

# Topology-Based Fuzzy Clustering (TFC)

Abhishek Jaiantilal and Atam P. Dhawan\*

Department of Electrical and Computer Engineering

New Jersey Institute of Technology, University Heights, Newark, NJ 07102, USA

---

## Abstract

Online clustering is of significant interest for real-time data analysis. Generic offline clustering methods such as K-Means, C-Means and others are computationally expensive. The computational burden of these methods increases non-linearly with the size of the data set. In addition these methods usually require a good amount of supervised knowledge yielding a non-unique solution. For real-time data analysis, there is an important tradeoff between accuracy and computational efficiency. An unsupervised one-pass clustering method Topology-Based Fuzzy Clustering (TFC), which efficiently adapts to data distribution using the topology of data, is proposed. TFC uses the method of Growing Neural Gas (GNG) method of creating linked sub-clusters and extends GNG by assigning a fuzzy membership to the sub-clusters, noting the link structure for creating clusters and influencing the learning nodes at each sub-clusters. The computational burden for TFC is proportional to the size of the initial data set and increases linearly with the addition of new data.

**Keywords:** fuzzy clustering, topology assisted clustering, incremental learning

---

## 1. Introduction

Computational time is a very important factor for the consideration of algorithms that need to do clustering in real-time. Iterative algorithms based on K-Means have an exponential time for calculation with regards to the number of the data samples to cluster [1,2]. Thus there exists a tradeoff between the samples sizes used for such clustering vs. the time taken for execution for K-Means with a good accuracy of clustering. Also the overdependence of K-Means on the initial cluster center is the reason of non-unique solutions.

Shape detection also is an important consideration when it comes to choice of the clustering algorithm. K-Means based algorithms and its variants like Gath-Geva [4] can detect clusters of Hyperellipsoidal shapes [1]. Shapes like Shell etc would need a different algorithm like C-Shells []. So detection of shaped cluster becomes very much application specific. Many of these algorithms would still not detect a very unusual arbitrary shape.

Traditionally clustering algorithms can be divided into 2 type's hierarchical and non-hierarchical clustering algorithms. In hierarchical, the number of clusters to be found is not required to be known priori. Hierarchical clustering, tries to create a partitioning, such that either a single cluster containing all data points or n clusters containing a single data point each. Hierarchical Clustering can be subdivided into agglomerative methods which work on creating clusters from 'n' data points, and divisive methods that create better clusters from a single cluster. As hierarchical clustering considers only local neighbors at each step, it is not possible to include priori knowledge about shaped clusters and their size. Also traditional hierarchical clustering is static as a result of which moving data points in a cluster to another cluster are not possible.

Non-hierarchical clustering does include prototype-based iterative clustering algorithms and self-organizing maps (SOM). Prototype-based iterative clustering algorithms do include 2 different measures on the inclusion of a data point in a cluster, which is either a binary measure (in or not-in a cluster, also called as a hard or crisp measure) or fuzzy measure (1 data point can be in multiple clusters). Fuzzy measure based clustering algorithms are helpful in data sets containing overlapping cluster boundaries. Such prototype based algorithms can be used to detect various shaped clusters like lines, planes, circles, ellipses, curves and other surfaces. Probability based measure has also been adapted for clustering, in algorithms like the Probabilistic C-Means(PCM).

Growing Neural Gas by Fritzke[6], is based on the Self Organization, with neural nodes added and deleted during the self-organization process. Growth is based on the earlier proposed growing cell structures by Fritzke [14] and the topology generation of Competitive Hebbian Learning by Martinetz and Schulten[7], combined to a new model. Few units are initialized and then newer units are inserted successively, on basis of local error measures that are calculated with observation of newer data. Such new units are inserted near the unit(s) that have the maximum such error.

Due to the peculiar construction of the nodes, GNG is resistant to shapes. Also a prominent feature of GNG is it takes linear time for execution vs. the number of data-samples presented to it. This is very useful in cases where estimation cannot be taken on the size of the sample test data for learning of the network. Due to the inherent plasticity of the GNG, newer patterns can occur over time. Thus GNG is promising in terms of adaptivity, execution time and shape-independence. GNG has been used to learn patterns in a supervised fashion [9], using

Radial basis function network, which requires a bit of learning every time a newer pattern is presented.

The primary goal of this paper is to present an algorithm that uses the best unsupervised properties of the GNG algorithm and compound it with aided user input. Thus the algorithm retains the properties of GNG, namely adaptivity, execution time and shape independence. And also adds the properties of learning that can be achieved as much in an unsupervised manner as possible but also allows the flexibility to supervisory learn certain data classes.

During our experimentations with GNG we found that the following characteristics are desirable for effective clustering in GNG.

1. Difference in Learning between nodes.

There are 2 approaches that can be taken for different learning rates; the first being a time dependent learning rate, and the second being a topology dependent learning rate. Time dependent learning rate causes decrease of the learning rate per node over time. The problem with the time dependent learning rate can be highlighted to it causing a fixed network over time which doesn't allow effective learning when newer patterns occur near the node with the least learning rate.

We define topology dependent learning rate as that which is dependent on the relative position of the node within a cluster. Thus basically defining a learning rate dependent indirectly on the relative distribution of the data through the cluster, with the most central node(s) of the cluster learning the least whereas the exterior nodes having the maximal learning rate. The intuition behind this is that the interior nodes are the nodes that are most descriptive of the cluster, and having a higher learning rate (compared to other nodes) is detrimental and would

cause unstable networks. Similarly the exterior nodes are the nodes that have the most chances of overlapping with other cluster, due to which it would be a better choice to make them learn more than the other nodes, so as to be able to include as much data points as possible near the cluster edges.

2. Noting what nodes belong to which cluster.

This is achieved by including another level of Nodes which act a place holder for each cluster. The edge structure constructed with GNG can be used to construct such clusters. Whenever an edge is deleted there is a chance of creation of 2 clusters from a single cluster. Similarly when an edge is created there is a chance of creation of a single cluster form 2 clusters. Noting such edge connections at each iteration, the cluster prototypes or placeholders can be dynamically rearranged during the iteration. We are be able to create an incremental merging and incremental splitting algorithms to check whether such merging or splitting occur during each iterations with minimal overhead on time.

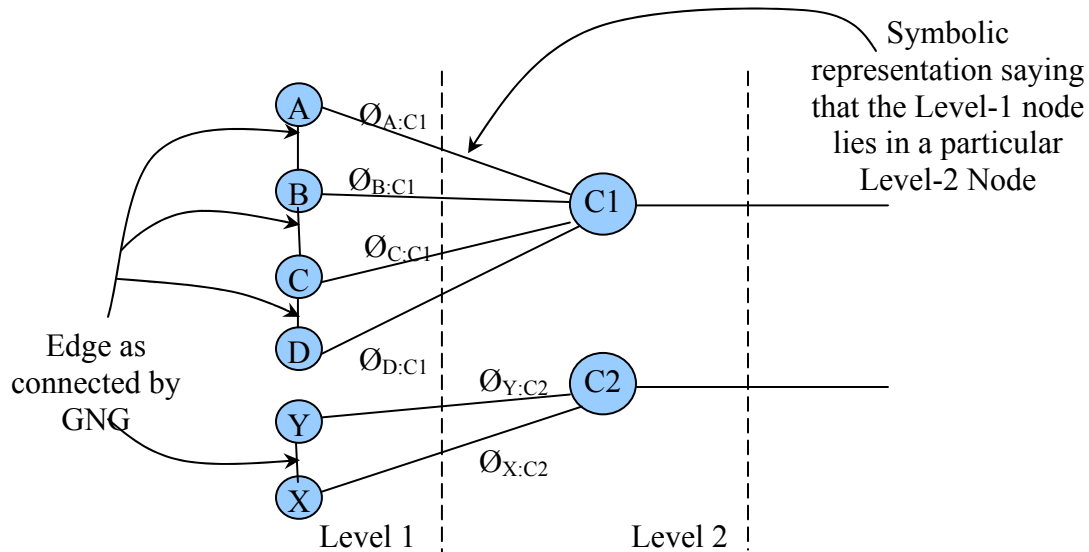
3. Incorporating Topology to ascertain the distribution of data in a cluster.

GNG creates the nodes and edge structure based on the occurrence of data (between nodes), thus indirectly representing the data through the nodes and edges. This topological structure is a connected undirected graph, and we propose later an algorithm that demonstrates usage of such topology to generate fuzzy membership for each such GNG generated node in a cluster.

## 2. Methodology

### 2.1 TFC System

The Network construction for TFC looks like the following:



**Figure 1 Numerical Value assignment between Level-1 and Level-2 Nodes.**

As shown in Figure 1, TFC consists of a 2-Layered structure, with the Level-1 nodes constructed by modified GNG algorithm. The Level-2 nodes are constructed on the basis of edge connection of the Level-1 nodes and thus serve as cluster prototype or placeholders.

