Fall 2005

Online clustering with single-pass topology based fuzzy clustering algorithm

Abhishek Jaiantilal
New Jersey Institute of Technology

Follow this and additional works at: https://digitalcommons.njit.edu/theses
Part of the Computer Engineering Commons

Recommended Citation

This Thesis is brought to you for free and open access by the Theses and Dissertations at Digital Commons @ NJIT. It has been accepted for inclusion in Theses by an authorized administrator of Digital Commons @ NJIT. For more information, please contact digitalcommons@njit.edu.
Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement.

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation.
The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.
ABSTRACT

ONLINE CLUSTERING WITH SINGLE-PASS TOPOLOGY BASED FUZZY CLUSTERING ALGORITHM

by
Abhishek Jaiantilal

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 that efficiently adapts to data distribution and evaluation is proposed. This method, Topology-Based Fuzzy Clustering (TFC), uses the topology of data to discover clusters. 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. This also gives a fuzzy estimation of data distribution within each cluster. The computational burden for TFC is proportional to the size of the initial data set and increases linearly with the addition of new data.

As TFC is based on GNG, it is an unsupervised algorithm. A supervised learning method is proposed that can be used in conjunction with TFC, to increases its accuracy with minimum computational burden. This adaptive algorithm is called the Adaptive Topology-Based Fuzzy Clustering (ATFC). In this study, the performance of ATFC and TFC is also evaluated against standard datasets.
ONLINE CLUSTERING WITH SINGLE-PASS TOPOLOGY BASED FUZZY CLUSTERING ALGORITHM

by
Abhishek Jaiantilal

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Engineering

Department of Electrical and Computer Engineering

January 2006
ONLINE CLUSTERING WITH SINGLE-PASS TOPOLOGY-BASED FUZZY CLUSTERING ALGORITHM

Abhishek Jaiantilal

Dr. Atam P. Dhawan, Thesis Advisor
Professor of Electrical and Computer Engineering, NJIT

Dr. Constantine Manikopoulos, Committee Member
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Sui-hoi E. Hou, Committee Member
Associate Professor of Electrical and Computer Engineering, NJIT
BIOGRAPHICAL SKETCH

Author: Abhishek Jaiantilal
Degree: Master of Science
Date: January 2006

Undergraduate and Graduate Education:

Master of Science in Computer Engineering,
New Jersey Institute of Technology, Newark, NJ, 2005

Bachelor of Engineering in Computer Engineering,
Dharamsinh Desai Institute of Technology, India, 2003

Major: Computer Engineering
To my parents and sisters
ACKNOWLEDGMENT

The current state of the thesis would have never been possible, were not for the constant urging of my thesis advisor, Dr. Atam P. Dhawan, to try my best. I will always remember every time I showed him a small modification to an existing algorithm, I would always be asked the question “What is new in it?” This question did serve as the main motivator to design my own algorithm and subsequently develop the whole thesis. I am extremely thankful to him for his special motivation, constant urging, and best of all; the idea that science invents things, engineering does both invention and optimization. Special thanks are given to Dr. Edwin Hou and Dr. Manikopolous for actively participating in my committee.

A lot of gratitude also goes to my friends Mohanbabu Natrajappa and Mahesh Rajamane who always took an interest in whatever I had to say about my thesis. I would especially like to thank Song Wang and Jacob Puthiamadathil for the time they spent with me digressing over mathematics, which is probably the reason I could come up with the proofs of all algorithms in the thesis.
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Background Information</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>Organization of the Document</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>LITERATURE REVIEW AND INFERENCES</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>C-means Family</td>
<td>7</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Hard C-Means (HCM)</td>
<td>9</td>
</tr>
<tr>
<td>2.1.2</td>
<td>Fuzzy C-Means (FCM)</td>
<td>10</td>
</tr>
<tr>
<td>2.1.3</td>
<td>Possibilistic C-Means (PCM)</td>
<td>12</td>
</tr>
<tr>
<td>2.1.4</td>
<td>Gath-Geva Algorithm</td>
<td>12</td>
</tr>
<tr>
<td>2.1.5</td>
<td>Gustaffson-Kessel Algorithm</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>Self Organization</td>
<td>15</td>
</tr>
<tr>
<td>2.2.1</td>
<td>Self-Organizing Map (SOM)</td>
<td>16</td>
</tr>
<tr>
<td>2.2.2</td>
<td>Neural Gas (NG)</td>
<td>18</td>
</tr>
<tr>
<td>2.2.3</td>
<td>Growing Neural Gas (GNG)</td>
<td>21</td>
</tr>
<tr>
<td>2.3</td>
<td>Radial Basis Function (RBF) Neural Network</td>
<td>26</td>
</tr>
<tr>
<td>2.3.1</td>
<td>RBF Training and Learning</td>
<td>27</td>
</tr>
<tr>
<td>2.3.2</td>
<td>RBF Classification</td>
<td>28</td>
</tr>
<tr>
<td>2.3.3</td>
<td>RBF Training with GNG (Supervised GNG)</td>
<td>29</td>
</tr>
<tr>
<td>2.4</td>
<td>Fuzzy Membership</td>
<td>30</td>
</tr>
<tr>
<td>Chapter</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>2.5 Insufficiencies in GNG</td>
<td>39</td>
<td></td>
</tr>
<tr>
<td>2.5.1 Difference in Learning between nodes</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>2.5.2 Noting what Nodes belong to which cluster</td>
<td>44</td>
<td></td>
</tr>
<tr>
<td>2.5.3 Ascertaining Data distribution from Topology</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>3 PROPOSED SYSTEM LAYOUT</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>3.1 System Layout</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>3.2 Brief Overview on Training, Learning and Testing</td>
<td>51</td>
<td></td>
</tr>
<tr>
<td>4. TOPOLOGY-BASED FUZZY CLUSTERING (TFC)</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>4.1 Motivation</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>4.2 Fuzzy and Reference Finding Algorithm</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>4.3 Effects with examples in 2D and 3D space</td>
<td>58</td>
<td></td>
</tr>
<tr>
<td>4.4 Mathematical Proof of the Algorithm</td>
<td>65</td>
<td></td>
</tr>
<tr>
<td>4.5 Working of the Reference Algorithm</td>
<td>68</td>
<td></td>
</tr>
<tr>
<td>4.6 Formalized Algorithm</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>4.7 Back-influencing of the Learning Rate</td>
<td>79</td>
<td></td>
</tr>
<tr>
<td>4.8 Storage in Terms of Overlapping Fuzzy Regions</td>
<td>81</td>
<td></td>
</tr>
<tr>
<td>4.9 Additional Algorithms</td>
<td>82</td>
<td></td>
</tr>
<tr>
<td>4.9.1 Incremental Merging Algorithm</td>
<td>83</td>
<td></td>
</tr>
<tr>
<td>4.9.2 Incremental Splitting Algorithm</td>
<td>84</td>
<td></td>
</tr>
<tr>
<td>4.10 Designation into Fuzzy Hyperspheres</td>
<td>87</td>
<td></td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS  
(Continued)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.11 Error Regions outside the Hyperspheres</td>
<td>94</td>
</tr>
<tr>
<td>4.12 Choice of Error Function for the Hyperspheres</td>
<td>101</td>
</tr>
<tr>
<td>4.13 Advantages and Disadvantages of TFC</td>
<td>102</td>
</tr>
<tr>
<td>4.14 Supervised Learning</td>
<td>103</td>
</tr>
<tr>
<td>4.15 Structure of ATFC Algorithm</td>
<td>104</td>
</tr>
<tr>
<td>4.15.1 Supervised Total Merging and Total Splitting</td>
<td>105</td>
</tr>
<tr>
<td>4.15.2 Learning through Misclassification</td>
<td>109</td>
</tr>
<tr>
<td>5. RESULTS AND DISCUSSIONS</td>
<td>116</td>
</tr>
<tr>
<td>5.1 IRIS Dataset</td>
<td>116</td>
</tr>
<tr>
<td>5.2 Simulated Dataset</td>
<td>127</td>
</tr>
<tr>
<td>5.2 NASA Dataset</td>
<td>135</td>
</tr>
<tr>
<td>6. CONCLUSIONS AND FURTHER WORK</td>
<td>146</td>
</tr>
<tr>
<td>6.1 Conclusions</td>
<td>146</td>
</tr>
<tr>
<td>6.2 Further Work</td>
<td>147</td>
</tr>
<tr>
<td>6.2.1 Hyperellipses</td>
<td>148</td>
</tr>
<tr>
<td>6.2.2 Better Supervised Learning and Choice of Parameters</td>
<td>151</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>154</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5.1</td>
<td>Misclassification rate for IRIS data using Various Algorithms</td>
</tr>
<tr>
<td>5.2</td>
<td>Misclassification rate for IRIS data using TFC and ATFC</td>
</tr>
<tr>
<td>5.3</td>
<td>Node Position from TFC and Expert Feedback</td>
</tr>
<tr>
<td>5.4</td>
<td>Simulated Dataset Distribution Parameters</td>
</tr>
<tr>
<td>5.5</td>
<td>Test Dataset Distribution Parameters</td>
</tr>
<tr>
<td>5.6</td>
<td>Simulated Dataset Learnt Node Position</td>
</tr>
<tr>
<td>5.7</td>
<td>Test Data Classification (Simulated Dataset)</td>
</tr>
<tr>
<td>5.8</td>
<td>Test Data Classification (MPRE301P) – TFC</td>
</tr>
<tr>
<td>5.9</td>
<td>Test Data Classification (MPRE301P) – ATFC</td>
</tr>
<tr>
<td>5.10</td>
<td>Choice of ATFC in Different Circumstances</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Effects of ( m ) on Fuzzy Memberships</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>Development of Self Organizing Maps over Multiple Iterations</td>
<td>17</td>
</tr>
<tr>
<td>2.3</td>
<td>Neural Gas Example</td>
<td>20</td>
</tr>
<tr>
<td>2.4</td>
<td>Growing Neural Gas Example</td>
<td>24</td>
</tr>
<tr>
<td>2.5</td>
<td>Radial Basis Function Network</td>
<td>26</td>
</tr>
<tr>
<td>2.6</td>
<td>Radial Basis Function (RBF) Network Learning from GNG</td>
<td>29</td>
</tr>
<tr>
<td>2.7</td>
<td>Fuzzy Membership Description for Height (Masai)</td>
<td>32</td>
</tr>
<tr>
<td>2.8</td>
<td>Fuzzy Membership Description for Height (Pgymy)</td>
<td>34</td>
</tr>
<tr>
<td>2.9</td>
<td>Fuzzy Membership Description for Speed</td>
<td>35</td>
</tr>
<tr>
<td>2.10</td>
<td>Fuzzy Membership Description for Braking Distance</td>
<td>36</td>
</tr>
<tr>
<td>2.11</td>
<td>Radial Basis Function and Axial Projection</td>
<td>37</td>
</tr>
<tr>
<td>2.12</td>
<td>Multiple RBF's with Axial Projection</td>
<td>38</td>
</tr>
<tr>
<td>2.13</td>
<td>Growth Dependence on Age for Topological Nodes</td>
<td>40</td>
</tr>
<tr>
<td>2.14</td>
<td>Topological Nodes Representing Learning</td>
<td>42</td>
</tr>
<tr>
<td>2.15</td>
<td>Local Weight Learning affecting Nodes as a taut rope</td>
<td>45</td>
</tr>
<tr>
<td>2.16</td>
<td>Area of Influence (Circular) for a 3-Node Structure</td>
<td>46</td>
</tr>
<tr>
<td>3.1</td>
<td>Addition of a New Layer</td>
<td>48</td>
</tr>
<tr>
<td>3.2</td>
<td>Numerical Value Assignment between Level-1 and Level-2 Nodes</td>
<td>49</td>
</tr>
<tr>
<td>4.1</td>
<td>2-Dimensional Topology Space with 2 Nodes</td>
<td>59</td>
</tr>
<tr>
<td>4.2</td>
<td>3-Dimensional Topology Space with 2 Nodes</td>
<td>59</td>
</tr>
</tbody>
</table>
**LIST OF FIGURES**  
*(Continued)*

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>2-Dimensional Topology Space with 3 Nodes</td>
<td>60</td>
</tr>
<tr>
<td>4.4</td>
<td>3-Dimensional Topology Space with 3 Nodes</td>
<td>61</td>
</tr>
<tr>
<td>4.5</td>
<td>Hyper-Sphere Projection (2D) and Representation (3D)</td>
<td>61</td>
</tr>
<tr>
<td>4.6</td>
<td>2-Dimensional Topology Space with 4 Nodes</td>
<td>62</td>
</tr>
<tr>
<td>4.7</td>
<td>3-Dimensional Topology Space with 4 Nodes</td>
<td>62</td>
</tr>
<tr>
<td>4.8</td>
<td>3-Dimensional Topology Space with 3 Nodes with Edge Connection</td>
<td>63</td>
</tr>
<tr>
<td>4.9</td>
<td>Comparison of Hyperspheres with Edge Connections (3D)</td>
<td>64</td>
</tr>
<tr>
<td>4.10</td>
<td>Hyper-Spheres Projection with Edge Connections</td>
<td>64</td>
</tr>
<tr>
<td>4.11</td>
<td>Effect of Edge Connection on Fuzzy Membership</td>
<td>65</td>
</tr>
<tr>
<td>4.12</td>
<td>System Layout with Fuzzy Membership</td>
<td>67</td>
</tr>
<tr>
<td>4.13</td>
<td>Path of Convergence (Internal)</td>
<td>68</td>
</tr>
<tr>
<td>4.14</td>
<td>Path of Convergence (External)</td>
<td>69</td>
</tr>
<tr>
<td>4.15</td>
<td>Topology Representation</td>
<td>71</td>
</tr>
<tr>
<td>4.16</td>
<td>5-Node Topology</td>
<td>72</td>
</tr>
<tr>
<td>4.17</td>
<td>Topology Constructed by GNG</td>
<td>79</td>
</tr>
<tr>
<td>4.18</td>
<td>Topology Constructed by TFC</td>
<td>80</td>
</tr>
<tr>
<td>4.19</td>
<td>Overlapping Fuzzy Hyperspheres</td>
<td>81</td>
</tr>
<tr>
<td>4.20</td>
<td>Generation of Fuzzy Hyperspheres</td>
<td>82</td>
</tr>
<tr>
<td>4.21</td>
<td>Detectable sample from Topology</td>
<td>88</td>
</tr>
<tr>
<td>4.22</td>
<td>Undetectable sample from Topology</td>
<td>88</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.23</td>
<td>Overlapping Cluster Regions</td>
<td>89</td>
</tr>
<tr>
<td>4.24</td>
<td>Flowchart for the Testing Routine (Representable Clusters)</td>
<td>91</td>
</tr>
<tr>
<td>4.25</td>
<td>Fuzzy Membership Projection by a Topology Network</td>
<td>92</td>
</tr>
<tr>
<td>4.26</td>
<td>Fuzzy Membership Projection with Maximal Membership</td>
<td>93</td>
</tr>
<tr>
<td>4.27</td>
<td>Overlapping Fuzzy Areas of Influence</td>
<td>94</td>
</tr>
<tr>
<td>4.28</td>
<td>Fuzzy Membership Projection with Smoothed Out Error</td>
<td>95</td>
</tr>
<tr>
<td>4.29</td>
<td>Overlapping Fuzzy Areas of Influence with Error considered</td>
<td>97</td>
</tr>
<tr>
<td>4.30</td>
<td>Plot of Hyperspheres with Nodes represented by ‘diamonds’</td>
<td>98</td>
</tr>
<tr>
<td>4.31</td>
<td>Plot of Hyperspheres with Nodes, Error fraction is 4</td>
<td>98</td>
</tr>
<tr>
<td>4.32</td>
<td>Plot of Hyperspheres with Nodes, Error fraction is 2</td>
<td>99</td>
</tr>
<tr>
<td>4.33</td>
<td>Flowchart for Testing Routine (Representable &amp; Unrepresentable Clusters)</td>
<td>100</td>
</tr>
<tr>
<td>4.34</td>
<td>Fuzzy Membership Projection with Linear Error Region</td>
<td>101</td>
</tr>
<tr>
<td>4.35</td>
<td>Adaptive Topology-Based Fuzzy Clustering</td>
<td>104</td>
</tr>
<tr>
<td>4.36</td>
<td>Pictorial View in 2 Dimension of a Topology (Before Expert’s Input)</td>
<td>106</td>
</tr>
<tr>
<td>4.37</td>
<td>Pictorial View in 2 Dimension of a Topology (After Expert’s Input)</td>
<td>106</td>
</tr>
<tr>
<td>4.38</td>
<td>Overlapping Clusters with High Memberships</td>
<td>111</td>
</tr>
<tr>
<td>4.39</td>
<td>Insertion of a Data point as a Node (1)</td>
<td>112</td>
</tr>
<tr>
<td>4.40</td>
<td>Insertion of a Data point as a Node (2)</td>
<td>113</td>
</tr>
<tr>
<td>4.41</td>
<td>Insertion of a Data point as a Node (3)</td>
<td>113</td>
</tr>
<tr>
<td>4.42</td>
<td>Insertion of a Data point as a Node (4)</td>
<td>114</td>
</tr>
<tr>
<td>5.1</td>
<td>Scatter plot for the IRIS Dataset</td>
<td>116</td>
</tr>
</tbody>
</table>
**LIST OF FIGURES**  
(Continued)

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2</td>
<td>GNG, Node-Data display in 3-Dimensional Topology</td>
<td>117</td>
</tr>
<tr>
<td>5.3</td>
<td>GNG, Node-Data display with Discriminant Line</td>
<td>118</td>
</tr>
<tr>
<td>5.4</td>
<td>GNG, Node-Data display from Another Vantage Point</td>
<td>119</td>
</tr>
<tr>
<td>5.5</td>
<td>Node Positions according to TPC</td>
<td>119</td>
</tr>
<tr>
<td>5.6</td>
<td>Output from TPC from another vantage point</td>
<td>120</td>
</tr>
<tr>
<td>5.7</td>
<td>Misclassified Data Points (Virginica as Versicolor)</td>
<td>121</td>
</tr>
<tr>
<td>5.8</td>
<td>Misclassified Data Points by ATFC</td>
<td>123</td>
</tr>
<tr>
<td>5.9</td>
<td>Only 2 Clusters created for IRIS Dataset by TFC</td>
<td>124</td>
</tr>
<tr>
<td>5.10</td>
<td>ATFC creates 3 Clusters for IRIS Dataset</td>
<td>126</td>
</tr>
<tr>
<td>5.11</td>
<td>Simulated Dataset with 3 Clusters</td>
<td>127</td>
</tr>
<tr>
<td>5.12</td>
<td>Simulated Dataset Topological Node Positions</td>
<td>129</td>
</tr>
<tr>
<td>5.13</td>
<td>Simulated Dataset Topological Node Positions with Breakage</td>
<td>131</td>
</tr>
<tr>
<td>5.14</td>
<td>Test Data Vector Positions Represented as ‘diamonds’</td>
<td>131</td>
</tr>
<tr>
<td>5.15</td>
<td>Fuzzy Area of Influence</td>
<td>132</td>
</tr>
<tr>
<td>5.16</td>
<td>Fuzzy Memberships represented by Circles</td>
<td>133</td>
</tr>
<tr>
<td>5.17</td>
<td>MPRE301P Signal (50Hz), Events of Interests</td>
<td>135</td>
</tr>
<tr>
<td>5.18</td>
<td>MPRE301P Signal (50Hz), Full Segment</td>
<td>136</td>
</tr>
<tr>
<td>5.19</td>
<td>MPRE301P Signal (50Hz), Full Segment without Events</td>
<td>137</td>
</tr>
<tr>
<td>5.20</td>
<td>3-Level Wavelet Decomposition on MPRE301P (50Hz)</td>
<td>138</td>
</tr>
<tr>
<td>5.21</td>
<td>Wavelet Tree for Figure 5.20</td>
<td>139</td>
</tr>
<tr>
<td>5.22</td>
<td>Unclassified Points Represented by Diamonds</td>
<td>139</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.23</td>
<td>Testing after Expert's comments</td>
<td>140</td>
</tr>
<tr>
<td>5.24</td>
<td>Testing after Expert's comments</td>
<td>141</td>
</tr>
<tr>
<td>5.25</td>
<td>Testing After Inclusion of more Data points by Expert</td>
<td>142</td>
</tr>
<tr>
<td>5.26</td>
<td>Overlapping Clusters with High Memberships</td>
<td>142</td>
</tr>
<tr>
<td>6.1</td>
<td>PCA Projected Axis – Global</td>
<td>150</td>
</tr>
<tr>
<td>6.2</td>
<td>PCA Projected Axis - Per Node</td>
<td>151</td>
</tr>
</tbody>
</table>
# LIST OF ALGORITHMS

