Multi-order Neurons for evolutionary higher order clustering and growth

Kiruthika Ramanathan and Sheng Uei Guan*

Abstract

This article proposes to use Multi-Order Neurons for clustering irregularly shaped data arrangements. Multi-order neurons are an evolutionary extension of the use of higher order neurons in clustering. Higher order neurons parametrically model complex neuron shapes by replacing the classic synaptic weight by higher order tensors. The multi-order neuron goes one step further and eliminates two problems associated with higher order neurons. Firstly, it uses evolutionary algorithms to select the best neuron order for a given problem. Secondly, it obtains more information about the underlying data distribution by identifying the correct order for a given cluster of patterns. Empirically, we observed that, when the correlation of clusters-found with ground-truth-information is used in measuring clustering accuracy, the proposed evolutionary multi-order neurons method can be shown to outperform other related clustering methods. The simulation results from the IRIS, WINE and GLASS datasets show significant improvement when compared to the results obtained using SOMs and higher order neurons. The paper also proposes an intuitive model by which multi order neurons can be grown, thereby determining the number of clusters in a given data.

1. Introduction

Self organizing maps (SOMs) (Kohonen, 1997) represent a type of *neural network* proposed for clustering. SOMs work by assigning a synaptic weight (denoted by the column vector $w^{(j)}$) to each *neuron*. The winning neuron j(x) is the neuron that has the highest correlation with the input x, i.e., it is the neuron for which $w^{(j)T}x$ is the largest:

$$j(x) = \arg_j \max \left\| w^{(j)T} x \right\| \qquad (1)$$

where the operator $\|.\|$ represents the Euclidean norm of the vector. The idea behind equation (1) is to select the neuron which exhibits maximum correlation with the input. Often used instead of equation (1) is the minimum Euclidean distance matching criterion (Kohonen, 1997)

$$j(x) = \arg_j \min \left\| w^{(j)} - x \right\| \qquad (2)$$

However, the use of the highest correlation or the *minimum distance matching criterion* implies that (a) the features of the input domain are spherical, i.e., deviations are equal in all dimensions and (b) the distance between features must be larger than the distance between points in a feature. These two implications of the data can be summarized in the equations below

$$1 \le m, n \le d, m \ne n, \lambda_m(\Sigma_{\mathrm{I}}) \approx \lambda_n(\Sigma_{\mathrm{I}}) \qquad (3)$$

$$\forall x, y \in \mathbf{I}, \left\| x^T y \right\| \textbf{if } \left(j(x) = j(y) \right) > \left\| x^T y \right\| \textbf{if} \left(j(x) \ne j(y) \right) \qquad (4)$$

In the above equations, the $\Sigma_{\mathbf{I}}$ operator represents the covariance of the data matrix **I**, λ_m is the m^{th} eigenvalue, d is the dimension of the input domain, x and y are arbitrary data vectors.

For an arbitrary set of data to fulfill these conditions it could be difficult, especially in cases where such distribution of data are difficult to visualize and detect due to the high dimensionality of the problem. Even when the distribution of the data is detected through visualization, when the conditions described in equations 3 and 4 are not satisfied, there is no guarantee that the use of an SOM will solve the problem.

Consider, for example, the arrangements of data clusters in two dimensions as shown in Figure 1. Table 1 summarizes the properties of these datasets in terms of their ability to satisfy equations 3 and 4. Due to the nature of the data there is no guarantee that the SOM will be able to cluster correctly the sets of data other than Data A.

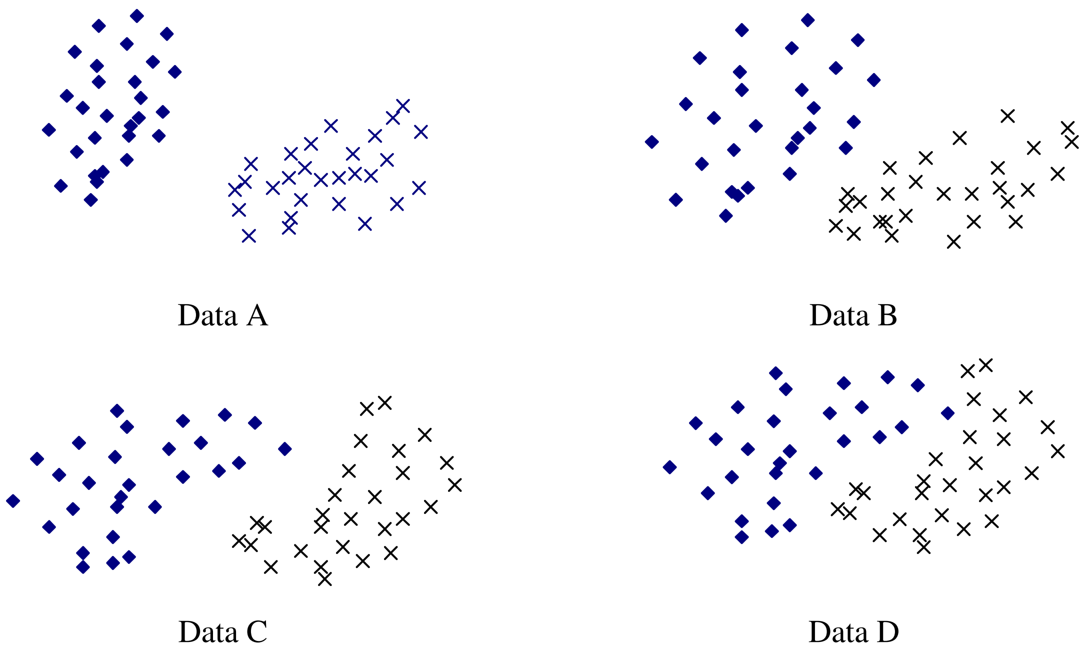


Figure 1. Artificially generated two-dimensional two class clusters illustrating the weakness of SOMs

Table 1. Summary of the properties of the data in Figure 1

Data Name	Spherical clusters (Satisfies equation 3)?	Clusters are sufficiently far apart (Satisfies equation 4)?
Data A	Yes	Yes
Data B	Yes	No
Data C	No	Yes
Data D	No	No

Several methods have been proposed to solve these problems. Second order metrics are a popular choice as the use of an inverse covariance matrix in the clustering metric captures linear directionality properties of a cluster (Lipson et al., 1998). The concept of second order curves was expanded in several cases to include second order shells (Krishnapuram et al., 1995). Kohonen discussed the use of weighted Euclidean distance (Kohonen, 1997) that captures different variances in the components of input signals. The use of the Mahlanobhis distance (Mao and Jain, 1996) was also considered. On the other hand, non-parametric techniques such as agglomeration (Blatt et al., 1996) attempt to find arbitrary shapes in the clusters. Improvements have been observed empirically in the application of these second-order and arbitrarily shaped clusters. However, their performance is dependent on several factors, including their ability to satisfy Equation 4, the distribution of data and the shape of the underlying data.