The links between the Level-1 and Level-2 nodes are the fuzzy memberships of the Level-1 nodes into the Level-2 nodes. The idea is: for a datapoint to be tested, all nearest nodes in Level-1 are found and their corresponding fuzzy memberships (from the links from Level-1 to Level-2) calculated. The Level-1 node with maximum membership for the datapoint is considered to be the winner node and the corresponding Level-2 node is considered the winner cluster. Construction of the Level-1 node from the Modified GNG algorithm and subsequent generation of cluster prototypes in the Level-2 is done as following in the TFC algorithm:

## 2.2 TFC Algorithm (Additions to the GNG are mentioned in 'bold')

1. Initialize the set A to contain two units  $C_2$  and  $C_1$

$$A = \{C_1, C_2, \dots, C_n\}$$

with reference vectors chosen randomly according to distribution "x".

Initialize the connection set  $C, C \in A \times A$  to the empty set:

$$C = 0$$

**Initialize the Cluster set Cluster,  $Cluster \in N \times A, N \in n$  (n is the set of real numbers, and  $N \leq size(x)$**

$$\mathbf{Cluster} = 0$$

**Initialize 2 units in Level-1 as A and B and make individual clusters from it**

$$\mathbf{Cluster} = \mathbf{Cluster} \cup \{(1,A), (2,B)\}$$

*Implying that A lies in Cluster numbered 1 and B in Cluster numbered 2.*

2. Generate at random an input signal "x".
3. Determine the winner  $S_1$  and the second-nearest unit  $S_2$  ( $S_1 \neq S_2$ ) by

$$S_1 = \min_{c \in A} \|x - W_c\|$$

$$S_2 = \min_{c \in A} \|x - W_c\|, S_1 \neq S_2$$

4. If a connection between  $S_1$  and  $S_2$  does not exist already, create it:

$$C = C \cup \{(S_1, S_2)\}$$

Set the age of the connection  $S_1$  between and  $S_2$  to zero ("refresh" the edge):

$$\text{Age}(S_1, S_2) = 0$$

5. **Check if  $S_1$  and  $S_2$  are already in the same cluster. (Step 5: Incremental Merging Algorithm).**

*If  $\exists i \in n, \exists \text{Cluster}(i, S_1) \wedge \exists \text{Cluster}(i, S_2) = \text{true}$  that means there already lie in the same cluster and so do nothing*

*In all other conditions there is none so make a new cluster and delete the existing clusters*

- a. Find the cluster numbers in which  $S_1$  and  $S_2$  lie.*

*Let  $M$  be the Cluster number in which  $S_1$  lies, then*

$$\exists A : \text{Cluster}(M, S_1) = \text{true}$$

*Similarly  $N$  is the cluster number in which  $S_2$  lie.*

$$\exists B : \text{Cluster}(N, S_2) = \text{true}$$

- b. Generate 2 sets each having the Level-2 nodes from the Cluster numbered  $M$  and numbered  $N$ .*

$$\text{Cluster}M = \{\text{Cluster}(M, i) \mid i \in A\}$$

$$\text{Cluster} = \text{Cluster} - \text{Cluster}M$$

*And for Cluster numbered  $N$*

$$\text{Cluster}N = \{\text{Cluster}(N, i) \mid i \in A\}$$

$$\text{Cluster} = \text{Cluster} - \text{Cluster}N$$

- c. Merge Cluster $M$  and Cluster $N$  into a single set*

$$\text{ClusterNew} = \text{Cluster}M \cup \text{Cluster}N$$

- d. Choose a cluster number say  $P$  not already used in Cluster*

$$\exists P : P \in n, P \notin i$$

$$i = \{\forall j \ \& \ x \in A \mid \text{Cluster}(j, x) = \text{true}\}$$

*e. Make a new cluster with the cluster number P*

$$ClusterP = \{(i, P) \mid \forall i \in A, Cluster(i, x), x \in n\}$$

*f. Add the new cluster created to the original cluster set*

$$Cluster = Cluster \cup ClusterP$$

5. Add the squared distance between the input signal and the winner to a local error variable:

$$\Delta E_{s_1} = \|x - W_{s_1}\|^2$$

6. Adapt the reference vectors of the winner and its direct topological neighbors by fractions  $e_b$  and  $e_n$ , respectively, of the total distance to the input signal:

$$\Delta W_{s_1} = e_b (x - W_{s_1}) \cdot \frac{1}{FuzzyMembership(S_1)}$$

For the Neighbors as,

$$\Delta W_i = e_n (x - W_i) \cdot \frac{1}{FuzzyMembership(S_2)}$$

For a unit  $c$  we denote with  $N_c$  the set of its direct topological neighbors:

$$N_c = \{i \in A \mid (c, i) \in C\}$$

The set of direct topological neighbors from  $S_1$  are  $N_{s_1}$

**[Note: A special case arises when initially there are 2 clusters each containing  $S_1$  and  $S_2$  and they would be merged at step 4, 5. In this case it's alright to assume the initial fuzzy membership of these units in the single cluster they create to be 1 for both, which is also true if taken mathematically.]**

7. Increment the age of all edges emanating from  $S_1$ :

$$\text{Age}(S_1, i) = \text{Age}(S_1, i) + 1$$

8. Remove edges with an age larger than  $A_{\max}$ . If this results in units having no more emanating edges, remove those units as well.
9. Now check if the clusters split due to edge deletion.

***Use the Incremental Splitting Algorithm.***

10. If the number of input signals generated so far is an integer multiple of a parameter  $\lambda$ , insert a new unit as follows:

- a. Determine the unit  $q$  with the maximum accumulated error:

$$q = \max_{c \in A} E_c$$

- b. Determine among the neighbors of  $q$  the unit  $f$  with the maximum accumulated error:

$$f = \max_{c \in A, c \neq q} E_c$$

- c. Add a new unit  $r$  to the network and interpolate its reference vector from  $q$  and  $f$ .

$$A = A \cup \{r\}, W_r = (W_q + W_f) / 2$$

- d. Insert edges connecting the new unit  $r$  with units  $q$  and  $f$ , and remove the original edge between  $q$  and  $f$ :

$$C = C \cup \{(r, q), (r, f)\}$$

$$C = C - \{(q, f)\}$$

- e. ***Merge the new node into the cluster containing  $q$  and  $f$ .***

$$\exists i \in n, \text{Cluster}(i, q) \wedge \text{Cluster}(i, f) = \text{true}$$

$$\text{Cluster} = \text{Cluster} \cup \{(i, r)\}$$

***Use the Incremental Merging Algorithm.***

- f. Decrease the error variables of  $q$  and  $f$  by a fraction  $\alpha$ :

$$\Delta E_q = -\alpha E_q \quad \Delta E_f = -\alpha E_f$$

g. Interpolate the error variable of  $r$  from  $q$  and  $f$ :

$$E_r = (E_q + E_f) / 2$$

10. Decrease the error variables of all units:

$$\Delta E_c = -\beta E_c \quad \forall c \in A$$

**11. Now calculate fuzzy membership for each node in its corresponding cluster**

*Use Equation 1, to find the fuzzy Membership.*

11. If a stopping criterion (e.g., net size or some performance measure) is not yet fulfilled continue with step 2.

There are some algorithms (Incremental Merging & Incremental Splitting Algorithms-mentioned in the algorithm) and formula (Fuzzy Membership) mentioned in the above algorithm. The fuzzy membership for each Level-2 node in the Level-2 cluster is found as follows:

First a so-called ‘Reference value’ is found. This variable is called the ‘reference value’ as it is the ‘reference or relative’ value of each node compared to the other nodes in a cluster based on the edge structure. The idea behind this was that the edge structure and relative distance between nodes can be used to find the central node(s) of the cluster.

So for each node a Reference value is found based on the edge connection of the nodes and the relative distance between the nodes and neighbors as:

$$\Phi'_p = \frac{\sum_{n=1}^k \Phi'_n * R_n}{\sum_{n=1}^k \Phi'_n * \sum_{n=1}^k R_n} \quad (1)$$

Where  $\Phi'_p$  = Reference Value of the Node itself

$n$  (=1 to  $k$ ) = Set of  $k$ -Neighbors of the Node in consideration, including the node itself.

$\Phi'_n$  = Reference value for the nth element in the {Neighbors} U {Node} set

$R_n$  = Average Distance between Nodes and its neighbors.

The above equation is iterated till it converges for all nodes in a cluster. The iterative form of the equation is mentioned below. Once it converges, the fuzzy memberships for each node is calculated as

$$fuzzy_{ic}(\Phi_{ic}) = \frac{\min(reference_c)}{reference_{ic}} \quad (2)$$

Where  $fuzzy_{ic}$  or  $\Phi_{ic}$  is the fuzzy membership of the  $i^{th}$  node in cluster  $c$ , and  $\min(reference_c)$  is the minimum of the reference values in cluster  $c$ .

