PARALLEL SELF-ORGANIZING, HIERARCHICAL NEURAL NETWORKS WITH FUZZY INPUT SIGNAL REPRESENTATION, COMPETITIVE LEARNING AND SAFE REJECTION SCHEMES

Seongwon Cho
Purdue University, School of Electrical Engineering

Okan K. Ersoy
Purdue University, School of Electrical Engineering

Follow this and additional works at: http://docs.lib.purdue.edu/ecetr
PARALLEL SELF-ORGANIZING, HIERARCHICAL NEURAL NETWORKS WITH FUZZY INPUT SIGNAL REPRESENTATION, COMPETITIVE LEARNING AND SAFE REJECTION SCHEMES

S. Cho
O. K. Ersoy

TR-EE 92-24
JUNE 1992

SCHOOL OF ELECTRICAL ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285
PARALLEL SELF-ORGANIZING, HIERARCHICAL NEURAL NETWORKS
WITH FUZZY INPUT SIGNAL REPRESENTATION,
COMPETITIVE LEARNING AND SAFE REJECTION SCHEMES

Seongwon Cho and Okan K. Ersoy

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907-1285
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF TABLES</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>x</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>xii</td>
</tr>
<tr>
<td>CHAPTER 1 - INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Statement of Research</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Thesis Organization</td>
<td>5</td>
</tr>
<tr>
<td>CHAPTER 2 - BACKGROUND</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Fuzzy Set Theory</td>
<td>6</td>
</tr>
<tr>
<td>2.2.1 Basic Concepts</td>
<td>7</td>
</tr>
<tr>
<td>2.2.2 Mathematical Operations on Fuzzy Sets</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Neural Networks</td>
<td>11</td>
</tr>
<tr>
<td>2.3.1 Basic Concepts</td>
<td>11</td>
</tr>
<tr>
<td>2.3.2 Multilayer Neural Networks</td>
<td>13</td>
</tr>
<tr>
<td>2.3.3 Delta Rule and Backpropagation</td>
<td>13</td>
</tr>
<tr>
<td>2.3.4 Parallel Self-Organizing Hierarchical Neural Network</td>
<td>18</td>
</tr>
<tr>
<td>2.3.5 Competitive Learning</td>
<td>20</td>
</tr>
<tr>
<td>CHAPTER 3 - COMPUTATION OF THE DEGREE OF MATCH IN FUZZY SYSTEMS</td>
<td>22</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>22</td>
</tr>
<tr>
<td>3.2 Computation of the Degree of Match</td>
<td>25</td>
</tr>
<tr>
<td>3.2.1 Imprecise A* and Imprecise A</td>
<td>26</td>
</tr>
<tr>
<td>3.2.2 Precise A* and Imprecise A</td>
<td>33</td>
</tr>
<tr>
<td>3.2.3 Precise A</td>
<td>40</td>
</tr>
<tr>
<td>3.2.4 The DM in [0 1] scale</td>
<td>40</td>
</tr>
<tr>
<td>3.3 Comparison to Other Computational Approaches</td>
<td>43</td>
</tr>
<tr>
<td>3.3.1 Possibility and Necessity</td>
<td>43</td>
</tr>
<tr>
<td>3.3.2 The Degree to which the Antecedent Includes an Assertion</td>
<td>46</td>
</tr>
<tr>
<td>3.4 Conjunction and Disjunction</td>
<td>47</td>
</tr>
<tr>
<td>3.5 Examples</td>
<td>48</td>
</tr>
<tr>
<td>3.6 Conclusions</td>
<td>61</td>
</tr>
</tbody>
</table>
4.1 Introduction ............................................. 63
4.2 Parallel, Self-Organizing, Hierarchical Neural Network with Competitive Learning and Safe Rejection Schemes ............................................. 65
   4.2.1 Competitive Learning and Rejection Schemes ................. 65
   4.2.2 Training ........................................... 67
   4.2.3 Two Safe Rejection Schemes ................................ 70
   4.2.4 Testing ........................................... 75
4.3 Theoretical Analysis ......................................... 78
   4.3.1 Theoretical Analysis of Safe Rejection Schemes ............. 78
   4.3.2 Theoretical Analysis of Competitive Learning with Multiple Learning Rates ............................................. 89
4.4 Experimental Results ........................................... 95
   4.4.1 Experiments with the Gaussian-Distributed Data Set .......... 95
   4.4.2 Experiments with the Flight Line C1 (FLC1) Data Set .......... 100
      4.4.2.1 Data Set .......................................... 100
      4.4.2.2 Simulation .......................................... 100
      4.4.2.3 Comparison to Other Neural Networks ..................... 105
   4.4.4 Comparison of Method I and Method II for Learning Reference Vectors ............................................. 110
      4.4.4.5 Comparison of a Single Scalar Learning Rate (SLR) and Multiple Learning Rates (MLR) ................................. 115
   4.4.3 Experiments with the Colorado Data Set ......................... 119
      4.4.3.1 Data Set .......................................... 119
      4.4.3.2 Simulation .......................................... 119
      4.4.3.3 Comparison to Other Neural Networks ..................... 120
4.5 Conclusions ................................................. 127

5.1 Introduction ............................................. 128
5.2 Automatic Derivation of the Membership Functions for Fuzzy Sets .... 129
   5.2.1 Basic Concepts ......................................... 129
   5.2.2 Algorithm ........................................... 130
   5.2.3 Example ........................................... 131
5.3 Fuzzy Input Signal Representation ................................ 131
   5.3.1 Fuzzy Input Signal Representation during the Training of the FCSNN ............................................. 135
      5.3.1.1 Modification of Fuzzy Sets .............................. 135
      5.3.1.2 Selection of Fuzzy Sets ................................ 138
      5.3.1.3 Conversion of the Training Vectors into the DM Vectors ........ 141
   5.3.2 Fuzzy Input Signal Representation during the Testing of the FCSNN ............................................. 143
5.4 Experimental Results ......................................................... 143
  5.4.1 Experiments with the Flight Line C1 (FLC1) Data Set .............. 144
    5.4.1.1 Data Set ...................................................................... 144
    5.4.1.2 Simulation ..................................................................... 144
    5.4.1.3 Comparison to Other Neural Networks.............................. 145
  5.4.2 Experiments with the Colorado Data Set ............................... 156
    5.4.2.1 Data Set ...................................................................... 156
    5.4.2.2 Simulation ..................................................................... 156
    5.4.2.3 Comparison to Other Neural Networks.............................. 157
  5.4.3 Experiments with the ATV Survey Data Set ............................. 162
    5.4.3.1 Data Set ...................................................................... 162
    5.4.3.2 Simulation ..................................................................... 162
    5.4.3.3 Comparison to Other Neural Networks.............................. 169
  5.5 Conclusions ........................................................................... 178

CHAPTER 6 - CONCLUSIONS ............................................................... 179
  6.1 Conclusions ................................................................. 179
  6.2 Possible Future Research ...................................................... 181

LIST OF REFERENCES ................................................................. 183

VITA ......................................................................................... 189
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Sample data base</td>
<td>51</td>
</tr>
<tr>
<td>4.1</td>
<td>Spectral bands of the FLC1 multispectral image data</td>
<td>102</td>
</tr>
<tr>
<td>4.3</td>
<td>The classification accuracy of the backpropagation network (4NN) as a</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>function of the number of iterations in the 8-class problem</td>
<td></td>
</tr>
<tr>
<td>4.4</td>
<td>The classification accuracy and the total number of SNN's of the PSHNN</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>with the delta rule as a function of the number of iterations in the 8-class</td>
<td></td>
</tr>
<tr>
<td></td>
<td>problem</td>
<td></td>
</tr>
<tr>
<td>4.5</td>
<td>The experimental results with the FLC1 multispectral image data</td>
<td>108</td>
</tr>
<tr>
<td>4.6</td>
<td>The classification accuracy and the total number of SNN's of the CSNN</td>
<td>112</td>
</tr>
<tr>
<td></td>
<td>when the reference vectors are computed for each class independently</td>
<td></td>
</tr>
<tr>
<td></td>
<td>versus all classes considered together</td>
<td></td>
</tr>
<tr>
<td>4.7</td>
<td>The number of rejected data of the Method I and Method II at each SNN</td>
<td>114</td>
</tr>
<tr>
<td></td>
<td>when the allowed number of iterations is 10</td>
<td></td>
</tr>
<tr>
<td>4.8</td>
<td>The classification performances of a single scalar learning rate and</td>
<td>118</td>
</tr>
<tr>
<td></td>
<td>multiple learning rates</td>
<td></td>
</tr>
<tr>
<td>4.9</td>
<td>10 dominant classes of the multispectral image data of Colorado area</td>
<td>122</td>
</tr>
<tr>
<td>4.10</td>
<td>The number of samples of each class of the multispectral image data of</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td>Colorado area</td>
<td></td>
</tr>
<tr>
<td>4.11</td>
<td>The classification accuracy of the backpropagation network (4NN) as a</td>
<td>124</td>
</tr>
<tr>
<td></td>
<td>function of the number of iterations in the 10-class problem</td>
<td></td>
</tr>
<tr>
<td>4.12</td>
<td>The classification accuracy and the total number of SNN's of the PSHNN</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td>with the delta rule as a function of the number of iterations in the 10-class</td>
<td></td>
</tr>
<tr>
<td></td>
<td>problem</td>
<td></td>
</tr>
<tr>
<td>4.13</td>
<td>The experimental results with the Colorado data set</td>
<td>126</td>
</tr>
<tr>
<td>5.1</td>
<td>The number of fuzzy sets derived for each attribute of the FLC1 data set</td>
<td>148</td>
</tr>
<tr>
<td>5.2</td>
<td>The classification accuracy and the total number of SNN's of the FCSNN</td>
<td>149</td>
</tr>
<tr>
<td></td>
<td>as a function of the number of iterations in the 4-class problem</td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>5.3</td>
<td>The classification accuracy and the total number of SNN's of the FCSNN as a function of the number of iterations in the 4-class problem when the vectors rejected in the last SNN are not classified</td>
<td>150</td>
</tr>
<tr>
<td>5.4</td>
<td>The classification accuracy of the backpropagation network (4NN) as a function of the number of iterations in the 4-class problem</td>
<td>151</td>
</tr>
<tr>
<td>5.5</td>
<td>The classification accuracy and the total number of SNN's of the PSHNN with the delta rule as a function of the number of iterations in the 4-class problem</td>
<td>152</td>
</tr>
<tr>
<td>5.6</td>
<td>The classification accuracy and the total number of SNN's of the CSNN as a function of the number of iterations in the 4-class problem</td>
<td>153</td>
</tr>
<tr>
<td>5.7</td>
<td>The classification accuracy and the total number of SNN's of the CSNN as a function of the number of iterations in the 4-class problem when the vectors rejected in the last SNN are not classified</td>
<td>154</td>
</tr>
<tr>
<td>5.8</td>
<td>The experimental results with the FLC1 multispectral image data (4-class)</td>
<td>155</td>
</tr>
<tr>
<td>5.9</td>
<td>The number of fuzzy sets derived for each attribute of the Colorado data set</td>
<td>158</td>
</tr>
<tr>
<td>5.10</td>
<td>The classification accuracy and the total number of SNN's of the FCSNN as a function of the number of iterations in the 10-class problem</td>
<td>159</td>
</tr>
<tr>
<td>5.11</td>
<td>The classification accuracy and the total number of SNN's of the FCSNN as a function of the number of iterations in the 10-class problem when the vectors rejected in the last SNN are not classified</td>
<td>160</td>
</tr>
<tr>
<td>5.12</td>
<td>The experimental results with the Colorado data set</td>
<td>161</td>
</tr>
<tr>
<td>5.13</td>
<td>6 dominant attributes of the ATV survey data set</td>
<td>164</td>
</tr>
<tr>
<td>5.14</td>
<td>6 dominant classes of the ATV survey data set</td>
<td>165</td>
</tr>
<tr>
<td>5.15</td>
<td>Survey sites of the ATV data set</td>
<td>166</td>
</tr>
<tr>
<td>5.16</td>
<td>The number of samples of each class of the ATV survey data set</td>
<td>167</td>
</tr>
<tr>
<td>5.17</td>
<td>The number of fuzzy sets derived for each attribute of the FLC1 data set</td>
<td>171</td>
</tr>
<tr>
<td>5.18</td>
<td>The classification accuracy and the total number of SNN's of the FCSNN as a function of the number of iterations with the ATV data set</td>
<td>172</td>
</tr>
<tr>
<td>5.19</td>
<td>The classification accuracy and the total number of SNN's of the FCSNN as a function of the number of iterations with the ATV data set when the vectors rejected in the last SNN are not classified</td>
<td>173</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>The graphical representation of %YOUNG</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Neural model with activation function</td>
<td>12</td>
</tr>
<tr>
<td>2.3</td>
<td>Single-layer neural network</td>
<td>14</td>
</tr>
<tr>
<td>2.4</td>
<td>Two-layer neural network</td>
<td>15</td>
</tr>
<tr>
<td>2.5</td>
<td>Block diagram of the PSHNN with 3 stage neural networks</td>
<td>19</td>
</tr>
<tr>
<td>3.1</td>
<td>Imprecise A* and imprecise A, DM (A*, A) = +1, 0 &lt; DM (A*, A) &lt; +1</td>
<td>29</td>
</tr>
<tr>
<td>3.2</td>
<td>Imprecise A* and imprecise A, DM (A*, A) = -1, -1 &lt; DM (A*, A) &lt; 0</td>
<td>31</td>
</tr>
<tr>
<td>3.3</td>
<td>Imprecise A* and imprecise A, 0 &lt; DM &lt; +1</td>
<td>32</td>
</tr>
<tr>
<td>3.4</td>
<td>Imprecise A* and imprecise A, DM = 0</td>
<td>34</td>
</tr>
<tr>
<td>3.5</td>
<td>Imprecise A* and imprecise A, -1 &lt; DM &lt; 0</td>
<td>35</td>
</tr>
<tr>
<td>3.6</td>
<td>Precise A* and imprecise A, DM = +1</td>
<td>37</td>
</tr>
<tr>
<td>3.7</td>
<td>Precise A* and imprecise A, DM = -1</td>
<td>38</td>
</tr>
<tr>
<td>3.8</td>
<td>Precise A* and imprecise A, 0 &lt; DM &lt; +1</td>
<td>39</td>
</tr>
<tr>
<td>3.9</td>
<td>Precise A* and imprecise A, DM = 0</td>
<td>41</td>
</tr>
<tr>
<td>3.10</td>
<td>Precise A* and imprecise A, -1 &lt; DM &lt; 0</td>
<td>42</td>
</tr>
<tr>
<td>3.11</td>
<td>Attribute values of Grass</td>
<td>54</td>
</tr>
<tr>
<td>3.12</td>
<td>Attribute values of Season</td>
<td>55</td>
</tr>
<tr>
<td>3.13</td>
<td>The hierarchical dependency diagram for the strategic corporate planning system</td>
<td>59</td>
</tr>
<tr>
<td>3.14</td>
<td>The non-hierarchical dependency diagram for the strategic corporate planning system</td>
<td>60</td>
</tr>
<tr>
<td>4.1</td>
<td>RADP and RADN for 2-D data</td>
<td>73</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>RAD for 2-D data</td>
<td>74</td>
</tr>
<tr>
<td>4.3</td>
<td>The distribution of test vectors</td>
<td>77</td>
</tr>
<tr>
<td>4.4</td>
<td>The decision surface of competitive learning for 2-D data</td>
<td>80</td>
</tr>
<tr>
<td>4.5</td>
<td>The relationship between $C(X, T)$ and $C(Y, T)$ when $X$ and $Y$ are distinct</td>
<td>84</td>
</tr>
<tr>
<td>4.6</td>
<td>The relationship between $C(X, T)$ and $C(Y, T)$ when $Y$ is a subset</td>
<td>85</td>
</tr>
<tr>
<td>4.7</td>
<td>rejection scheme of $X$</td>
<td>90</td>
</tr>
<tr>
<td>4.8</td>
<td>A rejection scheme $X$ is used before another rejection scheme $Y$</td>
<td>91</td>
</tr>
<tr>
<td>4.9</td>
<td>A rejection scheme $X$ is used after another rejection scheme $Y$</td>
<td>92</td>
</tr>
<tr>
<td>4.10</td>
<td>Parallel use of two rejection schemes $X$ and $Y$</td>
<td>96</td>
</tr>
<tr>
<td>4.11</td>
<td>The distribution of a Gaussian-distributed data set with the same number of samples per class</td>
<td>98</td>
</tr>
<tr>
<td>4.12</td>
<td>Training time versus energy with the Gaussian-distributed data set</td>
<td>99</td>
</tr>
<tr>
<td>4.13</td>
<td>The classification accuracy of the two methods with the Gaussian-distributed data set as a function of the number of iterations</td>
<td>101</td>
</tr>
<tr>
<td>4.14</td>
<td>The reference ground map of the FLC1 (when the spectral band is 0.80 - 1.00 mm)</td>
<td>113</td>
</tr>
<tr>
<td>4.15</td>
<td>The rejection boundaries constructed by Method I and Method II</td>
<td>117</td>
</tr>
<tr>
<td>5.1</td>
<td>Training time versus energy with two systems</td>
<td>132</td>
</tr>
<tr>
<td>5.2</td>
<td>The distribution of samples</td>
<td>133</td>
</tr>
<tr>
<td>5.3</td>
<td>The derived fuzzy sets of attribute $X1$</td>
<td>134</td>
</tr>
<tr>
<td>5.4</td>
<td>The derived fuzzy sets of attribute $X2$</td>
<td>136</td>
</tr>
<tr>
<td>5.5</td>
<td>The original fuzzy set $A$</td>
<td>139</td>
</tr>
<tr>
<td>5.6</td>
<td>Two fuzzy set $A1$ and $A2$ split from $A$</td>
<td>163</td>
</tr>
<tr>
<td>5.6</td>
<td>All-Terrain Vehicle (ATV)</td>
<td>163</td>
</tr>
</tbody>
</table>
ABSTRACT

Cho, Seongwon, Ph.D., Purdue University, August 1992. Parallel Self-Organizing, Hierarchical Neural Networks with Fuzzy Input Signal Representation, Competitive Learning and Safe Rejection Schemes. Major Professor: Okan K. Ersoy.

In this thesis we present parallel self-organizing, hierarchical neural networks with fuzzy input signal representation, competitive learning and safe rejection schemes (FCSNN). A computational scheme of the partial degree of match (DM) in fuzzy expert classification systems and a method for the automatic derivation of the membership functions for fuzzy sets are proposed. The derived fuzzy sets and the computational scheme of the degree of match are used together for the fuzzy representation of input information in neural networks, not only for improving the classification accuracy but also for being able to classify objects whose attribute values do not have clear boundaries. The original input is converted into multidimensional values using the fuzzy input signal representation scheme.

A new learning algorithm with competitive learning and multiple safe rejection schemes is proposed and used as the learning algorithm of the parallel self-organizing, hierarchical neural network (PSHNN) to get around disadvantages of both supervised learning algorithms and competitive learning algorithms. After reference vectors are computed using competitive learning in a stage neural network, the safe rejection schemes are constructed. The basic idea of safe rejection schemes is to reject the vectors so that there are no misclassified training vectors. Two different kinds of safe rejection schemes, RADPN and RAD, are developed and used together. The next stage neural network is trained with nonlinearly transformed values of only those training vectors that were rejected in the previous stage neural network.
Experimental results comparing the performance of the proposed learning algorithm with those of the backpropagation network and the PSHNN with the delta rule learning algorithm are discussed. The proposed learning algorithm produced higher classification accuracy and much faster learning. Learning of reference vectors were done by two methods and their classification accuracies were compared. When the reference vectors are computed separately for each class (Method II), higher classification accuracy is obtained as compared to the method in which reference vectors are computed together for all the classes (Method I). This conclusion has to do with rejection of hard vectors, and is the opposite of what is normally expected. Method II has the advantages of parallelism by which the reference vectors for all the classes can be computed simultaneously.

Experiments with the fuzzy input representation in comparison to the input representation of the original decimal values indicated the superiority of representation of the transformed inputs using the computational scheme of the partial degree of match.
CHAPTER 1
INTRODUCTION

1.1 Statement of Research

There has been much recent interest in the application of neural networks and artificial intelligence (AI) approaches to classification system areas such as speech/image recognition, remote sensing, medical diagnosis, decision making, and so on. In each area an input (attribute value) is received and matched (classified) to a corresponding output (class). Neural networks consist of a set of interconnected nodes. The output of each neural node is determined by the following equation:

$$\text{OUT} = F(\sum w_i x_i + \theta)$$  (1.1)

where $x_i$ is the input from the $i$-th input node and $w_i$ is a weight factor associated with the link between the $i$-th input node and an output node. $F$ is a nonlinear transfer function and $\theta$ is a threshold value.

Neural networks perform two processes to solve classification problems: learning (training) and recall (testing). Learning of a neural network is the procedure for determining the weight factors which will produce correct output responses for a given set of training input patterns, and can be implemented without explicitly programming classification rules. Recall with the neural network involves classifying new input patterns on the basis of the trained network.

Expert classification systems also solve problems by two steps: knowledge acquisition and reasoning. In knowledge acquisition step, expert knowledge is explicitly
expressed in the structured form of classification rules to construct a knowledge base. Given assertions provided by a user, an expert classification system reasons a conclusion (classification) based on the knowledge base in the reasoning step. However, knowledge acquisition is a major bottleneck in developing expert classification systems. Thus a recent focus of the artificial intelligence (AI) community has been on the development of efficient knowledge acquisition methodologies [Der85] [Rit86] [Kos91].

There are similarities and differences between neural networks and expert classification systems. Unlike statistical systems, neural networks and expert classification systems classify an object without a mathematical model of how outputs depend on inputs. In other words, they are model-free systems. Network learning can be compared to the knowledge acquisition step of expert classification systems. Learning and recall procedures of a neural network correspond to design of knowledge base and reasoning procedure of an expert classification system, respectively. While a neural network is trained directly by the training set consisting of finite input-output pairs, the knowledge base of an expert classification system is constructed indirectly through experts. Experts examine a set of input-output cases (training samples) and detect regularities in the set. Regularities extracted from the data become expert knowledge, usually called experience. Expert knowledge is used to construct the knowledge base of an expert classification system afterwards. However, often expert knowledge is implicit and it is known that a domain in which expert knowledge is implicit can hardly render a clear and complete specification of classification rules. In other words, it may not be possible or practical to identify implicit expert knowledge and represent it by machine executable rules. Thus, the knowledge acquisition step of an expert classification system may result in the loss of critical information in casting implicit expert knowledge into explicit rules.