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Pseudo-code for C-Means</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Self Organizing Map Algorithm</td>
<td>16</td>
</tr>
<tr>
<td>2.3 Neural Gas Algorithm</td>
<td>18</td>
</tr>
<tr>
<td>2.4 Growing Neural Gas Algorithm</td>
<td>21</td>
</tr>
<tr>
<td>4.1 Fuzzy and Reference Value finding Algorithm</td>
<td>57</td>
</tr>
<tr>
<td>4.2 Power Iteration Algorithm</td>
<td>66</td>
</tr>
<tr>
<td>4.3 Proposed Fuzzy Algorithm</td>
<td>74</td>
</tr>
<tr>
<td>4.4 Incremental Merging Algorithm</td>
<td>83</td>
</tr>
<tr>
<td>4.5 Incremental Splitting Algorithm</td>
<td>84</td>
</tr>
<tr>
<td>4.6 Testing Algorithm for Representable Data</td>
<td>90</td>
</tr>
<tr>
<td>4.7 Radius + Error Region Algorithm</td>
<td>96</td>
</tr>
<tr>
<td>4.8 Total Splitting Algorithm</td>
<td>107</td>
</tr>
<tr>
<td>4.9 Total Merging Algorithm</td>
<td>108</td>
</tr>
<tr>
<td>4.10 Single Step Adaptive Algorithm Variation-1 (SSAA)</td>
<td>110</td>
</tr>
<tr>
<td>4.11 Single Step Adaptive Algorithm (SSAA)</td>
<td>114</td>
</tr>
</tbody>
</table>
1.1 Motivation

Clustering is popularly defined as grouping of number of similar things or samples. And it has been used extensively in data analysis applied in fields like pattern recognition, linguistics, psychology and sociology. One of the most famous examples in pattern recognition is the Iris data set [4], a data set about 3 different varieties of Iris flowers by a biologist named Edgar Anderson. It shows the breadth of fields that clustering has influenced. An important measure in clustering is similarity between the samples, also known as the similarity measure, which might differ in each scenario from the distance between the samples to the shape it creates.

Clustering methods can be classified on different parameters, like requirement/non-requirement of pre-knowledge of clusters, or the type of cluster shapes to classify or sometimes even the batch processing method of the algorithm. In certain scenarios, like detecting abnormality patterns in the Electrocardiogram (ECG) graph of a patient; we would need an online clustering method that can record and classify abnormal events in real time that too sometimes without the pre-knowledge of the number of clusters. This can make many traditional algorithms like the K-Means class of algorithms, inadequate or ill-suited for the task directly; as they require pre-knowledge on the number of clusters, but can be adapted for such classification by using a two pronged approach of batch-processing at certain intervals. As much of the unknown is really uncertain, such methods do not perform very well in real circumstances.
Many of the research have also gone into the fact of using neural networks and fuzzy controllers to achieve good results in real time. But these methods also need some kind of priori knowledge about the clusters and need training. One of the problems with many methods like K-Means is the execution time depends not linearly but exponentially on the size of the data, which inhibits fast timing, when it comes to large amounts of data. The interval between which the clustering algorithm has to find cluster basically makes classification harder if not possible in real-time. The reason of which could be assigned to the fact that the number of samples during the interval could take toll on the timing. This is particularly true for K-Means family whose termination is the convergence rate which is partly dependent on the sample size.

It’s reasonable to infer that convergence based methods would suffer in offline clustering, primarily due to the fact that the guarantee of fast convergence is not a guarantee. The method chosen for such online clustering should be such that it can converge in a given number of steps or less, and should be computationally efficient. Typical online clustering also suffer problems due to the fact that it cannot represent efficiently all the data as it doesn’t have the privilege of calculating multiple iterations over a fixed number of data sample or the fact that it has to find clusters without previous knowledge of the structure and the number of clusters present.

My proposed algorithm, Topology based Fuzzy Clustering (TFC), is a single pass clustering algorithm that tries to generate clusters with help of a topology, with multiple nodes, generating algorithm called as Growing neural gas and by using a fuzzy value finding algorithm for the nodes. The cluster discovery process is also unsupervised and due to the fact that the growing neural algorithm with the help of the fuzzy value finding
algorithm can in fact in tandem is able to assign fuzzy values for the nodes for a cluster of any arbitrary shape help it achieve very high classification rates with the least amount of computation.

The origin of the growing neural algorithm draws its origin partly from the structure of biological neurons. Biological neurons have synapses between each other, and reinforce them whenever they or their neighbors get a signal for learning. Such a type of strengthening synapses was also proposed as a learning rule for neural networks. The regular GNG algorithm generates such a connected graph with connected nodes akin to neurons and connected synapses. This generates a kind of topology, and noting such a topology also supports decay of synapses so that memories can be forgotten, which is similar to biological neurons.

Topology-Based Fuzzy Clustering (TFC) enhances Growing Neural Gas (GNG) by adding more biological properties to the neurons: Namely the ability of difference in learning between nodes, which is not determinate on the age but on other factors like learning rate and topology. A global pattern is generated due to the fact that Neurons in a group do have dominant and sub-dominant nodes each forming their own local group and can be viewed as a hierarchy of nodes. Their learning is affected by their position in the global structure with the most dominant node learning the least, and the most non-dominant node doing most of the learning, and every other node learning between the maximum and the minimum rate.

Also examined is the effect of different shape of clusters on the proposed method and its ability to effectively do Online clustering. Comparisons with contemporary algorithms and some of the important factors like speed and complexities are also
considered. Results would be based and compared on the effectiveness of algorithm on well known datasets like IRIS as well as a specific dataset chosen from an existing NASA project [16].

I would be also discussing an Adaptive variant of TFC, called as Adaptive Topology-Based Fuzzy Clustering (ATFC), which can take supervised inputs and give better results in Classification and Clustering.

Finally, the implications of the algorithm with regards to size of the data, and important initial conditions are discussed. Also various parametric effects on the algorithm are considered and a methodology is built, so as to direct what parameters and approaches would work in certain conditions and what would not.

1.2 Background Information

With the later chapters discussing how different clustering algorithms work, with their relative advantages and disadvantages, I would like to keep this part of the chapter limited to a brief overview of ongoing research.

The field of Clustering has grown in the past many years and has been extensively used in fields like Chemistry, Electrical Engineering, Medicine, etc. With the advent of Fuzzy logic [12] and Neural Networks, concepts like Fuzzy Clustering and Radial basis Neural Networks has brought out newer avenues into ongoing research.

An important clustering family is the C-means (also known as K-Means) Clustering family of methods. A good discussion in Bezdek et al[1], is on fuzzy based K-means clustering also called as Fuzzy C-Means Clustering (FCM). Fuzzy Clustering Mechanism (FCM), has in particular given rise to many other Clustering Methods like the
Gaussian Mean Decomposition (GMD), Gustafson Kessel Method, Gath-Geva Algorithm etc [1]. These methods have their roots in Expectation Maximization (EM), which is used to approximate a probability function, and is typically used to compute maximum likelihood estimates given incomplete samples.

There have also been advances in the fields of Radial basis neural networks for clustering, like conversion of Radial functions into Fuzzy memberships to using a growing neural model in which neurons are added and deleted on specific conditions like error. Multidimensional clustering with fuzzy shells and c-shaped shells has also made possible to find variable shaped cluster.

Many of the Radial Basis Function Neural Networks have been extensively used to find Takagi-Sugeno-Kang fuzzy rules, which may be inferred back as fuzzy rules which can be incorporated and learnt by a fuzzy controller.

1.3 Organization of the Document

The document is divided into 6 main parts or chapters, namely (1) Introduction, (2) Literature review and Inferences, (3) System layout, (4) Topology based Fuzzy Clustering (TFC), (5) Results and Discussions and (6) Conclusions and future work.

Chapter 1 is concerned on the problem statement and the motivation for the document. Chapter 2 is mainly a review of existing Clustering techniques and proposes new ideas to increase the efficiency of GNG. Sections 2.1 to 2.4 discuss various algorithms and methodology used for classification. Section 2.5 is especially devoted on how to increase the efficiency of Growing Neural Gas algorithm.
Chapter 3 gives a brief overview on the system layout of the 2 layered network structure of TFC. Chapter 4 is devoted on the proposed Topology based Fuzzy Clustering (TFC) system’s working, with the learning and testing rules and the enhancements done to Growing Neural Gas Algorithm. Also covered in this chapter are the proof of various supporting algorithms and discussing about overlapping Hyper Spheres. Chapter 5 is devoted to the analysis of the proposed system on different datasets, namely the IRIS and the NASA dataset, and inferred conclusions on basis of the results. Chapter 6 is devoted to conclusions and further research for the better performance of TFC.
CHAPTER 2
LITERATURE REVIEW AND INFERENCE

2.1 C-Means Family

Any pattern recognition text would not be complete without the mention of the most famous clustering family of algorithms that is C-Means (or K-Means). Note for simplicity I would be referring this family of algorithms as C-Means everywhere in the text[2].

This family of algorithms is based on a variation of the expectation maximization technique. The input to this algorithm is in vector space. The objective of this technique is to increase the intra-cluster difference as much as possible. The central method involved in this difference calculation is based on the least squares for parameter estimation by Gauss, which was optimized later by Legendre.

The advantages of this algorithm are that; it is simple, with a series of iterations till convergence, and has been successfully adapted in many different ways to get better results. The variations of C-Means are hard clustering, fuzzy clustering, and possibilistic clustering. The reader is directed to see more about these variations is this seminal book by Bezdek[1] on fuzzy clustering. Some famous algorithms that have been based on this iterative method are the Fuzzy C-Means, Gustafsson-Kessel, Gath-Geva algorithm, etc. I would be discussing in depth these algorithms later.

The main part of the C-Mean’s algorithm is the minimization of an Objective function as noted in the following Equation 2.1.
The general formulation of the objective function to minimize in the C-Means is

\[ J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 \]  

(2.1)

Where \( \left\| x_i^{(j)} - c_j \right\|^2 \) is the distance measured between the point \( x_i^{(j)} \) and the cluster center is \( c_j \). C-means tries to minimize this objective function in Equation 2.1 by using the following algorithm.

**Pseudo-code for C-Means (Algorithm-2.1)**

Assuming that we have to find \( c \)-clusters,

1. **Step 1.** Choose arbitrary \( k \) points and assign them as the centroids of the initial group of clusters.
2. **Step 2.** Now for each object, assign it to the group that has the closest centroid with reference to the object.
3. **Step 3.** When all objects have been assigned, recalculate the positions of the \( C \) new-centroids. Repeat Steps 2 and 3 until the centroids no longer move.

Please note that the mathematical formulation for the different variations of the C-Means, is discussed later in the chapter. [Note on the termination condition: Termination can be achieved by comparing subsequent cluster centroids and analyzing if they have no large change during iterations.]
Some notes on the disadvantages of C-Means family:

- The initial number of cluster to be found should be known.
- Though it is proved to converge (terminate), some variations of this algorithm like Gath-Geva [5], are very sensitive to the initial cluster centers.
- As there is no fixed way to know the number of optimal cluster in a given dataset, multiple runs of the C-Means clustering has to be done, and a preferably human supervised analysis of the results to choose the optimal C (Number of clusters) needs to be done.
- Sparse and dense mixture models tend to generally give bad results with a large number of algorithms derived from C-Means, with notable exception of Gath-Geva.
- There is no unique solution due to the dependence on the choice of initial centers.

Let’s further see some algorithms from the C-Means family. Please note that \( v \) is also known as point prototype and is analogous to cluster centroids, whereas \( u \) is the degree to which a sample data is member of a cluster.

### 2.1.1 Hard C-Means (HCM)

Minimize \( J \) such that it delivers a C-Partition \( U \) of \( X \), \( D_{ij} \) is distance between \( V_i \) and \( X_j \). The equations for the point prototype and the sample data memberships are given as follows.

\[
    u_{ik} = \begin{cases} 
        1; & D_{ik} \leq D_{ij}, j \neq i \\
        0; & Otherwise 
    \end{cases} \forall i, k \tag{2.2a}
\]
The above equations (2.2a) and (2.2b) are iterated till difference in $\bar{v}$ is minimal over iterations. A note the choice of $P_{ik}$ as either 0 or 1 is a binary assumption, which implies that the point either lies inside the cluster or doesn’t lie inside the cluster.

2.1.2 Fuzzy C-Means (FCM)

The equations for the point prototype and the sample data fuzzy memberships are given as follows. Note the equations are similar to those mentioned for HCM.

$$u_{ik} = \left[ \sum_{j=1}^{c} \left( \frac{D_{ik}}{D_{ij}} \right)^{\frac{2}{m-1}} \right]^{-1} \quad \forall i, k, m > 1 \quad (2.3a)$$

$$v_{ik} = \frac{\sum_{k=1}^{n} u_{ik}^{m} x_{k}}{\sum_{k=1}^{n} u_{ik}^{m}} \quad \forall i \quad (2.3b)$$

Before discussing with Fuzzy C-means, the reader is assumed to have a relative good knowledge of Fuzzy Logic; otherwise which he/she is instructed to continue this section, after reading section 2.4.
The most obvious difference between Hard C-Means and Fuzzy C-Means is the way $U_{ik}$ is calculated in both. Please note that $U_{ik}$ can also be called as partition index. In Hard C-Means $U_{ik}$ is either 0 or 1, implying that the point can be in partition or not in partition, whereas in the Fuzzy C-Means a point can have different fuzzy membership for all the partitions, so it’s in the interval $[0,1]$.

There is also the introduction of $m$ whose value is greater than 1, which is a variable that specifies the fuzziness of the partition. An example would be best to explain the effects of $m$. The figure has been obtained from a software available online here [6].

Figure 2.1 Effects of ‘m’ on Fuzzy Memberships.

In general, more the fuzziness, that is larger the value of $m$, the greater overlap between the cluster occur. Lower value of $m$, will cause sharper boundaries. Its utmost important to choose a value of fuzziness that gives a good result by giving a better partition.
2.1.3 Possibilistic C-Means (PCM)

The equations for the point prototype and the sample data fuzzy probabilities are given as follows.

\[ u_{ik} = \left[ 1 + \left( \frac{D_{ik}^2}{w_i} \right)^{\frac{1}{m-1}} \right]^{-1} \quad \forall i, k, m > 1, w > 0 \quad (2.4a) \]

\[ v_{ik} = \frac{\sum_{k=1}^{n} u_{ik}^m x_k}{\sum_{k=1}^{n} u_{ik}^m} \quad \forall i \quad (2.4b) \]

On comparing (2.3a) and (2.4a), the major difference between Possibilistic C-Means and the Fuzzy C-Means is the introduction of \( w_i > 0 \), \( \forall i \). This value is also called as the weights \( w \), and is absent in both HCM and FCM. It is chosen initially by different methods which is comparable to the problem of finding initial cluster centers. It’s a direct version of the Expectation maximization algorithm that is used to optimize the maximum likelihood model of the data. The reader is guided to more details in [1].

2.1.4 Gath-Geva Algorithm

Gath-Geva[5] algorithm is a combination of the k-means algorithm and the fuzzy maximum-likelihood estimation(FMLE). Gath-Geva introduces a probabilistic measure, which is incorporated in the distance.
For the distance measure $D_{ik}$, is defined as
\[ D_{ik} = \frac{1}{\sum_{j=1}^{c} \left( \frac{1}{D_{ij}^2} \right)^{\frac{1}{(m-1)}}} = \forall i, k, m > 1 \quad (2.5a) \]

For the distance measure $D_{ij}$, is defined as $D_{ij} = (X_j - V_i)^T A (X_j - V_i)$, which depends a lot on $A$.

For example, when $A$ is an identity matrix, the distance calculated is Euclidean, and the algorithm turns into FCM. But for hyper ellipsoidal clusters, the $A$ measure is calculated from Fuzzy Maximum Likelihood Estimation (FMLE) as

- First finding the posterior probability for each point as,
\[
    h(i \mid X_j) = \frac{1 / D_{ij}^2}{\sum_{k=1}^{c} 1 / D_{kj}^2} \quad (2.5b)
\]

- And then the Distance measure
\[
    D_{ij}^2 = \frac{\det(F_i)}{P_i} \exp\left[ (X_j - V_i)^T F_i^{-1} (X_j - V_i) / 2 \right] \quad (2.5c)
\]

- Where Fuzzy Covariance matrix $F_i$ can be calculated as
\[
    F_i = \frac{\sum_{j=1}^{N} h(i \mid X_j)(X_j - V_i)(X_j - V_i)^T}{\sum_{j=1}^{N} h(i \mid X_j)} \quad (2.5d)
\]
This would estimate the priori and the posteriori probability and incorporate it in distance in such a way that Gath-Geva can effectively measure mixture of dense and sparse clusters. The only drawback is that Gath-Geva needs good initial cluster center to effectively cluster data. The distance measure can be used to detect not only spherical clusters, but even ellipsoidal clusters, which are not axially oriented but rather oriented in the manner the clusters really exist [5].

I would show examples of other algorithms that do clustering based on C-Means but with different way to choose the distance measure.

2.1.5 Gustaffson Kessel Algorithm

Gustaffson Kessel modifies the general C-Means by introducing another parameter inside the Optimization formulae as

$$D_{ij}^2 = \rho_i \cdot \text{det}(F_i)^{1/p} \left\| x_j - v_i \right\|^2$$  \hfill (2.6a)

Where $\rho_i = \sqrt{\text{det}(F_i)}$, $\rho_i > 0$

And the fuzzy covariance matrix as

$$F_i = \frac{\sum_{j=1}^{n} u_{ij}^m (x_j - v_i) (x_j - v_i)^T}{\sum_{j=1}^{n} u_{ij}^m}$$  \hfill (2.6b)

Gustaffson Kessel was formulated much before the probabilistic model of the Gath-Geva algorithm. Note that the primary difference between Gath-Geva and Gustafson-Kessel lies in the manner both algorithms compute the Distance.
The difference between FCM, Gath-Geva and Gustaffson Kessel can be summed up as their difference in finding Distance. The difference between FCM and PCM could be cited in terms of using Fuzzy membership Vs. Probability measure. And the difference between FCM and HCM could be cited as just as using Fuzzy measure Vs. Binary measure (0 or 1).

Please note that in case of Gath-Geva and other algorithms that are sensitive to initial choice of cluster-center/centroids, FCM is initially used to get a rough set of such centroids.

This has been a very mature field and there has been many algorithms developed on the similarity measures, one of them being least mean squares. Much research has also been done for the fine estimation of the initial centers, and their influence on the subsequent results from C-Means variations. This has been summed up very well in [1]. My experiments with a number of C-Means based algorithms showed me that Gath-Geva is usually the best performing C-Means variant with FCM and GK coming a distant second. The reader is also encouraged to look into this paper by Babuska et al [7], which shows the generation of Takagi-Sugeno Fuzzy models.