The whole process for finding the Fuzzy and Reference Values can be summed up as below:

### 2.3 Fuzzy and Reference Value finding Algorithm (Algorithm 1)

**Step1.** ReferenceValues = list of numbers between (0,1] corresponding to ReferenceValues for each node in the cluster

**Step2.** For each node in a cluster

$$ReferenceValue(i) = \frac{\sum_{n=1}^k ReferenceValue(n) \cdot R(n)}{\sum_{m=1}^k ReferenceValue(m) \sum_{p=1}^k R(p)}$$

Stop if  $ReferenceValue(i)_{t-1} - ReferenceValue(i)_t \leq \text{some small value}$

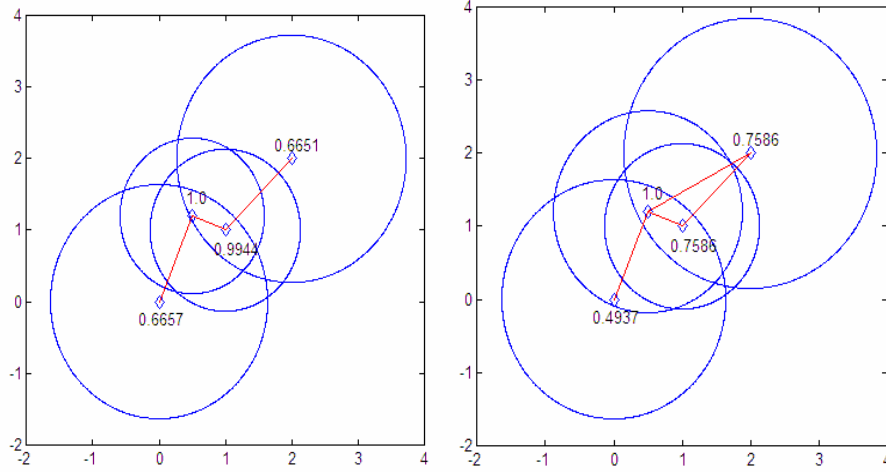
**Step3.** Now normalize in relation with the lowest ReferenceValue

$$FuzzyMembership(i) = \frac{\min(ReferenceValue)}{ReferenceValue(i)}, \forall 1 < i < size(ReferenceValue)$$

Stop

[Note:  $ReferenceValue(i)_{t-1}$  is the reference value at instant  $t-1$  and  $ReferenceValue(i)_t$  at instant  $t$ ]

Note that the derivation of the Fuzzy Membership from the Reference Value over here is done in a manner so that maximization of a single or multiple node(s) to 1 is done, though there can be alternate ways to set the fuzzy value of the node with highest reference value.



**Figure 2 Effects of the Edge connection**

The following figure-2 shows the effect of an edge creation, nodes are represented in diamonds with the edges in red and the average distance between nodes by blue circles. The mentioned values near each node are the corresponding fuzzy value associated. As can be seen from the figure, the edge creation causes the fuzzy membership increase for the nodes concerned and decrease for the nodes that are not affected.

## 2.4 Proof of Fuzzy and Reference Value finding Algorithm

From Golub and Van Loan [], the Symmetric Power Iterations can be stated as:

Given a unit 2-norm  $q^{(0)} \in \mathbb{R}^n$ , the power method produces a sequence of vectors  $q^{(k)}$  as follows:

```

for k=1,2,...
     $z^{(k)} = Aq^{(k-1)}$ 
     $q^{(k)} = z^{(k)} / \|z^{(k)}\|_2$ 
     $\lambda^{(k)} = [q^{(k)}]^T Aq^{(k)}$ 
end

```

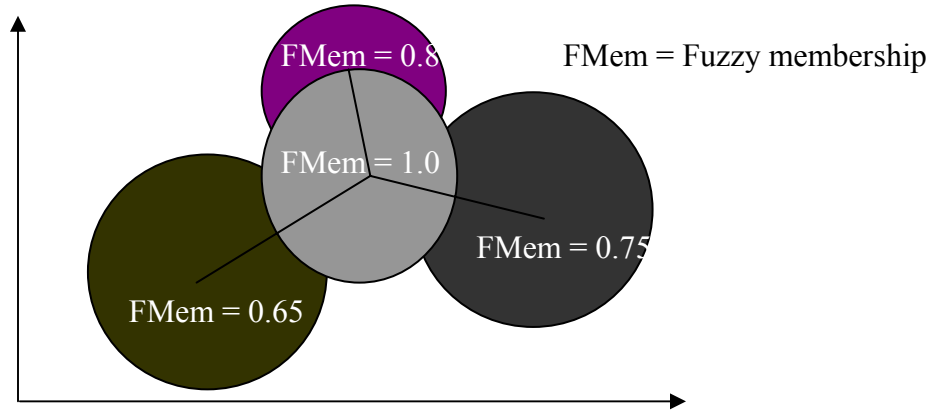
If  $q^{(0)}$  is not “deficient” and  $A$ 's eigen-value of maximum modulus is unique, the  $q^{(k)}$  converges to an eigenvector. The reader might observe that the update done to get the Reference Value is a variation of the Power Iteration. The ‘ $A$ ’ matrix mentioned in Power Iteration is analogous to the Edge Matrix  $C$ , as the update will depend directly on the node and its neighbor connection (which is indirectly derived from  $C$ ).  $q^{(k)}$  vector matrix is the Reference Value vector  $[\Phi'_1, \Phi'_2, \dots, \Phi'_n]$  (for  $n$ -nodes) derived at each iteration.

The whole process of finding the Fuzzy and Reference Value is incremental in nature, which reduces time on the computation whenever a new node/edge is created, or an older edge/node deleted or movement of a node occurs over iteration.

The fuzzy membership found for each node is used to influence the learning rate at each of the nodes and for testing. As seen in the step 6 of the TFC algorithm, the lesser the learning rate, the more a node is encouraged to learn. Thus the learning of the nodes is at the minimum a baseline learning rate (which will be the learning rate for a node of fuzzy membership 1). On comparing to the  $[0, \text{baseline learning}]$  rate in SOM, the effects of using a  $[\text{baseline}, \text{maximum}]$  learning rate would be that the nodes can always learn more pattern. And the edge matrix will, rather indirectly, influence the learning rate. This will also make learning a rather non-monotonic function compared to monotonic learning in SOM and Neural Gas.

## **2.5 Testing of TFC**

As I discussed earlier, the radius is calculated as an average distance between neighbors of a node. This is a region which I will call as Area of Influence, will cause fuzzy regions represented as hyperspheres, which are also overlapping in nature.



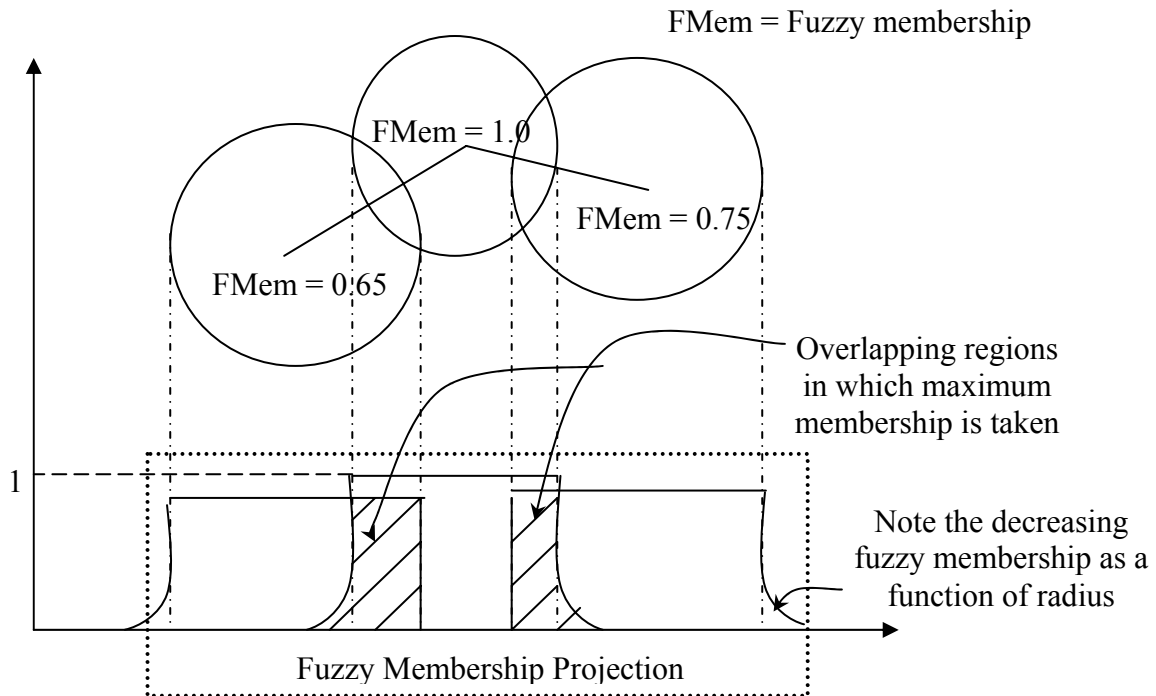
**Figure 3** Overlapping Fuzzy Hyperspheres.