In summary, neural networks can generalize their knowledge and respond to similar data they have not seen like an expert classification system. They are especially
The backpropagation technique is one of the most widely used learning algorithms and is used with feed-forward multilayered neural network architecture. However, the backpropagation model is an extremely slow learning method and performs biologically implausible error back propagation. Recently, the parallel, self-organizing, hierarchical neural network (PSHNN) was proposed as an alternative to the conventional multilayered neural network architecture for the purposes of decreasing system complexity, increasing classification accuracy, avoiding local minima, reducing learning and recall rates, and achieving true parallelism [ErHo90][Hon90]. However, the PSHNN still needs long learning times when iteration methods such as the delta rule and the backpropagation algorithm are used for determining the weight factors which produce the minimal error between the desired outputs and the actual outputs.

In addition, the backpropagation network and the PSHNN have previously treated only numerical data which is stated precisely whereas fuzzy expert classification systems, expert classification systems with fuzzy sets, can handle imprecise information in linguistic form as well as precise data. The data in a classification system can be often given imprecisely without their clear boundaries, particularly when the data is given by a human, not directly from a measurement machine. The cognitive activity of the brain, unlike the computational function of the binary computer, is based upon the relative grades of information acquired by our natural sensors [Gup88]. While driving on an icy road, for example, we perceive the driving environment in a relatively graded sense and act
accordingly. Information may appear in a precise numerical form (temperature of the body is 38.4°C). However, during the process of cognition, we perceive this temperature as near normal, in the form of an imprecise value. Thus, the data given to an expert classification system by a human is stated imprecisely. An example of imprecise data which is not given by a human but by a machine is the gray-level images given by a robot vision system when the objects are lying in poor, uneven or varying lighting conditions, or shrouded by smoke, chemical fumes or steam that often exist on a factory floor [GKN88].

In this thesis, we present parallel self-organizing, hierarchical neural networks with fuzzy input signal representation, competitive learning and safe rejection schemes (FCSNN) in order to keep the advantages of both neural networks and fuzzy expert classification systems as well as to get around their disadvantages. A computational scheme of the partial degree of match (DM) and a method for the automatic derivation of the membership functions for fuzzy sets are proposed. The derived fuzzy sets and the computational scheme of the degree of match are used together for the fuzzy input representation of neural networks, not only for improving the classification accuracy but also for being able to classify objects whose attribute values do not have clear boundaries. The fuzzy input representation scheme allows imprecise input and precise numerical input to be transformed into multidimensional numerical values.

A new learning algorithm with competitive learning and multiple safe rejection schemes is proposed and used as the learning algorithm of the parallel self-organizing, hierarchical neural network. After reference vectors are computed using competitive learning, the safe rejection schemes are constructed. The basic idea of safe rejection schemes is to reject the vectors in such a way that there are no misclassified training vectors. The next stage neural network is trained with non-linearly transformed values of only those training vectors that were rejected in the previous stage neural network. The
proposed algorithm produces higher classification accuracy and more speed than the backpropagation network and the PSHNN with the delta rule.

1.2 Thesis Organization

This thesis is organized into 6 chapters. Chapter 2 provides a brief description of fuzzy set theory and neural networks. Chapter 3 presents the computational scheme of the degree of match between an assertion and a classification rule in fuzzy expert classification systems. In Chapter 4, we propose the new learning algorithm and two safe rejection schemes, RADPN and RAD. The proposed learning method implemented on the PSHNN is compared with the backpropagation network and the PSHNN trained by the delta rule in terms of classification accuracy. In Chapter 5, the fuzzy input signal representation scheme is proposed as the preprocessing module of the parallel self-organizing, hierarchical neural networks with competitive learning and safe rejection schemes discussed in Chapter 4. The membership functions for fuzzy sets are derived based on the distribution of the training samples. The original input is transformed into multidimensional values using the derived fuzzy sets and the computational scheme of the degree of match presented in Chapter 3. The classification accuracies of the fuzzy input signal representation are compared with those of the original decimal representation. Chapter 6 covers conclusions and possible future research.
CHAPTER 2
BACKGROUND

2.1 Introduction

Well-known techniques used in AI systems which address imprecision and/or uncertainty include Bayesian probability theory [Dud76] [Dud81] [Pea86], MYCIN's model [Sho76], Dempster-Shafer theory [Sha76], and fuzzy set theory [AKS85] [ApRu85] [BaGu80] [FSTZ75] [HaKa86] [Jon85] [Kau75] [KMW87] [LeF74] [MaPr86] [MFT79] [Neg85] [TSPS85] [Zad75] [Zad79] [Zad83] [Zim87]. All methods except the last focus on precise but uncertain propositions. Fuzzy set theory can provide the basis for representing and capturing imprecise information. In this chapter, we first briefly describe the essentials of fuzzy set theory.

The second part of this chapter describes neural networks. Like expert classification systems, neural networks are also model-free estimators. Unlike the knowledge base of expert classification systems, neural networks are not composed of a set of explicitly programmed IF-THEN rules but consist of a set of interconnected nodes and a set of weight factors associated with the links between input nodes and output nodes. As a result, neural networks cannot directly encode structured knowledge. Instead we must present the system with a sufficiently large set of numerical input-output pairs. In unsupervised learning, the network also generates the appropriate classes.

2.2 Fuzzy Set Theory

In this section, we describe the basic concepts of fuzzy set theory and mathematical operations on fuzzy sets.
2.2.1 Basic Concepts

Fuzzy set theory generalizes the ordinary mathematical concept of a set by distinguishing a fuzzy set as "a class of objects with a continuum of grades of membership". The progression from membership to nonmembership of a particular set is commonly gradual rather than abrupt. A fuzzy set \( \%F \) is characterized by a membership function (possibility function) \( \mu_F \) mapping from \( X \) to the interval \([0, 1]\). The symbol \( U \) denotes a universe of discourse that contains all possible collection of subjects. \( X \) is a finite subset of \( U \) and is expressed as \( X = \{x_1, x_2, x_3, ..., x_n\} \).

For each element \( x_i \) in \( X \), \( \mu_F(x_i) \) is the degree to which \( x_i \) satisfies the properties of the fuzzy set \( \%F \) and is called a grade of membership (GOM) of \( x_i \) in \( \%F \). If \( x_i \) has 1 as its grade of membership, \( x_i \) fully satisfies the properties of the fuzzy set. When the properties of the fuzzy set are not at all satisfied, 0 is assigned. If \( \mu_F \) can assume only the extreme values of 0 and 1, the fuzzy set becomes equivalent to a non-fuzzy (ordinary) set. Mathematically, a fuzzy set \( \%F \) is a set of ordered pairs:

\[
\%F = \{ \mu_F(x_1)/x_1, \mu_F(x_2)/x_2, \mu_F(x_3)/x_3, ..., \mu_F(x_n)/x_n \} \\
= \{ \mu_F(x_i)/x_i \} \quad i=1,2,3, ..., n
\]  

(2.1)

Usually the \( \mu_F(x_i)/x_i \) whose \( \mu_F(x_i) \) is 0 is not expressed in the list. When \( X \) contains an infinite number of elements, we may represent \( \%F \) as

\[
\%F = \int_X \mu_F(X) / x
\]

(2.2)

A fuzzy variable is a variable whose value is a fuzzy set of a universe of discourse. A linguistic variable is a special kind of fuzzy variable whose values are words or sentences in a natural language or its subsets. The mathematical definitions of fuzzy set theory can be
linked to natural language by labelling the fuzzy sets with adjective descriptors such as "tall", "young", "heavy", "beautiful" etc. These adjective descriptors can also be modified with verbal hedges such as "very", "more or less", "somewhat", "almost" etc [Zad72]. If height is interpreted as a linguistic variable, then its term-set, that is, the set of linguistic values of a linguistic variable, might be

\[ T(\text{height}) = \{ \text{tall, short, very short, not tall, very short, more or less tall} \ldots \}. \]

Default effects of "very" and "more or less" are defined as [MFT79]

\[
\text{very } A = \int x \mu_A(x)^2 \, dx / x \quad (2.3)
\]

\[
\text{more or less } A = \int x \sqrt{\mu_A(x)} \, dx / x \quad (2.4)
\]

Example 2.1 Let \( X = \{ \text{Elizabeth, Mark, Michael} \} \). A fuzzy set \%KIND can be defined as

\[ \%KIND = \{ 0.7/\text{Elizabeth, 0.9/Mark, 0.2/Michael} \} \]

The above example defines a fuzzy linguistic term \%KIND which is a fuzzy set of universe of discourse describing people.

Example 2.2 Let \( X = \{ 10, 20, 25, 30, 40, 50 \} \). The universe of discourse is a segment of the integer number line representing ages of people. A fuzzy set \%YOUNG can be
defined as

\[ \%\text{YOUNG} = \{1/10, 1/20, 0.9/25, 0.8/30, 0.3/40, 0/50\} . \]

Figure 2.1 shows the graphical representation of \%\text{YOUNG}.

Example 2.3 A modified fuzzy set can be obtained using very
\[ A = \int \mu_A(x)^2 / x : \]

\[ \%\text{VERY YOUNG} = \{1/10, 1/20, 0.81/25, 0.64/30, 0.09/40, 0/50\} . \]

We can also define \%\text{VERY YOUNG} arbitrarily, not by derivation from \%\text{YOUNG} and \text{VERY}:

\[ \%\text{VERY YOUNG} = \{1/10, 0.9/20, 0.7/25, 0.4/30, 0/40, 0/50\} \]

2.2.2 Mathematical Operations on Fuzzy Sets

Let \%F, \%G be fuzzy sets in X. Common operations on fuzzy sets are as follows [Kau75].

Intersection: \%F \cap \%G = \int \min\{\mu_F(x), \mu_G(x)\} / x \hspace{1cm} (2.5)

Union: \%F \cup \%G = \int \max\{\mu_F(x), \mu_G(x)\} / x \hspace{1cm} (2.6)

Complement: 1 - \%F = \int \{1-\mu_F(x)\} / x \hspace{1cm} (2.7)

Algebraic product: \%F \cdot \%G = \int \mu_F(x)\mu_G(x) / x \hspace{1cm} (2.8)

Algebraic sum: \%F + \%G = \int \{\mu_F(x)+\mu_G(x)-\mu_F(x)\mu_G(x)\} / x \hspace{1cm} (2.9)
Figure 2.1 The graphical representation of %YOUNG.
\%F \supseteq \%G: \%G is contained in \%F \iff \int_x \mu_F(x) / x \supseteq \int_x \mu_G(x) / x \quad (2.10)

Fuzzy relation: A fuzzy relation \%R from X to Y is a fuzzy set in X × Y defined by \%R = \int_{x,y} \mu_R(x,y) / (x,y) \quad (2.11)

Composition: Let \%R and \%S be fuzzy relations in X × Y and Y × Z, respectively. The composition of \%R and \%S is a fuzzy relation in X × Z given by

\%R \circ \%S = \int_{x,z} \max_y \left[ \min \{ \mu_R(x,y), \mu_S(y,z) \} \right] / (x,z) \quad (2.12)

2.3 Neural Networks

In this section we describe the basic concepts of neural networks, neural network architectures (multilayer neural networks and PSHNN) and learning algorithms.

2.3.1 Basic Concepts

Neural networks are biologically inspired; that is, they are composed of elements that perform in a manner that is analogous to the most elementary functions of the biological neuron. A set of inputs labeled x_1, x_2, ..., x_n are applied to a neuron, each representing the output of another neuron. Each input is multiplied by a corresponding weight w_1, w_2, ..., w_n and all of the weighted inputs are then summed. A threshold value is added as an extra neuron to the network with a constant input of 1 and the weight is set equal to the amount of the threshold. The combined inputs determine, via a function \( F \), the activation value of the neuron. The output of a neuron is assumed to be the same as the activation value of the neuron. Figure 2.2 shows a model that implements this idea.

Although a single neuron can perform certain simple functions, the power of neural computation comes from connecting neurons into networks. The simplest network is a
\[ \text{OUT} = F(\sum \omega_i \chi_i + \theta) \]

Figure 2.2 Neural model with activation function.
group of neurons arranged in a layer (output layer) as shown on the right side of Figure 2.3. Note that the circular nodes on the left serve only to distribute the inputs. They perform no computation and constitute input layer. The literature is inconsistent in defining the number of layers in these networks. Some authors refer to the number of layers of neurons (including the nonsumming input layer), others to the layers of weights. The latter definition is more functionally descriptive, it is used throughout this proposal. By this definition, the network of Figure 2.3 is considered as a single-layer neural network.

2.3.2 Multilayer Neural Networks

Larger, more complex networks generally offer better computational capabilities. Multilayer neural networks have been proven to have capabilities beyond those of a single layer. Multilayer networks may be formed by simply cascading a group of single layers; the output of one layer provides the input to the subsequent layer. The layer(s) between the input layer and the output layer are called the hidden layer(s). Figure 2.4 shows two-layer neural network with one hidden layer.

2.3.3 Delta Rule and Backpropagation

The weight factors associated with the link between input nodes and output nodes are determined using a learning algorithm so that correct classifications are made for a given set of training samples.

The delta rule is one of the most important supervised learning algorithms used for single-layer neural networks with linear activation functions. It minimizes the squares of the differences between the actual and desired output values summed over the entire output units and all of the training samples. Each sample is represented as a pair of input/output vectors. For each sample the network first uses the input vector to produce its own actual output vector and then compares this with the desired output. If there is no difference, the
Figure 2.3 Single-layer neural network.
Figure 2.4 Two-layer neural network.
weights are not changed. Otherwise the weights are changed to reduce the difference as follows:

$$\begin{align*}
wij(t+1) &= wij(t) + \eta \left[ zd_k(j) - \left( \sum_{m=1}^{M} w_{mj}(t) x_{mk} + \theta_j \right) \right] x_{ik} \\
&= wij(t) + \eta \delta_{jk} x_{ik} 
\end{align*}$$

(2.13)

where

- \(wij(t+1)\) is the value of weight \(wij\) associated with the link between the \(i\)-th input unit and the \(j\)-th output unit after adjustment;
- \(wij(t)\) is the value of weight \(wij\) before adjustment;
- \(\eta\) is the learning rate coefficient;
- \(zd_k(j)\) is the desired output in the \(j\)-th output unit for sample \(k\);
- \(x_{ik}\) is the input in the \(i\)-th input unit for sample \(k\);
- \(M\) is the length of the input vector;
- \(\theta_j\) is the threshold value in the \(j\)-th output unit;
- \(\delta_{jk}\) represents the difference between the desired output and the actual output in the \(j\)-th output unit for sample \(k\);

The backpropagation algorithm, called the generalized delta rule, is a powerful method for training multilayer neural networks. It also minimizes the squares of the differences between the actual and desired output values but a nonlinear activation function in which the output of a unit is a nondecreasing and differentiable function of the net total output is used for each layer. Thus the actual output of the \(j\)-th node in each layer is

$$z_k(j) = F\{ \sum_{m=1}^{M} w_{mj}(t) \text{OUT}_{mk} + \theta_j \} = F\{\text{NET}_x(j)\}$$

(2.14)
where $F$ is a nonlinear function which is nondecreasing and differentiable; $\text{OUT}_{mk}$ is the output of the $m$-th neuron in the previous layer for the $k$-th sample; $M$ is the number of neurons in the previous layer.

The correction of the weight has the same form as the delta rule but $\delta_k$ is different for each layer. Each weight proceeding from the $i$-th neuron in the last hidden layer to the $j$-th neuron in the output layer is adjusted as follows:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_{jk} \text{OUT}_k$$

$$= w_{ij}(t) + \eta F'(\text{NET}_k(j)) \{ z_{dk}(j) - z_k(j) \} \text{OUT}_k$$  \hspace{1cm} (2.15)

where $\text{OUT}_k$ is the output value of the $i$-th neuron in the last hidden layer for the $k$-th sample; $F'$ represents the first derivative of $F$.

The value of $\delta_k$ needed for the $i$-th neuron in each hidden layer is obtained by the following equation:

$$\delta_k = F'(\text{NET}_k(i)) \left\{ \sum_{q=1}^{N} \delta_{qk} w_{iq} \right\}$$  \hspace{1cm} (2.16)

where $N$ is the number of neurons in the next layer.

For a single-layer neural network, only Equation (2.15) is used and it becomes

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_{jk} x_i$$

$$= w_{ij}(t) + \eta F'(\text{NET}_k(j)) \{ z_{dk}(j) - z_k(j) \} x_i$$  \hspace{1cm} (2.17)

For a single-layer neural network with linear activation function, $F' = 1$ and thus the backpropagation algorithm is same as the delta rule.
The delta rule and the backpropagation algorithm usually require many number of
iterations in order to determine the optimal weight factors which produce the minimal error
between the desired outputs and the actual outputs. Thus one of their serious drawbacks is
long learning times.

2.3.4 Parallel Self-Organizing Hierarchical Neural Network [Hon90] [ErHo90]

The PSHNN involves a number of stages, similar to a multilayer neural network.
Each stage can be a particular neural network, to be referred as the stage neural network
(SNN). Usually each SNN is a feed-forward single-layer neural network shown in Figure
2.3. Unlike a feed-forward multilayer network, each SNN is essentially independent of the
other SNN's in the sense that each SNN does not receive its input directly from the
previous SNN. At the output of each SNN, there is a rejection scheme which allows
acceptance or rejection of the input vectors.

Between two SNN's, there is a nonlinear transformation which transforms
nonlinearly certain input vectors rejected by the previous SNN. The transformed input
vectors are fed into the next SNN. Rejection schemes and nonlinear transformation are
probably the most original properties of the PSHNN, as distinct from other neural neural
networks. Figure 2.5 shows the block diagram of the PSHNN with 3 SNN's.

Examples of the nonlinear transform are real discrete Fourier transform (RDFT)
[Ers85], discrete symmetric cosine transform (DSCT) [Hu89], class-2 case-5 discrete
Fourier preprocessing transform (Class-2 Case-5 DFPT) [Hu89], shuffling and
complementing, Hadamard transform [ElRa81], and so on. The experiments performed in
comparison to multilayer neural networks with backpropagation training show the
superiority of the PSHNN with delta rule learning with a nonlinear activation function in
the sense of classification accuracy, training time [Hon90].
Figure 2.5  Block diagram of the PSHNN with 3 stage neural networks.
2.3.5 Competitive Learning

Competitive learning has been used for rapid training of neural networks since the algorithm is low computational complexity. There are several algorithms for competitive learning [AKC90] [Des88] [Koh88] [Koh89] [Koh90] [Kos91] [RuZi88]. One typical competitive learning algorithm can be described as follows:

\[
W_k(t+1) = W_k(t) + C(t) [X(t) - W_k(t)] \quad \text{if } k \text{ wins} \quad (2.18)
\]

\[
W_k(t+1) = W_k(t) \quad \text{if } k \text{ loses} \quad (2.19)
\]

where \(W_k(t+1)\) represents the value of the \(k\)-th reference vector after adjustment; \(W_k(t)\) is the value of the \(k\)-th reference vector before adjustment; \(X(t)\) is the training vector at time \(t\); \(C(t)\) is the learning rate coefficient. Usually slowly decreasing scalar time functions are used as the learning rates. At each instant of time, the winning reference vector, \(W_c(t)\), is the one which has the minimum Euclidean distance between the reference vector and \(X(t)\):

\[
\sqrt{\sum_i (x_i(t) - w_{ci}(t))^2} = \min_k \left\{ \sqrt{\sum_i (x_i(t) - w_{ki}(t))^2} \right\} \quad (2.20)
\]

where \(x_i(t)\) represents the \(i\)-th element of \(X(t)\); \(w_{ki}(t)\) is the \(i\)-th element of the \(k\)-th reference vector \(W_k(t)\); \(w_{ci}(t)\) is the \(i\)-th element of the winning reference vector \(W_c(t)\).

If \(X(t)\) and \(W_k(t)\) are normalized vectors, Equations (2.18) to (2.20) can be rewritten as follows:

\[
W_k(t+1) = \frac{W_k(t) + C(t) [X(t) - W_k(t)]}{\|W_k(t) + C(t) [X(t) - W_k(t)]\|} \quad \text{if } k \text{ wins} \quad (2.21)
\]

\[
W_k(t+1) = W_k(t) \quad \text{if } k \text{ loses} \quad (2.22)
\]
\[ \sum_{i} w_{ci}(t)x_{i}(t) = \max \left\{ \sum_{i} w_{ki}(t)x_{i}(t) \right\} \] (2.23)

where \( \| \cdot \| \) represents the Euclidean norm.

Kohonen developed several supervised competitive learning algorithms called the learning vector quantization (LVQ) [AKC90] [Koh88] [Koh89] [Koh90] [Kos91]. The LVQ gives higher classification accuracy than the unsupervised algorithm characterized by Equations (2.18) to (2.20) when it is used for pattern recognition. The first version, LVQ1, is as follows:

**STEP 1.** For a training vector \( X(t) \), the winning reference vector, \( W_{c}(t) \), is determined by Equation (2.20).

**STEP 2.** Update the reference vectors.

1) If \( W_{c}(t) \) has the same class as \( X(t) \),

\[ W_{c}(t+1) = W_{c}(t) + C(t) [X(t) - W_{c}(t)] \] (2.24)

2) If the class of \( W_{c}(t) \) is different from the class of \( X(t) \),

\[ W_{c}(t+1) = W_{c}(t) - C(t) [X(t) - W_{c}(t)] \] (2.25)

3) Otherwise,

\[ W_{k}(t+1) = W_{k}(t) \quad \text{for } k \neq c \] (2.26)
CHAPTER 3
COMPUTATION OF THE DEGREE OF MATCH IN FUZZY SYSTEMS

3.1 Introduction

The problem of object classification based on inspection of its attributes can be approached from an artificial intelligence (AI) perspective [Cox88], and classification systems using AI techniques are called expert classification systems.

Expert classification problems are characterized by a limited number of observations (attributes), a preenumerated set of possible solutions (classes), and heuristics or rules indicating which observations point to which solutions. Objects can be animated or lifeless concreta like persons, animals, industrial plants, buildings and machine devices, or abstracta like philosophies, opinions, methods, states and diagnoses. Attributes are direct or indirect measurements of quality and quantity or derived values. Many real-world problems including medical diagnosis, decision making, pattern recognition and fault-detection in technical devices and processes can be successfully described as classification problems.

Suppose we have a set of production rules for classifying objects based on the values of each of n descriptive attributes. Such classification rules map sets of attribute values into corresponding assigned classes. A set of classification rules can be expressed in the following form:

if $X_1$ is $A_{11}$ and $X_2$ is $A_{12}$ and ... and $X_n$ is $A_{1n}$ then Class is $C_1$
if $X_1$ is $A_{21}$ and $X_2$ is $A_{22}$ and ... and $X_n$ is $A_{2n}$ then Class is $C_2$
... 
... 
if $X_1$ is $A_{m1}$ and $X_2$ is $A_{m2}$ and ... and $X_n$ is $A_{mn}$ then Class is $C_m$
Each rule has \( n \) antecedent propositions (attributes) at most. Each of labels \( C_1, C_2, \ldots, C_m \) is the name of a class. \( X_1, X_2, \ldots, X_n \) are attribute names and \( A_{11}, A_{12}, \ldots, A_{1n}, A_{21}, A_{22}, \ldots, A_{2n}, \ldots, A_{m1}, A_{m2}, \ldots, A_{mn} \) are attribute values. These attribute values can be often stated precisely, but sometimes it is hard for experts to define their clear boundary. In other words, the antecedent propositions of classification rules may be precise or imprecise.