Lipson and Siegelmann (2000) proposed the generalized *higher order neuron* (HON) structure, a parametric algorithm which extended second order surfaces to arbitrary order hyper surfaces, thereby giving rise to a continuum of possible cluster shapes between the classic spherical – ellipsoidal unit and the fully non parametric agglomerative algorithm. To summarize, HON relaxes the shape restriction of classic neurons by replacing the weight vector or covariance matrix of a classic neuron with a general higher order tensor, which can capture multilinear correlations among the signals associated with the neurons. They can also capture shapes with holes or detached areas.

It must be noted that the higher order neuron proposed by Lipson and Siegelmann is different from that used by Balestrino and Verona (1994). While Balestino and Verona also used the term “Higher Order Neurons” to denote their work, their higher order neurons were not used in competitive learning and clustering. Lipson and Siegelmann show that the higher order neurons proposed are tensors and that their order pertains to their internal tensorial order rather then the degree of their inputs. The proposition of Lipson’s higher order neurons paves the way for a natural extension and it is this idea of higher order neurons that will be used in this paper.

At this note a difference also has to be established between higher order neurons as described by Lipson and Siegelmann and used in this paper, and high dimensional clustering methods Ben Hur et al. (2001), for instance, proposed a method of clustering using support vectors. The method maps the data into high dimensions and finds hyper-spheres in the high dimensions. The hypersphere data is then mapped back to determine the final clustering arrangement.

In contrast to the above described algorithm, HONs are parametric methods, where the clusters are represented by their eigen-tensors, which are derived from their covariance matrices. There is no higher dimensional mapping.

The usage of higher order neurons opens the possibility of parametrically extending neuron shapes to a continuum of shapes, both intuitively and using simulation results. Before the development of higher order neurons, arbitrary cluster shapes could only be identified by non parametric methods such as agglomerative clustering. The introduction of higher order neurons now enables the representation of clusters using the corresponding eigen-tensors. The clusters can now be trained with Hebbian learning. Also, unsupervised learning with higher order neurons can be extended to supervised (Lipson and Siegelmann, 2000) and semi supervised learning.

Despite the advantages of higher order neurons, preliminary experiments showed us that there was a significant difference in their performance across problems and across clusters in the same problem. We observed that the choice of neuron order plays an important part in the clustering accuracy of the neurons. In higher order neurons, this choice of neuron order is manual and hence could be suboptimal.

In this paper, we aim to solve the problem of neuron order by using evolutionary algorithms to extend the idea of higher order neurons. The proposal of evolutionary higher order neurons (eHONs or multi-order neurons) aims to serve two purposes. To illustrate the problem scope, let us consider the simple data distribution shown in Figure 2.

Due to the non-spherical nature of clusters 2 and 3, the use of SOMs may not be suitable in correctly clustering the patterns in Figure 2. It is difficult to visualize whether higher order neurons can be used to solve this problem. From Figure 2, we can see that cluster 2 is spherical in nature. This cluster alone can be clustered properly by an SOM. On the other hand, cluster 1 is elliptical in nature and cluster 3 has a shape like a banana. If the order of these clusters can be identified, then the clustering can be performed more accurately.

In higher order neurons, all the clusters in the problem are given only one order. Multiple orders, such as that required for the clusters in Figure 2, are not provided. As a result, the higher order neuron may not be able to correctly cluster a situation provided in 1.

The two purposes of eHONs are described as follows

1. To identify the correct order for a given set of data and cluster them according to the order
2. To identify the possibilities of multiple ordered clusters, assign a suitable order for each cluster and perform clustering.

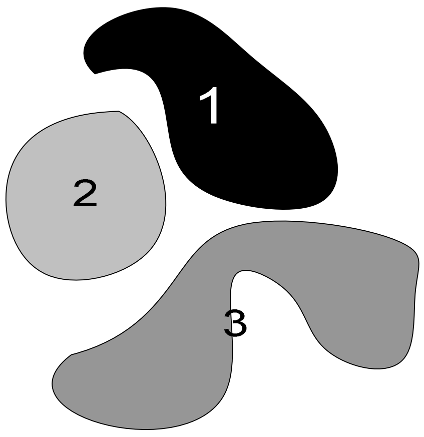


Figure 2. Hypothetical clusters to illustrate the performance scope of eHONs

The problem of order choice that occurs in HONs is thus circumvented by the use of eHONs. Evolutionary algorithms take into account that different clusters in the same problem may not be of the same order and use homogeneous coordinates to combine different orders of neurons together. As evolutionary higher order neurons are used to create a multi order solution to a problem, the terms *Evolutionary Higher Order Neurons* (eHONs) and *Multi Order Neurons* will be used interchangeably in this paper.

The rest of the paper is organized as follows: Section 2 explains the related theory of higher order neurons, and evolutionary algorithms. Section 3 describes the evolutionary learning of the higher order neurons. Section 4 describes the results of higher order neurons and evolutionary higher order neurons on real world data. Section 5 proposes a method for growing eHONs, enabling automatic detection of clusters. The article concludes with some remarks and future extensions.

2. Related theory

For the rest of the paper, we adopt the same notation as that of Lipson and Siegelmann, as given below:

x, w: Column vectors

W, D: Matrices

$x^{(j)}, D^{(j)}$: The *j*th vector/ matrix corresponding to the *j*th neuron

x_i, D_{ij} : Element of a vector/ matrix

x_H, D_H : Vector/ matrix in homogeneous coordinates

m : Order of the high-order neuron

d : Dimensionality of the input

N : The number of neurons in a cluster

The following terms appear in this paper and are as defined below:

Higher Order Neurons (HON) refers to the method proposed by Lipson and Siegelmann (2000), which represents a cluster by a neuronal order and tensorial structure. Section 2.1 describes the higher order neuronal structure and their training algorithm.

Self Organizing Maps (SOM) are a competitive neuronal structure for unsupervised learning introduced by Kohonen (1997)

Evolutionary Self organizing Maps (eSOM) are a model of self organizing maps proposed by Painho and Bacao (2000).

Evolutionary higher order neurons (eHONs), also termed as multi order neurons, are proposed in this paper as an evolutionary batch mode version of higher order neurons. They use evolutionary methods to train HONs, evolving not only the self organizing map structure, but also the order of each neuron.