As seen the hyperspheres with more fuzzy membership would dominate the overlapping area it shares with the other hyperspheres. Such hyperspheres can also be projected directly on the axes for construction of Takagi-Sugeno-Kang (TSK) based Fuzzy memberships functions.

Though this testing approach works well, it is still insufficient to find discover the outlier data, which lies out of the hyperspheres. The reason for it can be cited to that the nodes tend to suppress specialization and encourage local generalization. The hyperspheres though being valid prototypes for a sub-cluster inside a cluster, do not have hyperellipsoidal properties and suppress learning of such outliers, causing some outliers to be missed.

So for the discovery of such outliers, it is important to also consider the error associated with each node. The error so associated can be used to increase the size of the hyperspheres. If I mention the hypersphere with radius as ‘radius’ as ‘Radius region’ and the with radius equal to ‘radius+error’ as ‘Radius+error region’, than the fuzzy membership in the ‘Radius+error region’ should not be equal to the membership in ‘Radius Region’. It should rather be decreasing in regards with the distance from the cluster center. Such a value for a datapoint in the ‘Radius+error Region’ can be obtained by multiplying fuzzy membership (of the ‘Radius

region') with a monotonically decreasing function dependent on the distance of the datapoint from the center of the hypersphere. This can be summed up well in the next figure.



**Figure 4** Fuzzy Membership Projection with Smoothed Out Error.

**Pseudo code snippet for finding the Fuzzy membership for a datapoint :**

Functions used:

*Distance(X,Y) = gives the Euclidean distance between X and Y vectors*

*Radius(X) = gives Radius associated with node X*

*Error(X) = gives Error associated with node X.*

If  $\text{Distance}(\text{Node1 Center, Input Data}) \leq \text{Radius}(\text{Node1})$

Membership of Input Data = Membership of node 1

Else

If  $\text{Distance}(\text{Node 1 Center, Input Data}) \leq \text{Radius}(\text{Node1}) + \sqrt{\text{Error}(\text{Node1})}$

Membership of Input Data = Membership of node 1 x  $e^{\left(1 - \frac{\text{Distance}(X, \text{Center})}{\text{Radius} + \sqrt{\text{Error}}}\right)}$

Else

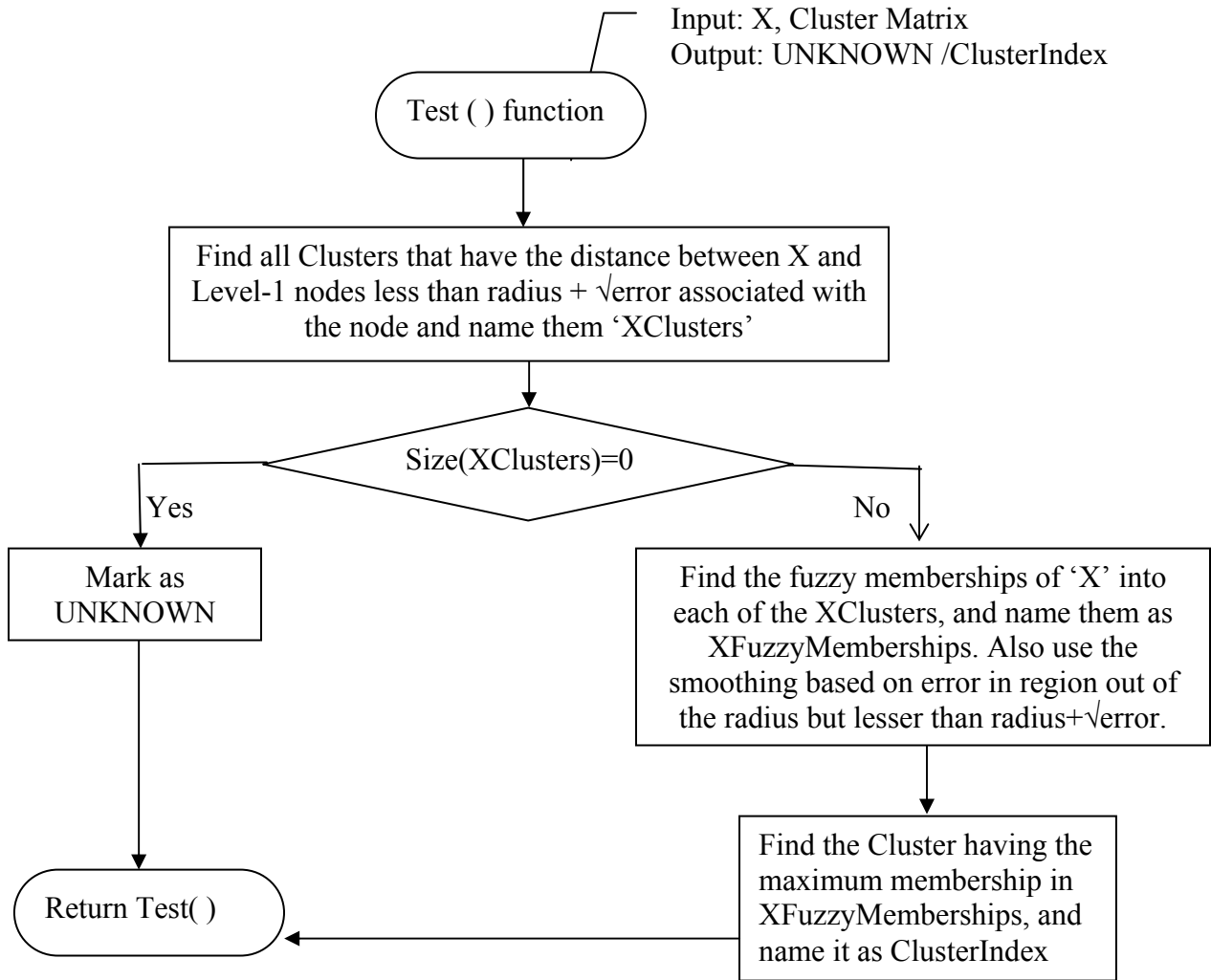
Membership of Input Data = 0

End

End

The earlier approach of using just the ‘Radius region’ is very stringent and is inefficient when it comes to the outlier data. The reason for which can be attributed to the very nature of the GNG method and our calculation of the radius based on the neighbor distance. To allow the calculation for the outliers, it’s imperative to choose a smoother function for the max-membership that we calculate. We would now present a method to find those outliers or Unrepresentable data points as shown in figure 5.

The flowchart shown in figure 5, is the testing routine that is employed to calculate the fuzzy membership and thereby deriving classification results. The above mentioned test() routine can classify almost all the outliers, though there might lie some outliers that might not get detected.



**Figure 5** Flowchart for the Testing Routine.

### 3. Results and Discussions

#### 3.1 IRIS Data

IRIS Dataset is a well known dataset used in pattern classification and was introduced by R. A. Fisher as an example for discriminant analysis. The dataset records four characteristics (sepal width, sepal length, petal width and petal length) of three species of Iris flower. The data used from Fisher, are the measurements of the sepal length and width, and petal length and width

in centimeters of fifty plants for each of three types of Iris flowers; Iris Setosa, Iris Versicolor and Iris Virginica.

Iris Setosa is completely classifiable, whereas there is an overlap between Iris Versicolor and Iris Virginica. Multiple runs were done using FCM, Gath-Geva and TFC to find a good classification rate (and so the best parameters) from each of the algorithm. FCM and Gath-Geva were made to cluster the dataset directly, whereas in case of TFC, the topology was first constructed and then tested again with the dataset. The results are tabulated in the following table 1. The table shows the misclassification or wrongly classified datapoint rate for each cluster.