2.2 Self Organization

Before going further on how my proposed online clustering method can be used to give comparative results to offline clustering methods, it’s important to touch some subjects that are necessary, and the most important of all of them is Self organization. Self organization had its root in biological neurons, which exhibited diminishing learning with more training. Initially though its discovery is credited to Teuvo Kohonen, it was
discussed earlier by Christoph von der Malsburg, among others but Kohonen’s method turned out to be most efficient.

2.2.1 Self Organizing Maps (SOM)

SOM’s which were developed by Kohonen, effectively provide a way of representing multidimensional data in much lower dimensional spaces - usually one or two dimensions. This process, of reducing the dimensionality of vectors, is essentially a data compression technique known as vector quantization. Here I am mentioning a brief primer for SOM’s and the best possible way to introduce SOM is via the SOM algorithm mentioned below.

**Self Organizing Map Algorithm (Algorithm 2.2)**

Initialize Map

While not reaching a termination Condition do

Select a sample

Get best matching unit (or neuron) based on the Euclidean distance

Make the neighbors and the unit learn.

End While

The basic part is learning, which is done by a competitive learning rule. This rule used in SOM is a “**winner-take-all**” competitive learning, meaning that the winner node (and its neighbors) learns, while the other nodes don’t. One thing not mentioned in the above algorithm, is that there is a value associated with learning at each neuron, which
basically; “decrease learning with increasing frequency of winning”. This would essentially serve as a dampener for learning, and reduce learning overtime for all the nodes. Also the number of the nodes is fixed.

**Figure 2.2** Development of Self Organizing Maps over multiple iterations.
The preceding Figure 2.2 shows the construction of a SOM over multiple iterations. The nodes (neurons) in the figure are connected by edges with its nearest neighbors. Also observable is the slow rate at which the network develops. It’s interesting to note that for samples more than 100000, the network error did not reduce that much to cause further movement of the neurons. So after a while, the network stabilizes and the algorithm can be terminated.

2.2.2 Neural Gas (NG)

Neural Gas as proposed by Martinez and Schulzen[9], is also used to find such vectors but differ from SOM of Kohonen using another measure for learning. And the algorithm is stated next.

The neural gas algorithm [9] sorts for each input signal “x” the units of the network according to the distance of their reference vectors to “x”. Based on this "rank order" a certain number of units is adapted. Both the number of adapted units and the adaptation strength are decreased according to a fixed schedule.

The complete neural gas algorithm is the following (Algorithm 2.3):

1. Initialize the set $A$ to contain $N$ units $C_i$
   
   $A = \{C_1, C_2, \ldots, C_N\}$

   with reference vectors $W_{C_i} \in \mathbb{R}^n$ chosen randomly according to expected distribution of “x” samples.

   Also initialize the time parameter $t$

   \[ t=0 \]
2. Get at random an input signal \( x \).

3. Order all elements of \( A \) according to their distance to \( x \), i.e., find the sequence of indices \( (i_0,i_1,\ldots,i_{N-1}) \) such that \( W_{i_0} \) is the reference vector closest to \( x \), \( W_{i_1} \) is the reference vector second-closest to \( x \) and \( W_{i_k}, k=0,\ldots,N-1 \) is the reference vector such that \( k \) vectors \( W_j \) exist with \( \|x-W_j\| < \|x-W_k\| \). Following Martinetz et al. [9] we denote with \( k_i(x,A) \) the number \( k \) associated with \( W_i \).

4. Adapt the reference vectors according to

\[
\Delta W_i = e(t) \cdot h_{\lambda}(k_i(x,A)) \cdot (x-W_i)
\]

with the following time-dependencies:

\[
\lambda(t) = \lambda_i \left( \frac{\lambda_f}{\lambda_i} \right)^{t/t_{\max}}
\]

\[
e(t) = e_i \left( \frac{e_f}{e_i} \right)^{t/t_{\max}}
\]

\[
h_{\lambda}(k) = e \left( \frac{k}{\lambda(t)} \right)
\]

5. Increase the time parameter \( t \):

\[
t = t+1
\]

6. if \( t < t_{\max} \), go to step 2.

Note the parameters to choose for the neural gas algorithm are \( \lambda_i, \lambda_f, e_i, e_f \).

The following Figure 2.3 shown describes the update process and growth of the network that happens during various stages of the Neural Gas algorithm. Also note that
the learning rate needs to be set individually for each neuronal node. This is pretty interesting as it doesn't follow explicitly, Kohonen's Self-Organizing constraints, that need a diminishing learning rate for the sake of stability of the network.

![Images of Neural Gas examples with varying signal counts](image)

Parametric values (from [8]): $\lambda_i = 10, \lambda_f = 0.01, e_i = 0.5, e_f = 0.005$

**Figure 2.3** Neural Gas Example [7].

As one can observe that like Self-Organizing Maps, the Neural Gas is a topology learning method. The advantage of the Neural Gas could be cited to that its dimensionless (without a fixed vector size) and could be made to learn topology. But do consider locally there does exist a dimension. The disadvantage of Neural Gas could be cited to the fact that Learning is not local but global at each iteration. The algorithm does need a lot of signals for effective topology learning and the user needs to be certain of the number of nodes that he/she would require beforehand.
2.2.3 Growing Neural Gas (GNG)

Growing Neural Gas as proposed by Fritzke is an unsupervised incremental clustering algorithm. The growth mechanism from the earlier proposed Growing Cell Structures [17] and the topology generation of Competitive Hebbian Learning [8] are combined to a new model. Starting with very few units, new units are inserted successively. To determine where to insert new units, local error measures are gathered during the adaptation process. Each new unit is inserted near the unit which has accumulated most error.

The complete *growing neural gas* algorithm is the following (Algorithm 2.4):

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

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

   with reference vectors chosen randomly according to distribution “x”.

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

   $$C = \emptyset$$

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

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

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:

$$\Delta W_{s_1} = e_b (x - W_{s_1})$$

$$\Delta W_i = e_n (x - W_i)$$

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

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

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

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 $A_{\text{max}}$. If this results in units having no more emanating edges, remove those units as well.

9. 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:

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

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

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

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

As seen the GNG differs from the Neural Gas, by using a constant learning rate, and knowledge of the neighbor nodes at all times. This gives the network more power in terms of faster updates of neighbors, and only causing localized updates with every new data point.
The following figure displays the final results after 40000 adaptation steps for a circular distribution. Do note that the choice of topological position of the initial 2 nodes don’t make a difference during the entire run of the algorithm, though might help in faster accurate learning at the start.

\[
\lambda = 300, \epsilon_b = 0.05, \epsilon_n = 0.0006, \alpha = 0.5, \beta = 0.0005, a_{\max} = 88
\]

**Figure 2.4** Growing Neural Gas Example [18].

Note that the parameters of GNG constant in time. Further, comparing it the Neural Gas model, as the number of nodes is being added during the running of the algorithm the decision on the number of nodes is not needed priori. This removes the restriction for multiple runs to estimate the best possible number of such nodes, which is needed in the case of NG and SOM. With GNG, insertion of new nodes continues until some user defined performance criteria are met like the mean error across all the nodes or alternatively if a maximum network size has been reached.
These properties of GNG makes it a potential algorithm for distributions in which priori knowledge is very limited, in places where the sample data set size is too high and the distribution has an unusual shape. It best works in cases of stationary or slowing moving distributions.

Note, that multiple runs are needed to determine the values of $\lambda$, $e_b$, $e_n$, $\alpha$, $\beta$, $A_{\text{max}}$. In general experimentation the values most affecting is the choice of $\lambda$ and $A_{\text{max}}$. The reader is cautioned to use different values of these parameters to see the effect. As a rule of thumb the value of $\lambda$ and $A_{\text{max}}$, should in be equal or near each other in magnitude or order.

The effects of $\lambda$ is that, the algorithm will generate less number of nodes at high values of $\lambda$ and more number of nodes at lower values of $\lambda$. But note that it is also influencing indirectly the representation of the localized distribution. The more the nodes, the finer the distribution is captured at the nodes, but a very small value of $\lambda$, would generate unnecessary number of nodes. The same can be said of $A_{\text{max}}$, which is the reason of breaking of edges. As the lesser $A_{\text{max}}$, the more broken graphs would be developed. The reader is instructed to specially take care when choosing the values of $\lambda$ and $A_{\text{max}}$.

It’s important to mention here that GNG is very good when it comes to stationary clusters, but in cases of jumping clusters, GNG might start wasting units. There has been related research by Fritzke on using a Utility factor with each node, so as to delete nodes that don’t contribute.
2.3 Radial Basis Function (RBF) Neural Network

An artificial neural network can be described as a set of interconnected simple computing units that are modeled crudely on the biological neurons present in living organisms. A brief description of neural network can be found in Haykin [10]. A very basic description of such a network could be described on the basis of their task of Optimization.

\[ XW + B \approx Y \]

Where X is the input and Y the output vectors; W is the matrix for weight and B is the matrix of Bias (or Offset) associated with the neurons in the network. Also note that the trained network would try to ensure that difference between XW+B and Y is below a certain threshold, for a large number of sample data.

There are many methods to find W matrix in neural networks, and one of them is using Radial Basis Network.

The general construction of a Radial Basis Function Network is given below.

![Radial Basis Function Network Diagram](image)

*Note: \( \sigma \) is the variance of the radial basis function at Laver-1.*
The primary working of the RBF Neural network could be explained by the Layer-1 causing non-linearization of X. Due to this non-linear transformation, the supposedly higher-dimension in which X is transformed to, brings about linearization.

Such non-linearization of the input would cause the Layer-1 output to become linearly classifiable. That is, the output from Layer-1 could be than separated into linearly separable areas. This is markedly difference from Multi-Layer Perceptron (MLP) which tries to generate linear-separation in all the layers. Due to the non-linearity imparted by the Layer-1, a 2 layered RBF neural network is functionally able to match and generate the same type of classification as an MLP with 3 layers.

2.3.1 RBF Training and Learning

Due to the property of RBF Neural network of being functionally equivalent to fuzzy rules [15], there has been much research into how to generate TSK based fuzzy rules from RBF’s [10]. Training and Learning is normally done in supervised manner, with each input and corresponding output presented to the system one by one, with the objective of penalizing the network on every misclassification and rewarding the network on every classification. Normally a strategy like Winner-take-all is used for training, and delta rule for learning [12].

The model talked before in Figure 2.5 on RBF Neural network is used for learning a single pattern, for multiple pattern classification, multiple Layer-2 nodes are constructed, also shown in the figure.

The most important in RBF is the centers and the variance of the centers for the Layer-1. These centers could be determined by using a Clustering algorithm like C-
Means. The choice of clustering algorithm does make a difference in the performance of the network. Also as the RBF Network is usually used in a supervised fashion during training, its utmost important to determine the number of clusters beforehand. Once the network has learned sufficiently, it can be tested on unknown data.

2.3.2 RBF Classification

1. Online Clustering is done with RBF using supervised training with some sample data points extracted from the dataset. The sample data with which it is trained is assumed to be a sufficient representative of the whole dataset.

2. Once supervised learning is performed than rest of the data is classified via unsupervised learning.

3. Such batch processing mode can be run for multiple iterations in form of teaching via supervised mode for a part of the data, then switching to unsupervised classification for some more data points, and finding out the misclassified data. Then again making the network re-learn the misclassified data and continuing the process.

The choice of initial cluster centers can be done using a Clustering algorithm. But this would not make it resistant to unknown data or data outside the range with which it was classified with. This proves to be a drawback of using RBF Neural Network directly with Online Clustering.

Fritzke had proposed a new type of method to choose the RBF’s Neural Network to help it generate Layer-1 nodes on fly and thereby providing it the ability to expand and learn new patterns online, but the number of Layer-2 nodes (or the number of clusters) remains constant.
2.3.3 RBF Training with GNG (Supervised GNG)

Multiple output units can be used for multiple clusters/pattern

![Diagram of RBF Network learning from GNG]

Figure 2.6 Radial Basis Function (RBF) Network learning from GNG.

In Fritzke’s model, he used a Growing Neural gas model to generate Layer-1 Nodes. The generation is though in supervised mode only. As you might recall from my earlier discussion, the major part during the GNG growth is learning and error calculation. Over here the learning and error calculation is directly dependent on the difference between expected and actual measurement. Though there is no way to find new clusters on fly as the number of Layer-2 neurons is fixed (which is also incidentally the number of clusters/patterns to be discovered).

So what are the advantages? It would be able to find newer pattern and would be resistant to over-generalization that arises when there are few Layer-1 nodes and there is the occurrence of large number of outliers. It’s a better version of the RBF, due to its ability to find outliers. But the disadvantages lie in its inability to find newer clusters; this can be ascribed to the static nature of the Layer-2 nodes.
2.4 Fuzzy Logic and Membership

This section would be devoted to a general understanding of fuzzy logic and application in the problem of clustering. I would be addressing the history of fuzzy logic as a set of statements and the history behind those statements.

"In the start there was 0 and 1".

Though this cannot be said of in biology, but is true in the terms of modern mathematics. Aristotle, whose contribution to mathematics and philosophy among other fields has been paramount, was also responsible for ingraining the mathematical philosophy of true and false, of 0 and 1, of in or out, of the binary world. The mathematical philosophy of many Arabic and Indian mathematicians whose were proponents of a more multi-valued mathematic was not used till recently.

"Then came a gray world"

In the seminal works by Lotfi Zadeh[13], showed the world again, the powerful mathematics of fuzziness or the gray world which lied between 0 and 1, as was ignored till date by Western mathematicians. Though he initially faced many critics who discounted his theory, most of them being Western mathematicians, fuzzy logic was slowly but surely adopted. Its initial adopters was not the west, it was the east countries like Japan who was always a proponent of a multi-valued world. A major resurgence in the west adoption to fuzzy logic came during the 80’s, like the fuzzy washing machine, which automatically loaded soap and water as per the wash-load weight and type.
“The world is gray now”

Once fuzzy logic was adopted as a proven mathematics by both the oriental and the occident, it has been adapted to aid solutions in real-world environments. Though originally themed for a control-engineer perspective, the fuzzy logic has found it place from computer algorithms to rule-based expert systems. Many recent theories on fuzzy logic have been on compounding and equating it on the lines of neural networks. As the reader might recall, neural networks is a field of science extensively used for environments that are very non-linear in nature, and where finding relations between variables in the environments is nearly impossible. A major complain about neural networks is its inability to give inference on why a particular solution is suggested by it.

The reader is encouraged to read Kosko [14] on the history and development of Fuzzy logic and its relation with neural networks.

“Fuzzy logic example”

Consider the example of defining height. How to define “tall”, “short” or “medium”? It’s not fixed at all! Consider that the average height of a Masai (a tribe in Africa) is 6 feet. A Masai will consider someone of 5’10” a short fellow. But consider the average height of a pygmy (another tribe in Africa) is around 4’3”, the same “short” Masai we talked of will be considered a very “tall” person by the pygmy.

The idea of this example is to show that height is relative and cannot be expressed in direct values as greater than 6’ or greater than 4’3”. Also it’s very fallacious not to include a 5’11” Masai in the medium height category. He is at least much taller than a 4’3” pygmy. Such an all-inclusion or all-exclusion policy also is detrimental in real
conditions. The bad effects can be seen if such a total-inclusion/exclusion policy is taken in the case of a boiler room measurement device. If the device can indicate only 2 conditions **good** or **bad** of a boiler, it won’t be a bit useful to know that boiler has suddenly turned from **good** to **bad** conditions. A gradual meter with many levels between **good** and **bad** conditions would be a lot helpful.

Now continuing the same example of height of a Masai, I will describe how fuzzy membership for the height of a person of Masai origin would look like.

![Fuzzy membership graph](image)

**Figure 2.7** Fuzzy Membership description for Height (Masai).

In the Figure 2.7, I am talking of height in terms of relative terms like short, medium and tall; and of the conditions like very short and somewhat short and somewhat medium. If we call “short”, “medium” and “tall” as variables, than “very” and
“somewhat” are called as literals, which enhance the meaning of these variables. Such literals are termed as hedges. By using variables like “short”, “medium” and “tall”, I have basically omitted all the crisp aspects of height that is fixed values like 6’ or 5’10”, is also similar to the way humans quantify such quantities like height. Its usefulness lies in, its reusability, that if height was described by Pygmies also as “short”, “medium” or “tall”, than the above membership graph and rules derived from it could have been directly used, with little or no modifications.

Before going further it’s imperative to discuss how to interpret relevant results from the graph.

Now let’s consider the problem of classifying a Masai via his/her height: In our normal parlance, we could have classified a Masai as having medium height if is somewhat short and somewhat medium. That is the region where the gray world exists. For the Masai example it lies somewhere between 5’8” an 6’. Though we would most probably term a Masai of height near 5’8” as short and someone near 6’ as medium.

In laymen terms we would generate a rule saying that “If a person is somewhat short and somewhat medium than he is somewhat medium”. Such types of rules are called as fuzzy rules. Fuzzy rules are divided into antecedents and consequents. The former part or the causal part after “if” is the antecedent, the latter part or the action part after “than” is called the consequent.

Going back to the same example on height, we can see the rules generated for the Masai tribe could be used even for the pygmy tribe as shown in Figure 2.8 next, but than the question arises what makes the difference? And where do crisp values of exact height come into picture?
The answer is the first process of any fuzzy logic based system is conversion of the crisp inputs into fuzzified values between 0 and 1. If we assume that the maximum value on the Y-axis is 1 and minimum 0, than 5'8" lies with the membership or the value 1 for the variable “short” whereas 0 for the variable “medium”. And similarly a person of height 5’10” would have a value off 0.5 into “short” and 0.5 into “medium”. This process of converting crisp values to fuzzy membership values is called as fuzzification.

Once converted multiple fuzzy functions can be correlated and an answer obtained. The process of correlation and de-fuzzification would be explained in another example.
Consider that we are modeling a car movement that has to occur somewhat autonomously, and we would like it to have auto braking capabilities. The auto braking abilities are dependent on the speed of the car. That is to say that if we are moving at a high speed the breaking distance would be large and so it would be wiser to brake much earlier, but the same braking distance would be less when speed is less and so a delay of braking would be feasible and would save some fuel.

So let’s define some fuzzy membership functions

![Fuzzy Membership](image)

**Figure 2.9** Fuzzy Membership description for Speed.

Preceding Figure 2.9 describes the fuzzy membership in regards with speed limits. Three variables are defined for the speed of the Car namely Slow, Medium and Fast. With reference to variables Slow, Medium and Fast, each of the variables is maximum at 30, 50 and 70 MPH.
The preceding figure shows the braking distances’ fuzzy membership. The variables “slow”, “medium” and “fast” are defined from the speed of the car. To cite an example say that if we are moving at slow speed than the breaking distance would be around 5m.

Consider that the car is moving at a speed of 40 mph. Relating this speed of the car to the speed membership graph, it would be counted as 0.7 memberships into slow and 0.3 membership as medium. If we correlate it to the braking distance membership graph, we can see that it related to the point where the braking distance should be 7m.

The example shown here is a very simplified form of multiple memberships with a single rule, which is “if we are slow and medium than the braking distance is slow”. Multiple rules do cause different effects; and the typical method to add the rules is using centroid or min-max method.
The reader is encouraged to look into the text of [14] for a more in depth analysis of fuzzy rules.

Now I would be concentrating on the part of fuzzy membership that is related to the thesis. The reader is asked not to confuse probability with fuzzy logic as they both rule different domains. For example, consider the example of a person in a room; he is either in the room or outside, the fuzzy membership is either 1 or 0, it has nothing to do with the probability of that person in the room or not. Similarly if he/she is standing on the door, than he is say 0.5 inside the room and 0.5 outside the room, it has nothing to do with probability at all.

