October, 2000

Clustering Irregular Shapes Using High-Order Neurons

H. Lipson
Hava Siegelmann, *University of Massachusetts - Amherst*

LETTER Communicated by Shimon Edelman

Clustering Irregular Shapes Using High-Order Neurons

H. Lipson
Mechanical Engineering Department, Technion—Israel Institute of Technology, Haifa 32000, Israel

H. T. Siegelmann
Industrial Engineering and Management Department, Technion—Israel Institute of Technology, Haifa 32000, Israel

This article introduces a method for clustering irregularly shaped data arrangements using high-order neurons. Complex analytical shapes are modeled by replacing the classic synaptic weight of the neuron by high-order tensors in homogeneous coordinates. In the first- and second-order cases, this neuron corresponds to a classic neuron and to an ellipsoidal-metric neuron. We show how high-order shapes can be formulated to follow the maximum-correlation activation principle and permit simple local Hebbian learning. We also demonstrate decomposition of spatial arrangements of data clusters, including very close and partially overlapping clusters, which are difficult to distinguish using classic neurons. Superior results are obtained for the Iris data.

1 Introduction

In classical self-organizing networks, each neuron j is assigned a synaptic weight denoted by the column-vector $\boldsymbol{w}^{(j)}$. The winning neuron $j(\boldsymbol{x})$ in response to an input $\boldsymbol{x}$ is the one showing the highest correlation with the input, that is, neuron j for which $\boldsymbol{w}^{(j)\boldsymbol{T}}\boldsymbol{x}$ is the largest,

$$j(\mathbf{x}) = \arg_j \max \|\mathbf{w}^{(j)T}\mathbf{x}\| \tag{1.1}$$

where the operator $\|\cdot\|$ represents the Euclidean norm of a vector. The use of the term $\boldsymbol{w}^{(j)\boldsymbol{T}}\boldsymbol{x}$ as the matching criterion corresponds to selection of the neuron exhibiting the maximum correlation with the input. Note that when the synaptic weights $\boldsymbol{w}^{(j)}$ are normalized to a constant Euclidean length, then the above criterion becomes the minimum Euclidean distance matching criterion,

$$j(\mathbf{x}) = \arg_j \min \|\mathbf{x} - \mathbf{w}^{(j)}\|, \quad j = 1, 2, \ldots, N. \tag{1.2}$$

However, the use of a minimum-distance matching criterion incorporates several difficulties. The minimum-distance criterion implies that the features of the input domain are spherical; matching deviations are considered

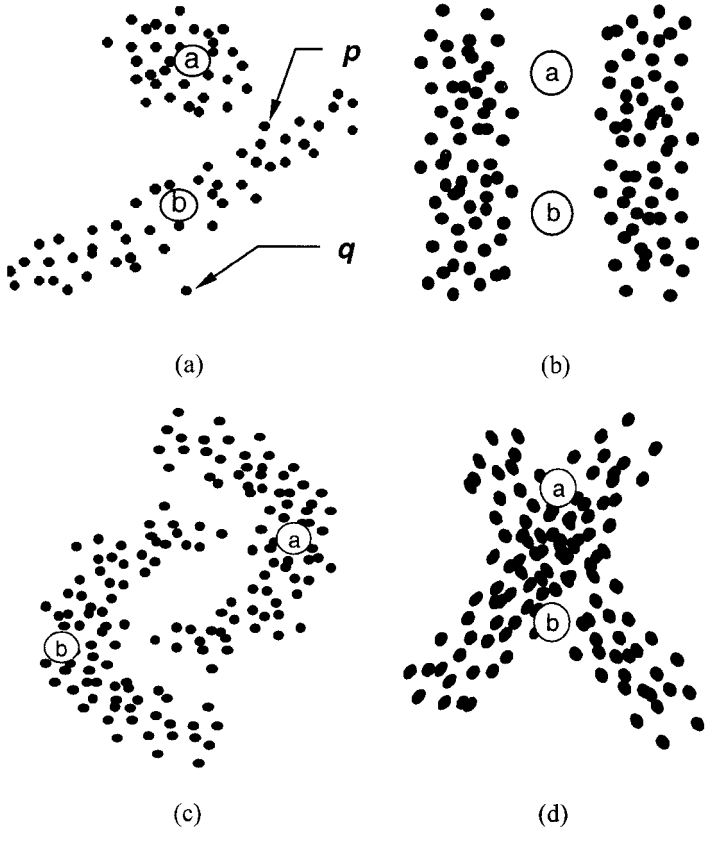


Figure 1: (a) Two data clusters with nonisotropic distributions, (b) close clusters, (c) curved clusters not linearly separable, and (d) overlapping clusters

equally in all directions, and distances between features must be larger than the distances between points within a feature. These aspects preclude the ability to detect higher-order, complex, or ill-posed feature configurations and topologies, as these are based on higher geometrical properties such as directionality and curvature. This constitutes a major difficulty, especially when the input is of high dimensionality, where such configurations are difficult to visualize and detect.

Consider, for example, the simple arrangements of two data clusters residing in a two-dimensional domain, shown in Figure 1. The input data points are marked as small circles, and two classical neurons (a and b) are located at the centers of each cluster. In Figure 1a, data cluster b exhibits a nonisotropic distribution; therefore, the nearest neighbor or maximum correlation criterion does not yield the desired classification because devia-

tions cannot be considered equally in all directions. For example, point p is farther away from neuron b than is point q, yet point p definitely belongs to cluster b while point q does not. Moreover, point p is closer to neuron a but clearly belongs to cluster b. Similarly, two close and elongated clusters will result in the neural locations shown in Figure 1b. More complex clusters as shown in Figure 1c and Figure 1d may require still more complex metrics for separation.

There have been several attempts to overcome the above difficulties using second-order metrics: ellipses and second-order curves. Generally the use of an inverse covariance matrix in the clustering metric enables capturing linear directionality properties of the cluster and has been used in a variety of clustering algorithms. Gustafson and Kessel (1979) use the covariance matrix to capture ellipsoidal properties of clusters. Davé (1989) used fuzzy clustering with a non-Euclidean metric to detect lines in images. This concept was later expanded, and Krishnapuram, Frigui, and Nasraoui (1995) use general second-order shells such as ellipsoidal shells and surfaces. (For an overview and comparison of these methods, see Frigui & Krishnapuram, 1996.) Abe and Thawonmas (1997) discuss a fuzzy classifier (ML) with ellipsoidal units. Incorporation of Mahalanobis (elliptical) metrics in neural networks was addressed by Kavuri and Venkatasubramanian (1993) for fault analysis applications and by Mao and Jain (1996) as a general competitive network with embedded principal component analysis units. Kohonen (1997) also discusses the use of adaptive tensorial weights to capture significant variances in the components of input signals, thereby introducing a weighted elliptic Euclidean distance in the matching law. We have also used ellipsoidal units in previous work (Lipson, Hod, & Siegelmann, 1998).

At the other end of the spectrum, nonparametric clustering methods such as agglomerative techniques (Blatt, Wiseman, & Domany, 1996) avoid the need to provide a parametric shape for clusters, thereby permitting them to take arbitrary forms. However, agglomerative clustering techniques cannot handle overlapping clusters as shown in Figure 1d. Overlapping clusters are common in real-life data and represent inherent uncertainty or ambiguity in the natural classification of the data. Areas of overlap are therefore of interest and should be explicitly detected.