**Table 1** Misclassification rate for IRIS data using Various Algorithms.

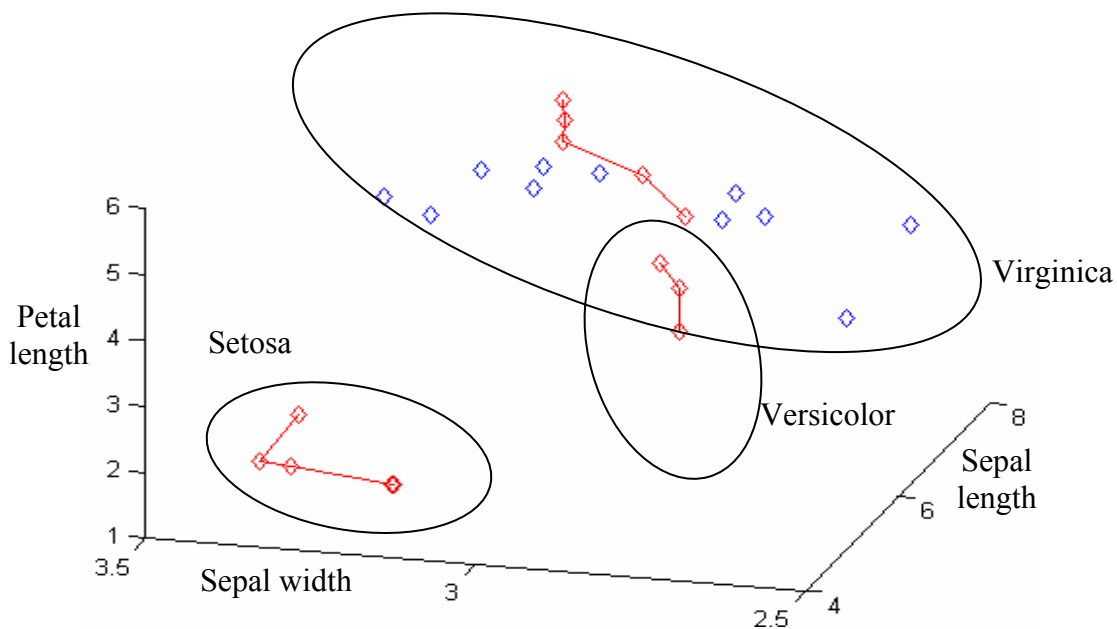
Misclassified data for IRIS dataset out of 150							
Classes	FCM			Gath-Geva			TFC
	m=1.5	m=2	m=3.5	m=1.5	m=2	m=3.5	$\lambda = 10, e_b = 0.04, e_n = 0.0006,$ $\alpha = 0.5, \beta = 0.0005, a_{\max} = 12$
Iris Setosa	0	0	0	0	0	0	0
Iris Versicolor	3	3	3	3	3	4	10
Iris Virginica	12	13	11	12	12	9	1
Total	15	16	14	15	15	13	11

From the Table 1 it's observable that TFC performs well for the IRIS data, with a lower misclassification rate than FCM and Gath Geva. Please note that unlike the other algorithms, TFC has performed only a single pass of the dataset. Now, the misclassified data in my algorithm is mostly in classifying Iris Virginica as the Iris Versicolor flower. This can be attributed to the hypersphere construction done by TFC. The hypersphere construction though representative enough is not able to rightly classify the outliers. These would be well classified if we were using a hyperellipsoidal construction of Level-1 node regions.

Figure 6 shows the plot of the 3 parameters (sepal width [2,4.4], sepal length [4.3,7.9] and petal length [1,6.9]). The fourth parameter, of petal width, though considered in the

experiment has not been shown in the figure 6. The nodes of the TFC are represented by red diamonds and connection in red edges. The datapoints that are misclassified are represented by blue diamonds.

This misclassification is caused due because of the peculiar nature of the position of the Iris Virgnica clusters' Level-1 nodes and its construction of hyperspheres, which is not able to distinct the area, because of the hyperellipsoidal nature of the misclassified datapoints.



**Figure 6** Misclassified Data points (Virginica as Versicolor).

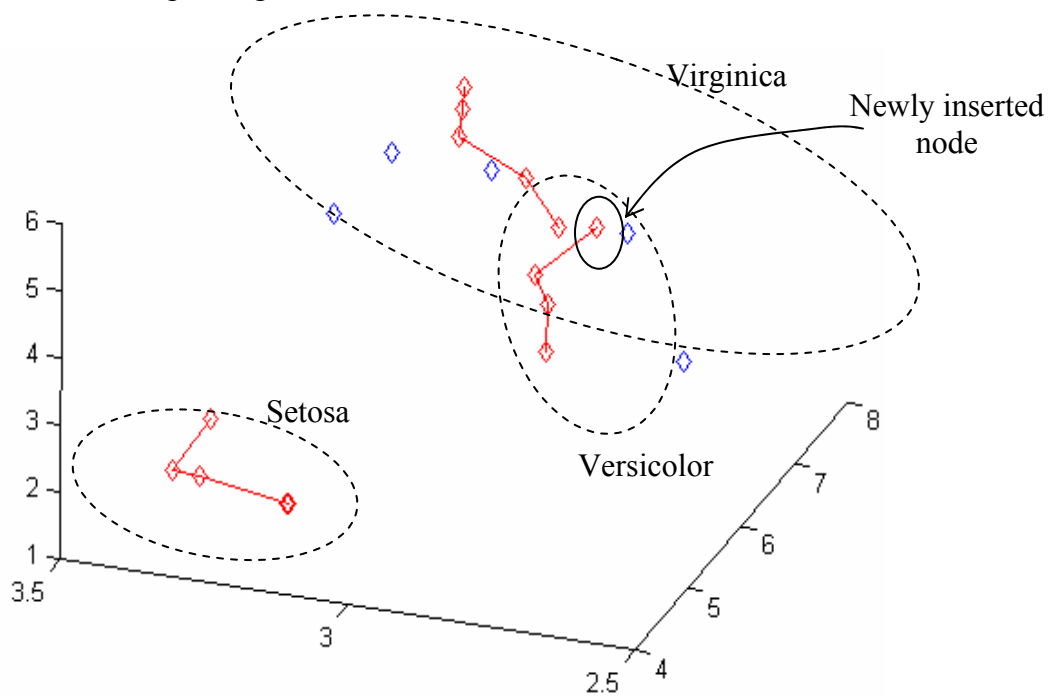
To get better results from TFC, we used a variant of the TFC called the Aided-TFC (ATFC) which added a new node to better quantify the misclassified datapoints. The idea behind ATFC was to add a node to the correct cluster based on the position of the misclassified point. With each addition of a new node the misclassification rate is calculated till a better result is obtained. As seen in Table 1, unsupervised TFC was not able to classify 10 datapoints from Iris Virginica and 1 datapoint from Iris Versicolor, ATFC was fed those misclassified points.

**Table 2** Classification rate for IRIS data using TFC and ATFC.

Classes	TFC		ATFC(for Iris Virginica only)	
	Misclassified	Classification	Misclassified	Classification

Iris Setosa	0	100%	0	100%
Iris Versicolor	10	80%	4	92%
Iris Virginica	1	98%	1	98%
Total	11	92.7%	5	96.7%

As seen by using ATFC is able to reduce the number of misclassification by 5 misclassifications. The adaptive insertion of a node at an appropriate place is the reason for the classification. The best possible way to understand where the misclassification has reduced is by observing the next Figure 7. Please note that the same parameters as used in the Figure 6 about the dataset were used to plot Figure 7.



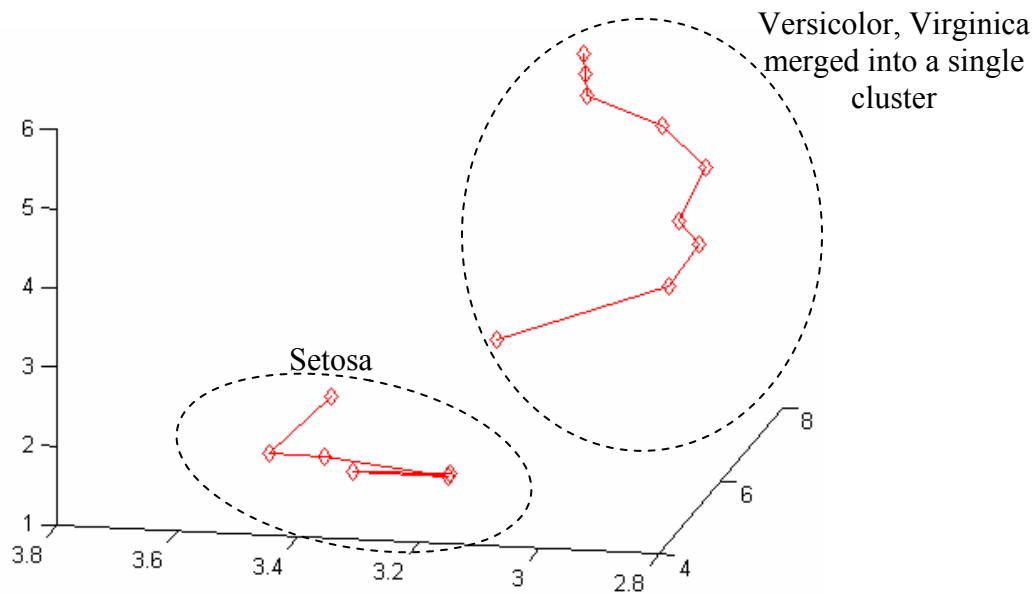
**Figure 7** Misclassified Data points by ATFC.