Though many researchers, including Kosko, have purported that probability is fuzzy logic in disguise [14]. Consider the example of a 2D cluster in space. And a radial basis function, to represent it.

![Figure 2.11 Radial Basis Function and Axial Projection.](image-url)
If we want to represent the recognition of the data within the cluster A, we would probably write a rule as

If data lies within A and B than Data is in Cluster A

The above case becomes complicated when boundary cases start to occur like 2 overlapping clusters. In those cases instead of defining hard boundaries soft boundaries are defined. But still how do we define a soft boundary?

One way to do is using some type of radial basis functions like Gaussian that can be the representation of fuzzy membership inside the cluster.
Note it's all up to the user to choose the type of membership functions he/she wants to use.

If the same clusters were to be represented by probability, we needed to calculate the priori and posteriori probability of each data point when considering both the clusters. This approach would constraint that the probabilities of a data point in both clusters equals 1 on addition. The flexibility of choosing a membership function is not present. A reader might ask why that would be necessary. The reason for which could be cited to the fact, that in order to make a neural network that can learn such a pattern, adjustment of the radial basis function would be needed. If we use probability to generate such rules, due to the probability based constraints, it won't be possible to make the network learn before causing any major re-learning for the other radial-basis function, as probability should always equal 1. Fuzzy membership because of its ability to be independent can be well adjusted without causing major re-learning.

The re-learning process is cumbersome and fuzzy logic due to its localized learning ability is well suited to reduce the re-learning phenomenon.

2.5 Insufficiency in GNG

From my early experiments with GNG, I found that efficiency could be increased if the following methods were incorporated into GNG

1. Difference in Learning between nodes.

2. Noting what nodes belong to which cluster.

3. Incorporating Topology to ascertain the distribution of data in a cluster.
Let’s see one by one why all these factors are necessary.

2.5.1 Difference in Learning between Nodes

It’s imperative that the learning rate should be dependent on some factor (will mention that afterwards) and not be constant for all the nodes, the reason for which can be cited as more a node has learned the more it should start to stabilize (or have lesser learning), or the network becomes more plastic in nature.

Now how to determine different learning rates for each node generated by GNG? Rather than gradually decreasing learning rate for all the nodes based on their age and the times they have been updated, its imperative to use another measure to change the learning rate.

The measure of changing the learning rate on basis of the time or age is not good as it wouldn’t effectively describe the learning in some parts of space where learning is to be done at a higher rate.

This would be most effectively communicated in the diagram below,

![Figure 2.13 Growth Dependence on Age for Topological Nodes.](image-url)
As one can see in the preceding Figure 2.13 that new data is increasingly being presented at the nodes having the Age=0. If the learning rate is damped (reduced) on the basis of age, than the effectively learning of the new data is somewhat stymied. So we need to disregard the damping of learning rate in time. Rather I propose something called as aggressive learning.

Let’s take the earlier example and show how aggressive learning could be used a factor for better learning. First let me state some factors and definitions for aggressive learning.

**Aggressive learning** is defined as a learning rate possible for each node, and the learning rate has a baseline learning rate and a maximum learning rate, rather than SOM’s damped learning rate that can vary between 0 to a baseline learning rate. In SOM’s defined learning, the learning rate for all nodes start from a high learning rate and decreases to 0 over time. Over here aggressive learning starts from a certain baseline learning rate and increases more according to the topologically assigned fuzzy rates.

It’s reasonable to assume that due to aggressive learning the network never becomes stable. This is both good and bad. Bad in the sense, it’s desirable for a network to stabilize. But stability also implies over-specialization and removal of the network’s ability to learn more patterns better, so over-specialization is bad in the case of online cluster discovery. Thus really the plasticity is preserved in the network due to Aggressive Learning.

Aggressive learning is good in sense that the learning rate being directly dependent on the topology would be plasticky but stabilized just enough that both learning new patterns and storing old pattern is balanced. The good advantages of
aggressive learning far outweigh the disadvantages; due to the learning rate always fluctuating in the hyperspace according to data-distribution causing a better form of learning.

The formulation and use of aggressive learning in my algorithm, is very much responsible for the cluster discovery. Let's see how different aggressive learning rates could be incorporated in our earlier model.

![Diagram of Topological Nodes](image)

**Figure 2.14** Topological Nodes Representing Learning.

In the preceding figure the learning at each node has been represented by a circle. The more the learning, the larger the circle, which influences the learning rate at different parts of topology.

Rather than being dependent on ages the topology should be taken as a baseline for the learning rates. If we consider that the above nodes are created by GNG and connected with edges, it's reasonable to assume that they represent the data distribution via the edge connection and the distance between each edge. The center of the topology graph should have the lowest learning rate and the exterior should have the lowest. The
reason being that; the center part should not migrate as much as the exterior nodes as exterior nodes are the ones where the merging with other clusters would happen the most.

This strategy would aggressively learn the pattern that are most near the cluster but just outside the cluster, whereas would make sure that the central nodes inside the cluster would learn those pattern also but less aggressively.

“So a question arises how to determine such internal nodes and external nodes?”

The algorithm to find such internal and external is discussed in the subsequent chapters. For the time being, assume that such an algorithm exists and I am able to give a reference value between the nodes of a cluster, based on the nearness to the virtual center and to the leaves of such a graph. Than the learning rate for each node could be directly incorporated based on these topological values as

\[
\text{Learning Rate} = \frac{\text{Baseline Learning Rate}}{\text{Topological Determined value from center}}
\]

Where, \(0 < \text{Topological Determined value from center} \leq 1\)

The learning rate could be thus determined for each node this way. Note that the ‘Topological Determined value from center is less than 1 for aggressive learning. The discussion in how to select the topological determined value is discussed later.

A better representation of Learning rate could be,

\[
\text{Learning Rate} = \text{Baseline Learning Rate} \times F (\text{Distance of the Node from Cluster center})
\]
Where,

F (Distance of the Node from Cluster center) = function that increases as the distance increases from the center.

It should be noted that the learning rate for a node should be more as farther the node is away from the center; the problem of different axial data distribution would cause more misclassifications later if strictly the measure of distance is taken. My subsequent algorithm on TPC rather takes into account the data distribution within the cluster into consideration and finds such a distance relative to the cluster center.

2.5.2 Noting what Nodes belong to which cluster

The original GNG algorithm has no provision to keep noting which of the nodes are connected with each other, thereby generating a cluster. The reason for which can be cited to Fritzke using the connection matrix C mainly to keep a note of neighbors, so as to use only localized update and save on computation time.

This edge matrix C can be still enhanced in such a way that such connected graph can be used to know what cluster they generate, and even the shape of such a cluster.

A question that arises is whether the construction of such cluster is valid? As we know node & edge deletion is pretty local, and the only time the edge number is increased between a node and its neighbors is during the update when the node is a winner for a given data sample. Due to this localized update the shape of the cluster is valid and can be used.

Another question is whether the whole cluster process is incremental or not? Yes, that is true, because in every iteration, there is the chance to either delete an edge or node
or addition of a node and edge. Algorithms can be designed that check the condition that 2 new clusters are created when a cluster edge is deleted. Similarly the condition of merging of such cluster on an edge addition can be checked. Such incremental way to generate and note the cluster nodes does reduce the number of computing resources required. There can be many ways to note the cluster; either as representing as a node prototype, or as a multi-layered structure in which Level-1 represents the modified growing neural gas and Level-2 representing the prototype cluster.

2.5.3 Ascertaining Data distribution from Topology

Topology from the GNG generates a connected undirected graph per cluster. A question that arises is, Is there any valid information in such a cluster graph? Yes, but the problem lies is how to determine any information from such a distribution.

Do take the analogy of GNG creating a taut-series of strings, whose each end try to pull in different directions on basis of data occurrence.

Figure 2.15 Local Weight Learning affecting Nodes as a taut rope.
Assume that edges in the preceding Figure 2.15 end in a node. Now with each iteration and presentation of sample data, each node is “pulled” in a particular direction. The distance between the nodes do determine how much they are pulled. Also due to the insertion process at every lambda intervals, the density of the nodes in a particular region is indirectly related to the density of the sample data in that region.

Using the distance measure between each node and its neighbors, and taking the average distance I can draw a hyper-circle around the targeted node that is shown in the next figure. The hyper circle is one way to draw to generate the area of influence for each node. The other alternatives are hyper-ellipses, which can be axially oriented or maximal-variance oriented.

![Diagram of node and hyper-circles](image)

**Figure 2.16** Area of Influence (Circle) for a 3-node structure.

Such type of circles can be drawn for every node generated, all due to the help of the edge information. Yet note that I haven’t talked about how to assign any measure to the node like the probability or a fuzzy membership to such a topology now. That would
be discussed in a later chapter, so be assured that it’s possible to give some numerical valid measure to the nodes however it is connected and oriented.

Note that such overlap of the hyper-spheres would be able to generate various degrees of fuzzy membership regions. And also due to the very nature of GNG of localized learning the value of the Radius and the Area of Influence for each Node would be more or less locally changed over single data learning.
3.1 System layout

Now I present the structure of my proposed Topology-Based Fuzzy Clustering (TPC) system in this section. Please note later chapters are devoted to Learning and Testing of the system and validity of the algorithm proposed. The structure shown here is just the overview of the system. Let me start by the incremental modifications done to the GNG algorithm.

1. **Addition of a new layer to describe a prototyped cluster**

![Diagram of the system layout](image)

*Figure 3.1 Addition of a New Layer.*

Please ignore the connection between Level-1 and Level-2 nodes and their meaning for the time being. Let the Level-1 nodes be the nodes that are generated from
the GNG algorithm. One place of interest was using a placeholder for prototyped cluster. This is done by adding another layer called as ‘Level-2’, which notes all the Level-1 nodes generated.

In the previous example X, Y nodes from Level-1 are connected with each other and so make a cluster C2 in Level-2. Same can be said of A, B, C and D, Level-1 nodes that make a cluster C1 as Level-2 node.

Note that the nodes in Level-1 and nodes in Level-2 are at different domains, namely Level-1 nodes are topology nodes, whereas the Level-2 nodes are the prototyped clusters. There is a sense of detachment between Level-1 and Level-2 node, that is the formation of Level-2 nodes is directly dependent on the Level-1 nodes, which is due to the fact there is interaction between Level-1 nodes, but Level-2 nodes don’t have that type of interactions. In short Level-2 nodes are mere placeholders for the clusters.

2. **Now assigning a numerical value to each connection from ‘Level-1’ nodes to ‘Level-2’ node**

![Figure 3.2 Numerical Value assignment between Level-1 and Level-2 Nodes.](image)
Ø represents that numerical value that connects the Level-1 node with the Level-2 node. This measure can be probability based measure or fuzzy based measure. The choice and the methodology to find Ø, plus what type of unit C1 and C2, would be discussed later in Chapter 4.

### 3.2 Brief Overview on Training, Learning and Testing

Let's see some brief overview on construction, training, learning and testing of the nodes of the network.

For any RBF network the basic phases are:

1. **Construction of nodes (either can be static or dynamic)**

   For example consider the construction in terms of Neural Gas (Algorithm-2.3) which has an initial number of nodes. Only these nodes need to learn. There is provision like dynamic growth in Growing Neural Gas (Algorithm-2.4). Growth though useful can cause unnecessary nodes being created and so would need a periodic pruning.

2. **Presentation of samples and Training of the nodes via learning**

   Consider the Radial Basis Function Network I talked of in the last chapter. The RBF network needs supervised learning, in which samples and required classes are presented to the network one by one. Such supervised learning does require manually knowing the classes' priori. Unsupervised learning is normally based on some heuristics or underlying pattern discovery process. Though it won't require knowing the classes' priori, the heuristic running the learning is assumed to be powerful enough
to find the underlying patterns. Thus the importance when migrating from a supervised to an unsupervised learning moves from manual knowledge to knowledge discovery.

3. Presentation of test data and relearning if necessary

Consider when the learning is finished, its very well suited to work on data-samples not used during the learning process. Such type of process would involve relearning when a sample data is misclassified. With each misclassification, some method used for learning are penalizing the node(s) responsible for the misclassification, and rewarding the node(s) responsible for correct classification.

For my system, the first phase is done in two places:

In Level-1, the nodes are constructed through a modified Growing Neural Gas algorithm and in Level-2; the nodes are constructed by using the topology into consideration. These Level-2 nodes are constructed by noting which edges are connected with other edges. The numbers of individual sub graphs created by the topology are the placeholders for each Level-2 nodes (or clusters). So I do note the number of clusters completed at each step, and create new clusters or nodes in the Level-2.

The second phase is done also in two places:

I would be using a technique called as aggressive learning, discussed earlier in Chapter Section 2.5.1, and use an algorithm (Algorithm 4.1) to manipulate the learning rate at each instant for every node. Samples are presented to the network one after the
other, and with each sample causing some perturbation of the values of the links between Level-1 nodes and Level-2 nodes.

The third phase related with testing would be discussed in a later chapter. Now assuming that I do have an algorithm to generate fuzzy membership for each Level-1 node into Level-2 node, it would be than be possible to use this membership during the learning phase. With lesser membership implying a larger learning rate, whereas higher membership implies a smaller learning rate; this has already been discussed in Chapter Section 2.5.1 as aggressive learning. I will use the aggressive learning phenomenon to change the learning rate at each step and use it as the learning rate for the Level-1 node.

Many questions would be arising through the readers’ mind and I would be addressing some of the questions below.

- What Level or Layer of nodes would be affected by learning?

Learning at each pattern is done in Level-1 node movement. Level-2 nodes are generated and deleted on edge creation and edge deletion cases only. Level-2 nodes are not affected at any point by the learning, only Level-1 nodes are affected.

- How would be the testing be done?

After the network has learned a lot amount of patterns, I can stop the learning and testing the network would on the basis of the radius and fuzzy membership (value of the link between Level-1 and Level-2 node). This is a user decided parameter, as theoretically, for total cluster discovery, the whole pattern data set should be available to the network.
• How are the Level 2 nodes constructed from Level-1 nodes?

Level-2 nodes are constructed noting the construction and deletion of edges at each iteration. The algorithms that would aid us are the Incremental Splitting Algorithm (Algorithm 4.5) and Incremental Merging Algorithm (Algorithm 4.4).

• What type of learning can Level-1 Nodes be associated with?

The Level-1 nodes can be said to follow winner-take-all strategy updating a region of nodes whenever a data occurs.
CHAPTER 4

TOPOLOGY BASED FUZZY CLUSTERING (TFC)

4.1 Motivation

As discussed earlier, GNG is most helpful in generating valid topologically connected nodes. But these are just connected graphs but without any metadata or information on how the data distribution of data or even how to determine the cluster shape. And let’s touch the issue of determining $\theta$ that I left in the earlier chapter.

That brings us important questions on how to define the cluster shape and the value of $\theta$ that I talked earlier and is there a relation between shape and $\theta$?

"Should I use a In or Out measure for each data point for a cluster?"

"Is it possible to use a probability based measure to define the shape? And would it be fast?"

"Or Could I just use fuzzy membership for the GNG-nodes for their respective cluster?"

Instead of calculating the probability, the best measure would be to use fuzzy membership.

Reason for which could be cited to:

1. Unlike Probability, there is no condition that sum of all probabilities is equal to 1.
2. Unlike Changing probability of a single data in a cluster requires re-calculating probabilities for all the sample-data; fuzzy membership can be done individually and normally without major recalculation.
So I decided the best method for inclusion would be applying a fuzzy membership to each Level-1 node inside the Level-2 cluster structure.

**Thought: “A wishful thinking would be ability to generate fuzzy membership for each connected node in cluster!”**

Yes it’s absolutely possible to generate valid fuzzy functions based on the topology. How? It can be answered from an observation, that fuzzy membership for a data sample can also be defined as its membership in different clusters. And it is fact that the cluster can be represented or prototyped by the cluster center. By knowing the cluster centers it’s possible to estimate the fuzzy membership for each Level-1 node.

So determining the cluster center is a major task and that’s where the topology comes into play. As our topology of nodes in a cluster is just connection of points (represented by vectors) by edges, an algorithm can be developed to find the center of such a graph. The next part of the chapter is devoted to finding such a center and effectively distributing fuzzy membership with the help of such a center.

Now I discuss something called as Reference value. Reference value is non-normalized form of fuzzy values as discussed below and is derived from the topology of a connected graph (or topology). It will try to differentiate the most periphery nodes from the most central nodes.

And is calculated as below:

Let nodes be represented by $N_1, N_2, N_3...N_m$ each having corresponding reference values be $\hat{O}_1, \hat{O}_2, \hat{O}_3... \hat{O}_m$ and Radius(Average distance between node and neighbors) $R_1, R_2, R_3..., R_n$. 
The following function, for each node, has to be evaluated till it converges

\[
\Phi_p' = \frac{\sum_{n=1}^{k} \Phi'_n * R_n}{\sum_{n=1}^{k} \Phi'_n \sum_{n=1}^{k} R_n}
\]  \hspace{1cm} (4.1)

Where \(n=1...k\) is the list of neighbors of \(P\), including \(P\) itself. And \(P\) varies from \(N_1, N_2, N_3...N_m\).

Also \(\Phi'\) is the Reference value.

After some iteration the above value will converge to some fixed values.

Due to the fact that there is the case of different number of neighbors for each of the nodes, the above formula has the tendency not to converge the values exactly to either 0 or 1 but would lie in the interval \([0,1]\). The Reference value is efficient to find the center of the topology cluster.

In the next step for each cluster, I calculate the fuzzy value from the reference value. The Reference value for the most outliers would be highest whereas would be lowest for the topologically based center. Now using this I can normalize all values against the center so that the center has a fuzzy membership of 1.

That can be done by using,

\[
fuzzy_{ic} = \frac{\min(\text{reference}_c)}{\text{reference}_{ic}}
\]  \hspace{1cm} (4.2)

Where \(fuzzy_{ic}\) is the fuzzy membership of the \(i^{th}\) node in cluster \(c\), and \(\min(\text{reference}_c)\) is the minimum of the reference values in cluster \(c\).
4.2 Fuzzy and Reference Finding Algorithm

This section is devoted to stating a formalized algorithm for finding the fuzzy membership in a topological network. The algorithm is able to give a topological estimation of fuzzy membership values for the nodes in the network based solely on radius and the neighbor list of each node. Formalized algorithm pseudo code to find fuzzy membership for nodes in a cluster is given below.

Fuzzy and Reference Value finding Algorithm (Algorithm 4.1)

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

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

\[
\text{ReferenceValues}(i) = \frac{\sum_{n=1}^{k} \text{ReferenceValues}(n) \cdot R(n)}{\sum_{n=1}^{k} \sum_{p=1}^{l} R(p)}
\]

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

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

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

Stop

*Note: ReferenceValue(i)_{t-1} is the reference value at instant t-1 and ReferenceValue(i) at instant t*

Please note for the Algorithm 4.1, the convergence time is totally dependent on the choice of the original values of Reference value. Though at convergence, all the values will be stabilize from whatever Reference value taken initially. Convergence is
always aided by choosing an initial relative value near to final relative. I would be talking later on how to increase the convergence rate by choosing such initial Reference values. The fuzzy membership that I found above is the desired $O$ that I was talking earlier.

Now before continuing further $O$, let me touch slightly on the subject of the validity of using the Equation-4.1 for finding fuzzy membership. Different way of assigning fuzzy values from Reference values can also be debated, which is perfectly alright. The reader is encouraged to try different methods to determine fuzzy membership from the Reference values. Please note though the merits of choosing a normalized minimum from Reference value is the straight forwardness and the ability to maximize the value of the so-called centroid node among the Level-1 node. The disadvantages would be no node should always be considered the centroid node in a cluster.