This work presents an attempt to generalize the spherical-ellipsoidal scheme to more general metrics, thereby giving rise to a continuum of possible cluster shapes between the classic spherical-ellipsoidal units and the fully nonparametric approach. To visualize a cluster metric, we draw a hypersurface at an arbitrary constant threshold of the metric. The shape of the resulting surface is characteristic of the shape of the local dominance zone of the neuron and will be referred to as the shape of the neuron. Some examples of neuron distance metrics yielding practically arbitrary shapes at various orders of complexity are shown in Figure 2. The shown hypersurfaces are the optimal (tight) boundary of each cluster, respectively.

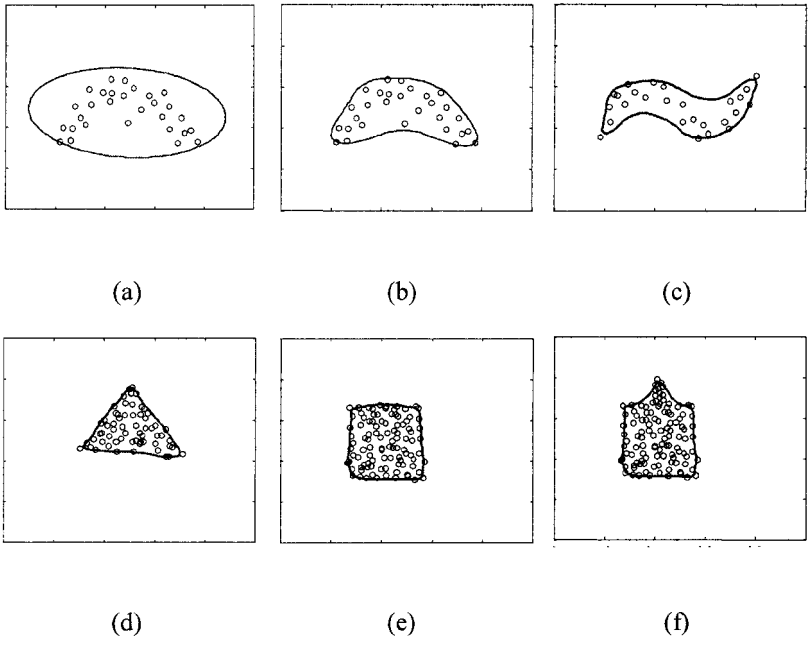
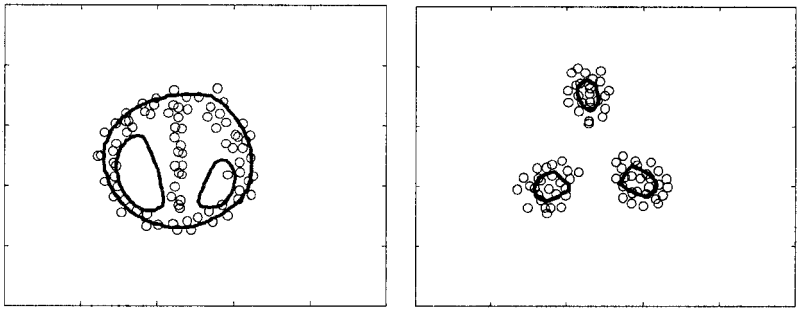


Figure 2: Some examples of single neuron shapes computed for the given clusters (after one epoch). (a–f) Orders two to seven, respectively.

In general, the shape restriction of classic neurons is relaxed by replacing the weight vector or covariance matrix of a classic neuron with a general high-order tensor, which can capture multilinear correlations among the signals associated with the neurons. This also permits capturing shapes with holes and/or detached areas, as in Figure 3. We also use the concept of homogeneous coordinates to combine correlations of different orders into a single tensor.

Although carrying the same name, our notion of high-order neurons is different from the one used, for example, by Giles, Griffen, and Maxwell, (1988), Pollack (1991), Goudreau, Giles, Chakradhar, & Chen (1994), and Balestrino and Verona (1994). There, high-order neurons are those that receive multiplied combinations of inputs rather than individual inputs. Those neurons were also not used for clustering or in the framework of competitive learning. It appears that high-order neurons were generally abandoned mainly due to the difficulty in training such neurons and their inherent instability due to the fact that small variations in weight may cause a large variation in the output. Some also questioned the usefulness of high-order statistics for improving predictions (Myung, Levy, & William, 1992). In this article we demonstrate a different notion of high-order neurons where



(a) (b)

Figure 3: A shape neuron can have nonsimple topology: (a) A single fifth-order neuron with noncompact shape, including two "holes." (b) A single fourth-order neuron with three detached active zones.

the neurons are tensors, and their order pertains to their internal tensorial order rather than to the degree of the inputs. Our high-order neurons exhibit good stability and excellent training performance with simple Hebbian learning. Furthermore, we believe that capturing high-order information within a single neuron facilitates explicit manipulation and extraction of this information. General high-order neurons have not been used for clustering or competitive learning in this manner.

Section 2 derives the tensorial formulation of a single neuron, starting with a simple second-order (elliptical) metric, moving into homogeneous coordinates, and then increasing order for a general shape. Section 3 discusses the learning capacity of such neurons and their functionality within a layer. Finally, section 4 demonstrates an implementation of the proposed neurons on synthetic and real data. The article concludes with some remarks on possible future extensions.

2 A "General Shape" Neuron

We adopt the following notation:

$\boldsymbol{x}, \boldsymbol{w}$	column vectors
$\boldsymbol{W}, \boldsymbol{D}$	matrices
$\boldsymbol{x}^{(j)}, \boldsymbol{D}^{(j)}$	the jth vector/matrix corresponding to the jth neuron/class
$\boldsymbol{x}_i, \boldsymbol{D}_{ij}$	element of a vector/matrix
$\boldsymbol{x}_H, \boldsymbol{D}_H$	vector/matrix in homogeneous coordinates

m order of the high-order neuron
d dimensionality of the input
N size of the layer (the number of neurons)

The modeling constraints imposed by the maximum correlation matching criterion stem from the fact that the neuron's synaptic weight $\boldsymbol{w}^{(j)}$ has the same dimensionality as the input $\boldsymbol{x}$, that is, the same dimensionality as a single point in the input domain, while in fact the neuron is modeling a cluster of points, which may have higher-order attributes such as directionality and curvature. We shall therefore refer to the classic neuron as a first-order (zero degree) neuron, due to its correspondence to a point in multidimensional input space.

To circumvent this restriction, we augment the neuron with the capacity to map additional geometric and topological information, by increasing its order. For example, as the first-order is a point neuron, the second-order case will correspond to orientation and size components, effectively attaching a local oriented coordinate system with nonuniform scaling to the neuron center and using it to define a new distance metric. Thus, each high-order neuron will represent not only the mean value of the data points in the cluster it is associated with, but also the principal directions, curvatures, and other features of the cluster and the variance of the data points along these directions. Intuitively, we can say that rather than defining a sphere, the high-order distance metric now defines a multidimensional-oriented shape with an adaptive shape and topology (see Figures 2 and 3).

In the following pages, we briefly review the ellipsoidal metric and establish the basic concepts of encapsulation and base functions to be used in higher-order cases where tensorial notation becomes more difficult to follow.