It's reasonable to assume that in general a heuristic estimation has to be done for including misclassified values with the ATFC algorithm.

Now the parameters  $\lambda = 10, e_b = 0.04, e_n = 0.0006, \alpha = 0.5, \beta = 0.0005, a_{\max} = 12$ , do create a better partition by creating 3 clusters for the IRIS dataset, but this is not the case every time with all the datasets. Due to the complex relations between the parameters, such a best case scenario is

somewhat rare with many datasets. In such case the experts aid, for merging and splitting the clusters would prove to be very useful.

Let's first see the output generated when the parameters chosen are  $\lambda = 10, e_b = 0.04, e_n = 0.0006, \alpha = 0.5, \beta = 0.0005, a_{\max} = 14$  in Figure 8. As seen from the Figure 8, the only change from the earlier parameters is increasing  $a_{\max}$  to 14 from an original value of 12. Due to this more nodes, the nodes that might have been deleted with a lower  $a_{\max}$  have survived, but also causing the Iris Versicolor and Iris Virginica clusters to merge. Supposing the expert/user knows about this anomaly, his feedback can be taken for each node and the cluster nodes (Level-2 Nodes) can be created or deleted. In this case there is a need to split the cluster and so create a new Level-2 node.



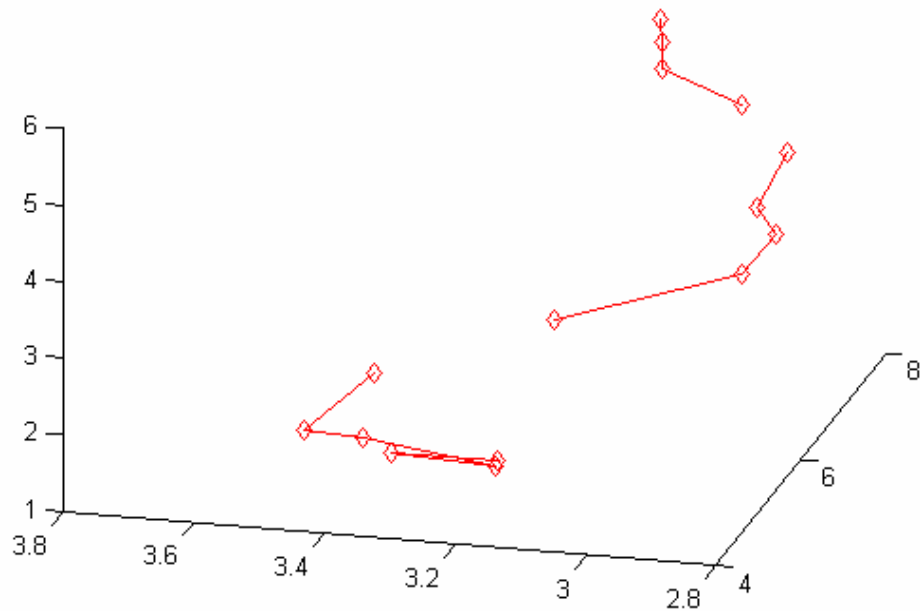
**Figure 8** Only 2 Clusters created for IRIS Dataset by TFC.

The expert is asked about the opinions on where the nodes and the data tabulated in the Table 3 given next. This variation of splitting and merging is added to the ATFC.

**Table 3** Node Position from TFC and Expert Feedback.

Node Number	Vector Positions				Cluster Number	Expert's Numbering Iris Setosa =1 Iris Virginica=2 Iris Versicolor=3
1	4.9931	3.3601	1.3828	0.1958	1	1
2	4.9626	3.1956	1.4246	0.2139	1	1
3	7.0123	3.0779	5.8370	2.0229	2	2
4	4.8475	3.1933	1.4349	0.2160	1	1
5	5.0667	3.4112	1.5114	0.2768	1	1
6	5.1531	3.5049	1.4796	0.2586	1	1
7	5.4213	3.4164	2.0999	0.4930	1	1
8	5.5824	3.1502	2.8594	0.8059	2	3
9	5.7497	2.8733	3.5546	1.0405	2	3
10	5.7807	2.8242	4.0769	1.2609	2	3
11	5.9938	2.8671	4.2543	1.3179	2	3
12	6.1525	2.8299	4.8814	1.6778	2	3
13	6.4510	2.9192	5.2274	1.7856	2	2
14	6.6609	3.0542	5.4676	2.0179	2	2
15	6.8517	3.0649	5.6623	2.0187	2	2

The expert aids the algorithm by showing that the Nodes numbered 8-12 cannot be considered to be in same class as the other nodes in the class and so makes a new class for them.



**Figure 9** ATFC creates 3 Clusters for IRIS Dataset.

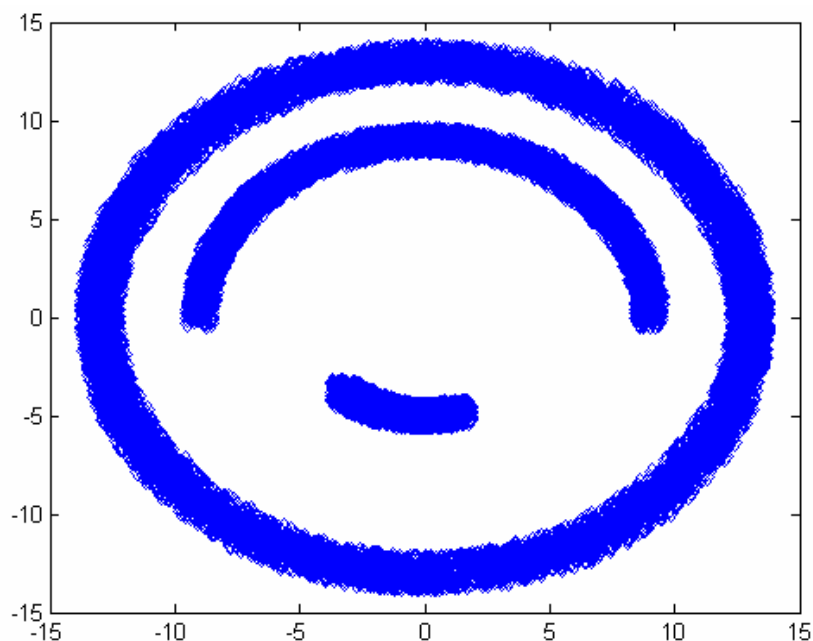
This would differentiate the Iris Versicolor from the Iris Virginica as shown in the preceding Figure 9. Due to the expert's opinion, the network has come to know the existence of all the clusters. The above example is strictly to show how parametric evaluation can be helped by inclusion of an expert's feedback.

Please do note that the above cluster breaking method can also be used to merge clusters. From this dataset result we can interpret the following results about TFC and ATFC.

1. Parametric evaluations are needed to set the correct parameters.
2. 'FLAT' Fuzzy Area of Influence does give good results.
3. ATFC can be used to garner the expert's comments and better TFC.

### 3.2 Simulated Data

The simulated dataset is a 2 Dimensional dataset, to show the power of representation imparted by Topology in TFC. The Simulated dataset is a set of 3 clusters, 1 semicircle, 1 arc and 1 circular distribution. The dataset representation in 2 Dimensional space is shown in the next Figure.



**Figure 10** Simulated Dataset with 3 Clusters.

A random distribution with a width of 1.5 for the full circle and a width of 1 for the semi circle and smaller chord was taken. The distance between the circle and the larger semicircle is 1.5. The distribution parameters are as follows:

**Table 4** Simulated Dataset Distribution Parameters.

Distribution	Center	Radius	Variation			in Radians		Distribution Size
			Width	Min	Max	Start	End	
Circle	(0,0)	13	1.5	11.5	14.5	0	$2\pi$	10000
Semicircle	(0,0)	9	1	8	10	0	$\pi$	5000
Arc	(0,0)	5	1	4	6	4	5	3000

Such a type of distribution was generated and then a test dataset was prepared. The test dataset had the parameters as shown next.

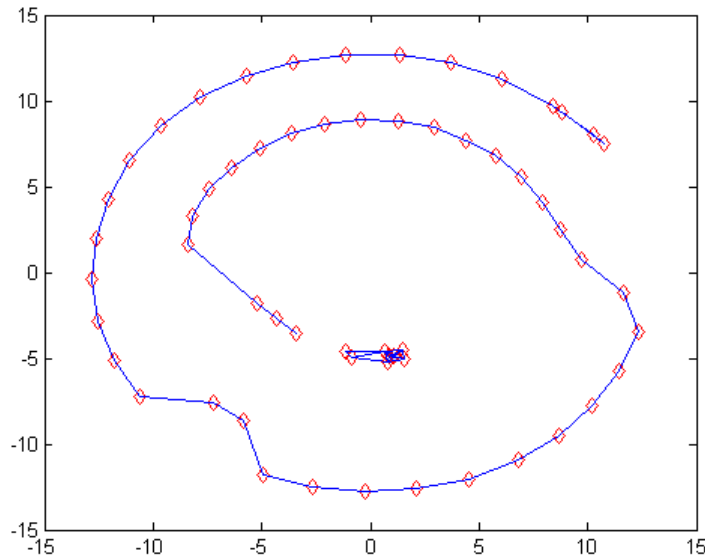
**Table 5** Test Dataset Distribution Parameters.