In order to perform classification of an object, an expert classification system needs assertions related to the antecedent part of classification rules. Assertions concerning an object are given by a user of the expert classification system. Some users can state it precisely and some users can not. The consequent parts (Class is \( C_1 \), Class is \( C_2 \), ..., Class is \( C_m \)) of all classification rules can be considered as precise propositions.

Imprecise information is represented well by fuzzy sets. Expert classification systems which incorporate fuzzy techniques into their knowledge representation schemes and/or reasoning processes are called fuzzy expert classification systems. The interpreter of a fuzzy system examines the working memory to determine which rules to match and their match strengths range along a continuum from "not at all" to "completely", rather than in the all-or-nothing fashion characteristics of conventional (non-fuzzy) systems. In other words, partial matching is allowed in fuzzy systems, and thus the interpreter of a fuzzy system estimates to what extent the antecedent part of a rule is satisfied by a working memory element. On the other hand, the interpreter of a conventional expert classification system examines merely whether the antecedent part is satisfied or not.

Some computational schemes have been proposed to estimate the degree of partial match between an imprecise assertion and the imprecise antecedent part of a rule [CFP82] [LeLa88] [OFY84]. Cayrol [CFP82] adopts two measures, possibility and necessity, for representing the partial match between an antecedent of a rule and an assertion. Leung [LeLa88] combines possibility and necessity into a single measure compatible with the
certainty factors used in MYCIN [Sho75]. Ogawa [OFY84] evaluates partial matching with a measure of the degree to which the antecedent of a rule includes an assertion which can also be combined with certainty factors.

The feasibility of combining these two latter approaches with certainty factors makes them advantageous when the information is both imprecise and uncertain. However, both methods have disadvantages when precise assertions are matched against imprecise propositions. In particular, the same computational scheme used to match imprecise assertions should also work for matches of precise assertions against imprecision propositions. Furthermore, the computed value should be equal to the grade of membership of the precise assertion in the fuzzy set representing the imprecise antecedent proposition. This follows because a precise assertion can be regarded as a completely contracted imprecise assertion. Neither Leung's or Ogawa's methods satisfy the above requirement.

In many expert classification systems, precise and imprecise information may be mixed. For example, the antecedent part of a rule is an imprecise proposition and a given assertion is a precise proposition. Moreover, the compound antecedent part of a rule can have precise and imprecise propositions together. There are four possible combinations of the antecedent of a rule and an assertion: precise antecedent and precise assertion, precise antecedent and imprecise assertion, imprecise antecedent and precise assertion, imprecise antecedent and imprecise assertion.

In this chapter, we discuss a computational scheme of the degree of match between the antecedent part and an assertion, which can be used consistently for all possible combinations of precise and imprecise propositions and satisfy the above requirement. The proposed computational scheme can be used successfully for estimating the certainty value of classification in fuzzy expert classification systems. The computed degree of match using our computational scheme can also be combined with a certainty factor.
This chapter consists of six sections. In Section 3.2, a computational scheme of the degree of match between the antecedent part of a rule and the assertions is proposed. In Section 3.3, we compare our scheme with the other methods. Section 3.4 deals with conjunctive and disjunctive antecedent parts. In Section 3.5, some examples are introduced to show the usefulness of our proposed computational scheme of the degree of match. Finally, conclusions are given in Section 3.6.

3.2 Computation of the Degree of Match [CEL92a]

In this section, a computational scheme of the degree of match (DM) between the antecedent part (A) of a classification rule and an assertion (A*) are proposed when imprecise and precise propositions are mixed freely in a fuzzy expert classification system. The following will briefly describes how the DM is computed.

In order to express the degree of match (DM), a value between -1 and +1 is assigned, where:

+1 means the antecedent part (A) is satisfied completely by an assertion (A*),
i.e., A = A* or A* is contained in A completely.

-1 means the antecedent part is not satisfied at all by A*, i.e., the negation of the antecedent part (1-A) is satisfied completely by A*.

0 represents the neutrality, i.e., A and 1-A are satisfied to the same degree by A*.

The DM will be expressed in terms of two new parameters CERT+(A*, A) and CERT-(A*, A) defined as follows:

\[
\text{CERT}+(A^*, A) = \max \{\min(\mu_d, \mu_n)\} \quad (3.1)
\]
CERT: \( (A^*, A) = \max \{ \min(1-\mu_d, \mu_n) \} \) \hspace{1cm} (3.2)

where \( \mu_d \) and \( \mu_n \) are the denominator and the numerator of the TR\((A^*, A)\), respectively and

\[
\text{TR}(A^*, A) = \text{Truth}(X \text{ is } A \mid X \text{ is } A^*) = \int \frac{\mu_{A^*}(x)}{\mu_A(x)}
\] \hspace{1cm} (3.3)

For any \( i \in [0,1] \) such that \( \mu_A(x) = i \) and \( x \in X \), \( \mu_{\text{TR}}(i) = \max \mu_{A^*}(x) \).

\( \text{CERT}+(A^*, A) \) measures to what extent \( A \) and \( A^* \) overlap, and hence it contributes to the positivity of the DM. \( \text{CERT}-(A^*, A) \) measures to what extent \( 1-A \) and \( A^* \) overlap and contributes to negativity of the DM. \( \text{CERT}+(A^*, A) \) and \( \text{CERT}-(A^*, A) \) can also be computed directly from \( A \) and \( A^* \):

\[
\text{CERT}+(A^*, A) = \max \{ \min(\mu_A, \mu_{A^*}) \}
\]

\[
\text{CERT}-(A^*, A) = \max \{ \min(1-\mu_A, \mu_{A^*}) \}
\] \hspace{1cm} (3.4) (3.5)

Clearly, \( 0 \leq \text{CERT}+(A^*, A) \leq 1 \) \( \leq \text{CERT}-(A^*, A) \leq 1 \) and \( \text{CERT}+(A^*, A) + \text{CERT}-(A^*, A) \geq 1 \). The following discussion will go into detail regarding how the DM can be calculated in terms of these parameters; first for the case where \( A^* \) and \( A \) are both imprecise and then for the case where \( A^* \) is precise and \( A \) is imprecise.

3.2.1 Imprecise \( A^* \) and Imprecise \( A \)

Figures 3.1 to 3.5 show graphical representations of imprecise \( A \), imprecise \( A^* \), \( 1-A \), their corresponding TR, \( \text{CERT}+(A^*, A) \) and \( \text{CERT}-(A^*, A) \) for varying degrees of overlap between \( A^* \) and \( A \). The original figures of \( A \) and \( A^* \) and \( 1-A \) are shown on the left side. The X-axis represents possible object values and the Y-axis represents the grades of membership of an object value in \( A, A^* \) and \( 1-A \). We see that \( \text{CERT}+(A^*, A) \) corresponds
to the maximum among the intersection points of $A^*$ and $A$. Similarly, $\text{CERT-}(A^*, A)$ is
the maximum among the points in which $A^*$ and $1-A$ intersect.

The right figures show the TR computed from $A$, $A^*$ and $1-A$ in the left figures. The $X$-axis and $Y$-axis of the right figures represent the grades of membership of $A$ and $A^*$, respectively. $\text{CERT+}(A^*, A)$ is the maximum among the intersection points of $\mu_n$ and
$\mu_d$, and $\text{CERT-}(A^*, A)$ is the maximal intersection point of $\mu_n$ and $1-\mu_d$. We see that
$\text{CERT+}(A^*, A)$ and $\text{CERT-}(A^*, A)$ can be computed from both the right and the left
figures, and the results are identical, leading to the same DM.

By separately analyzing the five different cases represented in Figures 3.1 to 3.5 it
becomes possible to infer the relationship between the DM and $\text{CERT+}(A^*, A)$ and $\text{CERT-}
(A^*, A)$. In the case of imprecise $A^*$ and imprecise $A$, we can conclude the following:

If $\text{CERT+}(A^*, A) \geq 0.5$ and $\text{CERT-}(A^*, A) \leq 0.5,$

\[
\text{DM}[1+1] = 2 \times \text{CERT+}(A^*, A) - 1
\]  

(3.6)

If $\text{CERT+}(A^*, A) \geq 0.5$ and $\text{CERT-}(A^*, A) \geq 0.5,$

\[
\text{DM}[1+1] = 2 \times \{\text{CERT+}(A^*, A) - \text{CERT-}(A^*, A)\}
\]  

(3.7)

If $\text{CERT+}(A^*, A) \leq 0.5$ and $\text{CERT-}(A^*, A) \geq 0.5,$

\[
\text{DM}[1+1] = -\{2 \times \text{CERT-}(A^*, A) - 1\}
\]  

(3.8)

where $\text{DM}[1+1]$ represents the degree of match in $[1+1]$ scale.

When $\text{CERT-}(A^*, A)$ is less than 0.5, only $\text{CERT+}(A^*, A)$ determines the DM.
Similarly, when $\text{CERT+}(A^*, A) \leq 0.5$, $\text{CERT-}(A^*, A)$ determines the DM. Otherwise, the
DM depends on the difference between $\text{CERT+}(A^*, A)$ and $\text{CERT-}(A^*, A)$. For a
boundary value $\text{CERT+}(A^*, A) = 0.5$ and $\text{CERT-}(A^*, A) = 0.5$, the DM can be computed
using any of three equations (3.6), (3.7) and (3.8) and the results are identical:
\[ \text{DM}[-1 +1] = 2 \times \text{CERT}^+(A^*, A) - 1 = 2 \times 0.5 - 1 = 0 \]
\[ \text{DM}[-1 +1] = 2 \times (\text{CERT}^+(A^*, A) - \text{CERT}^-(A^*, A)) = 2 \times (0.5 - 0.5) = 0 \]
\[ \text{DM}[-1 +1] = -(2 \times \text{CERT}^-(A^*, A) - 1) = -(2 \times 0.5 - 1) = 0 \]

Analogously, when \( \text{CERT}^+(A^*, A) = 1 \) and \( \text{CERT}^-(A^*, A) = 0.5 \), the DM is computed using either (3.6) or (3.7):

\[ \text{DM}[-1 +1] = 2 \times \text{CERT}^+(A^*, A) - 1 = 2 \times 1 - 1 = +1 \]
\[ \text{DM}[-1 +1] = 2 \times (\text{CERT}^+(A^*, A) - \text{CERT}^-(A^*, A)) = 2 \times (1 - 0.5) = +1 \]

When \( \text{CERT}^+(A^*, A) = 0.5 \) and \( \text{CERT}^-(A^*, A) = 1 \), the DM is computed using either Equation (3.7) or Equation (3.8):

\[ \text{DM}[-1 +1] = 2 \times (\text{CERT}^-(A^*, A) - \text{CERT}^-(A^*, A)) = 2 \times (0.5 - 1) = -1 \]
\[ \text{DM}[-1 +1] = -(2 \times \text{CERT}^-(A^*, A) - 1) = -(2 \times 1 - 1) = -1 \]

The basis for these equations relating the DM to \( \text{CERT}^+(A^*, A) \) and \( \text{CERT}^-(A^*, A) \) can be more easily understood by evaluating each of the cases in Figures 3.1 to 3.5. These five cases span the possible categories of overlap between \( A^* \) and \( A \) and are each reasonably accounted for by the above equations, as expanded upon below.

Case 1a. When \( A^* \) is contained in \( A \) completely (Figure 3.1) the degree of match is +1 by the above definition. For \( A^* \) and \( A \), \( \text{CERT}^-(A^*, A) \) is less than or equal to 0.5 and \( \text{CERT}^+(A^*, A) = 1 \). When \( A^* = A \), \( \text{CERT}^-(A^*, A) = 0.5 \) (point P) and \( \text{CERT}^+(A^*, A) = 1 \), and the DM should be equal to +1 clearly. As \( A^* \) contracts, \( \text{CERT}^-(A^*, A) \) point moves from P to Q and \( \text{CERT}^-(A^*, A) \) becomes less than 0.5. The DM in this situation is
Figure 3.1 Imprecise $A^*$ and imprecise $A$, $\text{DM}(A^*, A) = +1$, $0 < \text{DM}(A^{**}, A) < +1$. 
therefore correctly calculated using Equation (3.6) as follows:

\[ \text{DM}_{1-1+1} = 2 \times \text{CERT}^+(A^*, A) - 1 = 2 \times 1 - 1 = +1 \]

If A* moves to the right (A**) in Figure 3.1, CERT+(A*, A) decreases, CERT-(A*, A) increases, and thus the DM decreases. For A** and A, CERT+(A*, A) ≥ 0.5 and CERT-(A*, A) ≤ 0.5. Assume that CERT+(A*, A) = 0.9 and CERT-(A*, A) = 0.4 for A** and A. Then, the DM of A against A** is obtained by the following:

\[ \text{DM}_{1-1+1} = 2 \times \text{CERT}^+(A^*, A) - 1 = 2 \times 0.9 - 1 = +0.8 \]

Case 2a. When A* is completely contained in 1-A (Figure 3.2), the DM is -1 by the definition. If A* moves to the left (A*'), CERT-(A*, A) decreases, CERT+(A*, A) increases, and thus the DM increases. Examination of Figure 3.2 reveals that CERT+(A*, A) ≤ 0.5 and CERT-(A*, A) ≥ 0.5. The DM between A* and A is calculated using Equation (3.8) as follows:

\[ \text{CERT}^+(A^*, A) \leq 0.5 \text{ and } \text{CERT}-(A^*, A) = 1 \]

\[ \text{DM}_{1-1+1} = - [2 \times \text{CERT}-(A^*, A) - 1] = - (2 \times 1 - 1) = -1 \]

Case 3a. When A* or A** are more contained in A than in 1-A (Figure 3.3), their respective DMs will be positive. CERT-(A*, A) point lies between P (CERT-(A*, A) = 0.5) and R (CERT-(A*, A) = 1) and CERT+(A*, A) ≥ CERT-(A*, A) ≥ 0.5. The DM in this situation is calculated using Equation (3.7). For example, if CERT+(A*, A) = 0.9 and CERT-(A*, A) = 0.8,
Figure 3.2  Imprecise $A^*$ and imprecise $A$, $DM (A^*, A) = -1$, $-1 < DM (A^*, A) < 0$. 
Figure 3.3 Imprecise A* and imprecise A, 0 < DM < +1.
\[ DM_{[-1,+1]} = 2 \times \{ \text{CERT}^+(A^*, A) - \text{CERT}^-(A^*, A) \} = 2 \times (0.9 - 0.8) = +0.2 \]

Case 4a. Figure 3.4 is a special case of Figure 3.3 in which both \( \text{CERT}^+(A^*, A) \) and \( \text{CERT}^-(A^*, A) \) are 1. We can regard \( A^* \) of Figure 3.4 as the complete dilation of \( A^* \) of Figure 3.3. \( A^* \) of Figure 3.4 is interpreted as a fuzzy value \( \%\text{unknown} \) and the DM of \( A \) against \( A^* \) should be 0. Since both \( \text{CERT}^+(A^*, A) \) and \( \text{CERT}^-(A^*, A) \) are greater than 0.5 the DM in this situation is correctly calculated using Equation (3.7) as follows:

\[ DM_{[-1,+1]} = 2 \times \{ \text{CERT}^+(A^*, A) - \text{CERT}^-(A^*, A) \} = 2 \times (1 - 1) = 0 \]

Case 5a. Figure 3.5 is the opposite case of Figure 3.3. For this case, \( \text{CERT}^-(A^*, A) \geq \text{CERT}^+(A^*, A) \geq 0.5 \) and the DM should be negative. For example, if \( \text{CERT}^+(A^*, A) = 0.7 \) and \( \text{CERT}^-(A^*, A) = 1 \) the DM is calculated using Equation (3.7) as follows:

\[ DM_{[-1,+1]} = 2 \times \{ \text{CERT}^+(A^*, A) - \text{CERT}^-(A^*, A) \} = 2 \times (0.7 - 1) = -0.6 \]

3.2.2 Precise \( A^* \) and Imprecise \( A \)

Figures 3.6 to 3.10 shows the cases of imprecise \( A \)s and precise \( A^* \)s. Precise \( A^* \) can be considered as the complete contraction of imprecise \( A^* \) and hence the same computational schemes used for imprecise \( A^* \) and imprecise \( A \) should be able to work for precise \( A^* \) and imprecise \( A \).

For precise \( A^* \) and imprecise \( A \), \( \text{CERT}^-(A^*, A) = 1 - \text{CERT}^+(A^*, A) \), and thus the second situation \( (\text{CERT}^+(A^*, A) \geq 0.5 \) and \( \text{CERT}^-(A^*, A) \geq 0.5) \) of the conclusions stated in section 3.2.1 can not occur. In addition, Equation (3.8), \( DM = - \{ 2 \times \text{CERT}^-(A^*, A) - 1 \} \), can be changed as follows:
Figure 3.4 Imprecise $A^*$ and imprecise $A$, $\text{DM} = 0$. 
Figure 3.5 Imprecise $A^*$ and imprecise $A$, $-1 < DM < 0$. 
\[ DM[\cdot 1 + 1] = - \{ 2 \times (1 - \text{CERT}+(A^*, A)) - 1 \} = 2 \times \text{CERT}+(A^*, A) - 1. \]

Thus, when \( A^* \) is precise and \( A \) is imprecise, only the first equation (3.6), \( DM = 2 \times \text{CERT}+(A^*, A) - 1 \) is used for the computation of the DM.

For a precise \( A^* \) with one element and an imprecise \( A \), \( \text{CERT}+(A^*, A) \) corresponds to the GOM of \( A^* \) in a fuzzy set \( A \), and the DM represents the GOM of \( A^* \) in \( A \) whose scale is modified from \([0 \ 1]\) to \([-1 \ +1]\).

Case 1b. If \( A^* \) in Figure 3.1 contracts completely, it becomes a precise point (see Figure 3.6), and \( \text{CERT}^-(A^*, A) = 0 \) (point \( Q \)) and \( \text{CERT}^+(A^*, A) = 1 \). Since in Figure 3.6 \( A^* \) places in the point where the GOM in \( A \) is 1, \( A^* \) fully satisfies \( A \), and thus the DM = +1. The DM in this situation is correctly calculated using Equation (3.6) as follows:

\[ DM[\cdot 1 + 1] = 2 \times \text{CERT}+(A^*, A) - 1 = 2 \times 1 - 1 = +1 \]

Case 2b. When precise \( A^* \) is completely contained in \( 1-A \) (see Figure 3.7), the DM should be -1. In this situation \( \text{CERT}^+(A^*, A) = 0 \) and \( \text{CERT}^-(A^*, A) = 1 \). The DM is therefore correctly calculated as follows:

\[ DM[\cdot 1 + 1] = 2 \times \text{CERT}-(A^*, A) - 1 = 2 \times 0 - 1 = -1 \]

Case 3b. When precise \( A^* \) is more contained in \( A \) than in \( 1-A \) (see Figure 3.8), the DM is positive. For example, if \( \text{CERT}^+(A^*, A) = 0.7 \) (i.e., \( \text{CERT}^-(A^*, A) = 0.3 \)),

\[ DM[\cdot 1 + 1] = 2 \times \text{CERT}+(A^*, A) - 1 = 2 \times 0.7 - 1 = +0.4 \]
Figure 3.6 Precise $A^*$ and imprecise $A$, $DM = +1$. 
Figure 3.7 Precise $A^*$ and imprecise $A$, $DM = -1$. 
Figure 3.8 Precise $A^*$ and imprecise $A$, $0 < DM < +1$. 
Case 4b. In Figure 3.9 $A^*$ is equally contained in $A$ and 1-$A$, and the DM is 0 by definition. In this situation $\text{CERT}^+(A^*, A) = 0.5$ and $\text{CERT}^-(A^*, A) = 0.5$. The DM is correctly calculated as follows:

$$\text{DM}_{[-1 \to 1]} = 2 \times \text{CERT}^+(A^*, A) - 1 = 2 \times 0.5 - 1 = 0$$

Case 5b. When precise $A^*$ is more contained in 1-$A$ than in $A$ (see Figure 3.10), the DM should be negative. For example, if $\text{CERT}^+(A^*, A) = 0.3$ (i.e., $\text{CERT}^-(A^*, A) = 0.7$),

$$\text{DM}_{[-1 \to 1]} = 2 \times \text{CERT}^+(A^*, A) - 1 = 2 \times 0.3 - 1 = -0.4$$

3.2.3 Precise $A$

When $A$ is precise, partial matching is not allowed and the TR is $\{1/0, 0/1\}$ or $\{0/0, 1/1\}$ regardless of whether $A^*$ is imprecise or precise, and thus possible DM values are +1 and -1. For a precise $A$, $\text{DM}_{[-1 \to 1]} = 2 \times \text{CERT}^+(A^*, A) - 1$ also holds.

For $\text{TR} = \{1/0, 0/1\}$, $\text{CERT}^+(A^*, A) = 0$ and $\text{CERT}^-(A^*, A) = 1$, and the DM is equal to -1. For $\text{TR} = \{0/0, 1/1\}$, $\text{CERT}^-(A^*, A) = 1$ and $\text{CERT}^+(A^*, A) = 0$, and the DM is equal to +1.

3.2.4 The DM in [0 1] scale

If the degree of match (DM) ranges from 0 to 1, where 1 means the antecedent part is satisfied completely, 0 means the antecedent part is not satisfied at all and 0.5 represents the neutrality, the equations for the computation of the DM are changed into the following by adding 1 to the original DM equations (3.6), (3.7), (3.8) and dividing the result by 2:
Figure 3.9 Precise A* and imprecise A, DM = 0.
Figure 3.10 Precise $A^*$ and imprecise $A$, $-1 < DM < 0$. 
If \( \text{CERT}^+(A^*, A) \geq 0.5 \) and \( \text{CERT}^-(A^*, A) \leq 0.5 \),
\[
\text{DM}[0 \ 1] = \text{CERT}^+(A^*, A)
\]
(3.9)

If \( \text{CERT}^+(A^*, A) \geq 0.5 \) and \( \text{CERT}^-(A^*, A) \geq 0.5 \),
\[
\text{DM}[0 \ 1] = \{ \text{CERT}^+(A^*, A) - \text{CERT}^-(A^*, A) \} + 0.5
\]
(3.10)

If \( \text{CERT}^+(A^*, A) \leq 0.5 \) and \( \text{CERT}^-(A^*, A) \geq 0.5 \),
\[
\text{DM}[0 \ 1] = 1 - \text{CERT}^- (A^*, A)
\]
(3.11)

where \( \text{DM}[0 \ 1] \) represents the degree of match in \([0 \ 1]\) scale.