For a second-order neuron, define the matrix $\boldsymbol{D}^{(j)} \in \Re^{d\times d}$ to encapsulate the direction and size properties of a neuron, for each dimension of the input, where d is the number of dimensions. Each row i of $\boldsymbol{D}^{(j)}$ is a unit row vector $\boldsymbol{v}_i^T$ corresponding to a principal direction of the cluster, divided by the standard deviation of the cluster in that direction (the square root of the variance σ_i). The Euclidean metric can now be replaced by the neuron-dependent metric,

$$dist(\mathbf{x}, j) = \left\| \mathbf{D}^{(j)} \left(\mathbf{x} - \mathbf{w}^{(j)} \right) \right\|,$$

where

$$\mathbf{D}^{(j)} = \begin{bmatrix} \frac{1}{\sqrt{\sigma_1}} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sqrt{\sigma_2}} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{\sqrt{\sigma_d}} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_d^T \end{bmatrix}, \tag{2.1}$$

such that deviations are normalized with respect to the variance of the data in the direction of the deviation. The new matching criterion based on the new metric becomes

$$\begin{aligned} i(\mathbf{x}) &= \arg_j \min \left\| dist(\mathbf{x})^{(j)} \right\| \\ &= \arg_j \min \left\| \mathbf{D}^{(j)} \left(\mathbf{x} - \mathbf{w}^{(j)} \right) \right\|, \quad j = 1, 2, \ldots, N. \end{aligned} \tag{2.2}$$

The neuron is now represented by the pair $(\boldsymbol{D}^{(j)}, \boldsymbol{w}^{(j)})$, where $\boldsymbol{D}^{(j)}$ represents rotation and scaling and $\boldsymbol{w}^{(j)}$ represents translation. The values of $\boldsymbol{D}^{(j)}$ can be obtained from an eigenstructure analysis of the correlation matrix $\boldsymbol{R}^{(j)} = \Sigma \boldsymbol{x}^{(j)} \boldsymbol{x}^{(j)T} \in \Re^{d \times d}$ of the data associated with neuron j. Two well-known properties are derived from the eigenstructure of principal component analysis: (1) the eigenvectors of a correlation matrix $\boldsymbol{R}$ pertaining to the zero mean data vector $\boldsymbol{x}$ define the unit vectors $\boldsymbol{v}_i$ representing the principal directions along which the variances attain their extremal values, and (2) the associated eigenvalues define the extremal values of the variances σ_i. Hence,

$$\mathbf{D}^{(j)} = \lambda^{(j)^{-\frac{1}{2}}} \mathbf{V}^{(j)}, \tag{2.3}$$

where $\lambda^{(j)} = diag(\boldsymbol{R}^{(j)})$ produces the eigenvalues of $\boldsymbol{R}^{(j)}$ in a diagonal form and $\mathbf{V}^{(j)} = eig(\boldsymbol{R}^{(j)})$ produces the unit eigenvectors of $\boldsymbol{R}^{(j)}$ in matrix form. Also, by definition of eigenvalue analysis of a general matrix $\boldsymbol{A}$,

$$\mathbf{AV} = \lambda \mathbf{V}$$

and hence

$$\mathbf{A} = \mathbf{V}^T \lambda \mathbf{V}. \tag{2.4}$$

Now, since a term $\|\boldsymbol{a}\|$ equals $\boldsymbol{a}^T \boldsymbol{a}$, we can expand the term $\|\boldsymbol{D}^{(j)}(\boldsymbol{x} - \boldsymbol{w}^{(j)})\|$ of equation 2.2 as follows:

$$\left\| \mathbf{D}^{(j)} \left(\mathbf{x} - \mathbf{w}^{(j)} \right) \right\| = \left(\mathbf{x} - \mathbf{w}^{(j)} \right)^T \mathbf{D}^{(j)T} \mathbf{D}^{(j)} \left(\mathbf{x} - \mathbf{w}^{(j)} \right). \tag{2.5}$$

Substituting equations 1.2 and 2.3, we see that

$$\mathbf{D}^{(j)T} \mathbf{D}^{(j)} = \left[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d \right] \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \frac{1}{\sigma_d} \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_d^T \end{bmatrix} = \mathbf{R}^{(j)^{-1}}, \tag{2.6}$$

meaning that the scaling and rotation matrix is, in fact, the inverse of $\mathbf{R}^{(j)}$. Hence,

$$i(\mathbf{x}) = \arg_j \min \left[\left(\mathbf{x} - \mathbf{w}^{(j)} \right)^T \mathbf{R}^{-1} \left(\mathbf{x} - \mathbf{w}^{(j)} \right) \right], \quad j = 1, 2, \ldots, N, \quad (2.7)$$

which corresponds to the term used by Gustafson and Kessel (1979) and in the maximum likelihood gaussian classifier (Duda & Hart, 1973).

Equation 2.7 involves two explicit terms, $\boldsymbol{R}^{(j)}$ and $\boldsymbol{w}^{(j)}$, representing the orientation and size information, respectively. Implementations involving ellipsoidal metrics therefore need to track these two quantities separately, in a two-stage step. Separate tracking, however, cannot be easily extended to higher orders, which requires simultaneous tracking of additional higher-order qualities such as curvatures. We therefore use an equivalent representation that encapsulates these terms into one and permits a more systematic extension to higher orders.

We combine these two coefficients into one expanded covariance denoted $\boldsymbol{R}_H{}^{(j)}$ in homogenous coordinates (Faux & Pratt, 1981), as

$$\mathbf{R}_H{}^{(j)} = \sum \mathbf{x}_H^{(j)} \mathbf{x}_H^{(j)T}, \quad (2.8)$$

where in homogenous coordinates $\boldsymbol{x}_H^{(j)} \in \Re^{(d+1)\times 1}$ and is defined by $\mathbf{x}_H = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$.

In this notation, the expanded covariance also contains coordinate permutations involving 1 as one of the multiplicands, and so different (lower) orders are introduced. Consequently, $\boldsymbol{R}_H{}^{(j)} \in \Re^{(d+1)\times(d+1)}$. In analogy to the correlation matrix memory and autoassociative memory (Haykin, 1994), the extended matrix $\boldsymbol{R}_H{}^{(j)}$, in its general form, can be viewed as a homogeneous autoassociative tensor. Now the new matching criterion becomes simply

$$\begin{aligned} i(\mathbf{x}) &= \arg_j \min \left[\mathbf{X}^T \mathbf{R}_H^{(j)-1} \mathbf{X} \right] \\ &= \arg_j \min \left\| \mathbf{R}_H^{(j)-\frac{1}{2}} \mathbf{X} \right\|, \\ &= \arg_j \min \left\| \mathbf{R}_H^{(j)-1} \mathbf{X} \right\|, \quad j = 1, 2, \ldots, N. \end{aligned} \quad (2.9)$$

This representation retains the notion of maximum correlation, and for convenience, we now denote $\boldsymbol{R}_H{}^{-1(j)}$ as the synaptic tensor.

When we moved into homogenous coordinates, we gained the following:

- The eigenvectors of the original covariance matrix $\boldsymbol{R}^{(j)}$ represented directions. The eigenvectors of the homogeneous covariance matrix $\boldsymbol{R}_H{}^{(j)}$ correspond to both the principal directions and the offset (both

second-order and first-order) properties of the cluster accumulated in $\boldsymbol{R}_H{}^{(j)}$. In fact, the elements of each eigenvector correspond to the coefficient of the line equation $ax + by + \cdots + c = 0$. This property permits direct extension to higher-order tensors in the next section, where direct eigenstructure analysis is not well defined.

