unit in the BF layer, and δ is the vector of parameters controlling the shape of the function. Using this approximation, the activity at the BF layer can be computed using a direct function evaluation.

The behavior of the CSOM network during training can be implemented by the following algorithm.

CSOM algorithm

0. Set $t = 1$, distribute weights w_{ij} uniformly in $[w_-, w_+]$, and set all $c_{rk} = 0$
1. Decrease the 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 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 t^{th} input vector $\mathbf{A}$ and output vector $\mathbf{B}$
4. Find the coordinates $\mathbf{X}^*$ of the input $\mathbf{A}$ in the SOM grid (see interpolation step below)

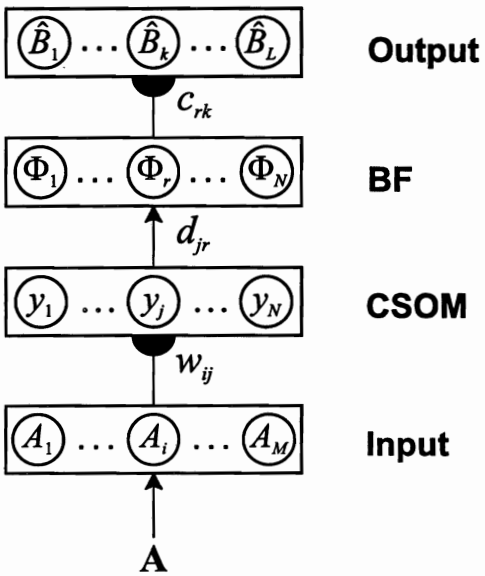

Figure 1: The CSOM network. A continuous learned coordinate transformation maps input $\mathbf{A}$ to a lattice of nodes at the CSOM layer, which is in turn mapped to a basis function (BF) layer. A second learned map transforms a combination of basis functions to the predicted output $\hat{\mathbf{B}}$.

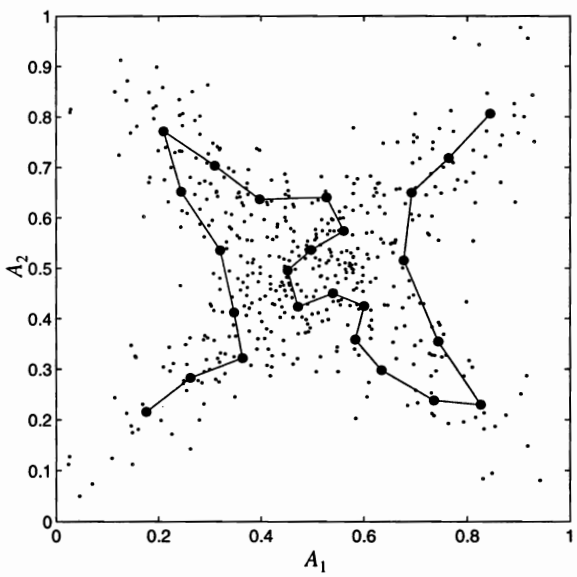

Figure 2: Using a one-dimensional SOM to represent a two-dimensional data distribution with a single feature. The map formation process of the SOM algorithm can be interpreted as defining a new variable that represents the data more concisely. In the winner-takes-all case of the traditional SOM, the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

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 1

During testing the same algorithm is applied with $\alpha = 0$, $\beta = 0$, and output $\hat{\mathbf{B}}$ for all inputs $\mathbf{A}$. 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 map activity (y_j) to drive the map adaptation (step 11). This can lead to a very slow initial adaptation of the map, which can be counteracted by a large initial learning rate ($1 < \beta_0$). Step 4 in the above CSOM algorithm implements a continuous transformation from the input space to the CSOM layer grid coordinates. This is accomplished using the following local linear coordinate system, based on [3]:

CSOM interpolation step

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

(b) Compute local bases ($\mathbf{L}$ and $\mathbf{P}$) for J:

b.1. Set $s = 1$

b.2. Let Ω_s equal to the set of indices of the neighbors of J along the map dimension s

b.3. Compute the projections η_j:
   $$\eta_j = \frac{(\mathbf{w}_j - \mathbf{w}_J).(\mathbf{A} - \mathbf{w}_J)}{(\mathbf{w}_j - \mathbf{w}_J).(\mathbf{w}_j - \mathbf{w}_J)}, j \in \Omega_s$$

b.4. Set $K = \arg\max_j \{\eta_j\}$

b.5. Compute the local basis vector in feature space $\mathbf{l}_s$:
   $$\mathbf{l}_s = \begin{cases} \mathbf{0} \text{ if } \eta_j < 0, \forall j \in \Omega_s, \text{ and } ||\Omega_s|| = 2 \\ \mathbf{w}_K - \mathbf{w}_J \text{ otherwise} \end{cases}$$

b.6. Compute the local basis vector in grid space $\mathbf{p}_s$:
   $\mathbf{p}_s = \mathbf{X}^K - \mathbf{X}^J$

b.7. If $s < S$, where S is the number of dimensions of the CSOM layer grid, add 1 to s and go to (b.2)

b.8. Define 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 local affine coordinates $\mathbf{u}$: $\mathbf{u} = (\mathbf{L}^T\mathbf{L} + \lambda\mathbf{I})^{-1}\mathbf{L}^T(\mathbf{A} - \mathbf{w}_J)$

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

c.3. Compute grid coordinates for the input X_s^*:

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

First the unit closest to the input is found (step a). The local system is then defined in terms of the neurons adjacent to the winner (J) in the CSOM layer grid. For each grid dimension the neighbor with the largest projection η_j is selected (steps b.2-b.4). If all projections η_j are negative then the grid coordinate along that grid dimension will be the same as the winner's. This is equivalent to setting the local basis vector for that grid dimension to $\mathbf{0}$ (step b.5). Once a neighbor for each map dimension has been selected, a S-dimensional local coordinate system $\mathbf{L}$ can be defined for the input space and another, $\mathbf{P}$, for the grid space (step b.8). Using the local system $\mathbf{L}$ it is possible to compute affine coordinates $\mathbf{u}$ (step c.1). These coordinates describe the input vector in the local system $\mathbf{L}$. The problem with singular matrices $\mathbf{L}^T\mathbf{L}$ usually can be avoided by an appropriate choice of grid-topology and regularization parameter λ [3].

The coordinate $\mathbf{u}$ can then be used with the local system $\mathbf{P}$, defined in grid coordinates, and the grid position of the winning node ($\mathbf{X}^J$) to create an initial representation ($\mathbf{v}$) of input $\mathbf{A}$ in the SOM grid (steps c.2). Finally this initial representation is truncated (step c.3) in order to keep the coordinate values within specified bounds. This last step is necessary to guarantee that there will always be some 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 packed together in some small area of the input space.

Basis Functions

The CSOM network can be implemented with a variety of radial basis functions (Figure 3). A key property of the radial basis functions adapted by CSOM is that they conform to the input distribution (Figure 4). The CSOM scheme also allows for the specification of radial basis functions in a single training phase. In the traditional approach to RBF networks [6] two training phases are used. In a first phase the centers of the basis functions are determined using some unsupervised algorithm (e.g., K-nearest neighbors). Next the standard deviations are computed as the average distance to the K-nearest neighbors. Alternative approaches such as the supervised growing cell structure [1] use the variable topology of a map to select which nodes to include in computing the standard deviations

of the basis functions. However, this still has to be done as a separate step. The above mentioned approaches define the basis functions in feature space coordinates. This requires information on the location, in the feature space, of the nearby nodes in order to compute the standard deviations. This information changes constantly during training unless the positions of the centers are kept fixed. CSOM defines the basis function in the CSOM layer grid coordinates. Because the location of the nearby nodes do not change in this coordinate system, the standard deviation of the basis functions can be specified from the start as a fixed parameter in the model.