The layout of the chapter further on would be, first to discuss the effects of fuzzy membership and Reference values and after that give proof of the algorithm. By the way, one of the most important parts of the algorithm is the use of $referenceM$ and $fuzzyM$ matrices to find the fuzzy regions and for testing. $fuzzyM$ is also used to influence the learning rates at certain regions in space.

4.2 Effects with examples in 2D and 3D space

Below are few examples of the Reference and the fuzzy values when there are 2,3,4,5 and more nodes. The examples are necessary to show the effects of the algorithm on topology. It will also show the ability of the algorithm to include vector prototypes of the node centers directly from the radius and the neighbor list.
The first 2 example are of 2 nodes in a 2-Dimensional and 3-Dimensional space respectively. Also the dimensionality is the size of the vector in which the data is represented.

![Figure 4.1](image1.png) 2-Dimensional Topology space with 2 Nodes.

![Figure 4.2](image2.png) 3-Dimensional Topology space with 2 Nodes.
In the Figure 4.1 & 4.2, it is observable that for a 2 node structure the fuzzy membership is 1. As Reference value is a measure of the most exterior to the most interior node, for a 2 node structure, both the nodes are at the same time the most interior and the most exterior nodes. Also please note that this 2-Node topology condition would be true for higher dimensions also.

The next 2 examples are of a 3-Node topology in 2-Dimensional and 3-Dimensional vector space respectively.

![Figure 4.3 2-Dimensional Topology space with 3 Nodes.](image)

Here, in 2-Dimensional space the node with the co-ordinate (0.5,1.2) is the internal node with the (0,0) and (2,2) nodes being the exterior node. As observable and reasoned earlier, the interior nodes will have the highest fuzzy membership and the lowest reference value, and is clearly the case as is shown in the Figure 4.3. Also note that the node (0.5,1.2) is more near to (0,0) and farther away from (2,2) in Euclidean
distance. Thus as node at (0,0) is more near to the center of the topology it should have higher membership than node at (2,2). This inference is also proved in the Figure 4.3, and the fuzzy membership of node at (0,0) is 0.68 which is greater than that at (2,2) of 0.65.

Figure 4.4 3-Dimensional Topology space with 3 Nodes.

The same thing happens in 3 Dimensional space, with the internal node having the highest memberships and the exterior nodes the lowest.

Figure 4.5 Hypersphere Projection(2D) and Representation (3D).
Figure 4.5 represents the projection of the hypersphere, drawn from Radius, on X-Y plane and then a representation of the hyperspheres in 3-Dimensional space.

**Figure 4.6** 2-Dimensional Topology space with 4 Nodes.

**Figure 4.7** 3-Dimensional Topology space with 4 Nodes.
Figure 4.6 and 4.7 represents the topology in 2-Dimensional and 3-Dimensional space with 4 nodes. Now let’s see the results when the topology is more complex in terms of the edge structure.

Comparing Figure 4.4 and Figure 4.8, there is a new connection between (0,0,0) and (2,2,2), the result of which the fuzzy membership of all the 3 nodes become 1. This can be equated to the fact that newer data comes in existence between the 2 nodes, due to which a new edge is created. Thus whenever an edge structure is created to quantify the data occurrence between 2 nodes, it would cause the fuzzy membership of the nodes also to change. Thus the edge structure and the neighbor structure are the most important components in the Equation 4.1 and are indirectly able to represent the underlying topology and the data distribution.
Figure 4.9 Comparison of Hyperspheres with Edge connections (3D).

Figure 4.9, is depicted in order to show the effects of edge connection. The edge is created between node at (0,0,0) and (2,2,2) and due to this the hyperspheres at the 2 nodes increases in size. This can be inferred as, edges are created on presence of newer data samples, and this should also in turn increase the hyper sphere size.

Figure 4.10 Hyperspheres Projection with edge connections.

Figure 4.10, shows the effects of the projection of the hyper-spheres on the X-Y plane. Note the increase in the projection area in the second figure part after edge insertion.
A question that would arise is whether the fuzzy membership would change on the edge connection?

The answer would be yes and is shown in the following Figure 4.11.

Figure 4.11 Effect of Edge Connection on Fuzzy Membership.

As seen in the above Figure 4.11, when a new connection is made between the nodes, the fuzzy membership of the node newly connected increases whereas the other exterior nodes’ membership decreases. This shows major change occurs when edges are connected, analogous to the density of occurrence of new nodes. That means, above would only occur in the cases when data samples start occurring between those nodes.

4.4 Mathematical Proof of the Algorithm

With any iterative algorithm, it’s imperative to prove whether the algorithm converges or not and to a lesser degree the rate of convergence. As observed the main components of the algorithm are the Edge matrix C and the Radius obtained from W matrix, with the neighbor list obtained from the C matrix. The proof of the algorithm lies in a standard
theorem in the matrix computation called as power iterations. I am also aided by the fact that edge matrix C is a symmetrical matrix.

The idea of Power iteration is that on repeated multiplication of a vector with a symmetric matrix A and with normalization at each step would cause convergence after some steps. Provided that the initial vector is not deficient and A’s eigenvalue of max modulus is unique. The rate of convergence is the ratio of the eigen value at step 2 and step 1.

The Power Iteration algorithm is stated below (Algorithm 4.2):

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

\[
\begin{align*}
\text{for } & k=1,2,\ldots, \\
q^{(k)} &= Az^{(k-1)} \\
q^{(k)} &= z^{(k)} / \|z^{(k)}\|_2 \\
\lambda^{(k)} &= \left[q^{(k)}\right]^T Aq^{(k)}
\end{align*}
\]

If \( q^{(0)} \) is not “deficient” and A’s eigen-value of maximum modulus is unique, the \( q^{(k)} \) converges to an eigenvector.

If you recall equation 4.1, the \( \Phi' \) value mentioned there is equivalent to the \( q^{(k)} \) vector, the A is equivalently a subset of the C matrix. As I am following all the above conditions the reference value finding algorithm would converge eventually. Practical examples show that the rate of convergence is very fast and is also helped by many facts like I am making use of the old values of reference values to calculate new values.
The new matrix formed is the fuzzy membership matrix for the each individual Level-1 nodes in a given cluster representing the Level-2 node.

![Diagram](image.png)

**Figure 4.12** System Layout with Fuzzy Membership.

Algorithm 4.1 will help finding the fuzzy memberships of these links

The above figure is a modified form of the system layout earlier discussed in the last chapter. The reference and fuzzy value finding algorithm would help calculating the fuzzy membership of the links between Level-1 Nodes and Level-2 nodes.

The reader would have realized by now that fuzzy memberships depend directly on reference values. So it’s very important to discuss what effects will be caused by reference value. Basically reference value finds the connection between the periphery nodes and the central nodes. The reason of using reference values stem from the reason that I need to find effectively the center of the cluster. The center could be found from the linkage of the Level-1 cluster nodes. The iterative update of the reference values would
ensure that the center would have the lowest reference value in comparison with the leaf nodes. Note that the inverse of the reference value finding equation is also convergent. Its just that using an inverse reference value would have caused values of ‘reference value’ greater than 1 and the normalization would still be required. So in short it’s just the terminology that differs.

4.5 Working of the Reference Algorithm

A small note on the iterative formula, that it will start updating from the periphery to the center. For example consider that the leaf nodes are assigned an arbitrary value of 0.1, that value will start updating the internal nodes of the graph and in the next iteration the nodes would start decreasing in such a way that the nodes would try to achieve the minimum reference value. The algorithm will start converge outwards from the central node to the leaf nodes when the central node achieves the minimum possible value for that cluster linkage configuration. In subsequent iterations the outer nodes start converging, till all nodes converge.

Initially all the nodes have random values, and in the first few iterations the outer values start converging till the central part the linkage structure converge to the minimum.

![Path of Convergence](image)

**Figure 4.13** Path of Convergence (Internal).
Now the convergence starts from internal nodes to external nodes

![Diagram of Path of Convergence](image.png)

**Figure 4.14** Path of Convergence (External).

Note though during this process the update might cause change in the values of the reference for internal nodes, but this change wouldn’t be as drastic as it was during the initial stages.

After some iteration the whole update process would converge and the reference values will settle down. As the reference values are dependent on the linkage and number of nodes connecting to it, it wouldn’t converge to a singular value. To put it in a single line “Reference values among the nodes of the same cluster show how much difference or similarity is there between the level nodes”.

So what affects the convergence of this algorithm? The choices of initial values of the reference value do help in convergence the algorithm. A fact that helps us in choosing the initial value of the reference values is that update is introduction of a new node would cause localized reconfiguration of the reference values which would probably be minute for existing nodes. So using this knowledge, instead of choosing arbitrary values
whenever we want to calculate the reference values, we just use old values of the
reference values and recalculate on the basis of those values. This will ensure very fast
convergence. For example when tried on a nominal node number of about 30, the
convergence dropped from about 3.5 average iterations to about 1.2 average iterations.
This saves a lot of calculations and even allows recalculating the reference values and
updating them on the occurrence of every new data.

As noted that the growing neural gas method does delete edges whenever the
number of nodes increase beyond a maximum limit and it deletes nodes which are
orphaned, that is not connected to other nodes. A reasonable method would be to judge
the perturbation caused in the fuzzy values. Rest assured that this perturbation doesn’t
cause much ripple when recalculating on the event of edge or node deletion.

Also this should be noted that the number of reference value update algorithm
iterations is linear in relation with the number of level-1 nodes, when the initial reference
values are chosen arbitrary. During actual working, due to inclusion of factors like usage
of old reference values, the number of iterations required is very nominal considered to
the graph size.

And due to the fact that a very small overhead of finding the reference values is
incurred at each iteration, it can be directly incorporated into the growing neural gas
algorithm with nominal overhead.

Let’s see the trace of the reference value at each iteration, for a 4-node topology
network. Note it’s represented as a vector referenceM.
Trace:

Iteration 1: referenceM = [ 0.3333  0.3333  0.3333  0.3333 ] (Initial value)
Iteration 2: referenceM = [0.5000  0.2500  0.3333  0.3333]
Iteration 3: referenceM = [0.5141  0.2539  0.3349  0.3349]
Iteration 4: referenceM = [0.5144  0.2540  0.3348  0.3348] (Converged value)

fuzzyM = [0.4937  1.0000  0.7586  0.7586]

If the first 2 iterations are observed one can notice that the initial value can be chosen at random. It doesn’t make a difference if the initial values are greater than the end value. And the values do converge at a very fast rate. The only condition for the initial values should be that its not deficient (remember the condition that was mentioned earlier when I talked of Power iteration). Its much better to use a non-zero value initially in-order that a special case like ‘divide-by-zero’ doesn’t occur.

The topology for the earlier example is shown below

Figure 4.15 Topology Representation.
Now let’s see another example with a more complex topology. Here I take an example of a 5-node topology network.

![Diagram of a 5-node topology network.](image)

**Figure 4.16** 5-Node Topology.

Below is the trace of the output at each iteration, for the reference value till convergence.

Iteration 1: \( \text{referenceM = [ 0.2500 0.2500 0.2500 0.2500 0.2500 ]} \)

Iteration 2: \( \text{referenceM = [ 0.5000 0.2500 0.3333 0.2500 0.5000 ]} \)

Iteration 3: \( \text{referenceM = [ 0.5141 0.2509 0.3262 0.2528 0.4962 ]} \)

Iteration 4: \( \text{referenceM = [ 0.5146 0.2516 0.3271 0.2531 0.4963 ]} \)

Iteration 5: \( \text{referenceM = [ 0.5146 0.2515 0.3270 0.2531 0.4963 ]} \)

\( \text{fuzzyM = [ 0.4888 1.0000 0.7691 0.9939 0.5068 ]} \)
As can be seen during the iterations, initially between Iteration 1 and Iteration 2 the convergence is inward with the 3\textsuperscript{rd} node (3\textsuperscript{rd} index in the reference vector) converging to a value of 0.3333. Then the convergence goes outwards with the outer nodes converging. There is a minor perturbation as can be seen in the values of 3\textsuperscript{rd} node during the rest of the iterations from 2 to 5.

4.6 Formalized Algorithm

Before I discuss further on stating a formalized algorithm, it's important for me to describe the mathematical formulation necessary for describing the various matrices, values and lists involved.

Mathematical formulation to find the fuzzy values in a cluster from the reference values

First find the lowest value from the reference values for a single cluster

\[ L_c = \{ L_c < x \& L_c \in \text{Refer}_c \ | \ x \in \text{Refer}_c - \{L\} \} \]

Lowest value (L) = minimum (reference value for each cluster)

\[ Fuzzy_c = \left\{ \frac{L_c}{\text{Refer}_c(i)} \mid 1 \leq i \leq \text{size(Refer}_c) \right\} \]

This means that the Fuzzy values are normalized inverse reference value in relation with the lowest reference value.

When used with the reference value finding update algorithm, leaf (exterior) nodes will try to achieve values that are maximum among the reference values of the
nodes in the same cluster whereas the internal nodes would try to achieve the lowest reference value.

The new algorithm (derived from Algorithm 2.4 on GNG) can now be written as (words in bold are the added/modified text to GNG)

Algorithm 4.3:

1. Initialize the set $A$ to contain two units $C_2$ and $C_1$
   
   $\mathcal{A} = \{C_1, C_2, \ldots, C_n\}$

   with reference vectors chosen randomly according to distribution “$x$”.

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

   $C = 0$

   Initialize the Cluster set $\mathcal{C}$, $\mathcal{C} \in \mathcal{N} \times \mathcal{A}$, $\mathcal{N} \in \mathbb{R}$ ($\mathbb{N}$ is the set of real numbers, and $\mathcal{N} \leq \text{size}(x)$

   $\mathcal{C} = 0$

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

   $\mathcal{C} = \mathcal{C} \cup \{(1, A), (2, B)\}$

   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

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

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.

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 \( N \).

\[ \text{Cluster}_M = \{ \text{Cluster}(M, i) | i \in A \} \]
\[ \text{Cluster} = \text{Cluster} - \text{Cluster}_M \]

And for Cluster numbered \( N \)

\[ \text{Cluster}_N = \{ \text{Cluster}(N, i) | i \in A \} \]
\[ \text{Cluster} = \text{Cluster} - \text{Cluster}_N \]
c. Merge ClusterM and ClusterN into a single set

$$\text{ClusterNew} = \text{ClusterM} \cup \text{ClusterN}$$

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

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

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

$$\text{Cluster} = \text{Cluster} \cup \text{ClusterP}$$

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

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

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:

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

For the Neighbors as,

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

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]
S1 and S2 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 S1:

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

8. Remove edges with an age larger than Amax. 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 discussed as Algorithm 4.5.

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:

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

c. Add a new unit \( r \) to the network and interpolate its reference vector from \( q \) and \( f \):

\[ A = A \cup \{ r \}, \quad W_r = \left( W_q + W_f \right) / 2 \]

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.

\[ \exists i \in n, \text{Cluster}(i, q) \land \text{Cluster}(i, f) = \text{true} \]
\[ \text{Cluster} = \text{Cluster} \cup \{(i, r)\} \]

Use the Incremental Merging Algorithm discussed as Algorithm 4.4.

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 4.1, 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.

Now once I have formalized the algorithm, I would be inferring the decisions taken for the various modifications done in the algorithm. The most important are the back-influencing learning rate, the incremental nature of cluster creation and deletion and the interpretation of the hyper-spheres that would be used for testing.
4.7 Back influencing of the learning rate

At each step the learning at each level-1 nodes is influenced by the fuzzy value associated with the node. As in my construction, every level-1 node is connected with one and only one level-2 node. As a result of which the learning of a level-1 node is directly influenced by its fuzzy membership in the given level-2 node. As a result of which the more the fuzzy value the lesser the learning whereas it would aggressively learn at places where the fuzzy membership of the node is less.

Though as our network is not following Kohonen's constraint on learning should be reduced for nodes that have learned more. In my case the learning is directly influenced on the basis of whether a node is an outlier or a central node. Outliers should tend to learn aggressively whereas the center nodes are already well defined, so the normal learning rate is more emphasized for them. Please refer to Chapter 2, Section 5 for a detailed view on Aggressive learning.

Let's see a brief example for IRIS data set which does

![Figure 4.17 Topology constructed by GNG.](image-url)
Figure 4.17, is showing the generation of connected nodes, for the IRIS dataset. Due to a uniform learning rate, there is a lagging edge which can lead to misclassification. The edges and the nodes should be passing through the dataset itself. Let’s see a brief result from my algorithm.

The results from my algorithm don’t show a lagging edge and node. This is because of aggressive learning by back influencing of learning rate. More results can be seen in the Chapter 6, devoted on results. I won’t be discussing more about the dataset or the testing method, because that’s reserved for another chapter. What I wanted to stress from this section is, aggressive learning does help in learning better the sample data.

Figure 4.18 Topology constructed by TFC.
4.8 Storage in terms of overlapping fuzzy regions

As I discussed in an earlier chapter, the radius is calculated as an average distance between neighbors of a node. This is a region often mentioned in this text as Area of influence. This area of influence will cause fuzzy regions represented as hyper-spheres, which are also overlapping in nature. The reader might have a question, whether this overlapping region is logically correct or not. Rest assured this would be alright in our condition as the learning is localized and is more or less binary in the region during testing, which we would discuss later is basically ‘Maximum of all the memberships of the sample data in all the clusters’.

![Figure 4.19 Overlapping Fuzzy Hyperspheres.](image)

The title of this chapter is about the storage that is associated with these hyper-spheres. This storage is in the form of the ability of the nodes to remember the data density and thus create edges based on local data occurrence. Also due to the ability of
the nodes to be able to create localized clusters that can be used to create bigger cluster of various

![Figure 4.20 Generation of Fuzzy Hyperspheres.](image)

Figure 4.20, is about generating fuzzy hyper-sphere from Level-1 nodes created by the IRIS dataset. The Hyper-sphere is generated around each node. Also the overlapping regions in space can be seen. Please note that there is always a chance that Hyper-sphere won't be able to detect outlier data, even if that outlier is used during the training process.

### 4.9 Additional Algorithms

This section is devoted to supplementary algorithms that would be necessary for construction of Level-2 nodes which are also the prototypes of the clusters. The algorithm discussed here are Incremental Splitting and Incremental Merging.
4.9.1 Incremental Merging

Now at each point there is the chance of new clusters to be created by either the splitting of older clusters, or by the merging of the clusters.

This merging and splitting criterion is checked for each new node. As the number of nodes in cluster is much lesser than the number of samples, finding out if the merge or the split has occurred is computationally feasible. Also note that this process is an incremental process which simplifies many things.

How to determine the splitting and merging of clusters?

The answer is either using a sophisticated approach like the breath-first search to find minimum spanned tree in the graph. A variation of this spanned tree can be also used that can be incremental in nature.

Another approach that can be taken to find if the merge and split criteria is met is to use a brute force technique, which is also very efficient, despite its brute-force approach. The idea is whenever a cluster has an edge removed between the nodes, the chance of split occurs in which use the following algorithm.

Incremental Merging Algorithm (Algorithm 4.4):

This algorithm is called whenever 2 nodes “a” and “b” have a new edge created between them.

1. find which clusters these 2 nodes “a” and “b” lie and name the indices as “P1” and “P2”
2. if $P1 = P2$, means that the nodes are already in the same cluster, so there is no need to continue more so Return;

3. if $P1 \neq P2$, means that the nodes are in separate clusters, so there would be merging involved
   a. For merging remove the original 2 clusters and insert a new cluster with the indices from both the clusters
   b. As merging is completed. Return

Please note that in both the above algorithm of merging and splitting the edge matrix or the matrix noting the connections, named as $C$, between nodes is extensively used. The above algorithm used for an incremental merging is computationally not so intensive. The reason for it being only 2 conditions being checked before merging is done, and the merging a direct merge.