Since \( \text{CERT}^- (A^*, A) = 1 - \text{CERT}^+(A^*, A) \) for all cases of Figures 3.6 to 3.10, the \( \text{DM}[0 \ 1] \) between a precise assertion \((A^*)\) and an imprecise antecedent part \((A)\) can be computed by \( \text{DM}[0 \ 1] = \text{CERT}^+(A^*, A) \), and thus the \( \text{DM}[0 \ 1] \) corresponds to the GOM of \( A^* \) in a fuzzy set \( A \). In addition, \( \text{DM}[0 \ 1] = \text{CERT}^+(A^*, A) \) also holds for a precise \( A \) regardless of whether \( A^* \) is imprecise or precise.

3.3 Comparison to Other Computational Approaches

In this section we compare our method with other methods.

3.3.1 Possibility and Necessity

The possibility and necessity measures used separately in Cayrol's method [CFP82] merge into one in Leung's method for the computation of degree of match. In Leung's method, the degree of partial match between an assertion \( A^* \) and the antecedent \( A \) of a rule is computed by the following [LeLa88]:

If \( \text{N}(A \mid A^*) > 0.5 \)
Then \( \text{DM}[0 \ 1] = \text{P}(A \mid A^*) \)
Else \( \text{DM}[0 \ 1] = \{ \text{N}(A \mid A^*) + 0.5 \} \times \text{P}(A \mid A^*) \)
(3.12)
where $P(A \mid A^*)$ is the possibility of $A^*$ given $A$ and $N(A \mid A^*)$ is the necessity of $A^*$ given $A$.

The following are the formulas of the possibility and necessity measures between $A^*$ and $A$ [CFP82] [LeLa88]:

\begin{align*}
P(A \mid A^*) &= \max \{ \min (\mu_A, \mu_{A^*}) \} \tag{3.13} \\
N(A \mid A^*) &= 1 - P(1-A \mid A^*) \tag{3.14}
\end{align*}

Note that the $DM_{[0,1]}$ ranges from 0 to 1 in Leung's computational scheme since both $P(A \mid A^*)$ and $N(A \mid A^*)$ ranges from 0 to 1.

We see that the possibility measure corresponds to $\text{CERT}^+(A^*, A)$ defined earlier and the necessity measure is equal to $1 - \text{CERT}^-(A^*, A)$. When $A^*$ is completely contained in $A$ (see Figure 3.1 and Figure 3.6), $\text{CERT}^-(A^*, A) \leq 0.5$ (i.e., $N(A \mid A^*) \geq 0.5$) and the computed $DM_{[0,1]}$ using Leung's scheme is equal to the $DM_{[0,1]}$ obtained using Equation (3.9) since $P(A \mid A^*) = \text{CERT}^+(A^*, A) = \max \{ \min (\mu_A, \mu_{A^*}) \}$ is used in both schemes.

However, the two schemes produce different $DM_{[0,1]}$ when imprecise $A^*$ is completely contained in $1-A$ (Figure 3.2). For example, when $A^* = 1-A$, the $DM_{[0,1]}$ should be 0 by definition. The $DM_{[0,1]}$ computed using Equation (3.10) or Equation (3.11) of the proposed scheme is 0:

\begin{align*}
\text{CERT}^+(A^*, A) &= 0.5, \quad \text{CERT}^-(A^*, A) = 1 \\
DM_{[0,1]} &= (\text{CERT}^+(A^*, A) - \text{CERT}^-(A^*, A)) + 0.5 = (0.5 - 1) + 0.5 = 0 \\
DM_{[0,1]} &= 1 - \text{CERT}^-(A^*, A) = 1 - 1 = 0
\end{align*}

On the other hand, the $DM_{[0,1]}$ computed using Equation (3.12) of Leung's scheme is
0.25, leading to the invalid result:

\[ P(A \mid A^*) = 0.5, \quad N(A \mid A^*) = 0 \]
\[ DM[0 \ 1] = [N(A \mid A^*) + 0.5] \times P(A \mid A^*) \]
\[ = (0 + 0.5) \times 0.5 \]
\[ = 0.25 \]

For precise \( A^* \) and imprecise \( A \) (see Figures 3.6 to 3.10), the \( DM[0 \ 1] \) should be equal to the GOM of \( A^* \) in \( A \) since \( A^* \) can be regarded as an element in a universe of discourse and \( A \) is a fuzzy set of the same universe of discourse. Our computational scheme satisfies the above requirement for every case of Figures 3.6 to 3.10. However, Leung's scheme does not satisfy the requirement when precise \( A^* \) is more contained in \( 1-A \) than in \( A \) (see Figure 3.10). Assume that \( CERT^+(A^*, \ A) = 0.2 \) (i.e., the GOM of \( A^* \) in \( A = 0.2 \)) in Figure 3.10. The \( DM[0 \ 1] \) computed using Equation (3.9) or Equation (3.11) is equal to the GOM:

\[ DM[0 \ 1] = CERT^+(A^*, \ A) = 1 - CERT^-(A^*, \ A) = 0.2 \]

On the other hand, the \( DM[0 \ 1] \) computed using Leung's scheme is not equal to the GOM as follows:

\[ P(A \mid A^*) = 0.2, \quad N(A \mid A^*) = 0.2 \]
\[ DM[0 \ 1] = [N(A \mid A^*) + 0.5] \times P(A \mid A^*) \]
\[ = (0.2 + 0.5) \times 0.2 \]
\[ = 0.14 \]
3.3.2 The Degree to which the Antecedent Includes an Assertion

In Ogawa's method, the degree to which \( A \) includes \( A^* \) is used to indicate the degree of partial match of \( A \) against \( A^* \) [OFY84]. The degree to which \( A \) includes \( A^* \), \( \text{INC}(A, A^*) \), is given by the following:

\[
\text{DM}[0 \ 1] = \text{INC}(A, A^*) = \frac{\Sigma \min\{\mu_A(x_i),\mu_{A^*}(x_i)\}}{\Sigma \mu_A(x_i)} \quad (3.15)
\]

When imprecise \( A^* \) is completely contained in \( A \) (see \( A^* \) and \( A \) in Figure 3.1), the \( \text{DM}[0 \ 1] \) should be equal to 1 as described in Section 3.2.1. However, the computed \( \text{DM}[0 \ 1] \) using Ogawa's method is less than 1 except the case of \( A = A^* \) (i.e., \( \text{CERT}+(A^*, A) = 1 \) and \( \text{CERT}-(A^*, A) = 0.5) \):

For every \( x_i \), it is true that \( \mu_A(x_i) \geq \mu_{A^*}(x_i) \).

Thus, \( \Sigma \min\{\mu_A(x_i),\mu_{A^*}(x_i)\} = \Sigma \mu_{A^*}(x_i) < \Sigma \mu_A(x_i) \) for every \( x_i \).

From Equation (3.15) and the above relationship, \( \text{INC}(A, A^*) < 1 \)

When \( A^* = 1-A \), \( \min\{\mu_A(x_i),\mu_{1-A}(x_i)\} \) is greater than or equal to 0 for any \( x_i \), and thus \( \text{INC}(A, A^*) \) is not equal to 0.

For \( A^* \) and \( A \) of Figure 3.3 and Figure 3.4, the \( \text{DM}[0 \ 1] \) should be less than 1 as described in Section 3.2.1. On the other hand, \( \text{INC}(A, A^*) = 1 \), which is absolutely invalid:

For every \( x_i \), it is true that \( \mu_A(x_i) \leq \mu_{A^*}(x_i) \).

Thus, \( \Sigma \min\{\mu_A(x_i),\mu_{A^*}(x_i)\} = \Sigma \mu_A(x_i) \) for every \( x_i \).

From Equation (3.15) and the above relationship, \( \text{INC}(A, A^*) = 1 \)
For precise A* and imprecise A (Figures 3.6 to 3.10), INC(A, A*) is not equal to the GOM of A* in A. For example, when \( \text{CERT}\times(\text{A*}, \text{A}) = 0.2 \) (i.e., the GOM of A* in A=0.2), INC(A, A*) is less than 0.2 by the following:

\[
\Sigma \min \{\mu_A(x_i), \mu_{A*}(x_i)\} = 0.2
\]
\[
\Sigma \mu_A(x_i) > 1
\]
\[
\text{INC}(A, A*) < 0.2
\]

3.4 Conjunction and Disjunction

If a rule has several propositions connected by conjunctive operators in its antecedent part, the degrees of match of those propositions should be combined in order to indicate the total degree of match of the rule:

\[
\text{Total DM} = f(\text{DM}_1, \text{DM}_2, \ldots, \text{DM}_i, \ldots, \text{DM}_n)
\]

where DM\(_i\) is the degree of match of the i-th antecedent proposition against its corresponding assertion. The default function for f is minimum.

If more than one rule have the same consequent part, they can be regarded as rules split from a disjunctive rule of which disjuncts are the antecedent parts of the rules. For example, a disjunctive rule if (X\(_1\) is A\(_{11}\) and X\(_2\) is A\(_{12}\)) or (if X\(_1\) is A\(_{31}\) and X\(_3\) is A\(_{33}\)) then Class is C can be reexpressed as two rules:

if X\(_1\) is A\(_{11}\) and X\(_2\) is A\(_{12}\) then Class is C

if X\(_1\) is A\(_{31}\) and X\(_3\) is A\(_{33}\) then Class is C.

The DM of a disjunctive rule with \( k \) disjuncts is computed based on each total DM of each


disjunct:

\[ \text{DM of disjunction} = g(\text{DM of disjunct}_1, \text{DM of disjunct}_2, \ldots, \text{DM of disjunct}_k) \]

The default function for \( g \) is maximum.

The final DM of each classification rule is used in order to indicate the certainty value of classification. The bigger the DM, the more is the possibility that an object is classified to the class.

3.5 Examples

The following are some examples of fuzzy expert classification systems for which our proposed computational scheme of the DM can be used successfully.

Example 3.1 Expert Risk Analysis System [KMW87]: This system is aimed at evaluation of potential risk of overexertion injury due to manual lifting tasks. Some typical classification rules are as follows:

(Rule 3): If Load size is Very Heavy

and Frequency of lift is High

Then Task risk is Extremely High.

(Rule 5): If Load size is Very Heavy

and Frequency of lift is Medium

and Horizontal distance away from the body is Close

and Height is Low

Then Task risk is High.
Rule 3 has two attributes and its risk class is Extremely High. Rule 5 has four imprecise attributes: Very Heavy, Medium, Close and Low. When the antecedent parts of rule 5 are satisfied, the assigned risk class is High.

Suppose the assertions given by a user are (Load size is 35 kg, Frequency of lift is High, Horizontal distance away from the body is 20 cm, Height is More-or-less Low). The final DM of rule 5 is \( f(\text{DM}(35 \text{ kg, Very Heavy}), \text{DM}(\text{High, Medium}), \text{DM}(20 \text{ cm, Close}), \text{DM}(\text{More-or-less Low, Low})) \) and the DM of rule 3 is \( f(\text{DM}(35 \text{ kg, Very Heavy}), \text{DM}(\text{High, High})) \) where DM\((x, y)\) means the degree of match between an assertion \( x \) and an attribute value \( y \) in the antecedent part of a rule, and \( f \) is a mathematical function for combining several DMs as discussed in Section 3.4. \( \text{DM}(35 \text{ kg, Very Heavy}) \) and \( \text{DM}(20 \text{ cm, Close}) \) are the degrees of match between a precise assertion and an imprecise antecedent proposition. \( \text{DM}(35 \text{ kg, Very Heavy}) \) can be computed directly from the GOM of 35 kg in a fuzzy set Very Heavy. In other words, if the GOM of 35 kg in Very Heavy is 0.9, the DM in \([-1, +1]\) scale is computed as follows:

\[
\text{DM}_{[-1,+1]} = 2 \times \text{CERT}_+ - 1 \\
= 2 \times \text{GOM} - 1 \\
= 2 \times 0.9 - 1 \\
= +0.8
\]

\( \text{DM}(35 \text{ kg, Very Heavy}) \) becomes 0.9 in \([0, 1]\) scale by the following:

\[
\text{DM}_{[0,1]} = \text{CERT}_+ \\
= \text{GOM} \\
= 0.9
\]
Similarly, DM(20 cm, Close) is determined by the GOM of 20 cm in a fuzzy set Close. Each of DM(High, Medium), DM(More-or-less low, Low) and DM(High, High) is the degree of match of an imprecise antecedent proposition against an imprecise assertion. The DM(x, y) can be computed using the computational scheme described in Section 3. If the classification system has only two rules as stated above and the DM of rule 3 is greater than the DM of rule 5, task risk will be Extremely High.

Example 3.2 Information Retrieval from Database: Data bases are widely used for storing and manipulating large collections of data. Retrieval from a data base using a query can be considered as a special expert classification system in which the query and records of data base correspond to a classification rule and a number of objects to be classified, respectively. The records are fetched and matched with the query (classification rule). For each record, we obtain its degree of match. If the matching degree obtained exceeds a prespecified threshold, the record is introduced into an output file, otherwise is neglected. The final output file contains all the records which sufficiently match the query with their corresponding degree of match.

Suppose we have a data base (Table 3.1) and a query as follows:

Query: Find all men who are tall and weigh over 150 lbs.

The query can be translated into a classification rule:

\[
\text{if } \text{Sex is Male and Height is Tall and Weight is } > 150 \text{ lbs} \\
\text{Then Class is Yes.}
\]

The first attribute value of the rule is precise and is satisfied by Michael, John and
<table>
<thead>
<tr>
<th>Name</th>
<th>Sex</th>
<th>Height</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michael</td>
<td>Male</td>
<td>More-or-less Tall</td>
<td>Heavy</td>
</tr>
<tr>
<td>John</td>
<td>Male</td>
<td>6'1&quot;</td>
<td>170 lbs</td>
</tr>
<tr>
<td>Mary</td>
<td>Female</td>
<td>Short</td>
<td>Light</td>
</tr>
<tr>
<td>Mark</td>
<td>Male</td>
<td>Very Tall</td>
<td>Moderate</td>
</tr>
<tr>
<td>Judy</td>
<td>Female</td>
<td>5'6&quot;</td>
<td>140 lbs</td>
</tr>
</tbody>
</table>
Mark. The second one, Tall, is imprecise and the third one, > 150 lb, is precise. For Michael, the final degree of match is \( f(\text{DM(Male, Male)}, \text{DM(More-or-less Tall, Tall)}, \text{DM(Heavy, > 150 lbs)}) \). DM(Male, Male) is the degree of match between a precise assertion and a precise antecedent proposition, which is equal to +1. DM(More-or-less Tall, Tall) is the degree of match of an imprecise antecedent proposition against an imprecise assertion. DM(Heavy, > 150 lbs) is the degree of match between an imprecise proposition and a precise one. For Judy, the degree of match is \( f(\text{DM(Female, Male)}, \text{DM(5'6", Tall)}, \text{DM(140 lbs, > 150 lbs)}) \). DM(Female, Male) = -1. DM(5'6", Tall) is the degree of match between a precise assertion and an imprecise antecedent proposition and can be computed from the GOM of 5'6" in a fuzzy set Tall. DM(140 lbs, > 150 lbs) is the degree of match between precise propositions. DM(140 lbs, > 150 lbs) is equal to -1 in \([-1 +1]\) scale and becomes 0 in \([0 1]\) scale.

Example 3.3 Knowledge-based decision making [Moc89a] [Moc89b]: Decision-making can be described as a classification problem, and thus knowledge-based decision making systems can be regarded as expert classification systems. For example, suppose that we are going to determine whether or not to cut the grass. The rule set for determining the need to cut is as follows:

Rule 1: If Grass is Long and Season is Growing

Then Need is Yes.

Rule 2: If Grass is Short or Season is Winter

Then Need is No.

Suppose that the sets of imprecise values of attributes Grass and Season are
(Long, Medium, Short) and (Growing, Non-growing), respectively. The graphical representations for attribute values are shown in Figures 3.11 and 3.12. Notice that Winter is a non-fuzzy subset of the universe of discourse describing Season. Possible class names for Need are Yes and No.

A careful user will measure and provide a precise information such as "Grass is 3.5 inches long" to the classification system. DM(3.5 inches, Long) and DM(3.5 inches, Short) are the DMs between precise assertions and imprecise antecedent propositions. However, a user is usually apt to provide an imprecise assertion such as "Long" rather than a precise assertion such as "3.5 inches". Then, the DM is computed from an imprecise assertion and an imprecise antecedent proposition.

Assume that the date is December 12. DM(December 12, Growing) is the DM of an imprecise antecedent proposition against a precise assertion and is equal to -1 in [-1, +1] scale. DM(December 12, Winter) is the DM between a precise assertion and a precise antecedent proposition and is equal to +1.

Rule 2 is a disjunctive rule and can be split into two rules:

Rule 2-1: If Grass is Short
Then Need is No.

Rule 2-2: If Season is Winter
Then Need is No.

Since DM(December 12, Winter) = +1, Rule 2-2 is satisfied completely and the decision made by this expert classification system is that there is no need to cut the grass. ♦
Figure 3.11 Attribute values of Grass.
Figure 3.12 Attribute values of Season.
Example 3.4 Strategic Planning [Hal87] [Hal74] [Moc89b]: Hall [Hal74] defines strategic planning as "long-range anticipatory decision making which is undertaken to affect the direction of the entire organization". The decisions to be taken have a strategic impact, and therefore the implications of making good or bad choice can be very severe indeed. Since strategic planning problems are complex and highly unstructured, and some strategic information is non-quantitative, strategic planning is an area in which a fuzzy expert system may be of use [Hal87]. Each rule in fuzzy expert systems for strategic planning has the form of "If X is A, Then Strategy is B". Our computational scheme of degree of match can be used to evaluate alternative strategies for the pursuit of a strategic objective.


Possible attribute values of Industry-type are Declining, Emerging, Fragmented, and Mature. The value of Industry-type is determined by a subsystem (module consisting of rules regarding Industry-type) and inputs to the subsystem (imprecise values of several factors ), which are given by users. Possible imprecise values of both Company-market-advantage and Company-finance are Weak, Average and Strong, and the values of two attributes are determined based on their corresponding subsystems and the values of relevant factors. Users need to provide an information about Company-strength-overall, Competitor-strength-overall, Company-organization-flexibility and Company-market-share, too. Figure 3.13 shows the hierarchical dependency diagram for the strategic corporate planning system. Notice that the hierarchical structure can be converted into the non-hierarchical form shown in Figure 3.14 using the induction method [Moc89a].
Each rule in the strategy modules of Figure 3.13 and Figure 3.14 recommends one of five alternative strategies. Rules in the strategy module of the non-hierarchical structure have greater number of attributes than rules in the strategy module of the hierarchical structure. The following is a sample rule stored in the strategy module of hierarchical knowledge base:

If Industry-type is Declining
and Company-strength-overall is Strong
and Competitor-strength-overall is Weak
and Company-market-advantage is Strong
and Company-organization-flexibility is High
and Company-finance is Average
and Company-market-share is Low

Then Strategy is Selective-investment.

This rule recommends Selective-investment as the strategy of the company. Suppose the values of seven factors are as follows:

Industry-type is (Declining with 0.9, Fragmented with 0.6,
Emerging with 0.3, Mature with 0.2)
Company-strength-overall is Average
Competitor-strength-overall is Weak
Company-market-advantage is (Weak with 1.0, Average with 0.5,
Strong with 0.3)
Company-organization-flexibility is High
Company-finance is (Average with 0.8, Strong with 0.4, Weak with 0.2)
Company-market-share is High
"Industry-type is (Declining with 0.9, Fragmented with 0.6, Emerging with 0.4, Mature with 0.3)" is the result of classification performed by a subsystem, Industry-type module. It represents that Industry-type is Declining with 0.9 possibility, Industry-type is Fragmented with 0.6 possibility, Industry-type is Emerging with 0.3 possibility and Industry-type is Mature with 0.2 possibility. "Company-market-advantage is (Weak with 1.0, Average with 0.5, Strong with 0.3)" and "Company-finance is (Average with 0.8, Strong with 0.4, Weak with 0.2)" are also the outputs of subsystems called Company-finance module and Company-market-advantage module, respectively. "Company-strength-overall is Average", "Competitor-strength-overall is Weak", "Company-organization-flexibility is High", and "Company-market-share is High" are provided directly by the user. Then, the DM of this rule can be computed by the following:

\[
\begin{align*}
&f [\max \{0.9 \times \text{DM(Declining, Declining)}, \\
&0.6 \times \text{DM(Fragmented, Declining)}, \\
&0.3 \times \text{DM(Emerging, Declining)}, \\
&0.2 \times \text{DM(Mature, Declining)}\}, \\
&\text{DM(Average, Strong)}, \\
&\text{DM(Weak, Weak)}, \\
&\max \{1.0 \times \text{DM(Weak, Strong)}, \\
&0.5 \times \text{DM(Average, Strong)}, \\
&0.3 \times \text{DM(Strong, Strong)}\}, \\
&\text{DM(High, High)}, \\
&\max \{0.8 \times \text{DM(Average, Average)}, \\
&0.4 \times \text{DM(Strong, Average)}, \\
&0.2 \times \text{DM(Weak, Average)}\}, \\
&\text{DM(High, Low)}].
\end{align*}
\]
Figure 3.13 The hierarchical dependency diagram for the strategic corporate planning system.
Figure 3.14  The non-hierarchical dependency diagram for the strategic corporate planning system.
If there are several rules that recommend Selective-investment as the strategy, their DMs are combined through a function $g$ (see Section 3.4), and the result represents the possibility that Selective-investment is appropriate strategy for the organization. The final results from processing all rules in the strategic module are reported to the user in descending or ascending order of possibility ratings. Each rating is accompanied by the associated strategy. For example, Strategy is (Selective investment with 0.9, Niche concentration with 0.6, Hold position with 0.5, Aggressive investment with 0.4, Get out of business with 0.2). The user must then exercise his own judgement to determine how to interpret these possibility ratings of alternative strategies and how to use them. The optimal strategy is usually that for which possibility rating is maximal.

3.6 Conclusions

A computational scheme of the degree of match between assertions and the antecedent parts of a rule was discussed. All four possible combinations of the antecedent of a rule and an assertion were considered, including 1) precise antecedent and precise assertion, 2) precise antecedent and imprecise assertion, 3) imprecise antecedent and precise assertion, and 4) imprecise antecedent and imprecise assertion. The proposed scheme can be used consistently to compute the degree of match for any type of combination and can be applied to a fuzzy expert classification system in order to indicate the certainty value of object classification.