2.1. Higher Order Neurons (HONs)

The Higher Order Neuron structure (Lipson and Siegelmann, 1998) generalizes the spherical and ellipsoidal scheme that is proposed by the self organizing maps and second order structures. The use of a higher order neuron structure gives rise to a continuum of cluster shapes between the classic spherical-ellipsoidal clustering systems and the fully parametric approach. Clusters of data in the real world usually have arbitrary shapes. The "shape" of a cluster is categorized using the order of the neuron representing it.

The constraint of the single order neuron (used by SOMs) comes from the fact that the weight $w^{(j)}$ of the neuron is of the same dimensionality as the input x, a single point in the domain. However, the neuron is modeling a cluster of points with attributes such as directionality and curvature.

To circumvent this restriction, the higher order neuronal structure augments the neuron by allowing it to map geometric and topological information, i.e., a second order neuron will allow for the orientation and size components of the cluster, thus introducing a new metric to the system. Each higher order neuron therefore aims to represent not only the mean value of the data points but also the principle directions, curvatures and other features of the cluster in these directions. Lipson and Siegelmann intuitively expressed a higher order metric as defining a multidimensional shape with an adaptive topology, as opposed to defining a sphere. Further, they also showed that higher order neurons exhibit stability and good training performance with Hebbian learning.

Tensorial multiplication and neuron shapes

A second order neuron is typically expressed by Lipson and Siegelmann as a matrix $D^{(j)} \in R^{dxd}$ that expresses the direction and size properties of a neuron for each dimension of the input, where d is the number of dimensions. The Euclidean distance metric in equation (2) can therefore be rewritten as

$$j(x) = \arg_j \min \left\| D^{(j)} \left(w^{(j)} - x \right) \right\| \qquad (5)$$

where $D^{(j)}$ can simply be expressed by a product of its eigenvalue and eigenvector matrices as below:

$$D^{(j)} = \begin{bmatrix} \frac{1}{\sqrt{\sigma_1}} & 0 & \dots & 0 \\ 0 & \frac{1}{\sqrt{\sigma_2}} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & \frac{1}{\sqrt{\sigma_d}} \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_d^T \end{bmatrix} \qquad (6)$$

In contrast to equation (2), the second order neuron in equation (5) is represented by 2 variables, $\left(D^{(j)}, w^{(j)} \right)$, where $D^{(j)}$ represents the rotation and scaling and $w^{(j)}$ represents translation. As $D^{(j)}$ can be obtained by the eigenstructure of the correlation matrix of the data associated with neuron *j*, equation 6 is rewritten as follows:

$$D^{(j)} = \lambda^{(j)-\frac{1}{2}} V^{(j)} \qquad (7)$$

Here, $\lambda^{(j)}$ refers to the eigenvalues of $D^{(j)}$ in diagonal form and $V^{(j)}$ produces its unit eigenvectors in matrix form.

Since the eigenvalues of a matrix A can be written as

$AV = \lambda V$,

$A = V^T \lambda V$

The expression $\left\|D^{(j)}\left(w^{(j)} - x\right)\right\|$ in equation (5) can be written as

$$\left\|D^{(j)}\left(w^{(j)} - x\right)\right\| = \left(w^{(j)} - x\right)^T D^{(j)^T} D^{(j)}\left(w^{(j)} - x\right) \qquad (8)$$

From 6, $D^{(j)^T} D^{(j)}$ can be written as

$$D^{(j)^T} D^{(j)} = \begin{bmatrix} v_1^T & v_2^T & \dots & v_d^T \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma_1} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_2} & & : \\ : & & . & 0 \\ 0 & \dots & 0 & \frac{1}{\sigma_d} \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ : \\ v_d^T \end{bmatrix} = R^{(j)-1} \qquad (9)$$

$R^{(j)}$ here is the inverse of the correlation matrix, $R^{(j)} = \sum x^{(j)} x^{(j)T} \in R^{dxd}$. The minimum Euclidean distance rule can therefore be written as

$$j(x) = \arg_j \min\left[\left(x - w^{(j)}\right)^T R^{(j)-1}\left(x - w^{(j)}\right)\right], \text{ j=1,2…,N} \qquad (10)$$

This use of the correlation matrix for clustering was used by (Gusfaffson and Kessel, 1979). Although the use of the correlation matrix somewhat simplifies the clustering process, it still requires the optimization of two terms, $R^{(j)}$ and $w^{(j)}$. Since separate optimization cannot be easily derived, Lipson and Siegelmann proposed the combination of the two parameters into one expanded covariance matrix in homogeneous coordinates as

$$R_H{}^{(j)} = \sum x_H^{(j)} x_H^{(j)T} \qquad (11)$$

where $x_H^{(j)} \in R^{(d+1)x1}$. This expanded matrix was proposed by Faux and Pratt (1981) and it contains coordinate permutations of $x_H = \begin{bmatrix} x \\ - \\ 1 \end{bmatrix}$, involving 1 as one of the multiplicands. The Euclidean distance matching criterion now can be written as

$$j(x) = \arg_j \min\left[X^T R^{(j)-1} X\right]$$

$$j(x) = \arg_j \min\left\|R^{(j)^{-1}} x\right\|, \qquad \text{j=1,2…N} \qquad (12)$$

Higher orders

Equation 12 can be extended beyond the second order neuron structure. The use of the ellipsoidal neuron (or the second order neuron) can capture only linear properties, but not curved directions. Higher order geometrics were introduced by Lipson and Siegelmann (2000) for the purpose of capturing more complex spatial configurations. The higher order extended covariance matrix requires the creation of a higher order correlation matrix. This can be obtained by multiplying the vector x_H by itself, in successive outer products, more than once. The process yields a covariance tensor, rather than a covariance matrix, such that $R_H^{(j)} \in R^{(d+1)x(d+1)}$. A covariance tensor of order m, for instance, can be obtained by $2(m-1)$ outer products of x_H by itself. Consequently, the inverse of the tensor can be used to compute the higher order correlation of x with the tensor, using a tensor multiplication ($\oplus$)

$$j(x) = \arg_j \min \left\| R^{(j)-1} \oplus x_H^{m-1} \right\|, \qquad \text{j=1,2...N} \qquad (13)$$

The metric in equation 12 will not necessarily be spherical, but may be disjoint or even take several other forms.