Examples

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

1-D Function Approximation

This task is to estimate a fifth-order chirp function (Figure 5). Simulations considered two input probability distributions, a uniform distribution and a polynomial distribution of degree four (equal to the degree of the chirp function frequency). The training set for each distribution consisted of input points A in the interval [0,1], with output $B = 0.5 + 0.5\sin(\omega(A)A)$, $\omega(A) = 40\pi A^4$.

Four models were compared: the traditional SOM, CSOM with gaussian basis functions (CSOM-G), CSOM with hat basis functions (CSOM-L), and gaussian radial basis function network (GRBF). All models had 150 nodes in the hidden layer (or the CSOM layer for CSOM). Each model was used in a fixed grid mode and an adaptive grid mode. For the adaptive grid mode, CSOM-G and CSOM-L had the same final map because the adaptation of the CSOM layer in CSOM is independent of the basis function used. 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 to the mean distance of the node's neighbors in the grid. Table 1 summarizes the performance for the different models and distributions after 20 training epochs, where each epoch corresponds to one pass through the training set. The results for CSOM-G and CSOM-L were significantly better than the traditional SOM. CSOM-G also performed as well as or better than GRBF.

2-D Function Approximation

This task is similar to the one used in [2]. The goal is to approximate the inverse functions of polynomials. The two input components are calculated as the following polynomial expressions of the two output components (Figure 6): $A_1 = B_1^2 + 5(B_1 + B_2) + 2.5$ and $A_2 = B_2^2 + 5(B_1 + B_2) + 2.5$. Three models were compared: SOM, CSOM-G and GRBF. All models had 64 nodes in the hidden layer (CSOM layer for CSOM). For SOM and CSOM the hidden nodes were arranged in a 8×8 grid. The GRBF network used the map learned by CSOM-G to compute the standard deviation of the gaussians as in the above example.

Table 2 summarizes the performance for the different models. The results for CSOM-G were considerably better than the traditional SOM and GRBF. Figure 7 shows the maps learned by the traditional SOM algorithm and CSOM, and the receptive fields of each basis function for GRBF and CSOMG-G.

Inverse Kinematics of a Two-Joint Arm

The task is learning the solution of the inverse kinematics of a two-dimensional two-joint arm. For a two-joint two-dimensional arm (Figure 8), given the end-effector position (x, y), the solution of the inverse kinematics problem returns joint angles (φ_1, φ_2) such that (x, y) and (φ_1, φ_2) are related through the following forward kinematics equations: $x = L_1 \sin\varphi_1 + L_2 \sin(\varphi_1 + \varphi_2 - \pi)$ and $y = L_1 \cos\varphi_1 + L_2 \cos(\varphi_1 + \varphi_2 - \pi)$. The training set consisted of input points $A = (x, y)$ with output $B = (\varphi_1, \varphi_2)$. The data was obtained by randomly generating pairs of joint angles (φ_1, φ_2) and then computing the associated end-effector position (x, y) using the above forward kinematics equations.

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). All models used 100 units in the map layer. For the two-dimensional models these units were arranged in a 10×10 grid.

Table 3 summarizes the performance for the different models. In both the one-dimensional and the two-dimensional cases, CSOM-G significantly outperformed the traditional SOM. Most remarkably, while the two-dimensional SOM version had a worse performance than the one-dimensional SOM, the same was not true for CSOM-G. Figure 9 shows the final map configuration for the SOM1 and CSOM-G1 models.

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

Conclusions

This paper introduced CSOM, a distributed version of the SOM capable of generating maps similar to those created with the original algorithm. Due to the continuous nature of the mapping, CSOM outperforms the traditional SOM algorithm in function approximation tasks. CSOM also implements a new approach for building basis functions adapted to the distribution of the input data using a simple parametric scheme. 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 other approaches, CSOM self-organizes the basis functions in a single training phase.

References

[1] Bernd Fritzke. Growing cell structures – a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[2] Josef Göppert and Wolfgang Rosenstiel. Interpolation in SOM: Improved generalization by iterative methods. In *Proceedings of ICANN'95*, Paris, France, 1995.