- The transition to homogeneous coordinates dispensed with the problem created by the fact that the eigenstructure properties cited above pertained only to zero-mean data, whereas $\boldsymbol{R}_H{}^{(j)}$ is defined as the covariance matrix of *non*-zero-mean data. By using homogeneous coordinates, we effectively consider the translational element of the cluster as yet another (additional) dimensions. In this higher space, the data can be considered to be zero mean.

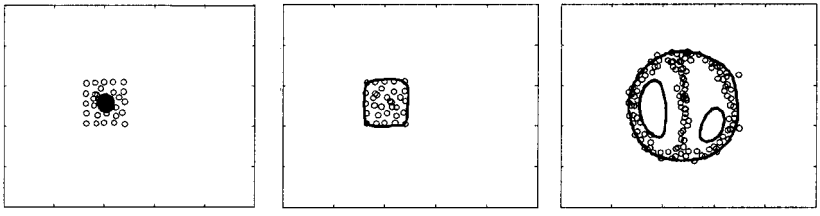
2.1 Higher Orders. The ellipsoidal neuron can capture only linear directionality, not curved directionalities such as that appearing, for instance, in a fork, in a curved or in a sharp-cornered shape. In order to capture more complex spatial configurations, higher-order geometries must be introduced.

The classic neuron possesses a synaptic weight vector $\boldsymbol{w}^{(j)}$, which corresponds to a point in the input domain. The synaptic weight $\boldsymbol{w}^{(j)}$ can be seen as the first-order average of its signals, that is, $\boldsymbol{w}^{(j)} = \sum \boldsymbol{x}_H$ (the last element $\boldsymbol{x}_{Hd+1} = 1$ of the homogeneous coordinates has no effect in this case). Ellipsoidal units hold information regarding the linear correlations among the coordinates of data points represented by the neuron by using $\boldsymbol{R}_H{}^{(j)} = \sum \boldsymbol{x}_H \boldsymbol{x}_H{}^T$. Each element of $\boldsymbol{R}_H$ is thus a proportionality constant relating two specific dimensions of the cluster. The second-order neuron can therefore be regarded as a second-order approximation of the corresponding data distribution. Higher-order relationships can be introduced by considering correlations among more than just two variables. We may consequently introduce a neuron capable of modeling a d-dimensional data cluster to an mth-order approximation.

Higher-order correlations are obtained by multiplying the generating vector $\boldsymbol{x}_H$ by itself, in successive outer products, more than just once. The process yields a covariance tensor rather than just a covariance matrix (which is a second-order tensor). The resulting homogeneous covariance tensor is thus represented by a $2(m-1)$-order tensor of rank $d+1$, denoted by $\boldsymbol{Z}_H^{(j)} \in \Re^{(d+1)\times\cdots\times(d+1)}$, obtained by $2(m-1)$ outer products of $\boldsymbol{x}_H$ by itself:

$$\boldsymbol{Z}_H^{(j)} = \mathbf{x}_H^{2(m-1)}. \tag{2.10}$$

The exponent consists of the factor $(m-1)$, which is the degree of the approximation, and a factor of 2 since we are performing autocorrelation, so the function is multiplied by itself. In analogy to reasoning that leads to equation 2.9, each eigenvector (now an eigentensor of order $m-1$) corre-



(a) (b) (c)

Figure 4: Metrics of various orders: (a) a regular neuron (spherical metric), (b) a "square" neuron, and (c) a neuron with "two holes."

sponds to the coefficients of a principal curve, and multiplying it by input points produces an approximation of the distance of that input point from the curve. Consequently, the inverse of the tensor $\mathbf{Z}_H^{(j)}$ can then be used to compute the high-order correlation of the signal with the nonlinear shape neuron, by simple tensor multiplication:

$$i(\mathbf{x}) = \arg_i \min \left\| \mathbf{Z}_H^{(j)^{-1}} \otimes x_H^{m-1} \right\|, \quad j-1, 2, \ldots, N, \tag{2.11}$$

where $\otimes$ denotes tensor multiplication. Note that the covariance tensor can be inverted only if its order is an even number, as satisfied by equation 2.10. Note also that the amplitude operator is carried out by computing the root of the sum of the squares of the elements of the argument. The computed metric is now not necessarily spherical and may take various other forms, as plotted in Figure 4.

In practice, however, high-order tensor inversion is not directly required. To make this analysis simpler, we use a Kronecker notation for tensor products (see Graham, 1981). Kronecker tensor product flattens out the elements of $X \otimes Y$ into a large matrix formed by taking all possible products between the elements of X and those of Y. For example, if X is a 2×3 matrix, then $X \otimes Y$ is

$$X \otimes Y = \begin{bmatrix} Y \cdot X_{1,1} & Y \cdot X_{1,2} & Y \cdot X_{1,3} \\ \hline Y \cdot X_{2,1} & Y \cdot X_{2,2} & Y \cdot X_{2,3} \end{bmatrix}, \tag{2.12}$$

where each block is a matrix of the size of Y. In this notation, the internal structure of higher-order tensors is easier to perceive, and their correspondence to linear regression of principal polynomial curves is revealed. Consider, for example, a fourth-order covariance tensor of the vector $\boldsymbol{x} = \{x, y\}$.

The fourth-order tensor corresponds to the simplest nonlinear neuron according to equation 2.10, and takes the form of the 2×2×2×2 tensor:

$$\langle \mathbf{x} \otimes \mathbf{x} \otimes \mathbf{x} \otimes \mathbf{x} \rangle = \langle \mathbf{R} \otimes \mathbf{R} \rangle = \begin{bmatrix} x^2 & xy \\ xy & y^2 \end{bmatrix} \otimes \begin{bmatrix} x^2 & xy \\ xy & y^2 \end{bmatrix}$$

$$= \left[\begin{array}{cc:cc} x^4 & x^3y & x^3y & x^2y^2 \\ x^3y & x^2y^2 & x^2y^2 & xy^3 \\ \hdashline x^3y & x^2y^2 & x^2y^2 & xy^3 \\ x^2y^2 & xy^3 & xy^3 & y^4 \end{array}\right]. \tag{2.13}$$

The homogeneous version of this tensor also includes all lower-order permutations of the coordinates of $\boldsymbol{x}_H = \{x, y, 1\}$, namely, the 3×3×3×3 tensor:

$$\mathbf{Z}_{H(4)} = \langle \mathbf{x}_H, \mathbf{x}_H, \mathbf{x}_H, \mathbf{x}_H \rangle = \langle \mathbf{R}_H, \mathbf{R}_H \rangle = \begin{bmatrix} x^2 & xy & x \\ xy & y^2 & y \\ x & y & 1 \end{bmatrix} \otimes \begin{bmatrix} x^2 & xy & x \\ xy & y^2 & y \\ x & y & 1 \end{bmatrix}$$

$$= \left[\begin{array}{ccc:ccc:ccc} x^4 & x^3y & x^3 & x^3y & x^2y^2 & x^2y & x^3 & x^2y & x^2 \\ x^3y & x^2y^2 & x^2y & x^2y^2 & xy^3 & xy^2 & x^2y & xy^2 & xy \\ x^3 & x^2y & x^2 & x^2y & xy^2 & xy & x^2 & xy & x \\ \hdashline x^3y & x^2y^2 & x^2y & x^2y^2 & xy^3 & xy^2 & x^2y & xy^2 & xy \\ x^2y^2 & xy^3 & xy^2 & xy^3 & y^4 & y^3 & xy^2 & y^3 & y^2 \\ x^2y & xy^2 & xy & xy^2 & y^3 & y^2 & xy & y^2 & y \\ \hdashline x^3 & x^2y & x^2 & x^2y & xy^2 & xy & x^2 & xy & x \\ x^2y & xy^2 & xy & xy^2 & y^3 & y^2 & xy & y^2 & y \\ x^2 & xy & x & xy & y^2 & y & x & y & 1 \end{array}\right]. \tag{2.14}$$