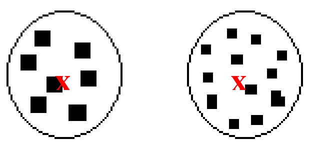
The HON training algorithm

The higher order unsupervised learning algorithm proposed by Lipson and Sieglamann (2000) is performed as follows

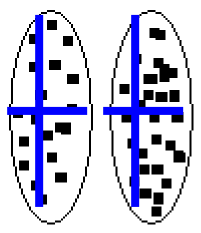
1. Select the number of clusters (or number of neurons) *K*, and the order of the neurons (*m*) for a given problem.
2. The neurons are initialized with $Z_H = \sum_{i=1}^{n} x_i^{2(m-1)}$. Z_H is the covariance tensor of the data, initialized to a midpoint value. In the case of a second order problem, the covariance tensor is simply the correlation matrix $Z_H = \sum_{i=1}^{n} x_i^T x_i$. For higher order tensors, this value is calculated by writing down x_H^{m-1} as a vector with all the m^{th} degree permutations of $\{x_1, x_2, ..., x_d, 1\}$and finding Z_H as the matrix summing the outer product of all these vectors. The value of the inverse of the tensor is found and normalized using its determinant f to obtain Z_H^{-1}/f.
3. The winning neuron for a given pattern is computed using
$$j = \arg_j \min \left\| Z_H^{-1} / f \oplus x^{(m\ \ 1)} \right\| \qquad (14)$$

4. The winning neuron is now updated using $Z_{H,new} = Z_{H,old} + \eta x_i^{2(m-1)}$, where η is the learning rate. The new values of Z_H and Z_H^{-1}/f are stored.
5. Steps 3 and 4 are repeated.

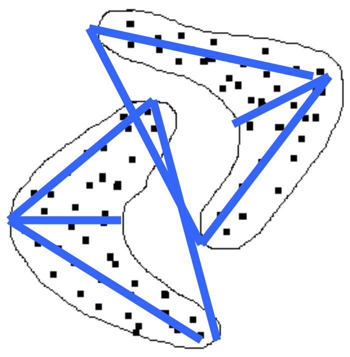
Ideally, while the first order neuron finds spherical shapes, and the second order neuron finds ellipsoidal shapes (with two principal directions), The third order neuron, which makes use of the covariance tensor (having a cubical shape), finds four principal directions and copes with banana shaped clusters. Figure 2 shows the neuron information of first-, second- and third-order neurons respectively.



Representation of a first-order neuron



Representation of a second-order neuron



Representation of a third-order neuron

Figure 3. The internal representation of a self organizing neuron is a point for a first order neuron (a), two eigenvectors (and values) for the second order neuron (b), and four eigenvectors (and values) for the third order neuron (c). More eigenvectors (and values) are encoded in the higher order tensors of neurons with a higher order.

2.2. Evolutionary Self Organizing Maps

Evolutionary algorithms have been used to find global solutions in many applications, including neural network applications for supervised learning (Yao, 1993). Inspired by this in (Painho and Bacao, 2000), the authors apply genetic algorithms to clustering problems with good effect. The genetic algorithm applied is simple and retains the form of SOMs, but with evolutionary representation of the weights. Genetic algorithms (Goldberg, 1989) make use of the Darwinian theory of the survival of the fittest. The parameters to be determined are encoded as an array of elements, also called a chromosome. The chromosomes are evaluated and allowed to reproduce according to their fitness values, so as to allow the development of new chromosomes with better fitness values.

More simply, since the objective is to maximize the value $\left\| \mathrm{w}^{(k)T} \mathrm{x} \right\|$ for each pattern x, a population of real coded chromosomes encode $\mathrm{w}^{(k)}$, for each cluster k. Each chromosome therefore consists of K*d elements, where K is the number of clusters and d is the dimension of the input data. The chromosomes are evaluated in batch mode, such as to maximize

$$\sum_{k=1}^{K} \sum_{x \in C_k} \left\| \mathrm{w}^{(k)T} \mathrm{x} \right\| \qquad \textbf{(15)}$$

Crossover and mutation are performed and a new generation of chromosomes is produced. The process is continued until the system stagnates or until a maximum number of epochs is reached.

3. Evolutionary Higher Order Neurons (eHONs)

We propose the evolutionary higher order neurons as an extension of the Higher Order Neuron Structure and the Evolutionary Self Organizing map. The idea of evolutionary higher order neurons is motivated by the following reasoning: While Lipson and Siegelmann's Higher Order Neurons are shown to exhibit improved performance; some of our simulations (presented later in section 4) showed that the order of the neuron plays an important part in the meaning of the clusters formed. A higher order neuron does not necessarily perform better than a lower order neuron. In the HON algorithm described in section 2, the order of the neuron is pre-specified by the user (Lipson et al., 2000).

Moreover, for a given dataset, only a single order of neuron is used to represent all the clusters in their work. We feel that this is a limitation to the algorithm. Data is usually distributed irregularly, with some classes having spherical, elliptical, banana-shaped or even higher order forms.

We propose in this paper a Messy Evolutionary Algorithm (Goldberg et al., 1991) based multi-order HONs. The algorithm is outlined below.

Batch version of HONs

Genetic Algorithms are used in Neural network applications with various scopes. Many of these GA based Neural networks (GANNs) are developed using batch mode training (Yao, 1993). A corresponding batch mode training version of HONs has also been developed to facilitate the process of developing eHONs.

The batch algorithm developed is similar to the online algorithm proposed by Lipson and Siegelmann (2000) in section 3.1. However, instead of choosing a winning neuron by using $j = \arg_j \min \left\| Z_H^{-1} / f \oplus x^{(m-1)} \right\|$, we implemented a batch minimization criterion such as the one used in equation 15 of the evolutionary SOMs in section 3.2. The algorithm focuses on minimizing Equation 16.

$$\sum_{k=1}^{K} \sum_{x \in C_k} \left\| \left(Z_H^k \right)^{-1} / f_k \oplus x^{(m-1)} \right\| \qquad \textbf{(16)}$$

Preliminary simulation results to compare the effect of the batch mode algorithm with the online version developed by Lipson showed that the batch mode implementation has similar results as the online version.

Chromosome Initialization

The initialization of chromosomes is done as outlined by the following steps.

1. Each data point is randomly assigned to one of the K clusters.
2. The covariance tensor of order *m*, Z_H^k, for the cluster $k \in K$ is initialized as

$$Z_H^k = \sum_{x \in C_k} x^{2(m-1)} \qquad \textbf{(17)}$$

Chromosome structure

Each chromosome is coded as an array of structures, each consisting of two components: the order of a chromosome and the value of the tensor, as given below:

```
struct chromosome
{
        neuron NEURON[K];
};
struct neuron
{
        int order ;
        int tensor[][] ;
} ;
```

A tensor, regardless of the neuron order, is flattened out into a Kronecker matrix (Graham, 1981) in a similar form as used by Lipson.