I would now present the Incremental splitting algorithm in the next section.

4.9.2 Incremental Splitting Algorithm (Algorithm 4.5):

Assuming that nodes “a” and “b” are disconnected, by removing the edge between them.

1. Make a list of existing nodes in the cluster and name it $GraphMatrix$

2. Make 2 individual lists namely $G1$ and $G2$, and insert indexes of nodes “a” and “b” in them. Remove “a” and “b” from $GraphMatrix$

3. Now note $G1$ as $OldG1$.

4. Now exhaustively search what nodes can be reached from each element in $G1$. 
a. If a node can be reached by an element in G1, remove it from GraphMatrix.

b. Insert that in G1.

5. Terminate the algorithm if G1 and OldG1 contain the same elements OR if GraphMatrix becomes empty ELSE go again to step 3.

6. At this point either GraphMatrix is either empty or has some elements.
   a. In case if GraphMatrix is empty than the cluster has not split.
   b. In case if GraphMatrix is non-empty, it implies that there has been splitting and the GraphMatrix is the second subgraph with the first being G1

7. let G2 = GraphMatrix plus node “b”

8. Returning Condition
   a. return “NO SPLIT” if G2 is empty
   b. return G1 and G2 as the splitted subgraphs, and make new clusters with reference to these 2 subgraphs.

As the name suggests, the incremental splitting algorithm would be able to construct newer clusters during the normal splitting operations done by edge deletion. If we analyze the computational speed of the algorithm, if there are “n” nodes in a cluster and the cluster splits into 2 clusters, than the computational time for the above algorithm would take $O(n^2)$. As the number of Level-1 nodes are very less as compared to the number of sample data presented, as a result of which the time taken for splitting a cluster...
is negligible. Next I would be presenting the proof about splitting causing at max 2 clusters.

**Proof: A splitting of a Cluster by an edge deletion causes at maximum 2 sub-graphs**

Consider a graph \( G \) with vertices \( V \) and set of connected undirected edges as

\[ E = \{(V_1, V_2) | V_1, V_2 \in V\}. \]

Let \( G \) contain 2 vertices \( A \) and \( B \) with an edge \( X \) between both of them

**Assumption:** Assume that deleting an edge creates \( n>2 \) graphs, each with vertices set \( V_{k1}, V_{k2}, V_{k3}, \ldots V_{kn} \) and Edges set \( E_{1}, E_{2}, E_{3}, \ldots \) which can also be written as,

\[ G_1 = \{V_{k1}, E_{1}\}, \ G_2 = \{V_{k2}, E_{2}\}, \ldots \ G_n = \{V_{kn}, E_{n}\} \]

This also implies that,

\[ V_{k1} \cup V_{k2} \cup \ldots \cup V_{kn} = V \] (set of vertices in original graph)

\[ E_{1} \cup E_{2} \cup \ldots \cup E_{n} = E - \{X\} \] (Set of all graphs in original graph without edge \( X \))

Now restore the earlier edge \( X \) (that we had deleted)

As \( X \in V \times V \) space, namely an edge must have only 2 vertices as endpoints.

This implication about 2 end vertices for an edge, affects also \( X \) and so restoring \( X \) would also restore a connection between 2 and only 2 vertices. These 2 vertices can either lie in the same graph or different graph.

Thus implying that on restoring \( X \) we would connect at max 2 graphs into a single graph. Thus all remaining graphs from \( n>2 \) would remain disconnected; which implies restoring \( X \) to its original vertices doesn’t return us back to the original state.
Thus implying that our initial assumption of \( n > 2 \) is false

**Thus the number of sub graphs caused on breaking a graph is \( n \leq 2 \).**

### 4.10 Designation into Fuzzy Hyper spheres

The construction of the nodes and the topology aided with the reference and fuzzy value algorithm, has its roots in the neighbors list for each node and the average radius associated from its interaction with its neighbors.

The Radius at each node is the average distance between the node and all its neighbors. As a radius is also incidentally though indirectly representing the amount of uncertainty or better its representing under-representation of a pattern in that part of space, it's imperative to correlate with the fuzzy membership to the global cluster it's contributing to. Though, it should be noted that the neighbor list is also an important factor for the final fuzzy membership.

**Interpretative analysis:**

Consider the following figure of a Cluster and the underlying Level-1 nodes. The Area of influence or radius is represented as Circles, whereas the centers are connected by straight lines. This area of influence is termed as the definitive area in which the fuzzy membership of the data is known. I would also be talking later of inclusion of error inside this fuzzy area of influence.

If the incoming data is represented by the ‘X’, it's reasonable to surmise that the by the distance between Level-1 nodes and ‘X’ should be lesser than the accumulated
Error at each of the nodes. In such cases the cluster shown in the Figure 4.21, by the level-1 node and corresponding Level-2 nodes would be enough to represent data ‘X’.

\[ X \text{ represents Data Sample} \]

**Figure 4.21** Detectable sample from Topology.

But in cases where the difference is too much even for the nearest level-1 node (to ‘X’) that means that the level-1 nodes are not enough to be representative to describe the data.

\[ X \text{ represents Data Sample} \]

**Figure 4.22** Undetectable sample from Topology.

If this happens with all the clusters that means that no cluster is representative of the incoming data. In such cases the data should be classified as “UNKNOWN” or
“NOT-REPRESENTATIVE”. A logical assumption might be that this data can be just lingering error or noise.

Note: as this algorithm is working like an accumulating algorithm, there is no guarantee that this new unclassified data is not a cluster and vice versa. There is always a tug of war between generalization and specialization that needs to be balanced, and its dependent on a very big extent on Lambda parameter of the GNG algorithm.

Now back to the assumption that the incoming data is representative enough than there are theoretical 2 possibilities. I would come back later on this point of non-representation and how to abate these effects of non-classification.

For the possibility that the sample data is representable, the incoming data might lie in overlapping Influence regions of 2 or more clusters or might be inside only 1 cluster.

Figure 4.23 Overlapping Cluster Regions.
For Figure 4.23, ‘X’ represents the test data. As ‘X’ is in influence regions of both clusters, it’s imperative to find the fuzzy membership of the nodes nearest to ‘X’ in the clusters controlling those nodes. In the above case, its Node A and Node B. Assume the cluster containing Node A is represented as “C” and that with Node B is represented by “D”.

Now I find the membership of X in both Cluster C and Cluster D and from that find in which cluster, X has the maximum membership, which is also the cluster in which X should lie.

This can be achieved by the following code and flowchart:

**Code snippet (Algorithm 4.6):**

For each Data

Find Membership of data in all the clusters

This can be done as finding memberships of the data in each of the level-1 nodes based on the distance and assigning node-membership when data lies in or on the hyper-spheres. And assigning “0” membership when the data is outside the hyper-sphere. Assigning the maximum membership from all calculated level-1 node for a given level-2 node(cluster) to the data is also the membership of the data in the level-2 node(cluster).

End

If all memberships are 0, return 0 and classify it as “UNKNOWN”

Else

*(code cont’d)*
Find the maximum of all memberships which is incidentally also the membership of the data in the cluster.

End

Flowchart:

Input: X, Cluster Matrix
Output: UNKNOWN or ClusterIndex

Test( ) function

Find all Clusters that have the distance between X and Level-1 nodes less than radius and name them ‘XClusters’

Size(XClusters)=0

Yes

Mark as UNKNOWN

Return Test( )

No

Find the fuzzy memberships of ‘X’ into each of the XClusters, and name them as XFuzzyMemberships

Find the Cluster having the maximum membership in XFuzzyMemberships, and name it as ClusterIndex

Figure 4.24 Flowchart for the Testing Routine(Representable Clusters).
Figure 4.24, is the flowchart for the Test() routine to find the fuzzy membership in cases where the sample data is representable enough by the network.

So when we assign fuzzy membership on basis of the topology and radius, we would be generating areas in space where the value of fuzzy membership is constant, which can be best expressed in the following diagram.

\[ \text{FMem} = \text{Fuzzy membership} \]

Please note that in the diagram given earlier, the nodes are represented in 2-Dimensional coordinate space. The projection of the fuzzy memberships of the nodes and affected by the radius is shown. If we take the projection of the fuzzy membership in both the X and Y axis, we can see that due to overlapping radius causing overlapping fuzzy memberships. So in those regions maximum membership among all overlapping nodes should be taken as the membership.

**Figure 4.25** Fuzzy Membership Projection by a Topology Network.
So for all overlapping fuzzy memberships the overlapping region is maxed out to the maximum membership from each node that is overlapping.

Now taking the earlier example, in which I showed the projection on X-axis, if the same projection is taken in n-dimensional a complex fuzzy membership region is generated. A sample 2D region is shown in next diagram. Note these fuzzy memberships can either be projected in each axial direction for creation of Max-Membership fuzzy rules, which can be advantageous for being independent of overlapping areas and also being able to generate rules that are most descriptive of the area. The major advantages I see here is there is a balance between the number of nodes (with the centers and width) overlapping regions and the density of input data.
Continuing with our example, we can describe the 2 Dimensional areas as

Due to the inherent max-memberships I am taking into consideration, the projection is straightforward and can serve an efficient system for generating Takagi-Sugeno based fuzzy clustering controller.

### 4.11 Error Regions Outside the Hyperspheres

The earlier approach though very stringent, is inefficient when it comes to the outlier data. The reason for which is the very nature of the Growing neural gas method and our calculation of the radius based on the neighbor distance. To augment the calculation for the outliers, it’s imperative to choose a smoother function for the max-membership that I calculate. I would now present a method to find those outliers or Unrepresentable data points.
To achieve that goal, it should be noted that with each node there is an mean squared error associated with each of the node. This error associated with the node also implies when I am associating a fuzzy membership with the node, I should exercise some amount of leeway with the node. So by accounting error into calculation, I generate hyper-spheres but with fuzzy membership outside these hyper-spheres decreasing as some function of error.

Let’s take the earlier 2-Dimensional example,

![Fuzzy Membership Projection with Smoothed Out Error](image)

**Figure 4.28** Fuzzy Membership Projection with Smoothed Out Error.

As you can see in Figure 4.28, instead of sharp partitions of max-memberships, I am smoothing the membership regions outside the hyper-spheres of each of the node as some function of error. Any decreasing monotonic function can be taken for this. Now for much of the experiments done by me this decreasing function was the exponentially
decreasing $e^k$ function. I used the following short code to assign the membership for the data for each node.

**Code snippet (Algorithm 4.7):**

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 $\text{Distance(Node 1 Center, Input Data)} \leq \text{Radius(Node 1 )}$

Membership of Input Data = Membership of node 1

Else

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

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

Else

Membership of Input Data = 0

End

End

The above is just a conditional statement code to verify if the data lies in the Radius region or the Radius+$\sqrt{\text{Error}}$ region or outside both of this region, and then assigning valid memberships. Please note that though I have taken exponentially decreasing $e^k$ over here, it's possible to take other monotonically decreasing function over here. More discussion about such a choice is discussed later. This above code does
minimum change to the fuzzy regions I have been talking of but just makes classification at outlying regions possible.

Let's see the effects of this decreasing Radius+\sqrt{Error} Hyper-sphere region on the fuzzy membership in the next figure

![Figure 4.29 Overlapping Fuzzy Areas of Influence with Error considered.](image)

Note: The area between the Radius region and Radius+\sqrt{Error} region is a decreasing fuzzy membership region. This would take care effectively the outlier data region. The inclusion of Error region would smoothen out the Hypersphere.

**Error Fraction:**

Before going further, I would like to stress 1 thing, its not important to take the whole error in the hyper-sphere calculations, one approach is to use a fraction of the
Error. I will be showing various results obtained when using different amounts of error fractions.

**Figure 4.30** Plot of Hyperspheres with Nodes represented by ‘diamonds’.

The total non-classified data points in the IRIS dataset without considering Error radius is 89 out of 150.

**Figure 4.31** Plot of Hyper-spheres with Nodes represented by ‘diamonds’

Error fraction is 4.
The total non-classified data points now drops to 29 out of 150. Still many data points are non-classified so it's necessary to increase the error radius.

![Image of Hyper-spheres with Nodes represented by 'diamonds'.](image.png)

**Figure 4.32** Plot of Hyper-spheres with Nodes represented by 'diamonds'. Error fraction is 2.

With an Error fraction of 2 the non-classified data drops to 5. The Error fraction has to be set to 1 for the maximum error radius and thereby total classification.

From this I can conclude that the inclusion of Error in calculating the Area of Influence of a Hyper-sphere would be a better measure to test than using just the Radius. The use of Error-fraction is up to the user, as the only task of the error fraction is to tighten or loosen the area outside the Hyper-sphere.

Now the original algorithm about testing can be modified to newer form to account for the Radius +\sqrt{Error} region, and is mentioned in the flowchart in figure 5.13. This algorithm will be than able to find both representable and unrepresentable Hyper-spheres.
Find the Cluster having the maximum membership in XFuzzyBemberships, and name it as ClusterIndex

Input: X, Cluster Matrix
Output: UNKNOWN /ClusterIndex

Find all Clusters that have the distance between X and Level-1 nodes less than radius + √error associated with the node and name them ‘XClusters’

Size(XClusters)=0

Yes

Mark as UNKNOWN

Return Test( )

No

Find the fuzzy memberships of ‘X’ into each of the XClusters, and name them as XFuzzyMemberships. Also use the smoothing based on error in region out of the radius but lesser than radius + √error.

Find the Cluster having the maximum membership in XFuzzyMemberships, and name it as ClusterIndex

Figure 4.33 Flowchart for the Testing Routine (Representable and Unrepresentable Clusters).
4.12 Choice of Error function for Hypersphere

As I had mentioned earlier that the only criteria, is choosing a monotonically decreasing function in the region between Radius to Radius+$\sqrt{\text{Error}}$. The choice is all up to the user on whether a strict or a lenient distinction of the error region is desired.

The effect of choice of such criteria is very much dependent on the results obtained when using particular datasets.

Let's see some examples based on when using Radius+$\sqrt{\text{Error}}$ with different types of smoothing functions.

![Diagram of Fuzzy Membership Projection with Linear Error Region.](image)

**Figure 4.34** Fuzzy Membership Projection with Linear Error Region.

When using the linear region the overlap would be more than when using an exponential decreasing function. So it's more or less on the user to choose a function that is best suited for his/her dataset and that which gives the best output.
4.13 Advantages and Disadvantages of TFC

Let me sum up the advantages and disadvantages of using TFC as a clustering algorithm, and what else enhancements remains much desired.

Advantages:

1. The sub-clusters are able to represent the local topology error quite well. Complex shapes can be represented by the hyperspheres and their connections via edge structure, analogous to a bead necklace with beads connected with a string.

2. Testing is a straightforward task, as only the Max fuzzy membership is taken into account.

3. Learning is straightforward as it is localized, and is not so computationally intensive compared to a Neural-Network based algorithm.

4. The execution time is linearized in comparison with the data, unlike the unknown time that might take for execution for a C-Means based algorithm.

Disadvantages:

1. Need to set multiple parameters.

2. Too much unsupervised in nature. Due to the complex interactions of parameters it’s not straightforward on how to choose a particular number of parameters to get a specific number of clusters.
Desired:

Ability to feed supervised information to the network, like the number of clusters and correctly classified data. The next section is devoted on how to extend TFC to a more adaptive algorithm called the Adaptive Topology-Based Fuzzy Clustering (ATFC) by inclusion of supervised Learning

4.14 Supervised Learning

Till this point, the reader has been exposed only to the unsupervised nature of the TFC Algorithm. The unsupervised nature of the TFC algorithm though comes at the cost of unmanageability on the part of the user, as an expert user cannot incorporate the knowledge he/she has about the cluster distribution to influence the learning of the TFC algorithm. This chapter section is devoted on converting the basic TFC algorithm to a more adaptive algorithm called as the Adaptive Topology-Based Fuzzy Clustering (ATFC) algorithm.

As discussed in the earlier section the most desired capabilities are the inclusion of supervised parameters. These parameters can be drawn down to one thing that is including Expert knowledge on Classified Data. This Expert knowledge can be further classified as

1. An Expert’s ability to classify clusters and ask the system to split or merge the clusters as deemed necessary.

2. An Expert should be able to comment on the classification of TFC and the system should be able to learn better on basis of the expert’s classification. This case arises when TFC does a misclassification.
Let's next see how these two factors are incorporated into ATFC.

4.15 Structure of ATFC Algorithm

As ATFC is just a supervised learning extension to the TFC algorithm, the algorithmic part is the same. The incremental change done is adding a supervised layer to influence the Level-1 nodes. A system overview is shown in the following Figure 4.35.

![Figure 4.35 Adaptive Topology-Based Fuzzy Clustering.](image)

As shown in Figure 4.35, the Adaptive nature of the system comes from including the expert's feedback. This feedback leads to the modification of the Level-1 node vector position in the TFC algorithm.

So how does the experts knowledge is incorporated in ATFC?

As I mentioned in the last section, the Expert should be able to merge and split clusters as deemed necessary. This can be done via observing the node positions.
4.15.1 Supervised Total Merging and Total Splitting

For example, at every certain interval, the Expert is asked opinions on what he/she thinks are the Clusters for the Level-1 nodes. A pseudo trace available to the Expert is shown next:

**Before Expert Knowledge:**

*Cluster* = 1

*Corresponding Level-1 Node Number* = 1, 2, 3, 4, 5, 6

*Vector Positions of Level-1 Nodes are also available.*

*Total Level-1 Nodes* = 6

*Total Clusters* = 1

After analyzing the Vector Positions of the Level-1 Nodes, the expert decides that the Cluster is incorrectly connected and there are in-fact 2 clusters as mentioned below.

**After Expert Knowledge (and due to ATFC):**

*Cluster 1*

*Corresponding Level-1 Node Number* = 1, 2, 3, 4

*Cluster 2*

*Corresponding Level-1 Node Number* = 5, 6

*Total Level-1 Nodes* = 6

*Total Clusters* = 2

So basically if we consider the topology we are creating two separated topology graph, by deleting the edges. Do examine the following Figure 4.36 for a pictorial view of such a split.
Continuing with the example, as shown in the Figure 4.36, the Topology is described before taking the Expert's input. As the expert has asked the topology to be split into 2 clusters, there is a need to delete all the connecting edges between both the clusters and the topology would look as shown in the following Figure 4.37.

Figure 4.36 Pictorial View in 2 Dimension of a Topology (Before Expert's Input).

Figure 4.37 Pictorial View in 2 Dimension of a Topology (After Expert's Input).
The above method is used whenever a supervised cluster splitting or merging is required. I would like to mention further more on Splitting and Merging next in such supervised learning.

During splitting, as multiple edges might be deleted, it’s important to note that disjoint sub-graphs are created; else there is a chance that the C (Edge) matrix might get bad. Also note, in splitting; only the C (Edge) matrix will be modified, due to which the method of splitting becomes very straightforward and can be written down in pseudo code as.

**Total Splitting Algorithm (Algorithm 4.8):**

1. Get the Expert list of the nodes and their respective clusters.
2. Create multiple lists of the clusters and nodes as per the Expert.
3. Make sure that only edges between nodes of the same clusters are retained, all other edges are reset to -1, meaning there is no connection between the clusters. This step can be done iteratively.

Using the Algorithm 4.8, a split topology like in Figure 4.37 can be generated. I would be next describing how a supervised merging can be done for the clusters.