Remark. It is immediately apparent that the matrix in equation 2.14 corresponds to the matrix to be solved for finding a least-squares fit of a conic section equation $ax^2 + by^2 + cxy + dx + ey + f = 0$ to the data points. Moreover, the set of eigenvectors of this matrix corresponds to the coefficients of the set of mutually orthogonal best-fit conic section curves that are the principal curves of the data. This notion adheres with Gnanadesikan's (1977) method for finding principal curves. Substitution of a data point into the equation of a principal curve yields an approximation of the distance of the point from that curve, and the sum of squared distances amounts to equation 2.11. Note that each time we increase order, we are seeking a set of principal curves of one degree higher. This implies that the least-squares matrix needs to be two degrees higher (because it is minimizing the squared error), thus yielding the coefficient 2 in the exponent of equation 210.

3 Learning and Functionality

In order to show that higher-order shapes are a direct extension of classic neurons, we show that they are also subject to simple Hebbian learning. Following an interpretation of Hebb's postulate of learning, synaptic modification (i.e., learning) occurs when there is a correlation between presynaptic and postsynaptic activities. We have already shown in equation 2.9 that the presynaptic activity $\boldsymbol{x}_H$ and postsynaptic activity of neuron j coincide when the synapse is strong; that is, ${\boldsymbol{R}_H}^{(j)^{-1}}\boldsymbol{x}_H$ is minimum. We now proceed to show that in accordance with Hebb's postulate of learning, it is sufficient to incur self-organization of the neurons by increasing synapse strength when there is a coincidence of presynaptic and postsynaptic signals.

We need to show how self-organization is obtained merely by increasing ${\boldsymbol{R}_H}^{(j)}$, where j is the winning neuron. As each new data point arrives at a specific neuron j in the net, the synaptic weight of that neuron is adapted by the incremental corresponding to Hebbian learning,

$$\mathbf{Z}_H{}^{(j)}(k+1) = \mathbf{Z}_H{}^{(j)}(k) + \eta(k)\mathbf{x}_H^{(j)^{2(m-1)}}, \tag{3.1}$$

where k is the iteration counter and $\eta(k)$ is the iteration-dependent learning rate coefficient. It should be noted that in equation 2.10, $\boldsymbol{Z}_H^{(j)}$ becomes a weighted sum of the input signals $x_H^{2(m-1)}$ (unlike $\boldsymbol{R}_H$ in equation 2.8, which is a uniform sum). The eigenstructure analysis of a weighted sum still provides the principal components under the assumption that the weights are uniformly distributed over the cluster signals. This assumption holds true if the process generating the signals of the cluster is stable over time, a basic assumption of all neural network algorithms (Bishop, 1997). In practice, this assumption is easily acceptable, as will be demonstrated in the following sections.

In principle, continuous updating of the covariance tensor may create instability in a competitive environment, as a winner neuron becomes increasingly dominant. To force competition, the covariance tensor can be normalized using any of a number of factors dependent on the application, such as the number of signals assigned to the neuron so far or the distribution of data among the neuron (forcing uniformity). However, a more natural factor for normalization is the size and complexity of the neuron.

A second-order neuron is geometrically equivalent to a d-dimensional ellipsoid. A neuron may be therefore normalized by a factor V proportional to the ellipsoid volume:

$$V \propto \sqrt{\sigma_1 \cdot \sigma_2 \cdot \cdots \cdot \sigma_d} = \prod_{i=1}^{d} \sqrt{\sigma_i} \propto \frac{1}{\sqrt{\det\left|\mathbf{R}^{(j)}\right|}}. \tag{3.2}$$

During the competition phase, the factors $\boldsymbol{R}^{(j)}\boldsymbol{x}/V$ are compared, rather than

just $\boldsymbol{R}^{(j)}\boldsymbol{x}$, thus promoting smaller neurons and forcing neurons to become equally "fat." The final matching criteria for a homogeneous representation are therefore given by

$$j(\mathbf{x}) = \arg_j \min \left\| \hat{\mathbf{R}}_H^{-1(j)} \mathbf{x}_H \right\|, \quad j = 1, 2, \ldots, N, \tag{3.3}$$

where

$$\hat{\mathbf{R}}_H^{(j)} = \frac{\mathbf{R}_H^{(j)}}{\sqrt{\det \left| \mathbf{R}_\mathbf{H}^{(j)} \right|}}. \tag{3.4}$$

Covariance tensors of higher-order neurons can also be normalized by their determinant. However, unlike second-order neurons, higher-order shapes are not necessarily convex, and hence their volume is not necessarily proportional to their determinant. The determinant of higher-order tensors therefore relates to more complex characteristics, which include the deviation of the data points from the principal curves and the curvature of the boundary. Nevertheless, our experiments showed that this is a simple and effective means of normalization that promotes small and simple shapes over complex or large, concave fittings to data. The relationship between geometric shape and the tensor determinant should be investigated further for more elaborate use of the determinant as a normalization criterion.

Finally, in order to determine the likelihood of a point's being associated with a particular neuron, we need to assume a distribution function. Assuming gaussian distribution, the likelihood $p^{(j)}(\mathbf{x}_H)$ of data point $\mathbf{x}_H$ being associated with cluster j is proportional to the distance from the neuron and is given by

$$p^{(j)}(\mathbf{x}_H) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{1}{2}\left\| \hat{\mathbf{Z}}^{(j)} \mathbf{x}_H \right\|^2}. \tag{3.5}$$

4 Algorithm Summary

To conclude the previous sections, we provide a summary of the high-order competitive learning algorithm for both unsupervised and supervised learning. These are the algorithms that were used in the test cases described later in section 5.

Unsupervised Competitive learning.

1. Select layer size (number of neurons N) and neuron order (m) for a given problem.

2. Initialize all neurons with $\mathbf{Z}_H = \sum \mathbf{x}_H^{2(m-1)}$ where the sum is taken over all input data, or a representative portion, or unit tensor or random numbers if no data are available. To compute this value, write down $\mathbf{x}_H^{m-1}$ as a vector with all mth degree permutations of $\{x_1, x_2, \ldots, x_d, 1\}$, and $\mathbf{Z}_H$ as a matrix summing the outer product of these vectors. Store $\mathbf{Z}_H$ and $\mathbf{Z}_H^{-1}/f$, for each neuron, where f is a normalization factor (e.g., equation 3.4).
3. Compute the winner for input x. First compute vector $\mathbf{x}_H^{m-1}$ from $\mathbf{x}$ as in section 2, and then multiply it by the stored matrix $\mathbf{Z}_H^{-1}/f$. Winner j is the one for which the modulus of the product vector is smallest.
4. Output winner j.
5. Update the winner by adding $\mathbf{x}_H^{2(m-1)}$ to $\mathbf{Z}_H{}^{(j)}$ weighted by the learning coefficient $\eta(k)$ where k is the iteration counter. Store $\mathbf{Z}_H$ and $\mathbf{Z}_H{}^{-1}/f$, for the updated neuron.
6. Go to step 3.