Kronecker notation

The Kronecker tensor product flattens out the elements of $X \oplus Y$ into a large matrix formed by taking all the possible products between the elements of X and those of Y. If X is a 2x2 matrix, then

$$X \oplus Y = \begin{bmatrix} Y.X_{1,1} & Y.X_{1,2} \\ Y.X_{2,1} & Y.X_{2,2} \end{bmatrix} \quad (18)$$

each term in 18 is a matrix of the size of Y. The Kronecker notation makes the internal structure of a higher order tensor easy to perceive and the eigenvalues of the tensor can be used in finding the equivalent of the minimum distance rule in equation 16.

Global search: Muation

Global search for eHONs is simulated by large range mutation. There are two criteria for large range mutation:

1. A random element in a tensor is mutated with a probability p_1.
2. The order of the tensor is changed with a probability p_2. The tensor is now reinitialized using Equation 17.

Fitness function

The fitness function is the optimization of the expression in equation 16. The expression is minimized so as to maximize the cluster tightness.

Algorithm summary

The eHON algorithm can be summarized as below:

1. Initialize a population of chromosomes denoting a set of multi-order tensors
2. While termination criterion is not met

    a. Evaluate each chromosome x using $f(x) = \sum_{k=1}^{K} \sum_{x \in C_k} \left\| (Z_H^k)^{-1} / f_k \oplus x^{(m-1)} \right\|$.

    b. Select fittest chromosome

    c. Perform mutation and reproduction

4. Experimental Results

Evaluation of the system is performed according to consistency with the ground truth information. The consistency with the ground truth information is measured by the number of misassigned patterns, i.e., the number of patterns not clustered together with their true class. Mutation is carried out as follows: Mutation of a random element in the tensor is performed with an arbitrary probability of 0.2. The order of a tensor is changed with a probability of 0.1. Arbitrary values of mutation were chosen as we were interested in the scope of the eHON. Other values of the mutation and crossover probability did not show a significant change in the accuracy of the eHON. Each chromosome could take an order of between 1 and 3. The relatively low order of neurons was preferred for this preliminary work as higher order neurons require more computational effort than those of lower orders. We therefore felt that, orders between 1 and 3 were sufficient to illustrate the scope of eHON.

Data and algorithm descriptions

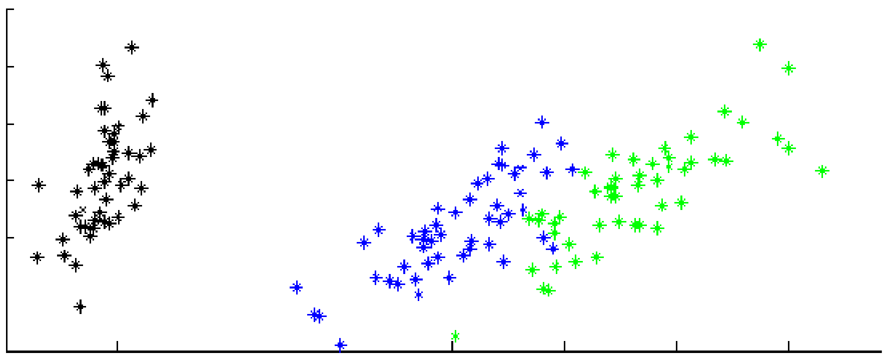
This section compares the results of four algorithms

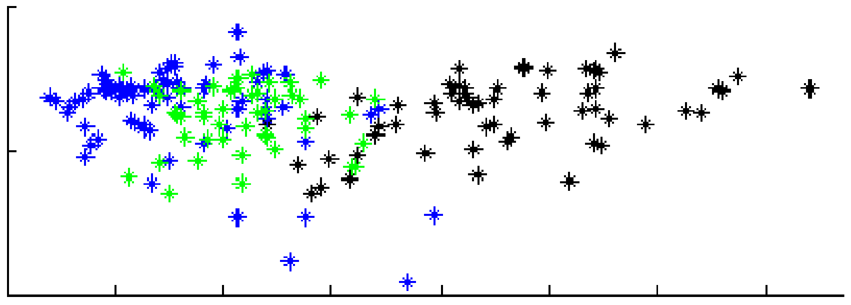
1. eHONs: Higher order self organizing neurons with evolutionary capabilities which identify the best possible order for a chromosome based on a population generated with various multi-order chromosomes.
2. HON: The higher-order neuron structure proposed by Lipson and Siegelmann (2000). Simulations were run with order 2 and order 3. Order 2 detects elliptical, oval and, to a certain extent, banana shapes, while order 3 detects other higher-order shapes to a certain extent
3. SOM: The self organizing map or HON of order 1 (Kohonen, 1997). This algorithm detects spherical clusters and, to some extent, oval shaped clusters.
4. eSOM: The self organizing map or HON of order 1 trained using evolutionary algorithms similar to that proposed by Painho and Bacao (2000). This algorithm also detects spherical clusters and, to some extent, oval shaped clusters.

The algorithms 1 to 4 will be applied to the following benchmark datasets from the UCI machine learning repository:

1. Iris dataset (4 dimensions, 3 clusters, 150 patterns)
2. Wine dataset (13 dimensions, 3 clusters, 178 patterns)
3. Glass dataset (9 dimensions, 6 classes, 214 patterns)

The figure below shows the 2-dimensional Principal component projections of these datasets.


(a)


(b)

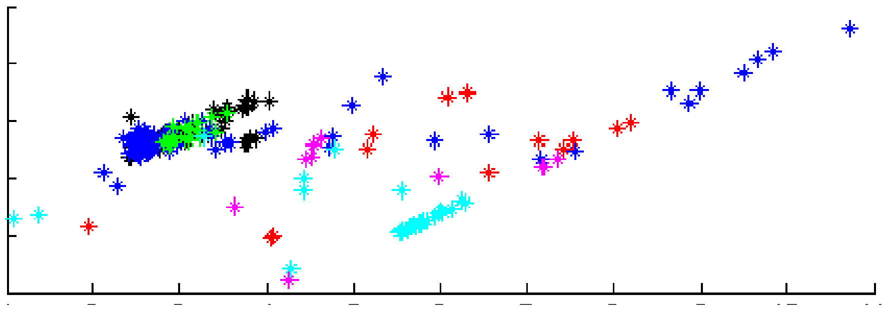

(c)

Figure 4. Distribution of data in (a) IRIS (3 clusters) (b) WINE (3 clusters) and (c) GLASS (6 clusters) datasets

Effect of eHONs (studies on the IRIS dataset)

The figure below compares the clustering of the IRIS data (with 3 clusters) using the SOMs, evolutionary SOMs, HONs and eHONs. It is observed that, over an average of 10 runs, the number of misassigned patterns of the data is as low as 3.5, using the eHON system, as opposed to 4 using the HON system.