The computed degree of partial match between a precise assertion ($A^*$) and an imprecise antecedent ($A$) using the proposed scheme is equal to the grade of membership of $A^*$ in a fuzzy set $A$, which is a requirement for a computational scheme of the degree of match violated by the other methods.

Furthermore, the DM computed by our computational scheme can be combined with the certainty factors (CF) used in MYCIN's model since it corresponds to the certainty
value. If each assertion is assigned a certainty factor $CFA_i$ as used in MYCIN's model and each rule is also assigned a certainty factor $CF_r$ in order to indicate the uncertainties of the assertion and the rule, the $DM[0 \ 1]$ of each proposition is modified by $CFA_i \times DM_i[0 \ 1]$. The final certainty value of classification is $CF_r \times f(CFA_1 \times DM_1[0 \ 1], CFA_2 \times DM_2[0 \ 1], \ldots, CFA_i \times DM_i[0 \ 1], \ldots, CFA_n \times DM_n[0 \ 1])$ where $DM_i[0 \ 1]$ is the $DM[0 \ 1]$ of the $i$-th antecedent proposition against its corresponding assertion with a certainty factor $CFA_i$. If a certainty factor is given in an imprecise form, the multiplication of fuzzy sets instead of $\times$ is used. The multiplication of fuzzy sets can be processed by the use of the extension principle [Zad75].
CHAPTER 4
PARALLEL, SELF-ORGANIZING, HIERARCHICAL NEURAL NETWORKS
WITH COMPETITIVE LEARNING AND SAFE REJECTION SCHEMES

4.1 Introduction

There are both advantages and disadvantages in well-known learning algorithms for neural networks. The backpropagation technique [RHW88] is one of the most widely used supervised learning algorithms, which usually has more accuracy than competitive learning in supervised classification problems. However, the backpropagation model is an extremely slow learning method and performs biologically implausible error back propagation. In comparison, competitive learning [AKC90] [Des88] [KKL90] [Koh88] [Koh89] [Koh90] [Kos91] [RuZi85] learns rapidly even though it usually has less accuracy than backpropagation in pattern recognition.

Recently, the parallel, self-organizing, hierarchical neural network (PSHNN) architecture was proposed for the purposes of decreasing system complexity, increasing classification accuracy, avoiding local minima, reducing learning and recall rates, and achieving true parallelism [ErHo89] [ErHo90] [EHB90] [Hon90].

However, the PSHNN still needs long learning times when supervised learning algorithms such as the delta rule and the backpropagation algorithm are used in each SNN. In addition, the classification performance of the PSHNN is strongly dependent on its rejection scheme. Thus, it is possible that we can improve the classification accuracy by developing better rejection schemes.

In this chapter, parallel, self-organizing, hierarchical neural network with competitive learning and safe rejection schemes (CSNN) is presented to get around the
disadvantages of both supervised learning algorithms and competitive learning algorithms. We first compute the reference vectors in parallel for all classes using competitive learning. Then, safe rejection boundaries are constructed in the training procedure so that there are no misclassified training vectors. The experimental results discussed in Section 4.4 show that the proposed network has more speed and accuracy than the multilayer neural network trained by backpropagation and the PSHNN trained by the delta rule.

Kohonen [KKL90] [Koh88] [Koh89] [Koh90] developed several versions of competitive learning algorithms. The main difference between our system and Kohonen's algorithms is safe rejection schemes. The reference vectors are used for classification by the nearest neighbor principle in Kohonen's methods. In our system, the decision surface of classification is determined by the rejection schemes in addition to the reference vectors.

Carpenter and Grossberg [CaGr87] [CaGr88] developed a number of neural network architectures based on adaptive resonance theory (ART). For example, ART I also uses competitive learning in order to choose the winning prototype (output unit) for each input vector. When an input vector is sufficiently similar to the winning prototype, the prototype represents the input correctly. Once a stored prototype is found that matches the input vector within a specific tolerance (the vigilance), that prototype is adjusted to make it still more like the input vector. If an input is not sufficiently similar to any existing prototype, a new classification category is formed by storing a prototype that is like the input vector. If the vigilance factor \( \rho \), with \( 0 < \rho < 1 \), is large, many finely divided categories are formed. On the other hand, a small \( \rho \) produces course categorization.

The CSNN system is different from ART I in that 1) all of the available output processing elements are used whereas in ART I the value of the vigilance factor determines how many are used, 2) the number of classes is predefined and each input vector is tagged with its correct class whereas in ART I the vigilance factor determines the number of classes, 3) an input vector is tested for similarity to the reference vectors by an elaborate
rejection scheme; if the input vector is rejected, it is fed into the next SNN. In ART I, the vigilance factor determines acceptance or rejection and a new classification category is created in case of rejection. In other words, the CSNN creates a new SNN whereas ART expands the dimension of its output layer for processing of the rejected training vectors, 4) the CSNN transforms nonlinearly the input vectors rejected by the previous SNN, etc.

This chapter is organized into five sections. Section 4.2 introduces the new learning algorithm with safe rejection schemes, as well as Method I and Method II of learning reference vectors. In Section 4.3, the theoretical analysis of the proposed system is described. In Section 4.4, the performance results obtained with the proposed algorithms in comparison to the backpropagation network and the PSHNN with the delta rule learning algorithm are presented. A comparative study of two methods for learning reference vectors is also described in Section 4. Section 4.5 is conclusions.

4.2 Parallel, Self-Organizing, Hierarchical Neural Network with Competitive Learning and Safe Rejection Schemes [CEL92b]

This section presents the training and the testing algorithms of the CSNN. In the training phase, the reference vectors and the safe rejection boundaries are obtained. Given an object with unknown classification, the testing algorithm is used to classify, based on the trained neural network.

4.2.1 Competitive Learning and Rejection Schemes

If neural networks are trained using only competitive learning algorithms, the reference vectors are used for classification by the nearest neighbor principle, namely by the comparison of the test vector $X$ with the reference vector $W$ in the nearest neighbor sense. The classification accuracy relies on how correctly the reference vectors are computed. However, it is difficult to compute the reference vectors which produce
globally minimum errors because reference vectors depends on initial reference vectors, learning rate, the order of training samples, etc.

To overcome the limitations of competitive learning algorithms, our system incorporates the rejection schemes. The purpose of the rejection scheme is to reject the "hard" vectors, which are difficult to classify and to accept the correctly classified vectors as much as possible. We train the next SNN with only those training vectors that are rejected in the previous SNN. During the training procedure, the correct classes are known, and we can check which ones are misclassified. However, this is not possible during the testing procedure. Thus, we need some criteria to reject error-causing vectors during both the training procedure and the testing procedure. For this purpose, we construct rejection boundaries for the reference vectors during the training procedure, and use them during both the training procedure and the testing procedure.

The classification performance of the CSNN depends strongly on how well the rejection boundaries are constructed because the decision surface of classification is to a large degree determined by the rejection boundaries. One promising way for the construction of rejection boundaries is to use safe rejection schemes. Two possible definitions for safe rejection schemes are as follows:

Definition 1: A rejection scheme is said to be "safe" if every training vector is classified correctly and rejected otherwise by each SNN so that there are no misclassified training vectors if enough SNN's are utilized.

Definition 2: A rejection scheme is said to be "unsafe" if there exists a misclassified training vector at the output of the total network.
4.2.2 Training

Assume that a training set of vectors with known classification is utilized. Each sample in the training set represents an observed case of an input-output relationship and can be interpreted as consisting of attribute values of an object with a known class. The training procedure is described as follows:

Initialize: \( m = 1 \)

STEP 1. For SNN\(_m\) (the \( m\)-th stage neural network), compute the reference vectors using a competitive learning method.

STEP 2. With the training vectors belonging to each class, construct safe rejection boundaries for reference vectors belonging to each class, as discussed in Section 4.2.3.

STEP 3. Determine the training vectors rejected by all safe rejection schemes. If there is no rejected training vector or the predetermined maximum number of SNN's are exceeded, stop the training procedure. Otherwise, go to STEP 4.

STEP 4. Transform nonlinearly the rejected training vectors.

STEP 5. \( m = m + 1 \). Go to STEP 1.

Assume a predetermined number of processing elements, each one provided with a reference vector \( W_k \). Their number may be a multiple \( L \) (say, ten times) of the number of classes considered. All of the available output processing elements are used in the CSNN like in Kohonen learning [KKL90] [Koh88] [Koh89] [Koh90], whereas in ART [CaGr87] [CaGr88], the value of the vigilance parameter determines how many are used. The variable \( L \) is determined by the total number of output processing elements and the number of classes:
\[ L = \frac{\text{the total number of processing elements}}{\text{the number of classes}} \]

In STEP 1 of the training procedure, we investigated two possible methods for the computation of the reference vectors. In Method I, all the reference vectors are computed together using the whole training data set. This is the way the reference vectors are computed in conventional competitive learning characterized by Equations (2.18) to (2.20).

In Method II, competitive learning is performed in parallel for all the classes as follows:

For the \( j \)-th class,
\[ \begin{align*}
W_j(t+1) &= W_j(t) + C_j(t) [X_j(t) - W_j(t)] & \text{if } i \text{ wins} \\
W_j(t+1) &= W_j(t) & \text{if } i \text{ loses}
\end{align*} \]

(4.1)

(4.2)

where \( W_j(t+1) \) represents the value of the \( i \)-th reference vector of class \( j \) after adjustment; \( W_j(t) \) is the value of the \( i \)-th reference vector before adjustment; \( X_j(t) \) is the training vector belonging to the \( j \)-th class used for updating the reference vectors at time \( t \); \( C_j(t) \) is the learning rate coefficient for the computation of the reference vectors of the \( j \)-th class.

In competitive learning, a reference vector is updated every time it wins the competition. The learning rate coefficient used during updating plays a critical role in determining error performance as well as speed of convergence. Too small learning rate allows very slow convergence. Too large learning rate causes oscillations between relatively poor solutions rather than convergence to a stable solution. Until now, slowly decreasing scalar time functions have usually been used as the learning rates in competitive learning.

Recently, competitive learning with multiple learning rate was proposed [KESL92]. Its basic idea is to update the reference vectors using multiple learning rates, each of which
is assigned to only one reference vector. Thus, the learning rate coefficient is not a scalar but a vector. The dimension of the learning rate vector is equal to the number of reference vectors. Each reference vector is updated independently of the others using its own learning rate. In other words, each learning rate is changed only when its corresponding reference vector wins the competition, and the learning rates of losing reference vectors are not changed. This learning method is believed to be biologically more plausible than conventional competitive learning methods with a scalar learning rate. The following equations describe competitive learning with multiple learning rates when it is used in Method I:

\[
W_k(t+1) = W_k(t) + C_k(t) [X(t) - W_k(t)] \\
W_k(t+1) = W_k(t)
\]

if \( k \) wins \hspace{1cm} \text{(4.3)}

if \( k \) loses \hspace{1cm} \text{(4.4)}

where \( W_k(t+1) \) represents the value of the \( k \)-th reference vector after adjustment; \( W_k(t) \) is the value of the \( k \)-th reference vector before adjustment; \( X(t) \) is the training vector used for updating the reference vectors at time at time \( t \). \( C_k(t) \) is the learning rate coefficient of the \( k \)-th reference vector.

If competitive learning with multiple learning rates is used in Method II, Equations (4.1) and (4.2) are changed to the following:

For the \( j \)-th class,

\[
W_i^j(t+1) = W_i^j(t) + C_i^j(t) [X^j(t) - W_i^j(t)] \\
W_i^j(t+1) = W_i^j(t)
\]

if \( i \) wins \hspace{1cm} \text{(4.5)}

if \( i \) loses \hspace{1cm} \text{(4.6)}

where \( C_i^j(t) \) represents the learning rate coefficient for the computation of the \( i \)-th reference vector of the \( j \)-th class.
When the reference vectors are computed separately for each class and in parallel, the learning speed is improved by a factor approximately equal to the number of classes, in comparison to conventional competitive learning. Method I is obviously more optimal when traditional competitive learning algorithms are used without rejection schemes. Interestingly, Method II gives better performance in terms of classification accuracy when rejection schemes are used, as discussed in Section 4.4.3.

4.2.3 Two Safe Rejection Schemes

In STEP 2 of the training algorithm, the rejection boundaries are constructed. The following are two safe rejection schemes to construct the safe rejection boundaries for the reference vectors belonging to the j-th class:

**RADPN (RADP and RADN)**

Initialize: \( k = 1 \), \( \text{RADP}_{ni} = w_{ni} \) and \( \text{RADN}_{ni} = w_{ni} \) for \( n = 1, 2, ..., I \) and \( i = 1, 2, ..., L \).

The variable \( w_{ni} \) represents the \( n \)-th element of a reference vector \( W_i \) where \( I \) is the dimension of the training vectors and \( L \) is the number of reference vectors that belong to the j-th class.

**STEP1.** For a training vector \( X_j(k) \) belonging to the j-th class, find the nearest reference vector \( W_i \) using Euclidean distance measure.

**STEP 2.** Compare \( x_{nj}(k) \), the \( n \)-th element of \( X_j(k) \), with \( w_{ni} \).

1) If \( x_{nj}(k) \) is bigger than \( w_{ni} \), check whether \( x_{nj}(k) \) is outside the previous rejection boundary \( \text{RADP}_{ni} \).
   
   A) If \( x_{nj}(k) > \text{RADP}_{ni} \), \( \text{RADP}_{ni} \) is modified to \( \text{RADP}_{ni} = x_{nj}(k) \).
   
   B) If \( x_{nj}(k) \leq \text{RADP}_{ni} \), \( \text{RADP}_{ni} \) is not changed.
2) If \(x_{nj}(k)\) is smaller than \(w_{ni}\), check whether \(x_{nj}(k)\) is outside the previous rejection boundary \(RAD_{ni}\).

   A) If \(x_{nj}(k) < RAD_{ni}\), \(RAD_{ni}\) is modified to \(RAD_{ni} = x_{nj}(k)\).
   B) If \(x_{nj}(k) \geq RAD_{ni}\), \(RAD_{ni}\) is not changed.

STEP 3. Check whether \(X_j(k)\) is the last training vector belonging to the \(j\)-th class.

1) If \(k = M_j\) where \(M_j\) is the number of the training vectors belonging to the \(j\)-th class, stop the procedure and save the current \(RAD_{ni}\) and \(RAD\).

2) If \(k < M_j\), \(k = k + 1\) and go to STEP1.

\[
RAD
\]

Initialize: \(k = 1\).

\[
RAD_{pi} = 0 \quad \text{for } i = 1, 2, ..., L.
\]

STEP 1. For a training vector \(X_j(k)\) belonging to the \(j\)-th class, find the nearest reference vector \(W_i\) using the following measure:

\[
R_{pi} = \sqrt[p]{\sum_{m}^i (x_{nj}(k) - w_{ni})^p}
\]

STEP 2. Compare \(R_{pi}\) with \(RAD_{pi}\).

1) If \(R_{pi} > RAD_{pi}\), \(RAD_{pi}\) is modified to \(RAD_{pi} = R_{pi}\).

2) If \(R_{pi} \leq RAD_{pi}\), \(RAD_{pi}\) is not changed.

STEP 3. Check whether \(X_j(k)\) is the last training vector belonging to the \(j\)-th class.

1) If \(k = M_j\) where \(M_j\) is the number of the training vectors belonging to the \(j\)-th class, stop the procedure and save the current \(RAD_{pi}\).

2) If \(k < M_j\), \(k = k + 1\) and go to STEP1.

The above procedures, \(RADPN\) and \(RAD\), can be executed in parallel for all classes \((j = 1, 2, ..., C\) where \(C\) is the number of possible classes) or can be executed serially. Figure 4.1 shows \(RADPN\) for a 2-dimensional training data set that has two classes. Only
one reference vector is defined for each class. Figure 4.2 shows an example (RAD\textsuperscript{2}) of RAD for the same data set as in Figure 4.1.

Each reference vector generates the interconnection weights between the input nodes and a particular output node identified with the reference vector. The output of an output node is set to 1 when a training vector is inside or on its rejection boundary. It has output 0 when a training vector is outside its rejection boundary.

For RADPN, a training vector $X(k)$ is judged to be inside or on the rejection boundary if it satisfies the following condition for every $n = 1, 2, \ldots, I$:

$$\text{RADN}_n \leq x_n(k) \leq \text{RADP}_n$$  \hspace{1cm} (4.7)

$\text{RADN}_n$ and $\text{RADP}_n$ represent the $n$-th elements of RADN and RADP of the reference vector identified with the output node, respectively. The variable $x_n(k)$ is the $n$-th element of $X(k)$. If at least one element of $X(k)$ does not satisfy Equation (4.7), $X(k)$ is said to be outside the rejection boundary.

For $X(k)$ to be inside or on the rejection boundary of RAD, it should satisfy the following condition:

$$\sqrt[p]{\sum_{n=1}^{I} (x_n(k) - w_n)^p} \leq \text{RADP}$$  \hspace{1cm} (4.8)

The variable $w_n$ represents the $n$-th element of a reference vector $W$. $X(k)$ exists outside the rejection boundary of RAD if the following condition is satisfied:

$$\sqrt[p]{\sum_{n=1}^{I} (x_n(k) - w_n)^p} > \text{RADP}$$  \hspace{1cm} (4.9)
Figure 4.1 RADP and RADN for 2-D data.
Figure 4.2 RAD for 2-D data.
If one or more of output nodes belonging to a class has output 1, the class output is set to 1. If none of reference vectors belonging to a class has output 1, the class output is set to 0. A training vector is rejected by the rejection scheme if more than one class have output 1. A training vector is not rejected if only one class has output 1. For example, in the data set of Figure 4.1, three training vectors of Class 1 and two training vectors of Class 2 in the common area of two rejection boundaries are rejected by RADPN. In Figure 4.2, six training vectors are rejected by RAD.

4.2.4 Testing

The following describes the testing procedure of the proposed system:

Initialize: \( m = 1 \)

STEP 1. Input the test vector to \( SNN_m \).

STEP 2. Check whether the test vector is rejected by every rejection scheme.

1) If it is rejected by all rejection schemes, find \( N \) nearest reference boundaries and perform the steps A and B below for every rejection scheme by which all class outputs are 0's.

   A) If \( N \) nearest reference boundaries belong to one class, classify the input as belonging to that class.

   B) If \( N \) nearest reference boundaries come from more than one classes, do not classify.

   C) If A and B are done for all rejection schemes, go to STEP 3.

2) If it is rejected by all rejection schemes and there is no rejection scheme by which all class outputs are 0's, go to STEP 4.

3) If it is not rejected by at least one rejection scheme, classify the input as belonging to the class whose output is 1. Stop the testing procedure.
STEP 3. Count the number of the classes to which the input is classified.

1) If there is only one such class, classify the test vector to the class. Stop the testing procedure.

2) If more than one class are chosen, do not classify the test vector. Go to STEP 4.

STEP 4. Check whether or not the current SNN is the last.

1) if it is the last SNN, then classify the test vector to the class of the nearest reference vector. Stop the testing procedure.

2) if it is not, go to STEP 5.

STEP 5. Transform nonlinearly the input vector.

STEP 6. m = m + 1. Go to STEP 1.

STEP 2 in the test procedure can be executed in parallel or serially for all safe rejection schemes since every rejection scheme works independently. The output of an output node is set to 1 when the test vector is inside or on its rejection boundary. It has output 0 when the test vector is outside its rejection boundary. For RADPN, the test vector X(k) is judged to be inside or on the rejection boundary if it satisfies Equation (4.7) for every n = 1, 2, ..., I. Otherwise, X(k) is said to be outside the rejection boundary. X(k) exists inside or on the rejection boundary of RAD if Equation (4.8) is satisfied. For X(k) to be outside the rejection boundary of RAD, it should satisfy Equation (4.9).

If one or more of output nodes belonging to a class has output 1, the class output is set to 1. If none of the output nodes belonging to a class has output 1, the class output is set to 0. A test vector is not rejected by the rejection scheme if only one class has output 1. A test vector is rejected if more than one class have output 1 or no class has output 1.

Every training vector exists inside or on at least one rejection boundary, and thus at least one class has output 1. However, this is not necessarily true for the test vectors. Some interesting cases are shown in Figure 4.3. The test vectors that fall in the area P are
- 3 reference vectors of Class 1
- 3 reference vectors of Class 2

Figure 4.3 The distribution of test vectors.
outside all rejection boundaries and are rejected because all classes have output 0. These vectors are expected to belong to Class 2. It is reasonable for the test vectors in the areas Q and R to be classified to Class 1 and Class 2, respectively, because they are far from the border between Class 1 and Class 2. However, they are also rejected by the rejection scheme. It is logical to classify the vectors in the areas P, Q and R and to reduce the burden of the next SNN instead of just rejecting them.

One promising way is to check N nearest rejection boundaries and to classify to the class to which all of N nearest boundaries belong. Among the rejection boundaries of the rejection scheme by which no class has output 1, we find N nearest rejection boundaries. If there are several rejection schemes all of whose class outputs are 0, we check for every rejection scheme to find out whether N nearest rejection boundaries belong to one class. Only when the marked classes by all rejection schemes coincide, we classify the test vector to the class. If there is only one rejection scheme whose N nearest rejection boundaries belong to one class, the test vector is classified to the class by the rejection scheme. Usually, \(1 \leq N \leq L\) where L is the number of the reference vectors of each class. The greater N is, the harder it is for the test vector to be classified to a class. In the experiments stated in section 4, \(N = 3\) was used.

If all the test vectors are required to be classified, the last SNN involves classifying the rejected test vector to the class of the nearest reference vector.

4.3 Theoretical Analysis

In this section, we describe the theoretical study of safe rejection schemes and competitive learning with multiple learning rates.

4.3.1 Theoretical Analysis of Safe Rejection Schemes

Competitive learning algorithms without rejection schemes, such as Kohonen's
algorithms [KKL90] [Koh88] [Koh89] [Koh90], can be regarded as those with unsafe rejection schemes. In Figure 4.4, the line AB is the decision surface between Classes 1 and 2. It also behaves like the rejection boundaries of both Class 1 and Class 2. The left side of the line AB is the rejection boundary of Class 1 and the right side is the rejection boundary of Class 2. Since there is no common area of two rejection boundaries, no vectors are rejected and all the training vectors are classified. We can see that the three training vectors labelled a, b and c are misclassified and thus the line AB is an unsafe rejection boundary.

The basic idea of the safe rejection scheme is to reject the vectors in such a way that there are no misclassified training vectors. In the following theorems, we prove that the two rejection schemes described in the previous section are safe rejection schemes.

Theorem 1: RADPN is a safe rejection scheme.