Distribution	Center	Radius	Variation			in Radians		Distribution Size
			Width	Start	End	Star	End	
Circle	(0,0)	13	1	11	15	0	$2\pi$	10
Semicircle	(0,0)	9	1	7.5	10.5	0	$\pi$	10
Arc	(0,0)	5	1	3.5	6.5	4	5	10

The parameters as set in ATFC are as follows:

$$\lambda = 300, e_b = 0.04, e_n = 0.0006, \alpha = 0.5, \beta = 0.0005, a_{\max} = 350, \text{Error Fraction} = 1.$$

The node positions and the edge connections are shown in the Figure 11.



**Figure 11** Simulated Dataset Topological Node Positions.

As seen the clusters are represented well, but merged. So it's important to break the clusters into more parts. A supervised input from the expert, would allow ATFC to break the clusters.

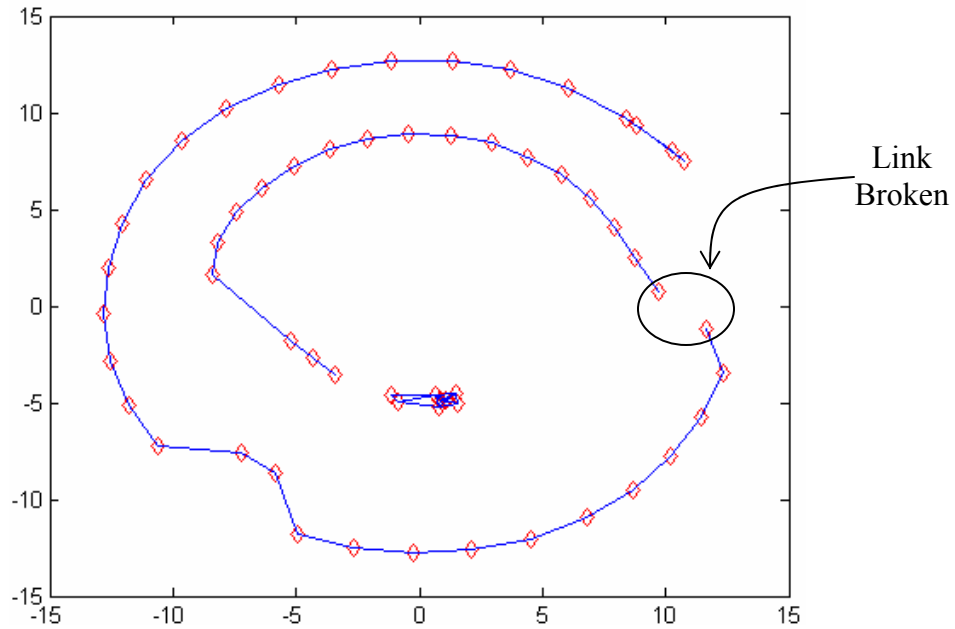
A total of 59 nodes have been generated. From the Node positions, the expert is able to tell that Node 34 to Node 51 does not belong to the same clusters, as a result of which they should be split from the cluster they belong to. It's not a requirement that the expert needs to examine the topology in terms of a 2 Dimensional or a 3 Dimensional graph, but it always helps to see it qualitatively. For higher dimension a tabular form as shown in Table 6 next, would be helpful, to note an expert's comments.

**Table 6** Simulated Dataset Learnt Node Position.

Node	Position X	Position Y	Cluster	
			ATFC	Expert
1	10.2582	8.0497	1	1
2	10.7786	7.535	1	1
3	1.5765	-5.0033	2	2
4	8.7875	9.3583	1	1
5	8.3885	9.6734	1	1
6	6.055	11.2485	1	1
7	3.7255	12.2506	1	1
8	1.3429	12.7109	1	1
9	-1.1125	12.6754	1	1
10	-3.5423	12.2709	1	1
11	-5.7342	11.4738	1	1
12	-7.8635	10.1757	1	1
13	-9.629	8.5295	1	1
14	-11.0848	6.542	1	1
15	-12.0458	4.2438	1	1
16	-12.6135	1.9775	1	1
17	-12.791	-0.434	1	1
18	-12.5274	-2.8771	1	1
19	-11.7703	-5.147	1	1
20	-10.585	-7.2445	1	1
21	-7.2139	-7.557	1	1
22	-5.8537	-8.623	1	1
23	-4.9192	-11.7842	1	1
24	-2.6732	-12.4653	1	1
25	-0.2543	-12.7644	1	1
26	2.0824	-12.6035	1	1
27	4.5217	-12.0305	1	1
28	6.7833	-10.961	1	1
29	8.6795	-9.507	1	1
30	10.1896	-7.7569	1	1

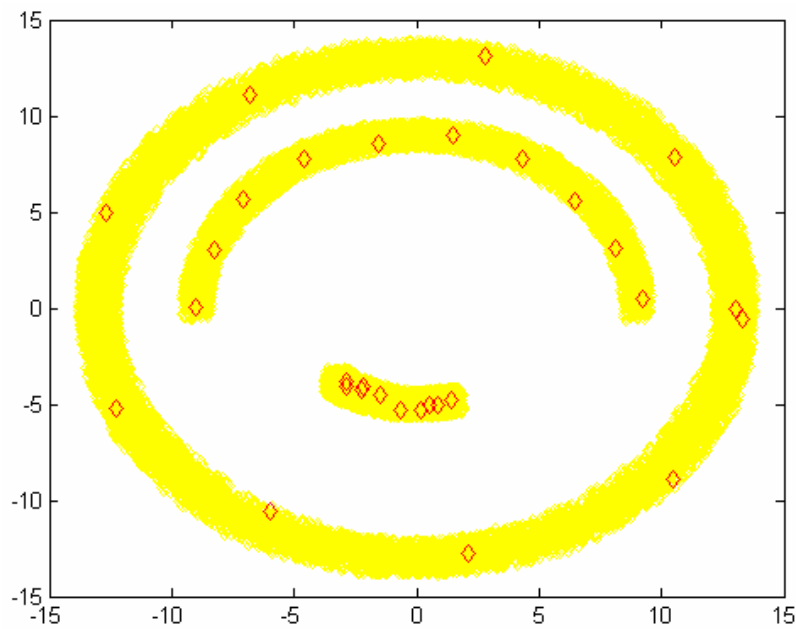
Node	Position X	Position Y	Cluster	
			ATFC	Expert
31	11.4584	-5.7416	1	1
32	12.3665	-3.4851	1	1
33	11.6753	-1.2015	1	1
34	9.7271	0.7566	1	3
35	8.7736	2.5237	1	3
36	7.9152	4.1021	1	3
37	6.9357	5.5277	1	3
38	5.7936	6.7572	1	3
39	4.4128	7.7138	1	3
40	2.9054	8.4262	1	3
41	1.2473	8.8105	1	3
42	-0.4473	8.8788	1	3
43	-2.089	8.6267	1	3
44	-3.6582	8.0833	1	3
45	-5.0852	7.2286	1	3
46	-6.3699	6.1385	1	3
47	-7.4041	4.8447	1	3
48	-8.1802	3.3244	1	3
49	-8.405	1.6199	1	3
50	-5.2101	-1.7774	1	3
51	-3.4284	-3.5355	1	3
52	-0.891	-4.9615	2	2
53	-1.1669	-4.5784	2	2
54	0.6341	-4.6132	2	2
55	0.7947	-5.2147	2	2
56	1.4546	-4.5012	2	2
57	-4.3193	-2.6564	1	3
58	0.8662	-4.9093	2	2
59	1.0713	-4.8665	2	2

After incorporating the expert's feedback in ATFC the corresponding topology is generated as shown in the next Figure 5.13.