Similarly, the eHON is also shown as being capable of identifying a suitable order for a given cluster. From the visualization of the IRIS data in Figure 4, and from the performance of the self organizing map, we can see that a single-order neuron is sufficient to correlate one of the classes with the ground truth information. However, the other two classes are closer to each other and therefore need higher-order neurons. The eHON system correctly identified order 2 to be suitable for these clusters while order 1 to be sufficient for the further cluster.

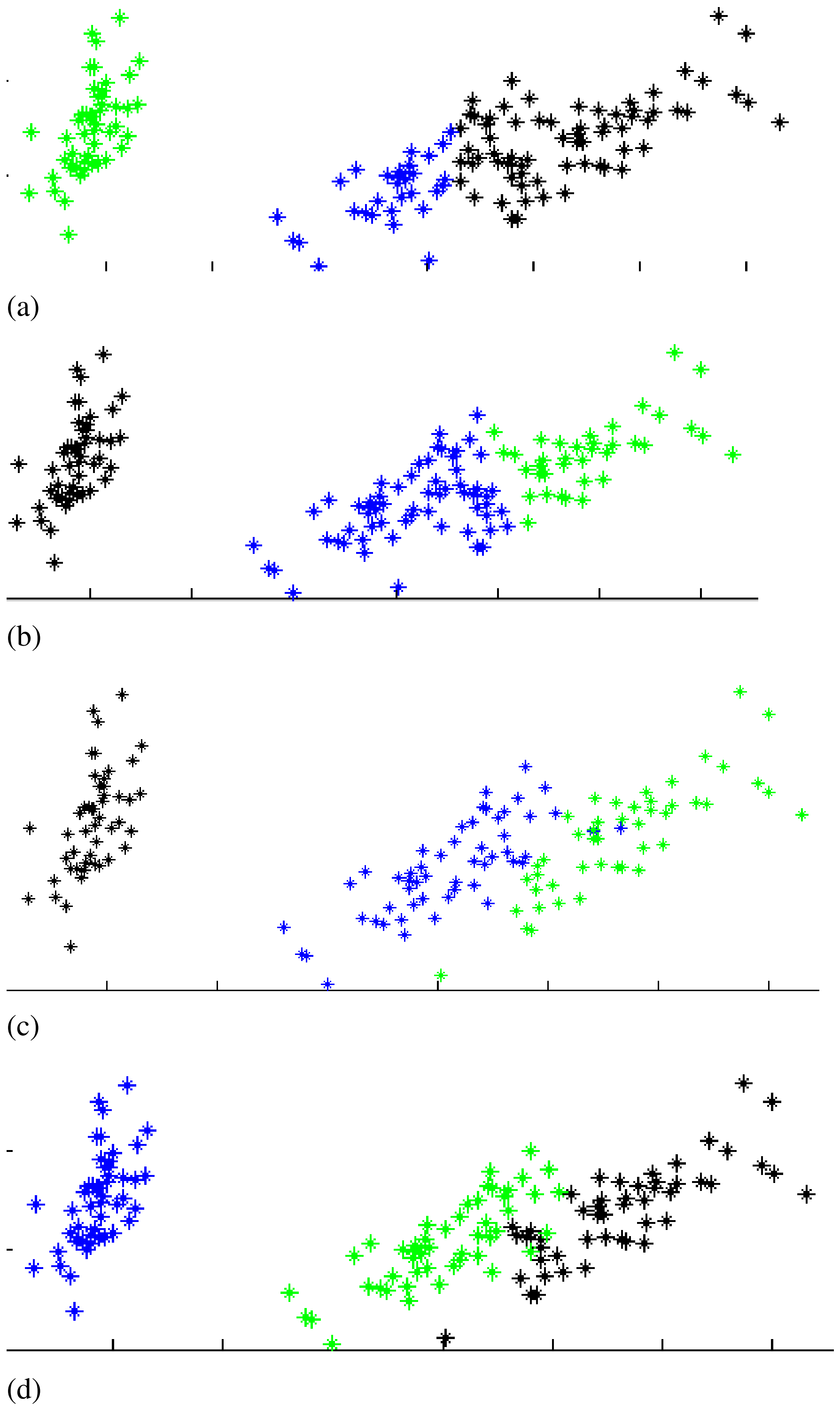


Figure 5. Clusters of the IRIS data using a. SOMs, b. eSOMs, c. HONs and d. eHONs

Correlation with ground truth information in real world data

In this section we present the correlation with ground truth information (the available class labels), for the IRIS, WINE and GLASS datasets. We begin the experimental analysis by observing the clustering accuracy across different classes of each dataset. The tables below show the correlation tables of the clusters found and the ground truth information (class labels) for the IRIS, WINE and GLASS datasets when SOMs and Higher Order Neurons are used. The number of mis-clustered patterns is marked in bold.

The tables can be read as follows. For instance, in Table 2, for the IRIS data clustered with neuron order 3, 50 patterns from class 1 are clustered together, with 0 patterns clustered with patterns of either class 2 or 3. However, 2 patterns from class 2 are clustered with patterns of class 3 and 5 patterns from class 3 are clustered with patterns of class 2 (as indicated in bold). The total number of mis-clustered patterns is noted below each table.

Table 2. Correlation between ground truth information and cluster information for the IRIS data using 2nd order 3rd order neurons

Neuron Order=2

	Class 1	Class 2	Class 3
Class 1	50	**0**	0
Class 2	0	48	**2**
Class 3	0	**2**	48

Total mis-clustered: 4

Neuron Order=3

	Class 1	Class 2	Class 3
Class 1	50	0	0
Class 2	0	48	**5**
Class 3	0	**2**	45

Total mis-clustered: 7

Table 3. Correlation between ground truth information and cluster information for the WINE data using SOMs (1st order neurons) , 2nd order and 3rd neurons.

Order=1

	Class 1	Class 2	Class 3
Class 1	27	0	**13**
Class 2	**31**	62	0
Class 3	**1**	**9**	35

Total Misclustered: 54

Order=2

	Class 1	Class 2	Class 3
Class 1	36	**8**	0
Class 2	0	34	0
Class 3	**23**	**29**	48

Total Misclustered: 60

Order=3

	Class 1	Class 2	Class 3
Class 1	35	**9**	**2**
Class 2	**10**	27	**8**
Class 3	**14**	**35**	38

Total Misclustered: 78

Table 4. Correlation between ground truth information and cluster information for the GLASS data using SOMs (1st order neurons) , 2nd order and 3rd neurons.