[3] Josef Göppert and Wolfgang Rosenstiel. Topology interpolation in SOM by affine transformations. In *Proceedings of ESANN'95*, 1995.

[4] Stephen Grossberg. Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134, 1976.

[5] T. Kohonen. *Self-Organization and Associative Memory.* Springer-Verlag, New York, second edition, 1988.

[6] J. Moody and C. J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–284, 1989.

[7] C. von der Malsburg. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetics*, 14:85–100, 1973.

Table 1: RMSE and gain compared to SOM for the chirp task (20 epochs)

Fixed grid	*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	*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 2: RMSE and gain for the inverse function of polynomials task (50 training epochs)

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

Table 3: RMSE and gain for the inverse kinematics task (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%

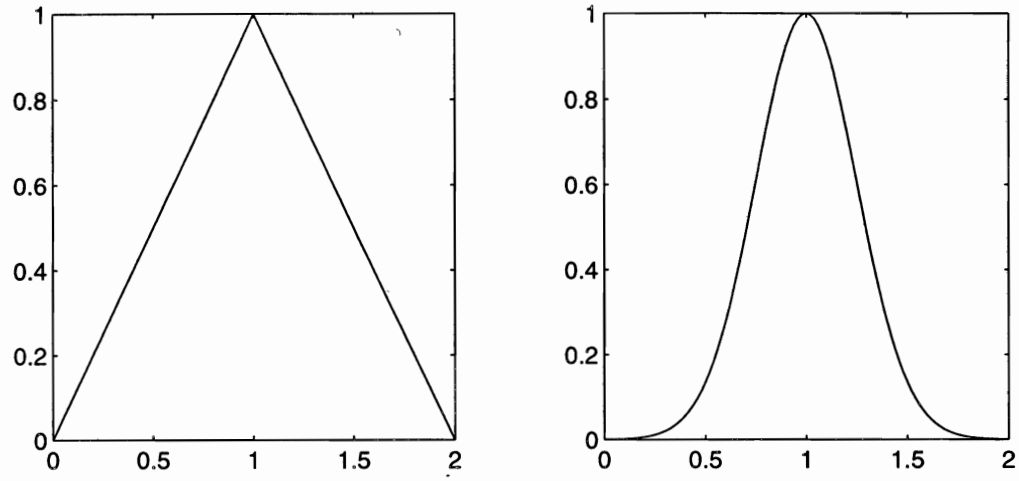

Figure 3: Examples of 1D basis functions: hat function (left) and gaussian (right).

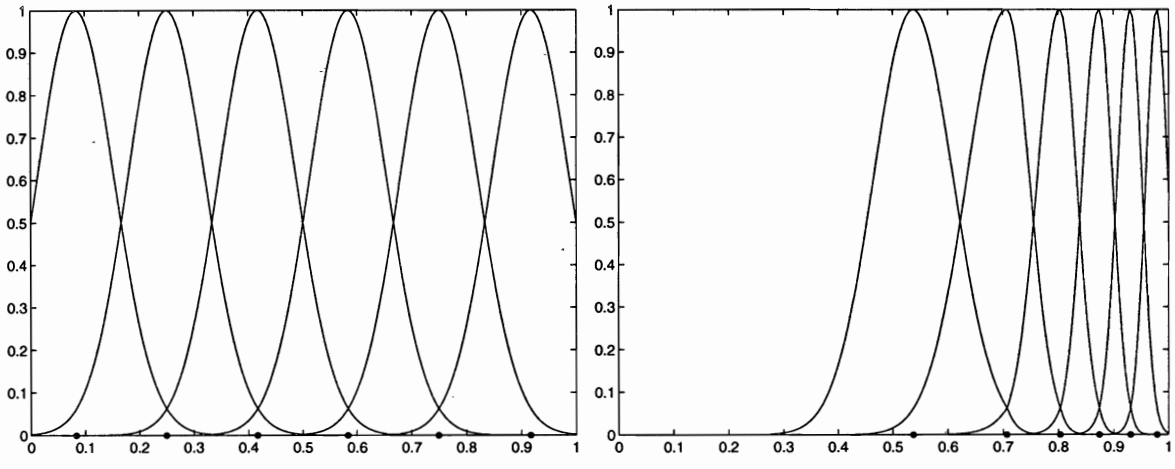

Figure 4: 1D gaussian basis functions for two different input distributions: uniform distribution (left) and polynomial distribution (right). The dark dots mark the position in the input space of each node's weight vector. In the uniform case the nodes are evenly spaced, and the basis functions are symmetric. 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 falls in (towards 1 in the horizontal axis). This provides improved coverage of the input space where the input density is higher.