Supervised Learning.

1. Set layer size (number of neurons) to number of classes, and select neuron order (m) for a given problem.
2. Train. For each neuron, compute $\mathbf{Z}_H^{(j)} = \sum \boldsymbol{x}_H^{2(m-1)}$ where the sum is taken over all training points to be associated with that neuron. To compute this value, write down $x_H{}^{m-1}$ as a vector with all mth degree permutations of $\{x_1, x_2, \ldots, x_d, 1\}$, and $\mathbf{Z}_H$ as a matrix summing the outer product of these vectors. Store $\mathbf{Z}_H{}^{-1}/f$ for each neuron.
3. Test. Use test points to evaluate the quality of training.
4. Use. For each new input x, first compute vector x_H^{m-1} from x, as in section 2, and then multiply it by the stored matrix $\mathbf{Z}_H^{-1}/f$. Winner j is the one for which modulus of the product vector is smallest.

5 Implementation with Synthesized and Real Data

The proposed neuron can be used as an element in any network by replacing lower-dimensionality neurons and will enhance the modeling capability of each neuron. Depending on the application of the net, this may lead to improvement or degradation of the overall performance of the layer, due to the added degrees of freedom. This section provides some examples of an implementation at different orders, in both supervised and unsupervised learning.

First, we discuss two examples of second-order (ellipsoidal) neurons to show that our formulation is compatible with previous work on ellipsoidal neural units (e.g., Mao & Jain, 1996). Figure 5 illustrates how

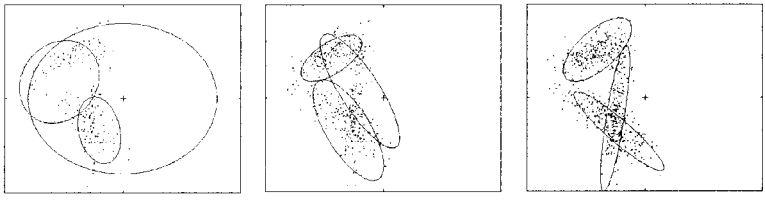


Figure 5: (a–c) Stages in self-classification of overlapping clusters, after learning 200, 400, and 650 data points, respectively (one epoch).

a competitive network consisting of three second-order neurons gradually evolves to model a two-dimensional domain exhibiting three main activation areas with significant overlap. The second-order neurons are visualized as ellipses with their boundaries drawn at 2σ of the distribution.

The next test uses a two-dimensional distribution of three arbitrary clusters, consisting of a total of 75 data points, shown in Figure 6a. The network of three second-order neurons self-organized to map this data set and creates the decision boundaries shown in Figure 6b. The dark curves are the decision boundaries, and the light lines show the local distance metric within each cell. Note how the decision boundary in the overlap area accounts for the different distribution variances.

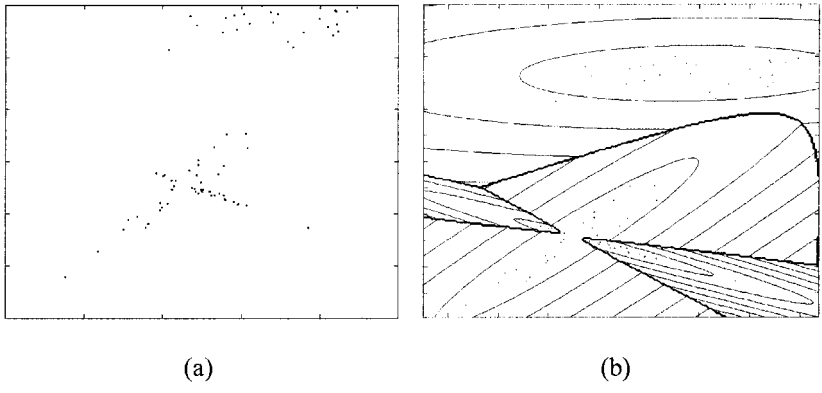


Figure 6: (a) Original data set. (b) Self-organized map of the data set. Dark curves are the decision boundaries, and light lines show the local distance metric within each cell. Note how the decision boundary in the overlap area accounts for the different distribution variances.

Table 1: Comparison of Self-Organization Results for IRIS Data.

Method	Epochs	Number misclassified or Unclassified
Super paramagnetic (Blatt et al., 1996)		25
Learning vector quantization / Generalized learning vector quantization (Pal, Bezdek, & Tsao, 1993)	200	17
K-means (Mao & Jain, 1996)		16
Hyperellipsoidal clustering (Mao & Jain, 1996)		5
Second-order unsupervised	20	**4**
Third-order unsupervised	30	**3**

The performance of the different orders of shape neuron has also been assessed on the Iris data (Anderson, 1939). Anderson's time-honored data set has become a popular benchmark problem for clustering algorithms. The data set consists of four quantities measured on each of 150 flowers chosen from three species of iris: *Iris setosa*, *Iris versicolor*, and *Iris virginica*. The data set constitutes 150 points in four-dimensional space. Some results of other methods are compared in Table 1. It should be noted that the results presented in the table are "best results;" third-order neurons are not always stable because the layer sometimes converges into different classifications.[1]

The neurons have also been tried in a supervised setup. In supervised mode, we use the same training setup of equations 2.8 and 2.10 but train each neuron individually on the signals associated with it, using

$$\mathbf{Z}_H^{(j)} = \sum \mathbf{x}_H^{2(m-1)}, \quad \boldsymbol{x}_H \in \boldsymbol{\Psi}^{(j)},$$

where in homogeneous coordinates $\mathbf{x}_H = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}$ and $\boldsymbol{\Psi}^{(j)}$ is the jth class.

In order to evaluate the performance of this technique, we trained each neuron on a random selection of 80% of the data points of its class and then tested the classification of the whole data set (including the remaining 20% for cross validation). The test determined the most active neuron

[1] Performance of third-order neurons was evaluated with a layer consisting of three neurons randomly initialized with normal distribution within the input domain with mean and variance of the input, and with learning factor of 0.3 decreasing asymptotically toward zero. Of 100 experiments carried out, 95% locked correctly on the clusters, with an average 7.6 (5%) misclassifications after 30 epochs. Statistics on performance of cited works have not been reported. Download MATLAB demos from http://www.cs.brandeis.edu/ lipson/papers/geom.htm.

Table 2: Supervised Classification Results for Iris Data.

Order	Epochs	Number Misclassified Average	Best
3-hidden-layer neural network (Abe et al., 1997)	1000	2.2	1
Fuzzy hyperbox (Abe et al., 1997)			2
Fuzzy ellipsoids (Abe et al., 1997)	1000		1
Second-order supervised	1	3.08	1
Third-order supervised	1	2.27	1
Fourth-order supervised	1	1.60	**0**
Fifth-order supervised	1	**1.07**	**0**
Sixth-order supervised	1	1.20	**0**
Seventh-order supervised	1	1.30	**0**

Notes: Results after one training epoch, with 20% cross validation. Averaged results are for 250 experiments.

for each input and compared it with the manual classification. This test was repeated 250 times, with the average and best results shown in Table 2, along with results reported for other methods. It appears that on average, supervised learning for this task reaches an optimum with fifth-order neurons.