Order=1

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Class 1	29	**4**	0	0	7	0
Class 2	**9**	28	**6**	0	0	0

Order=2

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Class 1	50	**47**	**11**	**8**	**4**	**4**
Class 2	0	**20**	0	**1**	**5**	**14**

Class 3	**22**	**24**	2	0	0	0
Class 4	0	0	0	10	**2**	**1**
Class 5	**1**	**17**	0	**3**	0	**2**
Class 6	**9**	**3**	**9**	0	0	**26**

Total misclustered: 119

Class 3	0	**0**	2	0	0	0
Class 4	**3**	**3**	0	2	0	**5**
Class 5	**2**	**5**	**1**	0	0	**1**
Class 6	**15**	1	**3**	**2**	0	5

Total misclustered: 176

Order=3

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6
Class 1	36	**34**	**11**	**3**	**5**	**2**
Class 2	**5**	11	**3**	**4**	0	**3**
Class 3	**13**	**10**	1	**1**	0	**5**
Class 4	**2**	**10**	**1**	2	**1**	**6**
Class 5	**10**	**3**	**1**	**2**	1	**6**
Class 6	**4**	**8**	0	**1**	**2**	7

Total misclustered = 157

The following observations are made from the tables above:

1. The best solution is obtained for the IRIS and WINE datasets with neuron order 2, and for the GLASS dataset with neuron order 1.
2. Different classes are better clustered with different orders of neurons. For instance, in the WINE dataset, order 1 is most suitable for class 2, while order 2 is most suitable for classes 1 and 3.

The objective of the eHONs is therefore to solve for the best order for each class using evolutionary algorithms. The following tables compare the total number of misclustered patterns obtained when using SOMs, HONs, eSOMS and eHONs. The eSOMs and eHON algorithms were run 10 times and the mean results are presented.

Table 5. Results of the IRIS data (best order: 1, 2, 2)

Algorithm	**Number of mis-assigned patterns**
HON *(Higher Order Neuron), order=2*	4
HON *(Higher Order Neuron), order=3*	7
SOM *(Self Organising map) HON order =1*	23
eSOMs *(Evolutionary evolved SOMs)*	16
eHONs *(to find the optimal order for a given class)*	3.5

Table 6. Results of the WINE data (best order: 2, 1, 2)

Algorithm	Number of mis-assigned patterns
HON *(Higher Order Neuron), order=2*	60
HON *(Higher Order Neuron), order=3*	78
SOM *(Self Organising map) HON order =1*	54
eSOMs *(Evolutionary evolved SOMs)*	57
eHONs *(to find the optimal order for a given class)*	49

Table 7. Results of the GLASS data (best order: 2, 1,1 , 1, 3, 2)

Algorithm	Number of mis-assigned patterns
HON *(Higher Order Neuron), order=2*	176
HON *(Higher Order Neuron), order=3*	157
SOM *(Self Organising map) HON order =1*	119
eSOMs *(Evolutionary evolved SOMs)*	115
eHONs *(to find the optimal order for a given class)*	110

Discussions

The above results show the importance of the use of multi order neurons in clustering, as opposed to higher order neurons. The use of the evolutionary algorithm picks out the best possible cluster order for each class, thereby enabling better correlation of clusters with the class labels, making the clusters more meaningful. This property is observed in all three datasets.

The introduction of cluster order serves a dual purpose. As shown in the tables above, there is an obvious improvement in clustering accuracy when multi order neurons are used. While the improvement is lucrative, there is a cost encountered in computation. Each higher order neuron requires the computation of the inverse of the higher order tensor, which is intensive in nature. This is a clear set back in the use of multi order neurons. Several other clustering algorithms such as AHC (Blatt et al., 1996) and cluster ensembles (Fred and Jain, 2005; Fred and Jain, 2002) can also be used instead to improve the clustering accuracy.

On the other hand, using the eHON, we also obtain the optimal order of a particular cluster of patterns, thereby also giving information on the shape of the cluster. We feel that this is an unique contribution of the eHON algorithm which will allow the algorithm to be integrated to ensemble clustering approaches. It is expected that cluster shape information can help to tune the clustering ensembles to the required shape and improve clustering accuracy.

Limitations

As the eHON model improves upon the model of the Higher Order Neurons, their computational cost is equivalent to that of the HONs. The computational complexity of the covariance tensor Z_H and the inverse covariance tensor Z_H^{-1} is high. Also, as the eHON makes use of several chromosomes, the covariance tensor of each new chromosome has to be evaluated.

The current implementation of the eHON algorithm takes into account the computational complexity and only implements neurons with orders 1, 2 and 3. However, with these orders alone, it is not possible to capture complex cluster shapes and overlapping clusters. Using the eHON algorithm to identify more complex cluster structures will involve higher computational complexity.

5. Growing Multi Order Neurons - a simplified model

While the use of eHONs enables the clustering system to evolve and tune, for a given problem, multiple orders of clusters, one problem that still remains. In all the problems discussed in section 4, the number of clusters to be formed is a predetermined value, found earlier and given to the system. However, determining the number of clusters in a data has been widely researched (Blackmore and Miikkulaihen, 1993; Dittenbach et al., 2000; Firtze, 1995). The algorithms described in these papers propose various ways to increase dynamically the number of clusters, using certain threshold conditions and stopping criteria. Dittenbach et al (2000) go one step further and create a hierarchical structure for clustering. Their setup not only grows the number of clusters but also creates a hierarchy, thereby identifying the relationship between clusters.

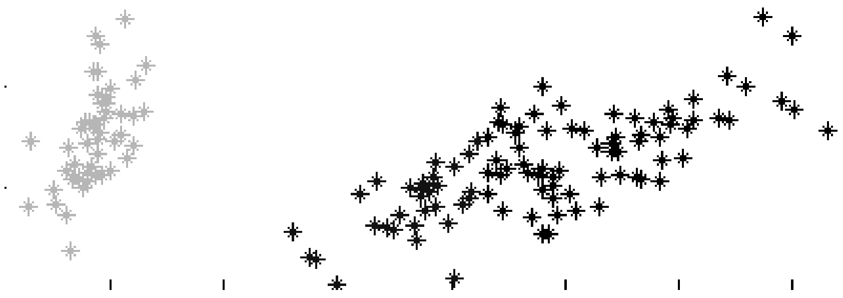
All the algorithms described above are implemented using Self Organizing maps (Kohonen, 1997) or other first order clustering methods. However, they can be converted into a batch version and adapted to the use of higher order neurons, thus creating a growing model for the number of clusters. The method developed will then be an incremental approach to increasing the number of clusters needed until the resultant number of clusters meets the error criterion.