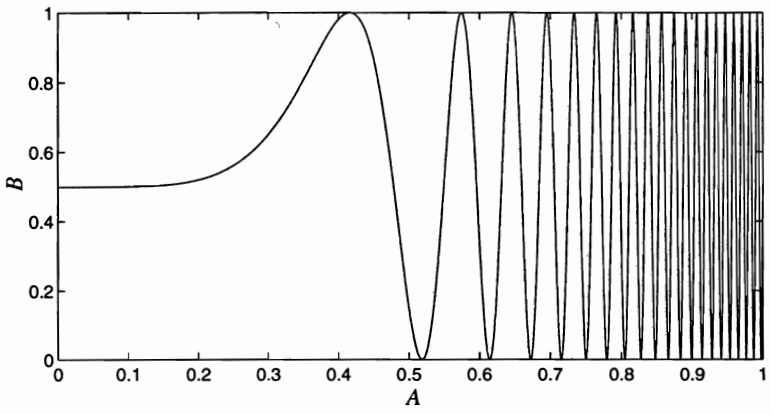

Figure 5: Fifth-order chirp signal used in the simulations. $B = 0.5 + 0.5\sin(\omega(A)A)$, $\omega(A) = 40\pi A^4$.

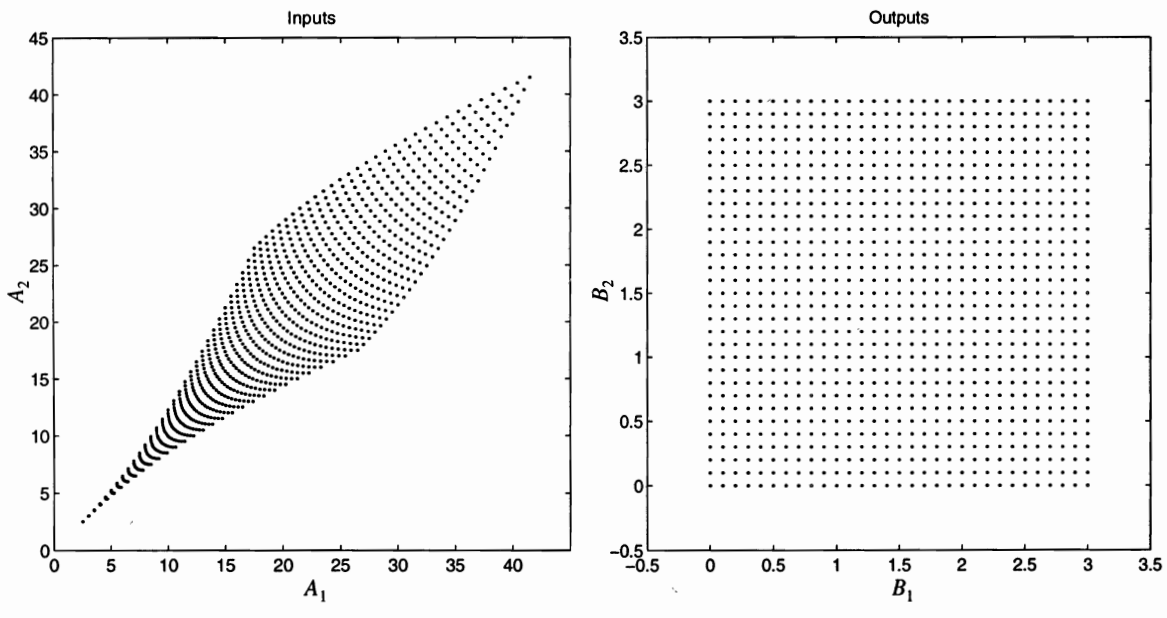

Figure 6: Input and output data distributions for the inverse function of polynomials task. A single dataset consisting of 961 observations was used for training and testing. The output data was obtained from a 31 by 31 uniform grid and the respective inputs computed according to $A_1 = B_1^2 + 5(B_1 + B_2) + 2.5$ and $A_2 = B_2^2 + 5(B_1 + B_2) + 2.5$.