Let me show the reverse of the example I described earlier,

**Before Expert Knowledge:**

*Cluster 1*

*Corresponding Level-1 Node Number =1,2,3,4*

*Cluster 2*
Corresponding Level-1 Node Number = 5, 6
Total Level-1 Nodes = 6
Total Clusters = 2

And the Expert suggests making 2 clusters as

After Expert Knowledge (and due to ATFC):

Cluster = 1
Corresponding Level-1 Node Number = 1, 2, 3, 4, 5, 6
Total Level-1 Nodes = 6
Total Clusters = 1

The simplest way to do such a merging would be taking both the clusters and finding out the nearest nodes between the clusters, and creating a newer edge between those two nodes. For our example the nearest nodes in both the cluster are either 2 and 6 or 4 and 5 (Please refer Figure 4.37). The algorithm to do such a merging is described next.

Total Merging Algorithm (Algorithm 4.9):

Step 1. Get the Expert list of the nodes and their respective clusters.
Step 2. Create multiple lists of the clusters and nodes as per the Expert.
Step 3. For each 2 clusters to be connected, find 2 nodes nearest to each other in both clusters, and connect with an edge. Do till the topology is as described by the expert.
As the reader can notice such an algorithm doesn’t preserve the TPC’s topology, which is alright as we do consider the expert’s feedback much more important compared to the unsupervised learning process.

A question that might arise in the reader’s mind would be the algorithmic complexity of the Total Split and Total Merge process. As during these procedures, we analyze whether a connection lies in case of Total Split (or find the distance in case of Total Merge) between each node and all the other nodes between them; would cause a total of \( n^2 \) such comparisons. Thus the algorithmic complexity of both the procedures is \( O(n^2) \).

4.15.2 Learning through Misclassification

The other problem with TFC was teaching the algorithm the misclassified data. Let’s first see what type of expert knowledge can be obtained.

Let’s consider that TFC with a reasonable good performing set of parameters is taken as the part of ATFC. Now the modifications to TFC can be done either in modifying the set of parameters or by modifying the position of the Nodes. Modifying the set of parameters is not recommended as it can cause invalidation of the distribution already learnt by the network. As the data distribution is stored in term of node position, it’s imperative to use the node positions as the place to modify.

As seen in the earlier sections on testing, the Test() function has the ability to tell the clusters in which the test data was classified in. The expert can point out the misclassified data points which can be incorporated back into the TFC. The procedure to modify TFC is given below.
Single Step Adaptive Algorithm (SSAA) Variation-1 (Algorithm 4.10):

1. For each misclassified Data (say $X_1, X_2, \ldots, X_n$), try to generate a new node at that data point and merge it with the cluster it should go with, as asked by the expert.

2. Now retest all the test data, to see if the new node at $X_k,(1 \leq k \leq n)$ has not caused more misclassification. Do this for each misclassified data point.

3. Find $X_p,(1 \leq p \leq n)$, such that it causes lesser misclassification than TFC and least misclassification for all $X_1, X_2, \ldots, X_n$ not including $X_p$.

Please Note that in Step-3, there can be more than one node that has the lowest misclassification rate. In that case, find the take the sum of the fuzzy values by those nodes and choose the node having the lowest sum fuzzy membership value. The inference for doing this is ‘the node to be added should cause least fuzzy membership for misclassified data’. The reason for this is by choosing the node having the lowest sum fuzzy membership for the misclassified data would give better cluster partitions.

There might be other measures also like using minimum fuzzy membership caused for the misclassified data, or using the minimum variance done to the fuzzy membership by the introduction of the node. The reader is encouraged to perform experiments to determine the best measure.

Please note as the SSAA merges a single node during a single run, multiple runs can be used to obtain the best merge partitions. Viable options are to perform SSAA for each cluster once, and save the obtained topology information.

Multiple runs over different misclassified points should be performed, with the best performing run to be taken as the final topology configuration. Problems with
multiple run would be the exhaustive testing required, whose time requirements would rise exponentially with the amount of misclassified data. My experiments later shows that SSAA is normally good enough to reduce the misclassification rate by a huge margin.

A question that might arise in the reader’s mind would be “whether the algorithm can mark certain inputs as another cluster, so as the expert can mark such clusters and increase the efficiency?”

As the reader might notice, SSAA can be used only when the cluster for the given data exists, than a merge of the data into the cluster occurs. It would not work in cases in which the cluster needed to integrate the data is not formed yet. In such cases when a new cluster has to be created for the data, the SSAA has to be modified to create such new clusters. I would first propose the algorithm, before merging into SSAA. The task of the algorithm is allowing the expert to mark certain inputs into a newer cluster, so as to aid cluster discovery.

What modifications would be necessary in the TFC to allow such cluster creation?

Topology till this point has always aided us, but here topology can cause havocs. The reason could be cited that a short line with 2 nodes will have a fuzzy membership of 1. Consider the following Figure 4.38.

![Overlapping Clusters with high memberships.](image)

**Figure 4.38** Overlapping Clusters with high memberships.
As can be seen in the Figure 4.38, due to the topology the cluster that is overlapping causes undue fuzzy membership around it. The lighter cluster has both the nodes of a fuzzy membership of 1. But the Area of Influence is much larger, this would be the cause of more than required number of inputs classified into the smaller clusters. This case would be real problem if user input are taken into consideration and such small clusters are created. Note that this would not be a problem when a good construction is done by the unsupervised TFC.

In such expert made clusters, it would be wise to ignore the overlapping error fraction and just take into consideration the exact radius, or even reduce the radius if needed.

How to create a new cluster with a single data point?
The other factor is the expert will be just giving 1 data point to be classified as a new cluster. An intuitive idea would be to generate a single cluster node at that data point. The problem is that the Reference and Fuzzy value algorithm (Algorithm 4.1) would not work with a single node. It’s necessary for even the smallest cluster to have at least 2 nodes. So assuming that a node is created at the point where the expert points out the new cluster, the second node has to be created near it. The way to do this is shown in the following figures.

Figure 4.39 Insertion of a Data point as a Node(1).
As depicted in the Figure 4.39, first find the nearest node in any nearby cluster. Such a node is shown connected with the data point by a dashed line in the Figure 4.39.

\[ X = \text{Data point} \]
\[ -- = \text{Virtual Edge to the Nearest Node} \]

**Figure 4.40** Insertion of a Data point as a Node(2).

As shown in Figure 4.40, insert a new node at exactly half the distance between data point ‘X’ and nearest node.

\[ X = \text{Data point} \]
\[ -- = \text{Virtual Edge to the Nearest Node} \]

**Figure 4.41** Insertion of a Data point as a Node(3).

As shown in the Figure 4.41, Also insert a new node at the data point at ‘X’ and connect to the node we created at Figure 4.40, which also makes a new cluster in itself. This is the method to create clusters when such data points are pointed to by the user. When multiple data points are to be created and are to be in the same cluster, all such
data points are to be created as a node and connected with each other as shown in the following Figure 4.42.

\[ \text{X, Y} = \text{Data point} \]
\[ \text{-- = Virtual Edge to the Nearest Node} \]

**Figure 4.42** Insertion of a Data point as a Node(4).

Now let me present the modified SSAA next:

**Single Step Adaptive Algorithm (SSAA) (Algorithm 4.11):**

1. For each misclassified Data (say \( X_1, X_2, \ldots, X_n \)), try to generate a new node at that data point and merge it with the cluster it should go with, as asked by the expert and proceed to Step 2. If Expert ask to integrate data point as a new cluster than proceed to Step 4.
2. Now retest all the test data, to see if the new node at \( X_k, (1 \leq k \leq n) \) has not caused more misclassification. Do this for each misclassified data point.
3. Find \( X_p, (1 \leq p \leq n) \), such that it causes lesser misclassification than TFC and least misclassification for all \( X_1, X_2, \ldots, X_n \) not including \( X_p \). Proceed to Step 7.
4. Find the Nearest Level-1 Node ‘P’ for ‘X’ and create a new Node ‘Y’ at half the distance between X and P. Vectorially, \( Y = X + \frac{(P - X)}{2} \)
5. Connect Y and X with an edge.
6. Create a new Cluster consisting of Y and X points.
7. End.
The above variation of SSAA would be able to both adapt if the Expert asks for the creation of a new cluster for the data point or if the Expert asks for merging the data point into an existing cluster.

In the next chapter I would be discussing results based on the methods discussed in this chapter.
CHAPTER 5
RESULTS AND DISCUSSION

5.1 IRIS Dataset

IRIS Dataset is perhaps the most famous dataset used in pattern classification. It 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. These data are commonly referred to as the "Fisher Iris Data".

![Figure 5.1 Scatter Plot for the IRIS Dataset.](image-url)
As seen in the Figure 5.1, of the 3 species of Iris flowers, only Iris Setosa is completely separated from the other classes, but there is overlap between the Iris Versicolor and the Iris Virginica.

Now let’s see some common outputs from the GNG algorithm and compare with the output from my algorithm. First let me show how the positions of nodes would be mentioned in comparison with data in 3D space. Note the 4th data point Petal Length is not considered in the scatter plot.

![Node-Data Display in 3-Dimensional Topology in GNG](image)

**Figure 5.2** Node-Data Display in 3-Dimensional Topology in GNG.

As shown in Figure 5.3, the nodes are represented in a lighter color. Let’s now continue with an example of the nodes as generated by the GNG. Please note that this example is represented for relation of the nodes in the 3D space, it’s just about the analytical interpretation of the nodes, and no mathematical inference should be born out of it.
Figure 5.3, is the results from the standard GNG algorithm with the parameters $\lambda = 10, e_b = 0.04, e_n = 0.0006, \alpha = 0.5, \beta = 0.0005, a_{\max} = 12$. The reader is asked to consider these as the standard parameters I used for IRIS dataset throughout this chapter, unless directed otherwise. From the figure, the reader is asked to observe the nodes marked for the Iris Versicolor, which are slightly lagging or off the place for the cluster. The position of the points should be more towards the denser part of the cluster than the sparse part. The reason for this could be cited to the static learning rate for all clusters, because of which even though the connection was broken between Iris Setosa cluster and Iris Virginica cluster, the node involved had not been able to learn as much as what is desired. In short it would be better qualitatively if the GNG node was in the midst of the cluster than lying in the outlier.

Now let’s see next the same results when see from another vantage point. This would give also a qualitative view on the node positions.
Figure 5.4  GNG, Node-Data Display from Another Vantage Point.

Figure 5.4, shows the Nodes from another vantage point, and as seen the position of the nodes by GNG is slightly off the desired positions. Let’s now see the position of the nodes as pointed out by my algorithm.

Figure 5.5  Node Positions according to TPC.
Let's see the same from another vantage point in Figure 5.6.

As we can interpret from all the above results, using my algorithm does seem to place the nodes more nearer to the position of the data, compared to GNG algorithm.

Now I will be presenting more results based on analyzing the IRIS Dataset with reference to the FCB, Gath-Geva and against my algorithm based on the Classification-Misclassification rates.

The methodology was finding the classes with m=3 for both FCM and Gath Geva, and using $\lambda = 10, e_p = 0.04, e_a = 0.0006, \alpha = 0.5, \beta = 0.0005, a_{\text{max}} = 12$, Error Fraction=1 for my algorithm. The results are presented in the following Table 5.1.
Table 5.1 Misclassification rate for IRIS data using Various Algorithms.

<table>
<thead>
<tr>
<th>Classes</th>
<th>FCM</th>
<th>Gath-Geva</th>
<th>TFC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m=1.5</td>
<td>m=2</td>
<td>m=3.5</td>
</tr>
<tr>
<td>Iris Setosa</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Iris Versicolor</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Iris Virginica</td>
<td>12</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>Total Misclassified</td>
<td>15</td>
<td>16</td>
<td>14</td>
</tr>
</tbody>
</table>

From the Table 5.1 its observable that TFC performs very well for the IRIS data, with a lower misclassification rate than FCM and GNG. The misclassified data in my algorithm is mostly in classifying Iris Virginica as the Iris Versicolor flower.

Figure 5.7 Misclassified Data points (Virginica as Versicolor).
This is because of the peculiar nature of the position of the Iris Versicolor clusters’ Level-1 node. This can be seen in the Figure 5.7. The nodes that are misclassified are represented by diamonds.

Now I will be presenting the advantages of using the ATFC algorithm on IRIS, and compare the results it with other methods like fuzzy rules and neural networks. As I came to know that the unsupervised TFC was not able to classify 10 data points from Iris Virginica and 1 data point from Iris Versicolor, ATFC’s Single Step Adaptive Algorithm (SSAA) (Algorithm 4.8) was fed those misclassified points.

**Table 5.2** Classification rate for IRIS data using TFC and ATFC.

<table>
<thead>
<tr>
<th>Classes</th>
<th>TFC</th>
<th>ATFC-SSAA(for Iris Virginica only)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Misclassified</td>
<td>Classification Rate</td>
</tr>
<tr>
<td>Iris Setosa</td>
<td>0</td>
<td>100%</td>
</tr>
<tr>
<td>Iris Versicolor</td>
<td>10</td>
<td>80%</td>
</tr>
<tr>
<td>Iris Virginica</td>
<td>1</td>
<td>98%</td>
</tr>
<tr>
<td>Total Misclassified</td>
<td>11</td>
<td>92.7%</td>
</tr>
</tbody>
</table>

As seen by using ATFC I am able to reduce the number of misclassification by 5 misclassifications. Please note that SSAA was performed only on the Iris Virginica Cluster above. 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 5.8.
The next thing done by me was using multiple runs of SSAA, the results were not so good. The reason for this could be cited to more than required overlapping being caused. The results also became worse when used on the Iris Versicolor (1 data point of class Iris Versicolor was misclassified as Iris Virginica). So it's reasonable to assume that in general a heuristic estimation has to be done for including misclassified values using SSAA with the ATFC algorithm.

Now the parameters $\lambda = 10, e_p = 0.04, e_s = 0.0006, \alpha = 0.5, \beta = 0.0005, a_{max} = 12$, do create a better partition creating 3 clusters, but this is not the case every time with all the datasets. Note the Error fraction is still kept 1. 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. As
the reader might recall, these algorithm called as Total Merging (Algorithm 4.9) and Total Splitting (Algorithm 4.8) are discussed in the last chapter.

Let's first see the output generated when the parameters chosen are $\lambda = 10, e_y = 0.04, e_y = 0.0006, \alpha = 0.5, \beta = 0.0005, a_{max} = 14$.

![Graph showing clusters](image)

**Figure 5.9** Only 2 Clusters created for IRIS Dataset by TFC.

As seen from the Figure 5.9, 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.

The expert is asked about the opinions on where the nodes and the data is tabulated in the Table 5.3 given next.
Table 5.3 Node Position from TFC and Expert Feedback.

<table>
<thead>
<tr>
<th>Node Number</th>
<th>Vector Positions</th>
<th>Cluster Number</th>
<th>Expert's Numbering</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Iris Setosa = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Iris Virginica = 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Iris Versicolor = 3</td>
</tr>
<tr>
<td>1</td>
<td>4.9931 3.3601 1.3828 0.1958</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4.9626 3.1956 1.4246 0.2139</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>7.0123 3.0779 5.8370 2.0229</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4.8475 3.1933 1.4349 0.2160</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5.0667 3.4112 1.5114 0.2768</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>5.1531 3.5049 1.4796 0.2586</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>5.4213 3.4164 2.0999 0.4930</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>5.5824 3.1502 2.8594 0.8059</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>5.7497 2.8733 3.5546 1.0405</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>5.7807 2.8242 4.0769 1.2609</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>5.9938 2.8671 4.2543 1.3179</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>6.1525 2.8299 4.8814 1.6778</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>6.4510 2.9192 5.2274 1.7856</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>6.6609 3.0542 5.4676 2.0179</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>6.8517 3.0649 5.6623 2.0187</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
The expert aids the algorithm by showing that the Nodes numbered 8-12 were misclassified and so makes a new class for them. This would differentiate the Iris Versicolor from the Iris Virginica as shown in the next Figure 5.10.

![Figure 5.10 TFC creates 3 Clusters for IRIS Dataset.](image)

Due to the expert’s opinion, the network has come to know the existence of all the clusters. The misclassification rate in above example was 20 out of 150 but 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 be used when there is more than 1 cluster. This example shows the effectiveness of TFC and ATFC against a standard dataset. 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.
5.2 Simulated Dataset

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.

![Simulated Dataset with 3 Clusters](image)

**Figure 5.11** 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:

- **Circle** centered at (0,0) with a radius of 13 and a width of 1.5 for the circumference. That is the radius of the distribution points varies from 11.5 to 14.5.

- **Semicircle** centered at (0,0) with a radius of 9 and a width of 1 for the circumference. That is the radius of the distribution points vary from 8 to 10.
Smaller Arc centered at (0,0) with a radius of 5 and a width of 1 for the circumference.

That is the radius of the distribution varies from 4 to 6.

**Table 5.4** Simulated Dataset Distribution Parameters.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Center</th>
<th>Radius</th>
<th>Variation Width</th>
<th>Variation Min</th>
<th>Variation Max</th>
<th>in Radians Start</th>
<th>in Radians End</th>
<th>Distribution Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle</td>
<td>(0,0)</td>
<td>13</td>
<td>1.5</td>
<td>11.5</td>
<td>14.5</td>
<td>0</td>
<td>2π</td>
<td>10000</td>
</tr>
<tr>
<td>Semicircle</td>
<td>(0,0)</td>
<td>9</td>
<td>1</td>
<td>8</td>
<td>10</td>
<td>0</td>
<td>π</td>
<td>5000</td>
</tr>
<tr>
<td>Arc</td>
<td>(0,0)</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>3000</td>
</tr>
</tbody>
</table>

Such a type of distribution was generated and then a test dataset was prepared.

The test dataset had the parameters as shown next.

**Table 5.5** Test Dataset Distribution Parameters.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Center</th>
<th>Radius</th>
<th>Variation Width</th>
<th>Variation Min</th>
<th>Variation Max</th>
<th>Start in Radians</th>
<th>End in Radians</th>
<th>Distribution Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circle</td>
<td>(0,0)</td>
<td>13</td>
<td>1</td>
<td>11</td>
<td>15</td>
<td>0</td>
<td>2π</td>
<td>10</td>
</tr>
<tr>
<td>Semicircle</td>
<td>(0,0)</td>
<td>9</td>
<td>1</td>
<td>7.5</td>
<td>10.5</td>
<td>0</td>
<td>π</td>
<td>10</td>
</tr>
<tr>
<td>Arc</td>
<td>(0,0)</td>
<td>5</td>
<td>1</td>
<td>3.5</td>
<td>6.5</td>
<td>4</td>
<td>5</td>
<td>10</td>
</tr>
</tbody>
</table>

The parameters as set in ATFC are as follows:

\[ \lambda = 300, e_b = 0.04, e_a = 0.0006, \alpha = 0.5, \beta = 0.0005, a_{\text{max}} = 350, \text{Error Fraction}=1 \]
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 5.6 next, would be helpful, to note an expert’s comments.
Table 5.6  Simulated Dataset Learnt Node Position.