Figure 7 shows classification results using third-order neurons for an arbitrary distribution containing close and overlapping clusters.

Finally, Figure 8 shows an application of third- through fifth-order neurons to detection and modeling of chromosomes in a human cell. The corresponding separation maps and convergence error rates are shown in Figures 9 and 10, respectively.

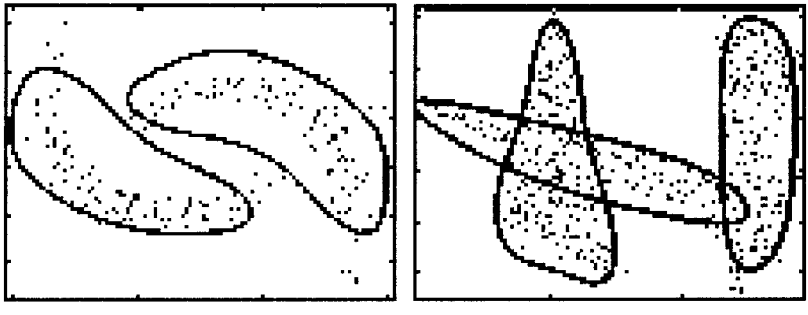


Figure 7: Self-classification of synthetic point clusters using third-order neurons.

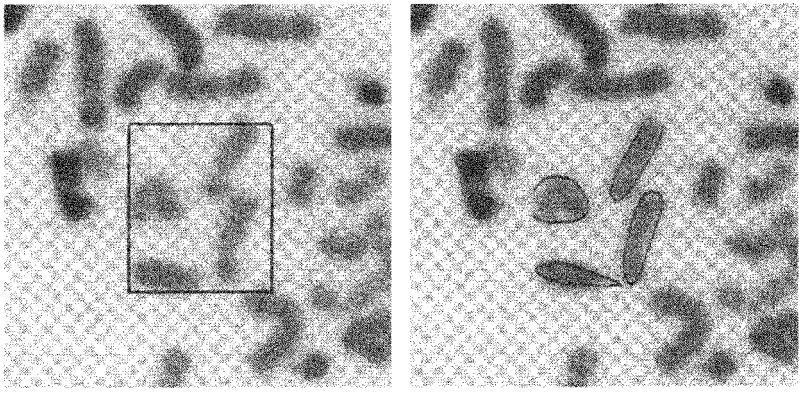


Figure 8: Chromosome separation. (a) Designated area. (b) Four third-order neuron self-organized to mark the chromosomes in designated area. Data from Schrock et al. (1996).

6 Conclusions and Further Research

We have introduced high-order shape neurons and demonstrated their practical use for modeling the structure of spatial distributions and for geometric feature recognition. Although high-order neurons do not directly correspond to neurobiological details, we believe that they can provide powerful data modeling capabilities. In particular, they exhibit useful properties for correctly handling close and partially overlapping clusters. Furthermore, we have shown that ellipsoidal and tensorial clustering methods, as well as classic competitive neurons, are special cases of the general-shape neuron. We showed how lower-order information can be encapsulated using tensor representation in homogeneous coordinates, enabling systematic continuation to higher-order metrics. This formulation also allows for direct extraction of the encapsulated high-order information in the form of principal curves, represented by the eigentensors of each neuron.

The use of shapes as neurons raises some further practical questions, which have not been addressed in this article and require further research, and are listed below. Although most of these issues pertain to neural networks in general, the approach required here may be different:

- **Number of neurons.** In the examples provided, we have explicitly used the number of neurons appropriate to the number of clusters. The question of network size is applicable to most neural architecture. However, our experiments showed that overspecification tends

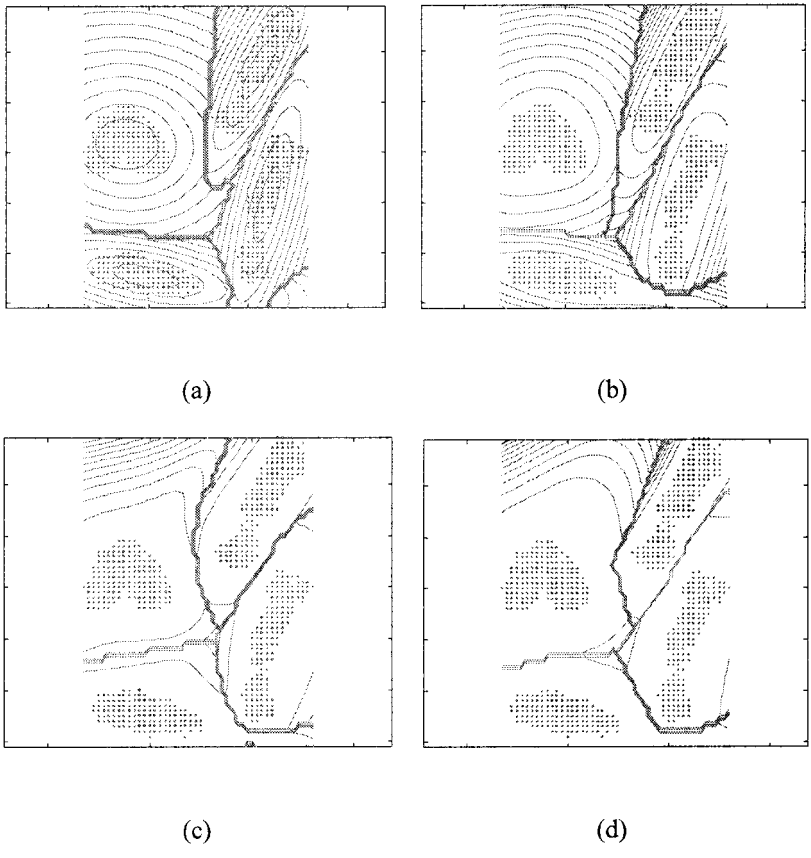


Figure 9: Self-organization for chromosome separation. (a–d) Second- through fifth-order, respectively.

to cause clusters to split into subsections, while underspecification may cause clusters to unite.

- **Initial weights.** It is well known that sensitivity to initial weights is a general problem of neural networks (see, for example, Kolen & Pollack, 1990, and Pal et al., 1993). However, when third-order neurons were evaluated on the Iris data with random initial weights, they performed with average 5% misclassifications. Thus, the capacity of neurons to be spread over volumes may provide new possibilities for addressing the initialization problem.
- **Computational cost.** The need to invert a high-order tensor introduces a significant computational cost, especially when the input dimension-