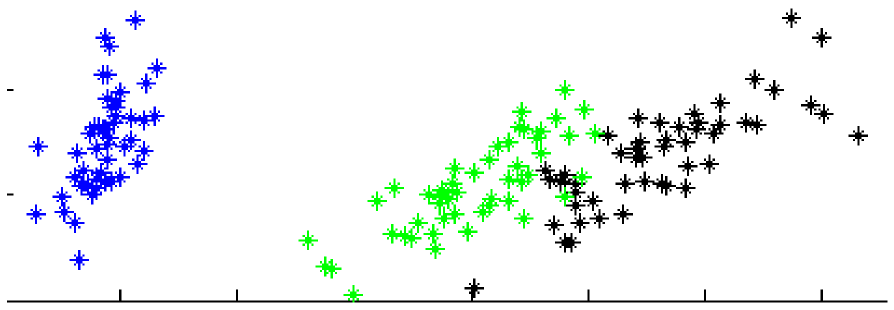
For illustration, we describe, in this section, a very simple extension to eHONs which can be used to determine the number of clusters. We begin the eHON algorithm with two clusters and fix the number of training iterations to be λ . We now determine the "spread" of each cluster, and the total spread of the eHON map. Effectively, this can be obtained by evaluating $j(x) = \sum_{x \in j} \left\| R^{(j)-1} \oplus x_H^{m-1} \right\|$ for each of the two clusters. The total spread of the eHON can therefore be evaluated as the sum of the spread of each of the clusters.

If the total spread after λ iterations is less than a given threshold τ, the cluster with the largest spread is reduced into two clusters. The eHON algorithm is repeated, now with the patterns in the largest cluster, to find two new clusters to represent the large cluster. The pseudocode of the growing eHON can therefore be summarized as below:

```
1. Set the number of clusters as two
2. Implement eHON with all the available patterns
3. IF the cluster spread is less than $\tau$
     a. Complete training
4. Else
     a. Repeat steps 1 to 3 with the patterns of the cluster with the largest spread.
```

The process can be illustrated using the IRIS data, as shown in Figure 6. Figure 6a shows two clusters of the IRIS data, of order one and order 2 respectively. In Figure 6b, the larger cluster is further reduced into two smaller clusters, each of order 2. The final clustering obtained has the same accuracy as that reported in Table 5.


(a)



(b)

Figure 6. The growing eHON model to incrementally determine the number of clusters

6. Conclusions

In this paper, we have motivated and introduced the concept of *multi order neurons* or *evolutionary higher order neurons* (eHONs). While higher order neurons are useful in clustering, their performance is heavily dependent on the pre-specified neuron order. It is desirable to determine the best order, not just for the data set but for individual clusters in the data set.

The global search capability of evolutionary algorithms makes them an ideal choice of search algorithms for a problem of this nature, therefore leading to the development of Evolutionary Higher Order Neurons (eHONs), also known as multi-order neurons. These eHONs work by identifying the best order a cluster can take, thereby identifying the complexity of each cluster. The clusters obtained are therefore better correlated with the class labels, making them more meaningful.

eHONs were designed by implementing an evolutionary algorithm to evolve the covariance tensors of neurons of multiple orders. The neurons are denoted as structures of tensors, each structure having as its components a. the order of the neuron and b. The kronecker form of the tensor. The evolutionary algorithm was performed using mutation, where the order of the neuron and the value of tensor elements were mutated with a given probability.

It was noticed that the use of the multi order neurons improves the performance of the best higher order neuron structure and is significantly better than the use of SOMs and single order neurons. The improvement in accuracy is as high as 13% when compared to the use of SOMs (IRIS data). From the results obtained, we can observe that the development of eHONs shows promise in the field of clustering.

We have also proposed a minor extension to the eHON model to grow the system such that the optimal number of clusters can be incrementally found. While the method proposed is mostly heuristic, the results on the IRIS dataset shows promise for further research using more sophisticated algorithms.

While eHONs is capable of finding optimal cluster order, its use is still confined to disjointed clusters of arbitrary shapes. Tensors of lower orders such as described in this paper will not be capable of dealing with overlapping clusters. However, as Lipson and Siegelmann (2000) have mentioned, tensors of very high orders are capable of dealing with overlapping clusters. Future work will therefore include adapting the eHON structure to deal with overlapping clusters, and to investigate the effect of eHONs as the system deals with higher dimensional complicated data.

REFERENCES

Balestrino A, Verona B F (1994), New adaptive polynomial neural network, Mathematics and Computers in simulation, 37, 189-194

Ben-Hur A, Horn D, Siegelmann H T, Vapnik V (2001), Support Vector Clustering, Journal of Machine Learning Research, 2, pp 125-137

Blackmore J and Miikkulaihen (1993), Incremental Grid Growing: encoding high dimensional structure into a two-dimensional feature map. IEEE Int'l Conf Neural Networks (ICNN'1993)

Blatt M, Wiseman S, Domany E (1996). Supermagnetic clustering of data, Physical Review Letters, 76/18, pp. 3251-3254

Dittenbach, M, Merkl D, Rauber A (2000), The growing Hierarchical Self organizing map, Int'l Joint Conference on Neural networks, IJCNN' 2000, 6, pp 15-19

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

Fred A , Jain A K, (2002), Data Clustering using Evidence Accumulation, IN Procedings of 16th Int Conference on Pattern Recognition, pp. 276-280

Fred A, Jain AK, (2005), Combining Multiple Clustering Using Evidence Accumulation, IEEE transactions on Pattern analysis and Machine Intelligence, 27(6), pp. 835-850

Fritze B (1995), Growing Grid: A self organizing network with constant neighborhood range and adaptation strength.

Goldberg D E (1989), Genetic Algorithms in Search, Optimization and Machine Learning: Addition Wesley

Goldberg D E, Deb K and Korb B (1991), Don't worry, be messy, Proceedings of the Fourth International Conference in Genetic Algorithms and their Applications, edited by R. Belew and L Booker, pp. 24-30

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

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

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

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, 29-60.

Lipson, H, Hod, Y, Siegelmann, H T (1998). Higher order clustering metrics for competitive learning neural networks. In proceedings of the Isreal-Korea Bi National Conference on New themes in computer aided geometric modeling. Tel-Aviv, Isreal

Lipson, H, Siegelmann, (2000), Clustering Irregular Shapes using higher order neurons, Neural Computation 12, pp. 2331-2353.

Mao, J and Jain, A (1996). A Self Organizing network for hyper ellipsoidal clustering (HEC), IEEE Transactions on Neural Networks, 7, pp. 16-39

Painho M, Bacao R (2000), Using Genetic Algorithms in Clustering Problems, Geocomputation 2000, available online at http://www.geocomputation.org/2000/GC015/Gc015.htm

The UCI Machine Learning repository: http://www.ics.uci.edu/~mlearn/MLRepository.html

Yao X (1993), A review of evolutionary artificial neural networks, International Journal of Intelligent Systens, 8(4), pp. 539-567.