Proof: Let us check for the training vectors that belong to the j-th class where j = 1, 2, ..., C, and C is the number of classes. When there is only one reference vector per class, the final RADPn1 and RADNn1 obtained after passing the training vectors to the RADPN rejection scheme corresponds to the maximum and the minimum values among the values of the n-th elements of the training vectors, respectively:

\[ \text{RADP}_{n1} = \max_k \{x_{nj}(k)\} \quad (4.10) \]
\[ \text{RADN}_{n1} = \min_k \{x_{nj}(k)\} \quad (4.11) \]

where \( x_{nj}(k) \) is the n-th element of the k-th training vector \( X_j(k) \) that belongs to the j-th class. Because the value of the n-th element of every training vector belonging to the j-th class falls between \( \min_k \{x_{nj}(k)\} \) and \( \max_k \{x_{nj}(k)\} \) for every n, the training vector is
Figure 4.4 The decision surface of competitive learning for 2-D data.
inside or on the rejection boundary formed by \( \text{RAD}_{pi} \) and \( \text{RAD}_{ni} \) \( (n = 1, 2, ..., i) \). Consequently, the output of class \( j \) is set to 1.

In the case of multiple reference vectors per class, the final \( \text{RAD}_{pi} \) and \( \text{RAD}_{ni} \) corresponds to the maximum and the minimum values among the values of the \( n \)-th elements of all training vectors for which the winner is the \( i \)-th reference vector of the \( j \)-th class. Every training vector belonging to the \( j \)-th class is inside or on at least one rejection boundary of the \( j \)-th class, and the output of class \( j \) is 1.

There are two possible cases concerning the outputs of the other classes. If none of the other classes have 1 as their outputs, the training vector will be classified correctly to class \( j \). If at least one of the other classes has output 1, the vector is rejected. Therefore, every training vector is rejected or has correct classification by \( \text{RAD}_{PN} \). If the number of SNN's are increased until no training vectors are rejected, then all the training vectors have to be correctly classified. 

(Q.E.D.)

Theorem 2: \( \text{RAD} \) is a safe rejection scheme.

Proof: Similar to Theorem 2, we check for the training vectors that belong to the \( j \)-th class where \( j = 1, 2, ..., C \). The final \( \text{RAD}_{pi} \) is the maximum distance in \( \text{RAD}_{P} \) from the \( i \)-th reference vector to the training vectors for which the winner is the \( i \)-th reference vector:

\[
\text{RAD}_{pi} = \max_k \left\{ \sqrt{\sum_{j=1}^{p} (x_{nj}(k) - w_{ni})^p} \right\} \tag{4.12}
\]

where \( x_{nj}(k) \) is the \( n \)-th element of the training vectors that belong to the \( j \)-th class and satisfies the following condition:
\[
\sqrt{\sum_{j=1}^{P} (x_{nj}(k) - w_{nj})^p} \leq \sqrt{\sum_{j=1}^{P} (x_{nj}(k) - w_{nh})^p} \quad \text{for } h = 1, \ldots, L \quad (4.13)
\]

Every training vector that belongs to the j-th class is inside or on at least the i-th rejection boundary of the j-th class, and the output of class j is set to 1. In this situation, every training vector that belongs to the j-th class is classified correctly to class j or rejected. Thus, RAD is a safe rejection scheme. Again, with a sufficient number of SNN's, all the training vectors are classified correctly. (Q.E.D.)

A training vector is classified correctly or rejected when we employ a safe rejection scheme. Therefore, a set of training vectors, T, can be divided into two disjoint subsets, a set of correctly classified vectors, C(X, T), and a set of rejected vectors, R(X, T):

\[
T = C(X, T) \cup R(X, T) \quad \text{and} \quad C(X, T) \cap R(X, T) = \emptyset \quad (4.14)
\]

The first argument of C(X, T) and R(X, T) represents the employed rejection scheme and the second one is the set of training vectors fed into the rejection scheme X.

Letting CN(X, T), RN(X, T) and TN represent the number of correctly classified vectors, the number of rejected training vectors and the total number of training vectors, respectively, we have the following equation:

\[
TN = CN(X, T) + RN(X, T) \quad (4.15)
\]

If a rejection scheme is unsafe, Equation (4.15) is modified to the following:

\[
TN = CN(X, T) + RN(X, T) + MN(X, T) \quad (4.16)
\]
where \( MN(X, T) \) is the number of misclassified vectors.

The rejection boundaries constructed by one rejection scheme is usually different from those constructed by another rejection scheme. Therefore, it is probable that some vectors rejected by a rejection scheme can be classified correctly by another rejection scheme. We can expect that the number of correctly classified training vectors is increased by using multiple safe rejection schemes.

Definition 3: Two rejection schemes \( X \) and \( Y \) are said to be "distinct" if the sets of correctly classified training vectors by \( X \) and \( Y \) may overlap but otherwise are different, i.e., \( C(X, T) \neq C(Y, T) \).

Definition 4: A rejection scheme \( Y \) is said to be a "subset rejection scheme" of another rejection scheme \( X \) if every training vector classified correctly by \( Y \) is also classified correctly by \( X \), i.e., \( C(X, T) \supseteq C(Y, T) \).

Figure 4.5 shows the relationship between \( C(X, T) \) and \( C(Y, T) \) when \( X \) and \( Y \) are distinct. Notice that the following is satisfied:

\[
CN(Y, R(X, T)) = CN(Y, T) - CN(X^*Y, T) > 0 \tag{4.17}
\]
\[
CN(X, R(Y, T)) = CN(X, T) - CN(X^*Y, T) > 0 \tag{4.18}
\]

where \( CN(X^*Y, TN) \) represents the number of the training vectors that is classified correctly by both \( X \) and \( Y \).

Figure 4.6 shows the relationship between \( C(X, T) \) and \( C(Y, T) \) when \( Y \) is a subset rejection scheme of \( X \). There are the following relationships:
Figure 4.5  The relationship between $C(X, T)$ and $C(Y, T)$ when $X$ and $Y$ are distinct.
Figure 4.6  The relationship between \( C(\mathbf{x}, T) \) and \( C(\mathbf{y}, T) \) when \( \mathbf{y} \) is a subset rejection scheme of \( \mathbf{x} \).
\[ \text{CN}(Y, R(X, T)) = 0 \]  \hspace{1cm} (4.19)  
\[ \text{CN}(X, R(Y, T)) = \text{CN}(X, T) - \text{CN}(Y, T) \geq 0 \]  \hspace{1cm} (4.20)

Theorem 3: Let \( X \) and \( Y \) be two different safe rejection schemes. The following relationship always holds true for \( X \) and \( Y \):

\[ \text{CN}(X + Y, T) \geq \max\{\text{CN}(X, T), \text{CN}(Y, T)\} \]  \hspace{1cm} (4.21)

where \( \text{CN}(X + Y, T) \) is the number of training vectors classified correctly by \( X \) or \( Y \) when both \( X \) and \( Y \) are used together.

Proof: When a rejection scheme \( X \) is used, the following relationship is satisfied:

\[ \text{TN} = \text{CN}(X, T) + \text{RN}(X, T) \]  \hspace{1cm} (4.22)

For another rejection scheme \( Y \), a similar equation is satisfied:

\[ \text{TN} = \text{CN}(Y, T) + \text{RN}(Y, T) \]  \hspace{1cm} (4.23)

If two rejection schemes \( X \) and \( Y \) are employed, the vectors rejected by one rejection scheme are fed into the next rejection scheme. If \( R(X, T) \), the set of vectors rejected by the rejection scheme \( X \), is fed into the rejection scheme \( Y \) again, the following equation is satisfied:

\[ \text{RN}(X, T) = \text{CN}(Y, R(X, T)) + \text{RN}(Y, R(X, T)) \]  \hspace{1cm} (4.24)
Substituting Equation (4.24) into (4.22), we obtain the following equation:

$$\text{TN} = \text{CN}(X, T) + \text{CN}(Y, R(X, T)) + \text{RN}(Y, R(X, T))$$  \hspace{1cm} (4.25)

The first two terms in the left side are the total number of correctly classified training vectors by the two rejection schemes X and Y:

$$\text{CN}(X + Y, T) = \text{CN}(X, T) + \text{CN}(Y, R(X, T))$$  \hspace{1cm} (4.26)

It is true from Equations (4.17) and (4.19) that

$$\text{CN}(Y, R(X, T)) \geq 0$$  \hspace{1cm} (4.27)

From Equations (4.26) and (4.27), the following inequality can be obtained:

$$\text{CN}(X + Y, \text{TN}) \geq \text{CN}(X, \text{TN})$$  \hspace{1cm} (4.28)

Similarly, when $R(Y, T)$ is fed into the rejection scheme X again, the following equations are satisfied:

$$\text{RN}(Y, T) = \text{CN}(X, R(Y, T)) + \text{RN}(X, R(Y, T))$$  \hspace{1cm} (4.29)

$$\text{TN} = \text{CN}(Y, T) + \text{CN}(X, R(Y, T)) + \text{RN}(X, R(Y, T))$$  \hspace{1cm} (4.30)

$$\text{CN}(X + Y, T) = \text{CN}(Y, T) + \text{CN}(X, R(Y, T))$$  \hspace{1cm} (4.31)

$$\text{CN}(X, R(Y, T)) \geq 0$$  \hspace{1cm} (4.32)

$$\text{CN}(X + Y, T) \geq \text{CN}(Y, T)$$  \hspace{1cm} (4.33)
From Equations (4.28) and (4.33), we infer the following result:

\[ CN(X + Y, T) \geq \max\{CN(X, T), CN(Y, T)\} \quad \text{(Q.E.D.)} \]

When a safe rejection scheme \( Y \) is a subset rejection scheme of another safe rejection scheme \( X \), Equation (4.21) becomes

\[ CN(X + Y, T) = \max\{CN(X, T), CN(Y, T)\} = CN(X, T) \quad (4.34) \]

When two safe rejection schemes \( X \) and \( Y \) are distinct, it is true from Equations (4.17) and (4.18) that

\[ CN(Y, R(X, T)) > 0 \quad (4.35) \]
\[ CN(X, R(Y, T)) > 0 \quad (4.36) \]

Thus

\[ CN(X + Y, T) > CN(X, T) \quad (4.37) \]
\[ CN(X + Y, T) > CN(Y, T) \quad (4.38) \]

In this case, Equation (4.21) becomes

\[ CN(X + Y, T) > \max\{CN(X, T), CN(Y, T)\} \quad (4.39) \]

Equation (4.39) shows that the classification accuracy of the neural network with two distinct safe rejection schemes is expected to be higher than the classification accuracy with
only one safe rejection scheme.

Two or more rejection schemes can be used in parallel rather than serially. In the case of serial use of X and Y, X can be used after Y or vice versa. Figures 4.7 and 4.8 show two rejection schemes X and Y used serially. Figure 4.9 shows the case of parallel use of two rejection schemes X and Y. During the training step, the ordering of X and Y is immaterial since there are no misclassified training vectors. However, during testing, the actual ordering of X and Y may affect the classification performance. In the case of parallel use of more than one rejection scheme, all the rejection schemes are used simultaneously, and each rejection scheme decides which input vectors to reject. During testing, if an input vector is accepted by one or more rejection schemes, but is classified to different classes by more than two rejection schemes, it is rejected.

We note that whether two rejection schemes are distinct depends on the data set. X and Y can be distinct for a data set, and X can be a subset rejection scheme of Y for another data set. In other words, there is no fixed relationship between the two rejection schemes. RADPN and RAD of a particular order are apt to be distinct because RADPN and RAD create different shapes of rejection boundaries as shown in Figures 4.1 and 4.2. On the other hand, RAD of a particular order, for example RAD^2, often becomes a subset rejection scheme of another order of RAD, for example RAD^5.

4.3.2 Theoretical Analysis of Competitive Learning with Multiple Learning Rates

In this section, we describe the theoretical analysis of competitive learning with multiple learning rates in terms of energy (cost). The associated energy function at the end of each iteration for competitive learning can be chosen as follows [RiSc88]:

\[
E(w_{kj}) = \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{L} M_{ik} |X_i - W_k|^2
\]

\[
= \frac{1}{2} \sum_{k=1}^{L} \sum_{i=1}^{N} M_{ik} \sum_{j} (x_{ij} - w_{kj})^2
\]
Figure 4.7  A rejection scheme $X$ is used before another rejection scheme $Y$. 
Figure 4.8 A rejection scheme $X$ is used after another rejection scheme $Y$. 
Figure 4.9 Parallel use of two rejection schemes X and Y.
where \( N \) is the number of samples; \( L \) is the number of reference vectors; \( X_i \) is an input vector; \( W_k \) is a reference vector; \( I \) is the dimension of input vectors; \( w_{kj} \) is the \( j \)-th element of \( W_k \); \( x_{ij} \) is the \( j \)-th element of \( X_i \). Here \( M_{ik} \) is the membership matrix which specifies whether or not input \( X_i \) activates node \( k \) as winner:

\[
M_{ik} = \begin{cases} 
1 & \text{if node } k \text{ wins when } X_i \text{ is input;} \\
0 & \text{Otherwise}
\end{cases}
\]

The change in energy due to the change in \( w_{kj} \) is

\[
\Delta E_{kj} = - \sum_i^N M_{ik} (x_{ij} - w_{kj}) \Delta w_{kj}
\]

The change in energy due to the change in \( W_k \) is the sum of \( \Delta E_{kj} \) over all elements of \( W_k \):

\[
\Delta E_k = - \sum_i^N M_{ik} \sum_j^I (x_{ij} - w_{kj}) \Delta w_{kj}
\]

Thus, the total energy change is the sum of \( \Delta E_k \) over reference vectors:

\[
\Delta E = - \sum_k^N \sum_i^L M_{ik} \sum_j^I (x_{ij} - w_{kj}) \Delta w_{kj}
\]

For conventional competitive learning whose learning rate is a scalar time function, the change in \( w_{kj} \) is given by Equation (2.18) as:

\[
\Delta w_{kj} = C(t) (x_{ij} - w_{kj})
\]

Substituting Equation (4.46) into Equations (4.44) and (4.45), we obtain the following equations:
\[ \Delta E_k = - \sum_{i=1}^{N} M_{ik} \sum_{j=1}^{I} C(t) (x_{ij} - w_{kj})^2 \]  
(4.47)

\[ \Delta E = - \sum_{k=1}^{L} \sum_{i=1}^{N} M_{ik} \sum_{j=1}^{I} C(t) (x_{ij} - w_{kj})^2 \]  
(4.48)

The learning rate coefficient \( C(t) \) is associated with \( X_i \). \( C(t) \) is dependent on the input presentation order of \( X_1, X_2, ..., X_N \). For example, if \( X_1 \) is passed to the neural network before \( X_2 \), \( C \) associated with \( X_1 \) is bigger than \( C \) associated with \( X_2 \). Thus, \( \Delta E_k \) depends on where the input vectors activating node \( k \) as winner are placed in the input sequence. If the input vectors contributing to the change of \( W_k \) are placed in the last part of the sequence, the associated \( C \)'s are too small and the corresponding \( \Delta E_k \) is also small. In this case, the system needs many iterations of the input sequence in order to reach a local minimum. There are typically many local minima, corresponding to different clusterings of the input data. Once the system with small \( \Delta E_k \) becomes stuck in a local minimum, it cannot get out of it and move towards progressively lower minima, and thus \( W_k \) resides far from the centroid of the input vectors activating node \( k \) as winner.

For conventional competitive learning with multiple learning rates, the change in \( w_{kj} \) is given by Equation (4.3) as:

\[ \Delta w_{kj} = C_k(t) (x_{ij} - w_{kj}) \]  
(4.49)

Substituting Equation (4.49) into Equations (4.44) and (4.45), we obtain the following:

\[ \Delta E_k = - \sum_{i=1}^{N} M_{ik} \sum_{j=1}^{I} C_k(t) (x_{ij} - w_{kj})^2 \]  
(4.50)

\[ \Delta E = - \sum_{k=1}^{L} \sum_{i=1}^{N} M_{ik} \sum_{j=1}^{I} C_k(t) (x_{ij} - w_{kj})^2 \]  
(4.51)

Each reference vector \( W_k \) has its own learning rate coefficient \( C_k(t) \). The coefficient \( C_k(t) \)
is changed (decreased) only when its corresponding reference vector $W_k$ wins the competition. Thus, $\Delta E_k$ is independent of where the input vectors activating node $k$ as winner are placed in the input sequence. Every reference vector are learned as if it is the reference vector of the system with only one output node. Since for the same input presentation and $W_k$ the change in energy $\Delta E_k$ is bigger than $\Delta E_k$ of the system with a single scalar learning rate, the system with multiple learning rates converges to a stable equilibrium more rapidly than the system with a single scalar learning rate. Furthermore, the system with multiple learning rates may be able kick it out of the higher local minima and towards progressively lower minima more possibly than the system with a single scalar learning rate.

4.4. Experimental Results

Experimental results are presented in order to test the theoretical results discussed in the previous sections. In Section 4.4.1, Gaussian-distributed data sets are used. Competitive learning with multiple learning rates is compared with competitive learning with a single scalar learning rate. In Sections 4.4.2 and 4.4.3, two sets of remote sensing data are used in the experiments and the classification performance of the CSNN is compared with those of the backpropagation network and PSHNN trained by the delta rule.

4.4.1 Experiments with the Gaussian-Distributed Data Set

In this section, the experimental results obtained with competitive learning with multiple learning rates in comparison to competitive learning with a single scalar learning rate are described.

Figure 4.10 shows a 2-dimensional Gaussian-distributed data set (GD1) which consists of 1200 samples with variance 40 and with centroids at $(10, 10)$, $(-10, 10)$, $(-10, -10)$, $(10, -10)$. There are four classes and each class has 300 samples. This data set was
Figure 4.10 The distribution of a Gaussian-distributed data set with the same number of samples per class.
used for the training of neural networks. The testing data set is 2400 samples with the same variance and centroids as the training data set.

For both competitive learning methods, the same parameters were used except the learning rates. The number of neurons in the input layer was 2. Only one reference vector was defined per each class. The initialization of the various reference vectors were carried out by choosing the first training sample belonging to each class. In the case of a single scalar learning rate, the learning rate coefficient \( C(t) \) was chosen as \( \frac{4}{t + \frac{3}{2}} \) where \( t = 1, 2, 3, \ldots \). In the case of multiple learning rates, the number of learning rates was 4 and the learning rate coefficient \( C_k(t) \) was chosen as \( \frac{1}{t} \) for every reference vector where \( t = 1, 2, 3, \ldots \). In order to have a fair comparison, \( C(t) \) needs to be decreased more slowly than \( C_k(t) \) since \( C(t) \) is associated with 1200 samples and \( C_k(t) \) is associated with around 300 (= 1200/4) samples.

Figure 4.11 shows the total energy values of competitive learning with multiple learning rates and competitive learning with a single scalar learning rate as a function of the number of iterations. MLR and SLR denote competitive learning with multiple learning rates and competitive learning with a single scalar learning rate, respectively. Both methods converge to the same local minimum. However, competitive learning with multiple learning rates converges to a stable equilibrium more rapidly than competitive learning with a single scalar learning rate.

Figure 4.12 shows the classification accuracy with the two methods as a function of the number of iterations. We observe that competitive learning method with multiple learning rates produces higher classification accuracy than competitive learning with a single scalar learning rate before the two methods converge to the same stable equilibrium. After the convergence, the classification accuracy of competitive learning with a single scalar learning rate is almost the same as that of competitive learning with multiple learning rates.
Figure 4.11 Training time versus energy with the Gaussian-distributed data set.
Figure 4.12  The classification accuracy of the two methods with the Gaussian-distributed data set as a function of the number of iterations.
4.4.2 Experiments with the Flight Line C1 (FLC1) Data Set

The data set for the second experiment is based on Multispectral Earth Observational remote sensing data called Flight Line C1 (FLC1). The geographic location of the FLC1 is the southern part of Tippecanoe County, Indiana. This multispectral image was collected with an airborne scanner in June 1966 at noon time. Figure 4.13 shows the reference ground map of the FLC1 when the spectral band is 0.80 - 1.00 μm.

4.4.2.1 Data Set

The FLC1 consists of 12 band signals. Table 4.1 shows the spectral bands of the data set. Each point of one spectral image represents one of 256 gray levels. Table 4.2 shows the 8 dominant classes of the data.

In this experiment, only 8 spectral bands out of 12 bands are used. 8 dominant farm products are chosen to represent 8 particular classes (alfalfa, corn, oats, red clover, soybean, wheat, bare soil, and rye).

Each data vector is generated by concatenating 8 corresponding band signal values together. Training and testing with the networks are done with 200 signal samples per class and 375 other signal samples per class, respectively. So the total numbers of the training and the testing samples are 1600 and 3000, respectively.

4.4.2.2 Simulation

In this experiment, the rejected vectors of a SNN were nonlinearly transformed with the shuffling and complementing method [ErHo90] [Hon90] and fed into the next SNN. This method was previously used with binary numbers. In this article, the shuffling and complementing method was used by converting each decimal number to the binary format, shuffling and complementing the binary number, and reconverting it back to the decimal format. RADPN and $\text{RAD}^p (p = 1, 2, ..., 6)$ were used in parallel as rejection
Figure 4.13  The reference ground map of the FLC1 (when the spectral band is 0.80 - 1.00μm).
Table 4.1 Spectral bands of the FLC1 multispectral image data.

<table>
<thead>
<tr>
<th>Band</th>
<th>Wave length(μm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.04 - 0.44</td>
</tr>
<tr>
<td>2</td>
<td>0.44 - 0.46</td>
</tr>
<tr>
<td>3</td>
<td>0.46 - 0.48</td>
</tr>
<tr>
<td>4</td>
<td>0.48 - 0.50</td>
</tr>
<tr>
<td>5</td>
<td>0.50 - 0.52</td>
</tr>
<tr>
<td>6</td>
<td>0.52 - 0.55</td>
</tr>
<tr>
<td>7</td>
<td>0.55 - 0.58</td>
</tr>
<tr>
<td>8</td>
<td>0.58 - 0.62</td>
</tr>
<tr>
<td>9</td>
<td>0.62 - 0.66</td>
</tr>
<tr>
<td>10</td>
<td>0.66 - 0.72</td>
</tr>
<tr>
<td>11</td>
<td>0.72 - 0.80</td>
</tr>
<tr>
<td>12</td>
<td>0.80 - 1.00</td>
</tr>
</tbody>
</table>
Table 4.2 8 dominant classes of the FLC1 multispectral image data.

<table>
<thead>
<tr>
<th>Class</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alfalfa</td>
</tr>
<tr>
<td>2</td>
<td>Corn</td>
</tr>
<tr>
<td>3</td>
<td>Oats</td>
</tr>
<tr>
<td>4</td>
<td>Red Clover</td>
</tr>
<tr>
<td>5</td>
<td>Soybean</td>
</tr>
<tr>
<td>6</td>
<td>Wheat</td>
</tr>
<tr>
<td>7</td>
<td>Bare Soil</td>
</tr>
<tr>
<td>8</td>
<td>Rye</td>
</tr>
</tbody>
</table>
schemes. During testing, if an input input vector was accepted by one or more rejection schemes, but was classified to different classes by more than two rejection schemes, it was rejected.