**Figure 12** Simulated Dataset Topological Node Positions with Breakage.

A set of 30 test data point were then presented to ATFC. The topological position of the data points are as follows:



**Figure 13** Test Data Vector Positions Represented as 'diamonds'.

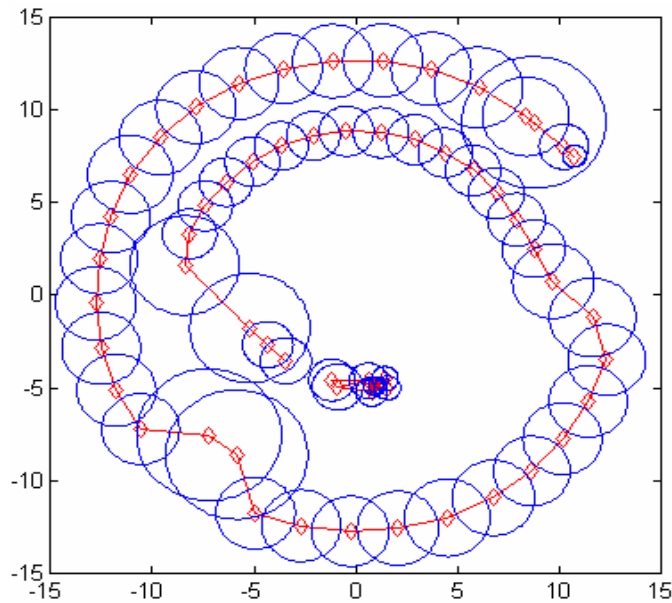
For the set of 30 Test datapoints, the following Misclassification rates were obtained, and as seen there were 2 datapoints that were wrongly classified in another class.

**Table 7** Test Data Classification (Simulated Dataset).

Class	Real Class	Calculated Classes from ATFC		
		Total Classification	Misclassified	Not Classified
Circle	10	10	0	0
Semicircle	10	12	2	0
Arc	10	8	0	2

The reason for the misclassification can be cited to the nodes still not rightly classified by the expert. The nodes that should be in Arc cluster are classified in to Semicircle cluster. Thus it's important to get a reliable expert for classification.

With a reliable expert opinion that Node 52 should also be included into the Cluster 2, the misclassification rate drops to 0. Let me also show the Fuzzy Area of Influence for each node vector in Figure 14.

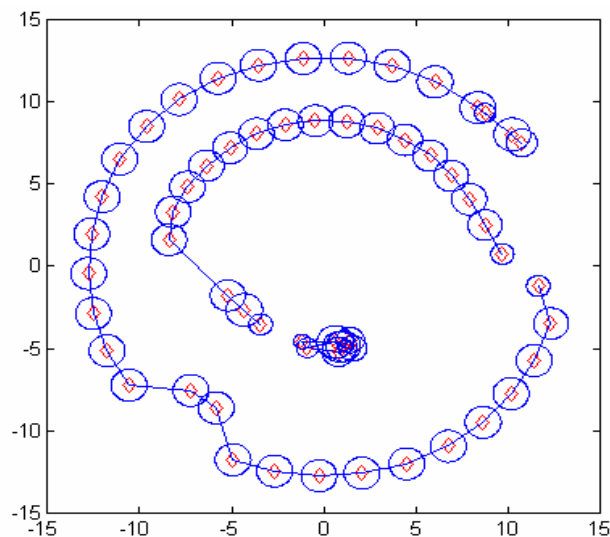


**Figure 14** Fuzzy Area of Influence.

As seen in the preceding Figure 5.15, the Area's of Influence are not the same for all the nodes. It's interesting to note that a majority of the nodes are separated with almost the same

distance due to the occurrence of a uniformly random distribution. Though there are some nodes with a bigger Area of Influence, which is due to the uncertainty in that area, which can be cited to nearby cluster influences. It's reasonable to assume that when enough data points are present, the Area of Influence will become the same for all the data points, and the network would more or less converge to fix values, though that is not a necessary case required for Testing the network.

Now let me present the Fuzzy Value that are shown at each point in terms of drawing circles around the point.



**Figure 15** Fuzzy Memberships represented by Circles.

In Figure 15, the Fuzzy Memberships for each node is represented by drawing a circle around the node. The higher, the fuzzy membership, and the bigger the circles are in size. As can be seen from the visual cue, the exterior nodes do have smaller circles compared to the other node which according to our interpretation is correct, as those nodes are the exterior of the clusters. So topologically the representation is what is desired.

Let's now see where ATFC (and TFC) lies in comparison with other Clustering Algorithms on the basis of topology. It's not correct to compare ATFC (or TFC) against standard

algorithms like Gath Geva or Fuzzy C-Means as the latter algorithms are good only for finding out hyperellipsoidal shapes, and not shapes like in the shape of a Torus or a Semicircular arc. The domain of ATFC and TFC becomes more than detection of simpler shapes, due to the power imparted by the topological construction, and it can compete with Fuzzy C-Means, and Fuzzy C-Shells at the same time and has advantages over both of them in terms of time of execution. We can sum up the advantages of TFC and ATFC in the following areas:

1. It is very good in estimating abnormal shapes (shapes like semicircular arc and torus shape).
2. ATFC can be used to further enhance the power of TFC's topology construction by including expert comments into the algorithm.
3. Using the node vectors as a placeholder for the expert's comments, the expert need to just point out groups/cluster in which those nodes should be in.
  - a. This would also allow an expert to comment in a more than 3 Dimensional topology.
  - b. This would also reduce the number of data points for which the Expert is asked to comment on.

This is due to the number of nodes is much lesser than the number of learnt data points.
4. As Topology construction is shape independent, ATFC (and TFC) is a viable algorithm for application to cluster non-elliptical shaped clusters, and theoretically any shaped cluster.

From our experiments, we found that TFC (and ATFC) can be used for a dataset if it follows the following criteria.

1. The dataset is very large in size.
2. The learning needs Adaptiveness like cluster creation, deletion and merging.
3. The dataset has clusters of any arbitrary size.

We do recommend the algorithm (ATFC, TFC) in its original state for different factors as follows:

**Table 8** Choice of ATFC in Different Circumstances.

Factors		Choice
1	Execution time	Yes, very good
2	Dataset Size-Large (Num. of Clusters $\lll$ Size of dataset)	Yes, very good
3	Dataset Size-Small (Num. of Clusters $\approx$ Size of dataset)	Yes, but better algorithms are available
4	Topology Shape	Yes, very good
5	Adaptiveness of Correcting Misclassification	Yes, but modification/ tweaking node radius might be required
6	Accuracy of Classification	Yes, but for medium to high accuracy. It's not always the best performing.
7	Adaptiveness of Learning new data	Yes, very good

Due to the linearized time for calculation a very large sample size can be used to calculate the Level-1 node centers without worrying about time factor, and still being adaptive in learning newer patterns. This makes it very much viable when we are using in a situation when newer clusters are presented to the system without the priori knowledge of the occurrence of the clusters. Thus the adaptive nature, linearized time factor and independence from shape makes it a potential algorithm for the domain of online clustering.

## References

1. Bezdek, J.C., Keller, J, Krishnapuram, R. & Pal, N. R. (1999). Fuzzy Models and Algorithms for Pattern Recognition and Image Processing.
2. J. B. MacQueen (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 281-297.
3. Hathaway, R.J. & Bezdek, J. C. (1986). On the asymptotic properties of fuzzy c-means cluster prototypes as estimators of mixture subpopulation centers, Communications in Statistics (A), 15(2), 505-513.
4. Gath, I. & Geva, A.B. (1989). Unsupervised Optimal Fuzzy Clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(7), 773-781.
5. Abonyi J., Babuska R., & Szeifert F. (2002). Modified Gath-Geva Fuzzy Clustering for Identification of Takagi-Sugeno Fuzzy Models. IEEE Transactions On Systems, Man, And Cybernetics—Part B: Cybernetics, 32(5), 612-621.
6. Fritzke, B. (1997). [Document posted on Web site Ruhr-Universität Bochum]. Retrieved Oct,2005 from World Wide Web:  
<http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/JavaPaper/node16.html>
7. Martinetz, T. M. & Schulten, K. J. (1991). A “neural-gas” network learns topologies. Artificial Neural Networks, Amsterdam, Elsevier, 397-402.
8. Haykins, S. (1993). Neural Networks: A Comprehensive Foundation (2<sup>nd</sup> ed.). Prentice Hall.
9. Fritzke, B. (1996). Automatic construction of radial basis function networks with the growing neural gas model and its relevance for fuzzy logic. Proceedings of the 1996 ACM symposium on Applied Computing, 624-627.
10. Yen, J. & Langari, R. (1998). Fuzzy Logic: Intelligence, Control and Information (1<sup>st</sup> ed.). Prentice Hall.
11. Jang, J.S.R. & Sun, C.T. (1993). Functional equivalence between radial basis function networks and fuzzy inference systems. IEEE Trans. on Neural Networks, 4(1), 156-159.
12. Fritzke, B. (1994). Growing Cell structures. Neural Networks, 7(9), 1141-1160.

13. Fritzke, B. (1997). [Document posted on Web site Ruhr-Universität Bochum]. Retrieved Oct,2005 from World Wide Web:  
<http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/JavaPaper/node19.html>
14. Fritzke, B. (1994) Growing cell structures - a self-organizing network for unsupervised and supervised learning. Neural Networks, 7(9), 1460.