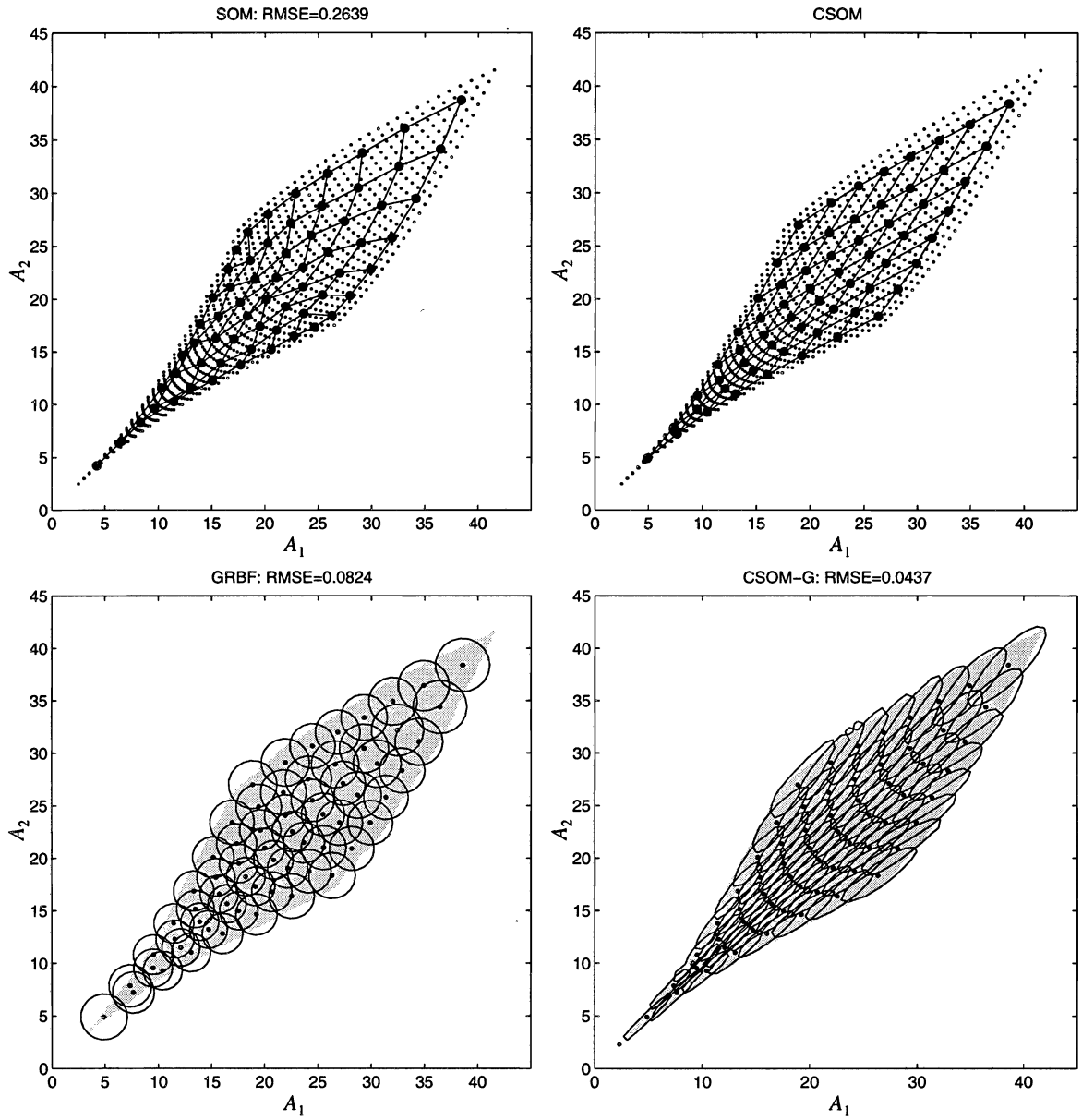

Figure 7: Final map configurations receptive field shapes for the inverse function of polynomials task. Top row: maps learned by the traditional SOM algorithm and CSOM. CSOM was capable of self-organizing a map similar to that of the traditional SOM. Bottom row: receptive field shapes of each basis function for GRBF and CSOM-G superimposed onto the input distribution (gray region). The curves represent a non-normalized activity level (Φ_j) of 0.5. The graphics show that CSOM does a better job at adapting the basis functions to the distribution of the input data.