Since the continuous number representation was used for the input vectors rather than binary representation, the number of neurons in the input layer was eight.

The reference vectors were computed separately for each class with the data samples belonging to the class. The reason for this is explained in Section 4.4.2.4. Competitive learning with multiple learning rates was used for learning the reference vectors. The reason for this is explained in Section 4.4.2.5. Ten reference vectors were defined for each class. The initialization of the various reference vectors were carried out by choosing the first ten training samples belonging to each class. Each of the multiple learning rates $C_i(t)$ was chosen as $\frac{9}{t+8}$ for all the classes.

We checked for every rejection scheme yielding no class output with 1 whether the classes of the three nearest rejection boundaries coincide. If so, we classified the input to that class.

The training accuracy was 100 % and the test accuracy was 90.53 % when the allowed number of iterations was 10. Only three SNN's were needed. In the last SNN, 0 training vectors and 30 test vectors were rejected. Those vectors were classified to the class of the nearest reference vector.

When the rejected vectors in the last SNN were not classified, the training accuracy and the testing accuracy were 100 % and 91.18 %, respectively.

The program was executed on a Gould/UNIX machine in about 3 minutes and 55 seconds in cpu time with 10 iterations and 3 SNN's. If the reference vectors are computed in parallel for each class, the training time is reduced by a factor approximately equal to 8.
4.4.2.3 Comparison to Other Neural Networks

In the case of the 4 layer backpropagation neural network (4NN), the length of the input layer and the hidden layers were fixed as 64, and the length of the output layer was 8. The initial weights were randomly chosen between -0.5 and 0.5. The best training accuracy was 98.93 % when the number of iterations was 1500, and the best testing accuracy was 89.50 % when the number of iterations was 800 (see Table 4.3).

In the case of the PSHNN with the delta rule learning algorithm, the best training accuracy was 98.06 % when the number of iterations was 400, and the best testing accuracy was 89.30 % when the number of iterations was 100 (see Table 4.4). The RDFT-based nonlinear transformation method [ErHo90] [Hon90] was used. Gray code representation was used and thus 64 nodes were required in the input layer. The number of neurons in the output layer was 8. The initial weights were randomly chosen in the range -0.5 to 0.5. The gain parameter was chosen as 0.05. Sigmoid function, \( \frac{1}{1 + \exp(-x)} \), was used for the nonlinear activation function. The program for the PSHNN with the delta rule run on the GOULD / UNIX machine required over 31 minutes in CPU time for training with 100 iterations.

We can see that the PSHNN with competitive learning and safe rejection schemes (CSNN) has higher classification accuracy than the backpropagation networks and the PSHNN with the delta rule. The learning speed of the CSNN is improved by a factor approximately equal to 63 (\( \approx 7.9 \times 8 \)), in comparison to the PSHNN with the delta rule when the reference vectors are computed in parallel for each class. Ersoy and Hong [ErHo90] [Hon90] estimated the learning speed of the PSHNN and the backpropagation networks. The 4 NN requires about 25 times longer training time than the PSHNN. Thus, the training time of the CSNN is about 1575 (\( = 63 \times 25 \)) times shorter than the time for the backpropagation network.
In the case of traditional competitive learning without rejection schemes characterized by Equations (2.18) to (2.20), the training and testing accuracies were 83.19 % and 78.53 %, respectively, when the number of iterations was 100. Ten reference vectors were defined for each class. The learning rate coefficient C(t) was chosen as $\frac{9}{t+8}$.

In the case of the Kohonen's learning vector quantization 1 (LVQ1) without rejection schemes characterized by Equations (2.24) to (2.26), the training and the testing accuracies were 90.19 % and 87.40 %, respectively, when the number of iterations was 100. Ten reference vectors were defined for each class. The learning rate coefficient C(t) was chosen as $\frac{9}{t+8}$.

Table 4.5 is the summary of the experimental results with the FLCL data set. CSNN (classified) denotes the case for which the rejected vectors in the last SNN were classified. CSNN (unclassified) denotes the case for which the rejected vectors in the last SNN were not classified.
Table 4.3  The classification accuracy of the backpropagation network (4NN) as a function of the number of iterations in the 8-class problem [ErHo90] [Hon90].

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train (%)</td>
</tr>
<tr>
<td>50</td>
<td>62.75</td>
</tr>
<tr>
<td>100</td>
<td>86.44</td>
</tr>
<tr>
<td>200</td>
<td>90.88</td>
</tr>
<tr>
<td>300</td>
<td>92.38</td>
</tr>
<tr>
<td>400</td>
<td>93.31</td>
</tr>
<tr>
<td>500</td>
<td>94.00</td>
</tr>
<tr>
<td>600</td>
<td>95.00</td>
</tr>
<tr>
<td>700</td>
<td>96.00</td>
</tr>
<tr>
<td>800</td>
<td>97.00</td>
</tr>
<tr>
<td>900</td>
<td>97.93</td>
</tr>
<tr>
<td>1000</td>
<td>98.31</td>
</tr>
<tr>
<td>1500</td>
<td>98.93</td>
</tr>
</tbody>
</table>
The classification accuracy and the total number of SNN's of the PSHNN with the delta rule as a function of the number of iterations in the 8-class problem [ErHo90] [Hon90].

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of SNN's</th>
<th>Classification Train (%)</th>
<th>Test (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>94.19</td>
<td>85.06</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>95.25</td>
<td>85.36</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>96.25</td>
<td>86.46</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>96.81</td>
<td>86.67</td>
</tr>
<tr>
<td>40</td>
<td>3</td>
<td>97.87</td>
<td>87.33</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
<td>97.25</td>
<td>88.01</td>
</tr>
<tr>
<td>60</td>
<td>3</td>
<td>97.37</td>
<td>88.47</td>
</tr>
<tr>
<td>70</td>
<td>3</td>
<td>97.38</td>
<td>88.87</td>
</tr>
<tr>
<td>80</td>
<td>3</td>
<td>97.38</td>
<td>88.90</td>
</tr>
<tr>
<td>90</td>
<td>3</td>
<td>97.44</td>
<td>89.13</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>97.75</td>
<td>89.30</td>
</tr>
<tr>
<td>200</td>
<td>3</td>
<td>97.63</td>
<td>88.67</td>
</tr>
<tr>
<td>300</td>
<td>3</td>
<td>97.87</td>
<td>88.60</td>
</tr>
<tr>
<td>400</td>
<td>3</td>
<td>98.06</td>
<td>88.06</td>
</tr>
<tr>
<td>500</td>
<td>3</td>
<td>98.00</td>
<td>88.40</td>
</tr>
<tr>
<td>600</td>
<td>3</td>
<td>98.06</td>
<td>87.90</td>
</tr>
</tbody>
</table>
Table 4.5 The experimental results with the FLC1 multispectral image data.

<table>
<thead>
<tr>
<th>Name of Neural Network</th>
<th>Training Data</th>
<th>Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iterations</td>
<td>Accuracy(%)</td>
</tr>
<tr>
<td>CSNN (classified)</td>
<td>10</td>
<td>100.00</td>
</tr>
<tr>
<td>CSNN (unclassified)</td>
<td>10</td>
<td>100.00</td>
</tr>
<tr>
<td>Backpropagation</td>
<td>1500</td>
<td>98.93</td>
</tr>
<tr>
<td>PSHNN with the delta rule</td>
<td>400</td>
<td>98.06</td>
</tr>
<tr>
<td>Competitive Learning</td>
<td>100</td>
<td>83.19</td>
</tr>
<tr>
<td>LVQ 1</td>
<td>100</td>
<td>90.19</td>
</tr>
</tbody>
</table>
4.4.2.4 Comparison of Method I and Method II for Learning Reference Vectors

In this section, we compare the classification accuracies of Method I and Method II for the computation of the reference vectors. In Method I, all reference vectors are computed together using the whole training data set. In Method II, the reference vectors of each class are computed with the training samples belonging to that class, independently of the reference vectors of the other classes.

Table 4.6 shows the experimental results with the FLCl multispectral image data. For every case, the same parameters were used except the learning rate. The number of iterations was 10. The number of neurons in the input layer was eight. Ten reference vectors were defined for each class. The initialization of the various reference vectors were carried out by choosing the first ten training samples belonging to each class. The rejected vectors of a SNN were nonlinearly transformed with the shuffling and complementing method [ErHo90] and fed into the next SNN. RADPN and RAD$_p$ ($p = 1, 2, ..., 6$) were used in parallel as rejection schemes. During testing, if an input input vector was accepted by one or more rejection schemes, but was classified to different classes by more than two rejection schemes, it was rejected. We checked for every rejection scheme by which no class has output 1 whether or not the classes of the three nearest rejection boundaries coincide. The rejected vectors in the last SNN were classified to the class of the nearest reference vector.

In the first algorithm of Method I (Conventional), the reference vectors were computed together for all the classes using the competitive learning algorithm characterized by Equations (2.18) to (2.20). The second algorithm of Method I was chosen as Kohonen's LVQ 1 algorithm [Koh88] [Koh90], which is used with the whole training data set for the computation of reference vectors. In these two cases, competitive learning with a single learning rate was used for learning the reference vectors and the learning rate coefficient C(t) was chosen as $\frac{14}{t + 13}$. 
In Method II, competitive learning with multiple learning rates was used for learning the reference vectors. The reference vectors were computed separately for each class using Equations (4.1) and (4.2). The scalar learning rate coefficient $C_i(t)$ was chosen as $\frac{14}{t+15}$ for each class.

We observe in Table 4.6 that Method II gives better classification accuracy and needs less number of SNN's than those of Method I. One reason for this is that Method II constructs a smaller common area bounded by the rejection boundaries, and thus the number of rejected input vectors is less than Method I. Figure 4.14 shows a simple case. There are two classes, and only one reference vector is computed for each class. The reference vector of Class 1 approaches the centroid of the samples belonging to Class 1 when Method II is used. On the other hand, when the reference vectors are computed by Method I, the reference vector of Class 1 resides in the left side of the real centroid because it approaches to the centroid of the samples, which are the combination of most Class 1 samples and a part of Class 2 samples, in the left side of the line AB. Similarly, the reference vector of Class 2 computed by Method II exists in the centroid of the samples belonging to Class 2 whereas the reference vectors computed by Method I is to the right of the centroid. Thus, the area of rejection determined by the two rejection boundaries constructed by Method I is bigger than the area of rejection determined by Method II.

Table 4.7 shows the number of rejected data at each SNN when the allowed number of iterations is 10. We observe in Table 4.7 that the number of rejected data in Method II is less than the number of rejected data in Method I.
Table 4.6  The classification accuracy and the total number of SNN's of the CSNN when the reference vectors are computed for each class independently versus all classes considered together.

<table>
<thead>
<tr>
<th>Name of Version</th>
<th>Number of Iterations</th>
<th>Classification No of SNN's</th>
<th>Accuracy Test (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method I (Conventional)</td>
<td>10</td>
<td>4</td>
<td>84.93</td>
</tr>
<tr>
<td>Method I (LVQ 1)</td>
<td>10</td>
<td>4</td>
<td>84.33</td>
</tr>
<tr>
<td>Method II</td>
<td>10</td>
<td>3</td>
<td>89.70</td>
</tr>
</tbody>
</table>
Figure 4.14  The rejection boundaries constructed by Method I and Method II.
Table 4.7 The number of rejected data of the Method I and Method II at each SNN when the allowed number of iterations is 10.

<table>
<thead>
<tr>
<th>Name of Version</th>
<th>SNN 0</th>
<th>SNN 1</th>
<th>SNN 2</th>
<th>SNN 3</th>
<th>SNN 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method I (Conventional)</td>
<td>1600</td>
<td>708</td>
<td>264</td>
<td>57</td>
<td>0</td>
</tr>
<tr>
<td>Method I (LVQ 1)</td>
<td>1600</td>
<td>712</td>
<td>237</td>
<td>62</td>
<td>0</td>
</tr>
<tr>
<td>Method II</td>
<td>1600</td>
<td>324</td>
<td>24</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
4.4.2.5 Comparison of a Single Scalar Learning Rate (SLR) and Multiple Learning Rates (MLR)

In this section, we compare the classification accuracies of competitive learning with a single scalar learning rate and competitive learning with multiple learning rates when the two methods are used in Method II.

For both cases, the same parameters were used except the learning rate. The reference vectors were computed separately for each class. The number of iterations was 10. The number of neurons in the input layer was eight. Ten reference vectors were defined for each class. The initialization of the various reference vectors were carried out by choosing the first ten training samples belonging to each class. The rejected vectors of a SNN were nonlinearity transformed with the shuffling and complementing method [ErHo90] and fed into the next SNN. RADPN and RADP ($p = 1, 2, \ldots, 6$) were used in parallel as rejection schemes. During testing, if an input vector was accepted by one or more rejection schemes, but was classified to different classes by more than two rejection schemes, it was rejected.

We checked for every rejection scheme by which no class has output 1 whether or not the classes of the three nearest rejection boundaries coincide. The rejected vectors in the last SNN were classified to the class of the nearest reference vector.

In the case of scalar learning rate, the reference vectors were computed for each class using Equations (4.1) to (4.2). The scalar learning rate coefficient $C_i(t)$ was chosen as $\frac{14}{t+13}$ for each class.

In the case of multiple learning rates, the reference vectors were computed for each class using Equations (4.5) to (4.6). Each element of the vector learning rates $C_i(t)$ was chosen as $\frac{9}{t+8}$ for every reference vector of all the classes. In order to have a fair comparison, $C_i(t)$ needs to be decreased much more slowly than $C_i(t)$ since $C_i(t)$ is associated with more samples than $C_i(t)$. 

Figure 4.15 shows the total energy values of competitive learning with multiple learning rates and competitive learning with a single scalar learning rate as a function of the number of iterations when the FLC1 data was used. MLR and SLR denote competitive learning with multiple learning rates and competitive learning with a single scalar learning rate, respectively. We observe in Figure 4.15 that competitive learning with multiple learning rates converges to a deeper local minimum.

Table 4.8 shows the classification accuracy with the two methods. CSNN with MLR and CSNN with SLR denote the CSNN with multiple learning rates and the CSNN with a single scalar learning rate, respectively. It is observed in Table 4.8 that competitive learning with multiple learning rates produces higher classification accuracy than competitive learning with a single scalar learning rate. The higher classification accuracy can be attributed to convergence to a deeper equilibrium.
Figure 4.15 Training time versus energy with two systems.
Table 4.8 The classification performances of a single scalar learning rate and multiple learning rates.

<table>
<thead>
<tr>
<th>Name of Neural Network</th>
<th>Training Data</th>
<th>Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iterations</td>
<td>Accuracy(%)</td>
</tr>
<tr>
<td>Method II with MLR</td>
<td>10</td>
<td>100.00</td>
</tr>
<tr>
<td>Method II with SLR</td>
<td>10</td>
<td>100.00</td>
</tr>
</tbody>
</table>

SLR: scalar learning rate
MLR: multiple learning rates
4.4.3 Experiments with the Colorado Data Set

The data set used in this section is a multispectral earth observation remotely sensed data covering a mountainous area in Colorado [BeSe89].

4.4.3.1 Data Set [BeSe89]

The Colorado data set consists of 7 data channels obtained from the following 4 data sources:

1) Landsat MSS data (4 data channels)
2) Elevation data (in 10 m contour intervals, 1 data channel)
3) Slope data (0-90 degrees in degree increments, 1 data channel)
4) Aspect data (1-180 degrees in 1 degree increments, 1 data channel)

The area used for classification is a mountainous area in Colorado. It has 10 ground cover classes which are listed in Table 4.9. Each channel comprises an image of 135 rows and 131 columns, all of which are co-registered.

Ground reference data were compiled for the area by comparing a cartographic map to a color composite of the Landsat data and also to a line printer output of each Landsat channel [BeSe89]. By this method, 2019 ground reference points (11.4 % of the area) were selected. Ground reference consists of two or more homogeneous fields in the imagery for each class. For each class, the largest field was selected as a training field. The other fields were used for testing. Overall, 1188 pixels were used for training and 831 pixels for testing the classifiers. The number of the samples from each class are shown in Table 4.10.

4.4.3.2 Simulation

In this experiment, the shuffling and complementing method was used for nonlinear
transformation of rejected vectors, as described in Section 4.4.1.2. RADPN and RADP \( p = 1, 2, \ldots, 6 \) were used in parallel as rejection schemes. During testing, if an input vector was accepted by one or more rejection schemes, but was classified to different classes by more than two rejection schemes, it was rejected.

Since the continuous number representation was used for the input vectors, the number of neurons in the input layer was seven.

Competitive learning with multiple learning rates was used for learning the reference vectors. The reference vectors were computed separately for each class with the data samples belonging to the class. Ten reference vectors were defined for each class. The initialization of the various reference vectors were carried out by choosing the first ten training samples belonging to each class. Each of the multiple learning rates \( C_i(t) \) was chosen as \( \frac{3}{i+2} \) for every reference vector of all the classes.

We checked for every rejection scheme yielding no class with output 1 whether or not the class of the three nearest rejection boundaries coincide. If so, we classified the input to that class.

When the reference vectors were computed with only a single pass (no iteration) of the training vectors, the training accuracy was 98.14 % and the testing accuracy was 57.94 %. In the last SNN, 54 training vectors and 23 test vectors were rejected. Those vectors were classified to the class of the nearest reference vector.

If the rejected vectors in the last SNN were not classified, the training accuracy and the testing accuracy were 100 % and 58.17 %, respectively.

4.4.3.3 Comparison to Other Neural Networks

In the case of the 4 layer backpropagation neural network (4NN), the best training accuracy was 94.11% when the number of iterations was 2500 (see Table 4.11 [ErHo90] [Hon90]). The best testing accuracy was 54.27 % when the number of iterations was
2000 (see Table 4.11 [ErHo90] [Hon90]). The length of the input layer and the hidden layers were fixed as 56, and the length of the output layer was 10. The initial weights were randomly chosen between -0.5 and 0.5.

In the case of the PSHNN using the delta rule learning algorithm, the best training accuracy was 95.79 % when the number of iterations was 20 (see Table 4.12 [ErHo90] [Hon90]). The best testing accuracy 57.03 % was obtained with 130 iterations (see Table 4.12 [ErHo90] [Hon90]). The shuffling and complementing method was used for the nonlinear transformations. Gray code representation was used, and thus the length of the input layer was 56. The number of neurons in the output layer was 10. The initial weights were randomly chosen in the range -0.5 to 0.5. The gain parameter was chosen as 0.05. For the activation function, sigmoid function was used.

The experimental results with the Colorado data set are summarized in Table 4.13. CSNN (classified) denotes the case for which the rejected vectors in the last SNN were classified. CSNN (unclassified) denotes the case for which the rejected vectors in the last SNN were not classified. We observe that the proposed system produces higher classification accuracy and more speed than the backpropagation network and the PSHNN with the delta rule.
Table 4.9 10 dominant classes of the multispectral image data of Colorado area.

<table>
<thead>
<tr>
<th>Class</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Water</td>
</tr>
<tr>
<td>2</td>
<td>Colorado blue spruce</td>
</tr>
<tr>
<td>3</td>
<td>Montane/Subalpine meadow</td>
</tr>
<tr>
<td>4</td>
<td>Aspen</td>
</tr>
<tr>
<td>5</td>
<td>Ponderosa pine</td>
</tr>
<tr>
<td>6</td>
<td>Ponderosa pine/Douglas fir</td>
</tr>
<tr>
<td>7</td>
<td>Engelmann spruce</td>
</tr>
<tr>
<td>8</td>
<td>Douglas fir/White fir</td>
</tr>
<tr>
<td>9</td>
<td>Douglas fir/Ponderosa pine/Aspen</td>
</tr>
<tr>
<td>10</td>
<td>Douglas fir/White fir/Aspen</td>
</tr>
</tbody>
</table>
Table 4.10 The number of samples of each class of the multispectral image data of Colorado area.

<table>
<thead>
<tr>
<th>Class</th>
<th>Train (1188)</th>
<th>Test (831)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>408</td>
<td>195</td>
</tr>
<tr>
<td>2</td>
<td>88</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>45</td>
<td>42</td>
</tr>
<tr>
<td>4</td>
<td>75</td>
<td>65</td>
</tr>
<tr>
<td>5</td>
<td>105</td>
<td>139</td>
</tr>
<tr>
<td>6</td>
<td>126</td>
<td>188</td>
</tr>
<tr>
<td>7</td>
<td>224</td>
<td>70</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>44</td>
</tr>
<tr>
<td>9</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>10</td>
<td>60</td>
<td>39</td>
</tr>
</tbody>
</table>
Table 4.11 The classification accuracy of the backpropagation network (4NN) as a function of the number of iterations in the 10-class problem.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train (%)</td>
</tr>
<tr>
<td>50</td>
<td>61.86</td>
</tr>
<tr>
<td>100</td>
<td>71.04</td>
</tr>
<tr>
<td>200</td>
<td>81.06</td>
</tr>
<tr>
<td>300</td>
<td>85.19</td>
</tr>
<tr>
<td>400</td>
<td>88.04</td>
</tr>
<tr>
<td>500</td>
<td>89.06</td>
</tr>
<tr>
<td>600</td>
<td>90.31</td>
</tr>
<tr>
<td>700</td>
<td>90.91</td>
</tr>
<tr>
<td>800</td>
<td>91.84</td>
</tr>
<tr>
<td>900</td>
<td>92.25</td>
</tr>
<tr>
<td>1000</td>
<td>92.42</td>
</tr>
<tr>
<td>1500</td>
<td>93.68</td>
</tr>
<tr>
<td>2000</td>
<td>93.86</td>
</tr>
<tr>
<td>2500</td>
<td>94.11</td>
</tr>
</tbody>
</table>
Table 4.12  The classification accuracy and the total number of SNN's of the PSHNN with the delta rule as a function of the number of iterations in the 10-class problem.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of SNN's</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Train (%)</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>93.28</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>95.37</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>95.79</td>
</tr>
<tr>
<td>30</td>
<td>5</td>
<td>95.53</td>
</tr>
<tr>
<td>40</td>
<td>5</td>
<td>95.49</td>
</tr>
<tr>
<td>50</td>
<td>4</td>
<td>95.71</td>
</tr>
<tr>
<td>60</td>
<td>4</td>
<td>95.20</td>
</tr>
<tr>
<td>70</td>
<td>5</td>
<td>95.20</td>
</tr>
<tr>
<td>80</td>
<td>5</td>
<td>94.45</td>
</tr>
<tr>
<td>90</td>
<td>4</td>
<td>94.45</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>94.45</td>
</tr>
<tr>
<td>130</td>
<td>4</td>
<td>94.45</td>
</tr>
<tr>
<td>200</td>
<td>4</td>
<td>95.20</td>
</tr>
</tbody>
</table>
Table 4.13 The experimental results with the Colorado data set.