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

<table>
<thead>
<tr>
<th>Node</th>
<th>Position X</th>
<th>Position Y</th>
<th>Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>ATFC</td>
</tr>
<tr>
<td>31</td>
<td>11.4584</td>
<td>-5.7416</td>
<td>1</td>
</tr>
<tr>
<td>32</td>
<td>12.3665</td>
<td>-3.4851</td>
<td>1</td>
</tr>
<tr>
<td>33</td>
<td>11.6753</td>
<td>-1.2015</td>
<td>1</td>
</tr>
<tr>
<td>34</td>
<td>9.7271</td>
<td>0.7566</td>
<td>1</td>
</tr>
<tr>
<td>35</td>
<td>8.7736</td>
<td>2.5237</td>
<td>1</td>
</tr>
<tr>
<td>36</td>
<td>7.9152</td>
<td>4.1021</td>
<td>1</td>
</tr>
<tr>
<td>37</td>
<td>6.9357</td>
<td>5.5277</td>
<td>1</td>
</tr>
<tr>
<td>38</td>
<td>5.7936</td>
<td>6.7572</td>
<td>1</td>
</tr>
<tr>
<td>39</td>
<td>4.4128</td>
<td>7.7138</td>
<td>1</td>
</tr>
<tr>
<td>40</td>
<td>2.9054</td>
<td>8.4262</td>
<td>1</td>
</tr>
<tr>
<td>41</td>
<td>1.2473</td>
<td>8.8105</td>
<td>1</td>
</tr>
<tr>
<td>42</td>
<td>-0.4473</td>
<td>8.8788</td>
<td>1</td>
</tr>
<tr>
<td>43</td>
<td>-2.089</td>
<td>8.6267</td>
<td>1</td>
</tr>
<tr>
<td>44</td>
<td>-3.6582</td>
<td>8.0833</td>
<td>1</td>
</tr>
<tr>
<td>45</td>
<td>-5.0852</td>
<td>7.2286</td>
<td>1</td>
</tr>
<tr>
<td>46</td>
<td>-6.3699</td>
<td>6.1385</td>
<td>1</td>
</tr>
<tr>
<td>47</td>
<td>-7.4041</td>
<td>4.8447</td>
<td>1</td>
</tr>
<tr>
<td>48</td>
<td>-8.1802</td>
<td>3.3244</td>
<td>1</td>
</tr>
<tr>
<td>49</td>
<td>-8.405</td>
<td>1.6199</td>
<td>1</td>
</tr>
<tr>
<td>50</td>
<td>-5.2101</td>
<td>-1.7774</td>
<td>1</td>
</tr>
<tr>
<td>51</td>
<td>-3.4284</td>
<td>-3.5355</td>
<td>1</td>
</tr>
<tr>
<td>52</td>
<td>-0.891</td>
<td>-4.9615</td>
<td>2</td>
</tr>
<tr>
<td>53</td>
<td>-1.1669</td>
<td>-4.5784</td>
<td>2</td>
</tr>
<tr>
<td>54</td>
<td>0.6341</td>
<td>-4.6132</td>
<td>2</td>
</tr>
<tr>
<td>55</td>
<td>0.7947</td>
<td>-5.2147</td>
<td>2</td>
</tr>
<tr>
<td>56</td>
<td>1.4546</td>
<td>-4.5012</td>
<td>2</td>
</tr>
<tr>
<td>57</td>
<td>-4.3193</td>
<td>-2.6564</td>
<td>1</td>
</tr>
<tr>
<td>58</td>
<td>0.8662</td>
<td>-4.9093</td>
<td>2</td>
</tr>
<tr>
<td>59</td>
<td>1.0713</td>
<td>-4.8665</td>
<td>2</td>
</tr>
</tbody>
</table>
After incorporating the expert's feedback in ATFC the corresponding topology is generated as shown in the next Figure 5.13.

**Figure 5.13** 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 5.14** 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 5.7 Test Data Classification (Simulated Dataset).

<table>
<thead>
<tr>
<th>Class</th>
<th>Real Class</th>
<th>Calculated Classes from ATFC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Total Classification</td>
</tr>
<tr>
<td>Circle</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Semicircle</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>Arc</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

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 5.15.

![Figure 5.15 Fuzzy Area of Influence.](image-url)
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 5.16 Fuzzy Memberships represented by Circles.

In Figure 5.16, 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. I 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). In short all types of shaped cluster are detectable, provided there exists some kind of differentiation between those shapes.

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 linearized in time, ATFC (and TFC) is a viable algorithm for application to cluster non-elliptical shaped clusters, and theoretically any shaped cluster.

5.3 NASA Dataset

The NASA Dataset is a dataset that was previously used in ALEC by Dai et al[16], and was obtained from the reading from a fuel tank. ALEC is primarily designed to detect events and mark those events adaptively based on user inputs.

The events of interest in this dataset are

1. PV07 Commanded Open.
2. PV02 Commanded Open/PV02 Open Command Discontinued.
3. SV02 Commanded Open/SV02 Command Discontinued.

Figure 5.17 MPRE301P Signal (50Hz), Events of Interests.
The dataset shown in Figure 5.17 is a smaller segment of the MPRE301P at 50Hz showing the events of interests that need to be detected. The whole dataset named MPRE301P shown in Figure 5.18, had 9052 samples and starts at 9,229,000 ms and ends at 9,410,020 ms and sampled at 50Hz. Only the last segment consisting of 5,000 data points is used in this section and is shown in Figure 5.17. The results are almost similar for the remaining sections of the signal.

![Figure 5.18 MPRE301P Signal (50Hz), full segment.](image)

The interesting thing about this dataset compared to the other datasets discussed yet, is the objective here is the discovery of a very less amounts of events in a large amount of signals. It's like comparing to find a needle in a haystack. The amount of data size can cause problems for C-Means based algorithms, as other than certain events, the whole dataset is a valid dataset, and doesn't make a difference if we consider the whole
dataset (without the events) as a single cluster or multiple clusters. Thus the finding of cluster is rather an indirect process, a more important objective is finding the abnormalities or events.

The last segment I am working on looks like the following Figure 5.19, without the events.

![Figure 5.19 MPRE301P Signal (50Hz), full segment without events.](image)

The reader is asked to refer to Dai[16] on the methodology, he used to find the events in the dataset, which is known as Adaptive Hierarchical Fuzzy Clustering (AHFC). The brief steps from Dai[16] are mentioned below:

1. Select a data segment size. An optimum data segment size for this case would be 1000 samples.

2. Perform three-level wavelet decomposition.

3. Get the average value of this data segment to obtain the additional features from the signal.
4. For the initial unsupervised learning phase, do hierarchical clustering to find initial clusters and initialize database using the predefined event list.

5. Verify initial clusters and database with domain experts.

6. For subsequent learning phase, perform step 1-3 on the incoming signals. Once the signal information has been processed and collected, perform the fuzzy k-means clustering and calculate the membership function with previous clusters. New clusters may be added based on leader-follower algorithm and flag the event if it is not in the database for user review.

7. Repeat step 6 until all the signal data has been processed.

According to Dai[16], a three-level decomposition using db2 wavelet was applied to the data segment, and it looks like in the following Figure 5.20 with the wavelet tree shown in Figure 5.21.

![Figure 5.20 3-Level Wavelet Decomposition on MPRE301P (50Hz).]
The vector constructed consisted of $A_3, D_3, D_2, D_1$ a set of 4 parameters.

My next task was to try to classify the clusters using ATFC, so ATFC was first trained with the first 2000 samples and then tested on the rest of the 3000 samples. The parameters as used with ATFC are as follows

$$\lambda = 15, e_p = 0.04, e_n = 0.0005, \alpha = 0.5, \beta = 0.0005, a_{\text{max}} = 22$$

Initially the clusters were not detected. There were 2 data points that were not classified at all. Those points are pointed by Blue diamonds in the next Figure 5.22.

\[ S = A_1 + D_1 \]
\[ = A_2 + D_2 + D_1 \]
\[ = A_3 + D_3 + D_2 + D_1 \]
The reason for these Unclassified points is because of the noise-resistance of the ATFC algorithm, due to which even though the Error Fraction of 1 was taken the points were too off to be counted in a cluster. The expert (in this case me) was notified of both the unclassified data and possible events missed.

**Table 5.8 Test Data Classification (MPRE301P) – TFC.**

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Number of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4998</td>
</tr>
<tr>
<td>Unclassified</td>
<td>2 at (9334920, 9335000)</td>
</tr>
</tbody>
</table>

SSAA (algorithm 4.11) was used in the case of missed events and fed back to the algorithm, and the 3000 sample points were again tested. The results are shown in the following Figure 5.23, with the red diamonds showing the now newly classified data points. The expert has pointed an event at 9334920.

![Figure 5.23 Testing after expert's comments.](image-url)
Table 5.9 Test Data Classification (MPRE301P) – ATFC.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Number of points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4998</td>
</tr>
<tr>
<td>2</td>
<td>7 at (9326760, 9334880, 9334960, 9335000,</td>
</tr>
<tr>
<td></td>
<td>9362420, 9362440, 9392360)</td>
</tr>
</tbody>
</table>

Of the 7 events highlighted by ATFC, except the event at 9326760, all other events correspond to the opening of the valve. The above results now are due to the supervised inclusion of the expert’s comments. Let’s see the results when the expert’s comments on another data point 9311680.

Figure 5.24 Testing after expert’s comments.

The preceding Figure 5.24 corresponds to the events detected in a new cluster when the data point at 9311680 is converted to a cluster by SSAA. When a second set of supervised inputs were fed back by the expert into the ATFC by the expert more events
are found by ATFC as shown in Figure 5.25 (both the events at 9311680 and 9334920 are pointed by expert).

![Graph showing data points](image)

**Figure 5.25** Testing After Inclusion of more Data points by Expert.

Though the topology pointing is a powerful mechanism it is a clumsy mechanism in exact pointing of events, as a lot of supervised modifications have to be used. Let me discuss why this occurs.

![Diagram of overlapping clusters](image)

**Figure 5.26** Overlapping Clusters with high memberships.
The preceding Figure 5.26 shows when a node is inserted by an expert in an area, and we do create 2 new nodes in a particular area. The fuzzy membership of those nodes is 1. Due to which the fuzzy memberships in the region around the 2 nodes starts being dominated by the nodes. The error fraction also does not help in this case, so for such supervised node/cluster insertion, it’s important to disregard the error fraction. It might be needed to decrease the distance between the newly inserted nodes in order to decrease the misclassification rate.

So in such supervised cases, the error fraction has to be disregarded and/or the distance between the newly inserted node/cluster has to be decreased. Heuristics can be built to find the appropriate node distance but with a higher execution time of the algorithm because of requiring more testing.

In the above case, though ATFC is able to detect many new events; many events that are classified as an event by the system are not really events per se the expert. Thus for this dataset, ATFC gives a mixed amount of results. There is always a lot of leeway that is imparted by ATFC in this case but at the cost of increased misclassification. The advantages and disadvantages of ATFC for such a dataset can be summed as below:

1. Topology position can be exploited for such Event based mechanisms.
2. The Node width (Radius) has to be modified intelligently such that misclassifications do not increase with cluster creation. A per-se cluster width (radius and error) for the event cluster has to be done rather than a globally defined value by the unsupervised TFC.
3. Again it beats AHFC (which is based on FCM) by a faster execution time.
4. Supervised ATFC can be adapted with least modifications for different online clustering scenarios.

From my personal experiments, 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.

Personally, from my experimentations, I would recommend the algorithm (ATFC, TFC) in its original state for different factors as follows:

**Table 5.10** Choice of ATFC in Different Circumstances.

<table>
<thead>
<tr>
<th>Factors</th>
<th>Choice</th>
</tr>
</thead>
<tbody>
<tr>
<td>1    Execution time</td>
<td>Yes, very good</td>
</tr>
<tr>
<td>2    Dataset Size-Large</td>
<td>Yes, very good</td>
</tr>
<tr>
<td>(Num. of Clusters&lt;&lt;Size of dataset)</td>
<td></td>
</tr>
<tr>
<td>3    Dataset Size-Small</td>
<td>Yes, but better algorithms are available</td>
</tr>
<tr>
<td>(Num. of Clusters = Size of dataset)</td>
<td></td>
</tr>
<tr>
<td>4    Topology Shape</td>
<td>Yes, very good</td>
</tr>
<tr>
<td>5    Adaptiveness of Correcting Misclassification</td>
<td>Yes, but modification/ tweaking node radius might be required</td>
</tr>
<tr>
<td>6    Accuracy of Classification</td>
<td>Yes, but for medium to high accuracy. It’s not always the best performing.</td>
</tr>
<tr>
<td>7    Adaptiveness of Learning new data</td>
<td>Yes, very good</td>
</tr>
</tbody>
</table>
Thus, ATFC (and TFC) is very good in different circumstances as results in this chapter shows, and depicted in Table 5.10. This table can be used as an aid when the choice of ATFC to a particular dataset is to be debated.
CHAPTER 6
CONCLUSIONS AND FURTHER STUDY

6.1 Conclusions

As seen by the performance evaluations and accuracy of the algorithm, it's easy to conclude that TFC does give very good results for Online Clustering.

Its advantages can be cited to:

1. TFC has a linearized time for calculating the clusters in an unsupervised manner.
2. TFC has the advantages of being able to learn different cluster shapes through the use of Topology.
3. Through ATFC, the algorithm becomes adaptive to data by allowing expert's feedback on the type of the data encountered.

Due to the linearized time for calculation a very large sample size can be used to calculate the Level-1 node centers. 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 and linearized time factor makes it a potential algorithm for the domain of online clustering. The only iterative process in TFC is the Reference and Fuzzy Value finding Algorithm (Algorithm 4.1) which is called whenever a new data is sampled. This algorithm is very fast due to the use of older Fuzzy and Reference values to find the newer values. Normal experimentations
also show that the maximum amount of iterations for convergence is normally very nominal.

The primary aim when I started designing this algorithm was to include topology to aid construction of different shapes and creation of the nodes. This process is linearized which gives a better estimation of execution time than an iterative algorithm like C-Means.

TFC and ATFC are good in circumstances where the dataset is very large in size and not priori known. It can be also used with dataset of different shapes and due to the adaptiveness, can be used to incorporate expert’s opinions. Its accuracy would be more or less as good if not better, compared to FCM or any other C-Means algorithm, but with better results on Shaped clusters. The complexity involved in programming and understanding TFC (and ATFC) might be an inhibiting factor, as it involves a lot of different methods and tweaks for getting good results. The parameters that need to be set for TFC can be too much difficult to set in some cases. In such cases expert comments through ATFC can be helpful. The reader can also refer Figure 5.10, to come to a conclusion whether or not ATFC is proper for his/her application scenario.

6.2 Future Work

The goal of TFC was primarily finding out unknown patterns. Its real power is its linearized time to find those clusters and its ability to detect the number of clusters at runtime rather than a user chosen parameter. Its weak point is also its unsupervised pattern learning as a lot of parameter has to be tested to get the best cluster classification. Also its inherent unsupervised learning could be enhanced by getting feedback from the user at
runtime as so to better the classification done by the nodes. Though through the variant of
the algorithm, ATFC, supervised learning can be incorporated into the model, but it
leaves many factors to be desired. These factors to better this algorithm could be the
following things.

- Better way to choose the parameters of Lambda and Amax.
- Better use of the Area of influence.

The reason I mention the Area of Influence is because the Area of influence can
be a reason for misclassification. Consider the case when I got a high rate of
misclassification due to my use of hyperspheres. I do present 2 methods to find
hyperellipses that can be used to lower this misclassification rate but with a higher
computational time.

6.2.1 Hyperellipses

Hyperellipses can be used to generate regions instead of hyperspheres. So what would be
the advantages of using hyperellipses instead of hyperspheres and is there some logic in
using hyperellipses?

It’s perfectly valid to use hyperellipses as data is oriented not as a hypersphere but
data arranged like hyperellipses. So how can one generate such hyperellipses? There are
2 possible answers to this question.

Using Error as Hyperellipse Axis ratios:

The answer lies in orientation of these hyperellipses. These hyperellipses can be
axially oriented and trying to use the Error in each direction as the ratio of the Ellipse
axis in each Axis.
Right now the algorithm incorporating error into distance; this error is non-directional. Usage of a directional error should be used to calculate error in each direction. Using the ratio of the error in each direction can be equated to the radius we calculate through the Neighbor positions of a node.

Mathematically,

If error in each direction is represented as an Array ‘ErrDirec’, Then first obtain the ratios of error in each direction by equating to the maximum value in ‘ErrDirec’.

$$\text{Normalized } \text{ErrDirec} = \frac{\text{ErrDirec}}{\max(\text{ErrDirec})}$$

Now using this Normalized ErrDirec Ratios, it’s easy to plug in to the hyperellipse formula.

$$\frac{(x - N_x)^2}{\text{ErrDirec}^2_x} + \frac{(x - N_y)^2}{\text{ErrDirec}^2_y} + \frac{(x - N_z)^2}{\text{ErrDirec}^2_z} + \ldots + \frac{(x - N_k)^2}{\text{ErrDirec}^2_k} = r^2 \quad (6.1)$$

Where $N_x = \text{Node Position on ‘x’ axis}$, $N_y = \text{Node Position on ‘y’ axis}$ and so on.

$\text{ErrDirec}_x = \text{Normalized Error in ‘x’ axis}$ and so on.

$r^2 = \text{Square of the Average Radius}$.

Equation 6.1 can be used for validity of where a Test data point lies.

What’s the problem with this approach?

The directional error use though more intuitive is not really the right ellipse to create. The reason is the best ellipse is not axially oriented, it’s more or less oriented on the axis of maximal error or variance. So how do we turn the hyperellipse so that the best case scenario occurs?
There are 2 ideas that can be used for finding such variance oriented hyperellipses.

1. Using a PCA [20] based method to find the axes on to which the variability is maximum, and use them as the axes of the hyperellipses.

As shown in Figure 6.1, the hyperellipse is oriented in the direction of the maximal variation on basis of PCA. All the hyperellipse are oriented in the same way on the same axes. This would give a globally correct picture, but does not ensure the orientation when it comes to localized variation. I will now propose a per node hyperellipse construction with per node orientation but at a higher computational cost.

**Figure 6.1** PCA Projected Axis – Global.
2. The second idea is to use PCA but on a per-node basis rather than a global basis.

**Figure 6.2** PCA Projected Axis-Per Node.

As shown in figure 6.2, the nodes are orientated as per the local variance axes found through PCA.

How to how to find such axes?

For each node there is a set of neighbors. So considering each such neighbor’s positions as vectors, PCA can be applied to a set of such vectors and the variance axes can be found.

### 6.2.2 Better Supervised Learning and Choice of Parameters

As seen TFC has a lot of adaptiveness present, but it’s more or less application oriented when it comes to supervised insertion of new nodes/clusters. A better supervised learning would be to decrease the distance between the nodes, in order that minimum misclassification occurs.
Such supervised learning should be concerned with finding out the number of test data points affected by increasing or decreasing the width of the cluster from by supervision. Those data points should be again tagged by the expert to tell the ATFC algorithm if the classification done is correct or not. Such increase/decrease in the distance between the nodes should be done till the misclassification decreases to the minimum.

ATFC inherits the many parameters setting from GNG algorithm. To a regular user, the choice of such parameters at run time is very difficult. My experiments with ATFC showed that in general the following method can be used to choose such parameters.

1. Choose $e_b, e_n, \alpha, \beta$ at some rates as discussed in Chapter 5. These parameters are generally the most easy to set for ATFC.

2. Choose an arbitrary value of $\lambda, a_{\text{max}}$ with either the values around 5th (or more) the size of the smallest cluster to find. These parameters the most difficult to set. The reader might have to do some experimentation before an a good set of parameters can be found.

3. Once $\lambda, a_{\text{max}}$ are set, the user can again tinker with $e_b, e_n, \alpha, \beta$ to get a better resolution.

4. Now the user can test the sample data to find errors. If the unclassified data is too high in numbers, the user needs to decrease the size of the error fraction.

The reader is invited to suggest better ways to improve choosing such parameters for ATFC, which would also help enhancing GNG and TFC in the process.
At this point I would like to emphasize that ATFC was designed to be a single-pass algorithm from the start, where the user has full abilities to judge the running time of the algorithm. Even in its single pass variation, ATFC has a high classification rate with added benefits of adaptivity on newer inputs, linearized execution time, detection of varied shaped clusters, and inclusion of supervised learning. A multi-pass variation which can increase the classification rate to the maximum is also possible, and the reader is encouraged to pursue that option.

Now at this thesis I would recommend the reader to try improving the adaptiveness of ATFC and classification abilities as future goals for the ATFC algorithm.
REFERENCES