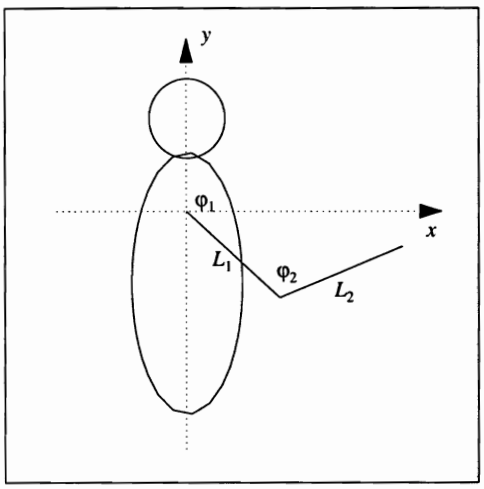

Figure 8: Coordinate definitions for the inverse kinematics of a two-dimensional two-joint arm task.

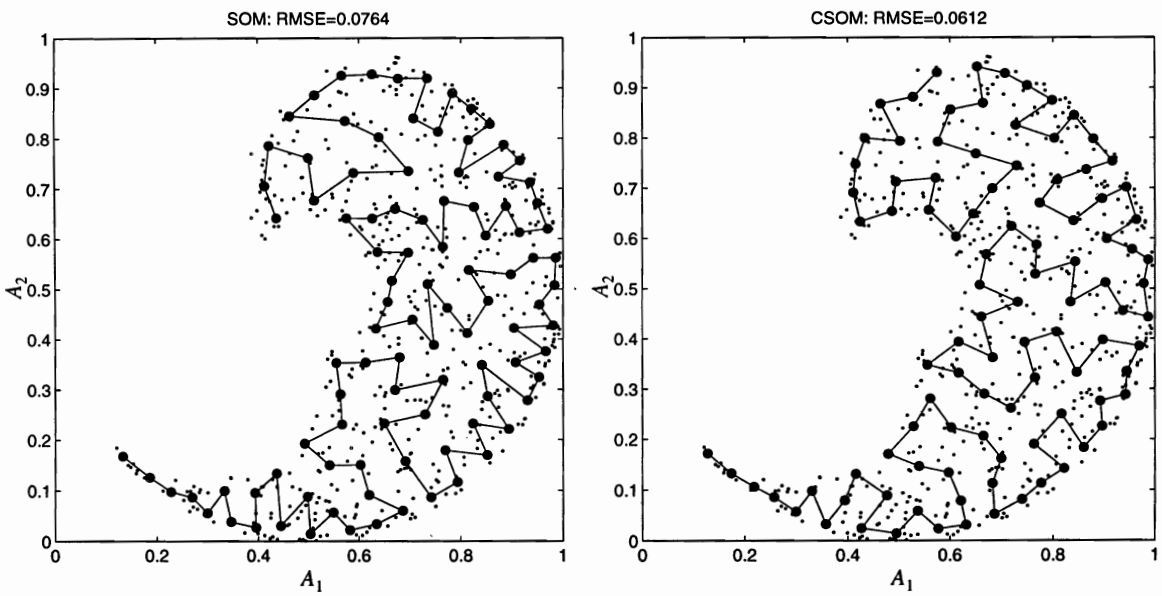

Figure 9: Final map configuration superimposed onto the input distribution for the one-dimensional models used in the inverse kinematics of a two-dimensional two-joint arm task. As illustrated by the graphics, CSOM-G1 was capable of self-organizing a map similar to the one created with the traditional SOM. Both maps do a good job at quantizing the input space.