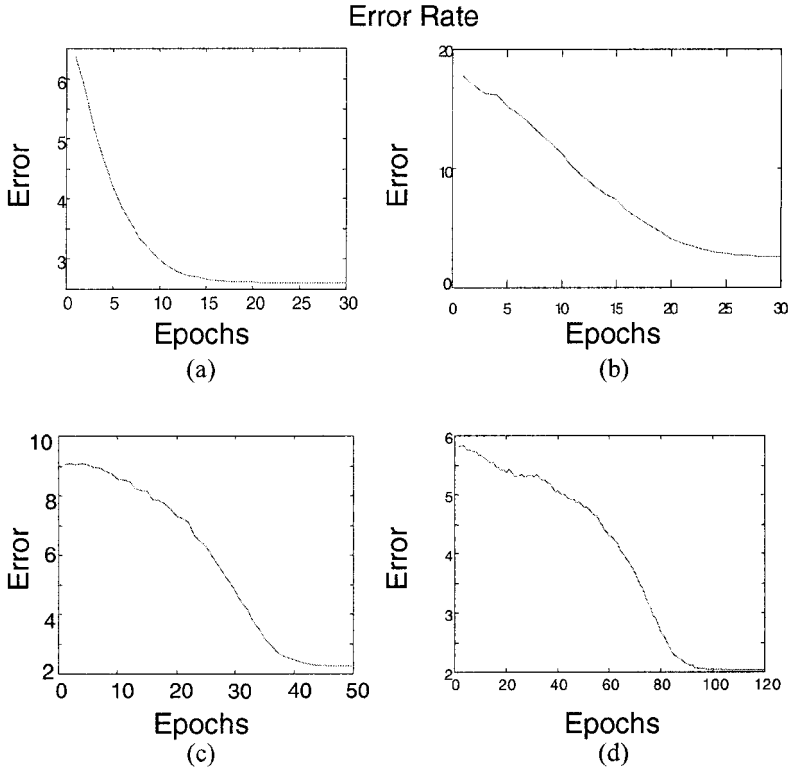


Figure 10: Average error rate in self-organization for chromosome separation of Figure 9, respectively.

ality is large. There are some factors that counterbalance this computational cost. First, the covariance tensor needs to be inverted only when a neuron is updated, not when it is activated or evaluated. When the neuron is updated, the inverted tensor is computed and then stored and used whenever the neuron needs to compete or evaluate an input signal. This means that the whole layer will need only one inversion for each training point and no inversions for nontraining operation. Second, because of the complexity of each neuron, sometimes fewer of them are required. Finally, relatively complex shapes can be attained with low orders (say, third order).

- **Selection of neuron order.** As in many other networks, there is much domain knowledge about the problem that comes into the solution through choice of parameters such as the order of the neurons. However, we also estimate that due to the rather analytic representation

of each neuron, and since high-order information is contained with each neuron independently, this information can be directly extracted (using the eigentensors). Thus, it is possible to decompose analytically a cross-shapes fourth-order cluster into two overlapping elliptic clusters, and vice versa.

- **Stability versus flexibility and enforcement of particular shapes.** Further research is required to find methods for biasing neurons toward particular shapes in an attempt to seek particular shapes and help neuron stability. For example, how would one encourage detection of triangles and not rectangles?

- **Nonpolynomial base functions.** We intend to investigate the properties of shape layers based on nonpolynomial base functions. Noting that the high-order homogeneous tensors give rise to various degrees of polynomials, it is possible to derive such tensors with other base functions such as wavelets, trigonometric functions, and other well-established kernel functions. Such an approach may enable modeling arbitrary geometrical shapes.

Acknowledgments

We thank Eitan Domani for his helpful comments and insight. This work was supported in part by the U.S.-Israel Binational Science Foundation (BSF), the Israeli Ministry of Arts and Sciences, and the Fund for Promotion of Research at the Technion. H. L. acknowledges the generous support of the Charles Clore Foundation and the Fischbach Postdoctoral Fellowship. We thank Applied Spectral Imaging for supplying the data for the experiments described in Figure 8 through 10.

Download further information, MATLAB demos, and implementation details of this work from http://www.cs.brandeis.edu/~lipson/papers/geom.htm.

References

Abe, S., & Thawonmas, R. (1997). A fuzzy classifier with ellipsoidal regions. *IEEE Trans. on Fuzzy Systems, 5*, 358–368.

Anderson, E. (1939). The Irises of the Gaspe Peninsula. *Bulletin of the American Iris Society, 59*, 2–5.

Balestrino, A., & Verona, B. F. (1994). New adaptive polynomial neural network. *Mathematics and Computers in Simulation, 37*, 189–194.

Bishop, C. M. (1997). Neural networks for pattern recognition. Oxford: Clarendon Press.

Blatt, M., Wiseman, S., & Domany, E. (1996). Superparamagnetic clustering of data. *Physical Review Letters, 76/18*, 3251–3254.

Davé, R. N. (1989). Use of the adaptive fuzzy clustering algorithm to detect lines in digital images. In *Proc. SPIE, Conf. Intell. Robots and Computer Vision, 1192*, 600–611.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Faux, I. D., & Pratt, M. J., (1981). *Computational geometry for design and manufacture*. New York: Wiley.

Frigui, H. & Krishnapuram, R. (1996). A comparison of fuzzy shell-clustering methods for the detection of ellipses. *IEEE Transactions on Fuzzy Systems, 4*, 193–199.

Giles, C. L., Griffin, R. D., & Maxwell, T. (1988). Encoding geometric invariances in higher-order neural networks. In D. Z. Anderson (Ed.), *Neural information processing systems*. American Institute of Physics Conference Proceedings.

Gnanadesikan, R. (1977). *Methods for statistical data analysis of multivariate observations*. New York: Wiley.

Goudreau, M. W., Giles C. L., Chakradhar, S. T., & Chen, D. (1994). First-order versus second-order single layer recurrent neural networks. *IEEE Transactions on Neural Networks, 5*, 511–513.

Graham, A. (1981). *Kronecker products and matrix calculus: With applications*. New York: Wiley.

Gustafson, E. E., & Kessel, W. C. (1979). Fuzzy clustering with fuzzy covariance matrix. In *Proc. IEEE CDC* (pp. 761–766). San Diego, CA.

Haykin, S. (1994). *Neural networks: A comprehensive foundation*. Englewood Cliffs, NH: Prentice Hall.

Kavuri, S. N, & Venkatasubramanian, V. (1993). Using fuzzy clustering with ellipsoidal units in neural networks for robust fault classification. *Computers Chem. Eng., 17*, 765–784.

Kohonen, T., (1997). *Self organizing maps*. Berlin: Springer-Verlag.

Kolen, J. F., & Pollack, J. B. (1990). Back propagation is sensitive to initial conditions. *Complex Systems, 4*, 269–280.

Krishnapuram, R., Frigui, H., & Nasraoui, O. (1995). Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation—Parts I and II. *IEEE Transactions on Fuzzy Systems, 3*, 29–60.

Lipson, H., Hod, Y., & Siegelmann, H. T. (1998). High-order clustering metrics for competitive learning neural networks. In *Proceedings of the Israel-Korea Bi- National Conference on New Themes in Computer Aided Geometric Modeling*. Tel-Aviv, Israel.

Mao, J., & Jain, A. (1996). A self-organizing network for hyperellipsoidal clustering (HEC). *IEEE Transactions on Neural Networks, 7*, 16–29.

Myung, I., Levy, J., & William, B., (1992). Are higher order statistics better for maximum entropy prediction in neural networks? In *Proceedings of the Int. Joint Conference on Neural Networks IJCNN'91*. Seattle, WA.

Pal, N., Bezdek, J. C., & Tsao, E. C.-K. (1993). Generalized clustering networks and Kohonen's self-organizing scheme. *IEEE Transactions on Neural Networks, 4*, 549–557.

Pollack J. B. (1991). The induction of dynamical recognizers. *Machine Learning, 7*, 227–252.

Schrock, E., duManoir, S., Veldman, T., Schoell, B., Wienberg, J., Feguson Smith, M. A., Ning, Y., Ledbetter, D. H., BarAm, I., Soenksen, D., Garini, Y., & Ried, T. (1996). Multicolor spectral karyotyping of human chromosomes. *Science, 273*, 494–497.

Received August 24, 1998; accepted September 3, 1999.