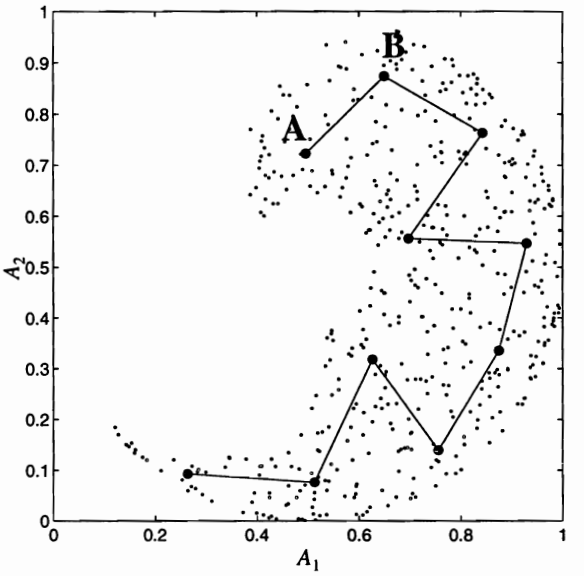

Figure 10: Input distribution and map topology for a 1D map with 10 units trained with the same data used in the inverse kinematics of a two-dimensional two-joint arm task.

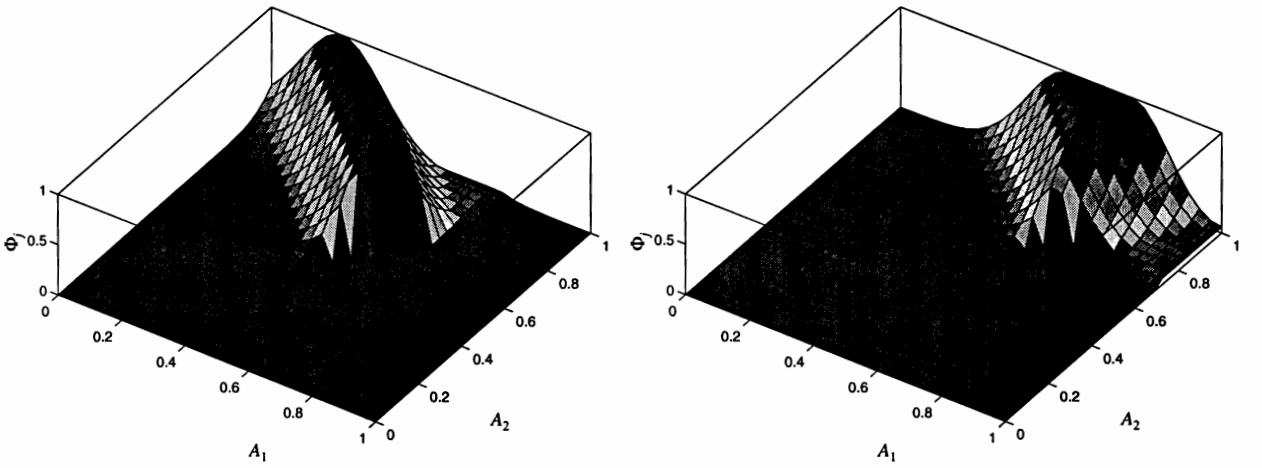

Figure 11: Gaussian basis functions for nodes A and B in Figure 10. Because of the dimensionality reduction, the basis functions are in fact ridge-like functions oriented along the axes defined by the map connectivity.