<table>
<thead>
<tr>
<th>Name of Neural Network System</th>
<th>Training Data</th>
<th>Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iterations</td>
<td>Accuracy(%)</td>
</tr>
<tr>
<td>CSNN (classified)</td>
<td>1</td>
<td>98.14</td>
</tr>
<tr>
<td>CSNN (unclassified)</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>Backpropagation</td>
<td>2500</td>
<td>94.11</td>
</tr>
<tr>
<td>PSHNN with the delta rule</td>
<td>20</td>
<td>95.79</td>
</tr>
</tbody>
</table>
4.5 Conclusions

We presented parallel, self-organizing, hierarchical neural networks with competitive learning and safe rejection schemes. The purpose of safe rejection schemes is to reject the input vectors in such a way that there are no misclassified training vectors.

Whereas in traditional competitive learning, the reference vectors themselves are used for the classification by the nearest neighbor principle, the decision surface of classification is additionally and to a large degree determined by the rejection boundaries in the CSNN. Thus, the classification performance of the CSNN depends to a large degree on the rejection schemes used. RADPN and RAD were described as two safe rejection schemes.

When multiple safe rejection schemes are used, the classification accuracy is higher than when only one safe rejection scheme is used. The experimental results indicated the superiority of the CSNN in comparison to the backpropagation network and the PSHNN with the delta rule learning algorithm, both in terms of classification accuracy and speed of processing.

When the reference vectors were computed separately for each class (Method II), higher classification accuracy was obtained and less number of SNN's were needed in comparison to Method I in which the reference vectors are computed together for all the classes. In addition, Method II can be performed in parallel, resulting in much faster learning.

Competitive learning with multiple learning rates converged to a stable equilibrium more rapidly, and produced higher classification accuracy than competitive learning with a single scalar learning rate when Method II was used.
CHAPTER 5
PARALLEL, SELF-ORGANIZING, HIERARCHICAL NEURAL NETWORKS WITH FUZZY INPUT SIGNAL REPRESENTATION, COMPETITIVE LEARNING AND SAFE REJECTION SCHEMES

5.1 Introduction

The parallel self-organizing, hierarchical neural networks with competitive learning and safe rejection schemes (CSNN) presented in the previous chapter can handle only precise numerical data, as is the case with the backpropagation network and the PSHNN with the delta rule. In this chapter, we describe parallel self-organizing, hierarchical neural networks with fuzzy input signal representation, competitive learning and safe rejection schemes (FCSNN), not only for improving the classification accuracy but also for being able to classify objects whose attribute values do not have clear boundaries.

The CSNN with fuzzy input signal representation is referred to as the FCSNN. The fuzzy input signal representation scheme is included in the CSNN as a preprocessing module. It transforms imprecise input in linguistic form and precisely stated numerical input into multidimensional numerical values. The transformed input is processed in the postprocessing module, the CSNN, of the FCSNN.

This chapter is composed of 5 sections. In section 5.2, we address the automatic derivation of the membership functions for fuzzy sets based on the distribution of a set of samples. In section 5.3, the original input is transformed into multidimensional numerical values using the derived fuzzy sets and the computational scheme of the degree of match. In section 5.4, the comparison in classification performance between the FCSNN and other neural networks is discussed. Three sets of data are used in the experiments. Section 5.5 covers conclusions.
5.2 Automatic Derivation of the Membership Functions for Fuzzy Sets

Generally, fuzzy systems work well when we can use experts' experience to articulate the memberships for fuzzy sets. When we cannot do this, we should derive the membership functions directly from the observational data, which are the basis for an expert to formulate his knowledge.

This section presents a method of how to derive the grade of memberships for fuzzy sets to be used in fuzzy systems from a set of observational samples automatically.

5.2.1 Basic Concepts

Differently from conventional (non-fuzzy) expert systems, fuzzy systems need the information about fuzzy sets used in them. In developing such systems, we should define the membership functions for fuzzy sets.

The question of how the membership functions for fuzzy sets are defined has often been asked by those interested in fuzzy set theory. Some researchers point out that the memberships need to be defined subjectively by an expert and/or a user. The empiricists constructs membership functions for fuzzy sets from a statistical survey of the appropriate group (expert and/or users). In other words, the membership functions of fuzzy sets may be defined subjectively by an expert and/or a user, or constructed based on the opinions of the appropriate group as to whether an element is a member of a fuzzy term or not [HSK86] [HeCa76] [Tur86].

We propose a method for deriving membership functions for fuzzy sets objectively, based on the distribution of samples in the observational data set, and not on the opinions of individuals. The observational data consists of a set of samples. Each sample represents an observation of an input-output relationship and consists of several attribute values of an object and the class of the object. For each attribute, a limited number of fuzzy sets and their membership functions are derived. Each derived fuzzy set represents an
imprecise attribute value.

5.2.2 Algorithm

The following is the procedure to be performed for each attribute in order to derive the membership functions for fuzzy set from a set of samples:

Initialize : $i = 1$

STEP 1: For samples belonging to class $i$ ($i = 1, 2, \ldots, n$), construct the histogram of the attribute. The variable $n$ is the number of possible classes. The X-axis and the Y-axis of the histogram represent the universe of discourse of the attribute and the number of samples falling in each interval, respectively.

STEP 2: Compute the degree to which the $j$-th interval, $x_j$, satisfies the properties of the fuzzy set $A_i$.

$$
\mu_{A_i}(x_j) = \frac{1}{N_{e_i}} \sum_{k=1}^{l} \min(N_j, N_k)
$$

(5.1)

where $\mu_{A_i}(x_j)$ represents the grade of membership of $x_j$ in $A_i$; $N_j$ and $N_k$ are the number of samples falling in the $j$-th interval and the $k$-th interval, respectively; $l$ is the number of intervals; $N_{e_i}$ is the total number of samples belonging to class $i$.

STEP 3: If $i = n$, go to STEP 4. Otherwise, $i = i + 1$ and go to STEP 1.

STEP 4: Compute the similarity between $n$ fuzzy sets corresponding to $n$ classes by using the membership functions computed in STEP 2. Examples of similarity
measures are the Euclidean distance and the Hamming distance.

STEP 5: If the similarity is less than a chosen threshold, then the similar fuzzy sets are merged into a new fuzzy set. The GOM's of the new fuzzy sets are recomputed by Equation (5.1).

The maximum number of possible fuzzy sets for an attribute is n. STEP 1 to STEP 5 is executed repeatedly for every attribute. Thus, if there are m attributes, the maximum total number of fuzzy sets is m times n.

5.2.3 Example

Figure 5.1 shows 2-dimensional Gaussian-distributed data which consists of 1200 samples with variance 20 and with centroids at (10, 10), (-10, 10), (-10, -10), (10, -10). There are four classes and each class has 300 samples.

Figure 5.2 shows two fuzzy sets derived for attribute X1 when the Hamming distance, \(d(A_i, A_j) = \sum |\mu_{A_i}(x_k) - \mu_{A_j}(x_k)|\), where \(A_i\) and \(A_j\) are fuzzy sets for the same attribute, is used as the similarity measure. The Hamming distance 3.0 was chosen as the threshold to determine whether two fuzzy sets are similar or not. The first fuzzy set of attribute X1, \(A_{i1}\), has the highest GOM at \(X1=-10\) and the second fuzzy set of attribute X1, \(A_{i2}\), at \(X1=10\). Figure 5.3 shows two fuzzy sets \(A_{21}\) and \(A_{22}\) derived for X2. \(\mu_{A_{21}}(-10) = 1\) and \(\mu_{A_{22}}(10) = 1\).

5.3 Fuzzy Input Signal Representation

How to represent the input information as numbers is an important issue in neural networks. Neither the decimal representation nor the binary representation is necessarily optimal for generalization. In this section, we propose a new representation scheme using
Figure 5.1 The distribution of samples.
Figure 5.2 The derived fuzzy sets of attribute X1.
Figure 5.3 The derived fuzzy sets of attribute X2.
the fuzzy sets generated from the training data set and the computational scheme of the partial degree of match in fuzzy systems.

5.3.1 Fuzzy Input Signal Representation during the Training of the FCSNN

The following is the procedure for the fuzzy input signal representation of the training vectors as the preprocessing step of the training of the FCSNN:

STEP 1. Derive the membership functions for the fuzzy sets from the training data set using the algorithm presented in Section 5.2.2.

STEP 2. Divide each fuzzy set into two new fuzzy sets. Section 5.3.1.1 describes how this step is carried out.

STEP 3. Select the K fuzzy sets based on the class separability of the fuzzy sets. The details are discussed in Section 5.3.1.2.

STEP 4. Convert the training vectors into the DM vectors using the computational scheme of the degree of match and the fuzzy sets selected in STEP 3. Section 5.3.1.3 describes this step in detail.

The DM vectors are input to the CSNN, the postprocessing module of the FCSNN.

5.3.1.1 Modification of Fuzzy Sets

Usually the membership functions for the fuzzy sets generated from the training data are not nondecreasing functions. One example is shown in Figure 5.4.

The point P and Q are different values in the X-axis but has the same grade of
Figure 5.4 The original fuzzy set A.
membership in the fuzzy set A. Thus, if we use such a membership function directly for
the fuzzy input signal representation in neural networks, there is a high chance that one
object located in P and another object in Q are regarded as the same objects. If the object in
P is rejected or misclassified in the post processing module, the object in Q is also rejected
or misclassified. One promising method to cope with such a problem is the modification of
the membership functions. The modified membership functions should satisfy the
following conditions:

1) The modified membership of P is different from the one of Q.
2) The modified memberships are proportional to the original memberships.
3) The original memberships of P and Q can be recovered from the modified
memberships.

We propose one method for the modification of fuzzy sets that satisfy the above
conditions as follows:

STEP 1. Find the value, M, in the X-axis whose GOM in the fuzzy set, A, is largest.

\[ \mu_A(M) = 1 \]  

(5.2)

STEP 2. Split the original fuzzy set into two.
1) If the value, x, in the X-axis is greater than or equal to M, the GOM of x in the
first new fuzzy set, A1, is 1 and the GOM in the second new fuzzy set, A2, is
equal to the GOM of x in the original fuzzy set:

\[ \mu_{A1}(x) = 1 \text{ and } \mu_{A2}(x) = \mu_A(x) \text{ for } x \geq M \]  

(5.3)
2) If the value, \( x \), in the X-axis is less than \( M \), the GOM in the second new fuzzy set, \( A_1 \), is equal to the GOM of \( x \) in the original fuzzy set and the GOM of \( x \) in the second new fuzzy set, \( A_2 \), is 1:

\[
\mu_{A_1}(x) = \mu_A(x) \quad \text{and} \quad \mu_{A_2}(x) = 1 \quad \text{for} \quad x < M
\]  

(5.4)

Figure 5.5 shows two new fuzzy sets modified from the fuzzy set shown in Figure 5.4.

5.3.1.2 Selection of Fuzzy Sets

When we use the fuzzy input representation scheme in neural networks, the number of neurons in the input layer is equal to the total number of fuzzy sets. After every fuzzy set is split into two new fuzzy sets, the maximum total number of fuzzy sets is \( 2 \times m \times n \) where \( m \) is the number of attributes (the dimension of the original input vector) and \( n \) is the number of classes. However, due to the hardware limitation of practical neural networks, the number of available neurons in the input layer is sometimes limited. In such circumstances we need to reduce the number of fuzzy sets used for the input representation.

In this section, we select those fuzzy sets for which the class separability are the largest. Within-class scatter and between-class scatter are used to formulate the criteria of class separability, similarly to Fisher’s discriminant function in statistical pattern recognition [Fuk72]. The features (fuzzy sets) with little class separability do not necessarily contribute to classification and may even cause misclassification in certain cases. Thus, choosing the fuzzy sets with large class separability for the fuzzy input signal representation may help increase classification performance. The following is the procedure for the computation of class separability of a fuzzy set:
Figure 5.5 Two fuzzy set A1 and A2 split from A.
STEP 1. Compute the sample mean value, $m_i$, for each class $i$ and a fuzzy set $A_{jk}$ with the memberships of the $j$-th attribute values of input vectors belonging to class $i$.

$$m_i = \frac{1}{N_i} \sum_{m=1}^{N_i} \mu_{A_{jk}}(x_{mj})$$ (5.5)

where $x_{mj}$ represents the $j$-th attribute value of an input vector $X_m$ belonging to class $i$, $A_{jk}$ is the $k$-th fuzzy sets of the $j$-th attribute and $N_i$ is the total number of input vectors belonging to class $i$.

STEP 2. Compute the sample variance, $\sigma^2$, for each class $i$ and the fuzzy set $A_{jk}$ with the memberships of the $j$-th attribute values of input vectors belonging to class $i$.

$$\sigma^2 = \frac{1}{N_i} \sum_{n=1}^{N_i} [\mu_{A_{jk}}(x_{nj}) - m_i]^2$$ (5.6)

STEP 3. Compute the within-class scatter, $S_w$, for the fuzzy set $A_{jk}$. The within-class scatter shows the scatter of samples around their class mean values, and is expressed by

$$S_w = \sum_{i=1}^{n} P_i \sigma^2$$ (5.7)

where $n$ is the number of classes and $P_i$ is the a priori probability.

$$P_i = \frac{N_i}{\sum_{k=1}^{n} N_k}$$ (5.8)
STEP 4. Compute the mean value of the mixture, $m_o$, for the fuzzy set $A_{jk}$.

$$m_o = \sum_{i=1}^{n} P_i m_i$$

(5.9)

STEP 5. Compute the between-class scatter, $S_b$, as follows:

$$S_b = \sum_{i=1}^{n} P_i [m_i - m_o]^2$$

(5.10)

STEP 6. Finally, the class separability for the fuzzy set $A_{jk}$ is obtained by computing the ratio of the between-class scatter and the within-class scatter:

$$S = \frac{S_b}{S_w}$$

(5.11)

STEP 7. Repeat STEP 1 thru STEP 6 for every fuzzy set.

5.3.1.3 Conversion of the Training Vectors into the DM Vectors

Once the fuzzy sets to be used for the input representation are selected, the training vectors are converted into the DM vectors using the computational scheme of the degree of match in fuzzy systems presented in Chapter 3. For the $j$-th attribute, if the number of the selected fuzzy sets is $K_j$, the corresponding attribute value of an original training vector is converted into the $K_j$-dimensional DM values as follows:

1) In the case of precise attribute values, the DM values are obtained using the following equations:
In \([-1 \ +1]\) scale,
\[
DM_{[-1 \ +1]} = 2 \times \text{CERT}^+(x_{mj}, A_{jk}) - 1
\]  \(\text{(5.12)}\)

In \([0 \ 1]\) scale,
\[
DM_{[0 \ 1]} = \text{CERT}^+(x_{mj}, A_{jk}) = \mu_{A_{jk}}(x_{mj})
\]  \(\text{(5.13)}\)

where \(x_{mj}\) represents the \(j\)-th attribute value of an input vector \(X_m\); \(A_{jk}\) is the \(k\)-th fuzzy sets of the \(j\)-th attribute; \(\mu_{A_{jk}}(x_{mj})\) is the GOM of \(x_{mj}\) in \(A_{jk}\).

2) In the case of imprecise attribute values, the DM values are obtained using the following equations:

In \([-1 \ +1]\) scale,
- If \(\text{CERT}^+(x_{mj}, A_{jk}) \geq 0.5\) and \(\text{CERT}^-(x_{mj}, A_{jk}) \leq 0.5\),
  \[
  DM_{[-1 \ +1]} = 2 \times \text{CERT}^+(x_{mj}, A_{jk}) - 1
  \]  \(\text{(5.14)}\)
- If \(\text{CERT}^+(x_{mj}, A_{jk}) \geq 0.5\) and \(\text{CERT}^-(x_{mj}, A_{jk}) \geq 0.5\),
  \[
  DM_{[-1 \ +1]} = 2 \times \{\text{CERT}^+(x_{mj}, A_{jk}) - \text{CERT}^-(x_{mj}, A_{jk})\}
  \]  \(\text{(5.15)}\)
- If \(\text{CERT}^+(x_{mj}, A_{jk}) \leq 0.5\) and \(\text{CERT}^-(x_{mj}, A_{jk}) \geq 0.5\),
  \[
  DM_{[-1 \ +1]} = \{2 \times \text{CERT}^-(x_{mj}, A_{jk}) - 1\}
  \]  \(\text{(5.16)}\)

In \([0 \ 1]\) scale,
- If \(\text{CERT}^+(x_{mj}, A_{jk}) \geq 0.5\) and \(\text{CERT}^-(x_{mj}, A_{jk}) \leq 0.5\),
  \[
  DM_{[0 \ 1]} = \text{CERT}^+(x_{mj}, A_{jk})
  \]  \(\text{(5.17)}\)
- If \(\text{CERT}^+(x_{mj}, A_{jk}) \geq 0.5\) and \(\text{CERT}^-(x_{mj}, A_{jk}) \geq 0.5\),
  \[
  DM_{[0 \ 1]} = \{\text{CERT}^+(x_{mj}, A_{jk}) - \text{CERT}^-(x_{mj}, A_{jk})\} + 0.5
  \]  \(\text{(5.18)}\)
- If \(\text{CERT}^+(x_{mj}, A_{jk}) \leq 0.5\) and \(\text{CERT}^-(x_{mj}, A_{jk}) \geq 0.5\),
  \[
  DM_{[0 \ 1]} = 1 - \text{CERT}^-(x_{mj}, A_{jk})
  \]  \(\text{(5.19)}\)
Note that the attribute values of the training samples are usually stated precisely.

The DM vector is obtained by concatenating all of the DM values for each input vector. The membership functions of the selected fuzzy sets and the order of the concatenation for the DM vector should be saved in order to be used for the fuzzy input signal representation of the testing vectors. The size of the DM vector is equal to the total number of fuzzy sets selected using the procedure in Section 5.3.1.2.

\[
\text{Dimension of the DM vector} = \sum_{j=1}^{m} K_j \tag{5.14}
\]

where \( K_j \) is the number of the selected fuzzy sets of the \( j \)-th attribute and \( m \) is the total number of attributes.

5.3.2 Fuzzy Input Signal Representation during the Testing of the FCSNN

The fuzzy input signal representation during the testing is simple. The steps for the derivation of the membership functions, the modification of fuzzy sets and the selection of the fuzzy sets are not needed during the testing. The testing vectors are converted into the DM vectors at the same concatenation order as the DM vectors of the training vectors.

In the case of precise attribute values, we compute the DM values according to Equations (5.12) and (5.13). In the case of imprecise attribute values, we compute the DM values according to Equations (5.12) to (5.13).

5.4 Experimental Results

Some experimental results are presented in order to compare the fuzzy input signal representation and the original decimal input representation. Two sets of remote sensing data, FLC1 data and Colorado data, and a survey data for the ATV safety are used in the
experiments.

3.4.1 Experiments with the Flight Line C1 (FLC1) Data Set

The data set for the first experiment is FLC1 data set described in Section 4.4.1.1. However, the 4 dominant classes are used in this experiment. The experiments in terms of 8 classes result in the same type of results as the 4-class case [ErHo90] [Hon90]. The FCSNN is compared with the PSHNN with the delta rule, the backpropagation network and the CSNN in terms of classification performance.

5.4.1.1 Data Set

In this experiment, 8 spectral bands out of 12 bands are used. 4 farm products are chosen to represent 4 particular classes (alfalfa, corn, oats, red clover) among 8 farm products.

Training and testing with the networks are done with 200 signal samples per class and 375 other signal samples per class, respectively. So the total numbers of the training and the testing samples are 800 and 1500, respectively.

5.4.1.2 Simulation

In this experiment, the rejected vectors of a SNN were nonlinearly transformed with the shuffling and complementing method [ErHo90] [Hon90] and fed into the next SNN, as described in Section 4.1.2. RADPN and RAD$^p$ ($p = 1, 2, ..., 6$) were used in parallel as rejection schemes.

When the membership functions for fuzzy sets were derived, the Hamming distance was used as the similarity measure. The Hamming distance 4.0 was chosen as the threshold to determine whether two fuzzy sets were similar or not. Table 5.1 shows the number of fuzzy sets derived from 4-class FLC1 data set. After each fuzzy set was split
into two, the total number of fuzzy set was 44. Only 26 fuzzy sets with the largest class separability values were used for the fuzzy input representation. Thus, the number of neurons in the input layer was 26.

Competitive learning with multiple learning rates was used for learning the reference vectors. The reference vectors were computed separately for each class with the data samples belonging to the class. Ten reference vectors were defined for each class. The initialization of the various reference vectors were carried out by choosing the first ten training samples belonging to each class. Each of the multiple learning rates $C_i(t)$ was chosen as $\frac{9}{14+8}$ for all the classes.

We checked for every rejection scheme yielding no class with output 1 whether the classes of the three nearest rejection boundaries coincide. If so, we classified the input to that class. The vectors rejected in the last SNN were classified to the class of the nearest reference vector.

Table 5.2 shows the classification accuracy of the FCSNN and the total number of SNN's as a function of the number of iterations in the 4-class problem. The training accuracy was always 100 % and the best testing accuracy was 95.00 % when the number of iterations was 6 and the number of SNN's was 2.

Table 5.3 shows the number of the testing vectors rejected in the last SNN and the testing classification accuracy when the rejected vectors were not classified. The best testing accuracy was 95.32 % with 10 iterations.

5.4.1.3 Comparison to Other Neural Networks

In the case of the 4 layer backpropagation neural network (4NN), the length of the input layer and the hidden layers were fixed as 64, and the length of the output layer was 4. The initial weights were randomly chosen between -0.5 and 0.5. The best training accuracy was 99.88 % when the number of iterations was 900, and the best testing
accuracy was 92.80% when the number of iterations was 700 (see Table 5.4).

In the case of the PSHNN with the delta rule learning algorithm, the best training accuracy was 99.00% when the number of iterations was 200, and the best testing accuracy was 92.13% when the number of iterations was 90 (see Table 5.5). The RDFT-based nonlinear transformation method [ErHo90] [Hon90] was used. The input representation was based on the Gray code representation, and thus 64 nodes were required in the input layer. The number of neurons in the output layer was 8. The initial weights were randomly chosen in the range -0.5 to 0.5. The gain parameter was chosen as 0.05. Sigmoid function, \[
\frac{1}{1 + \exp(-x)},
\] was used for the nonlinear activation function.

In the case of the CSNN, the training accuracy was always 100.00% and the best testing accuracy was 94.47% when the number of iterations was 6. Table 5.6 shows the classification accuracy of the CSNN and the total number of SNN's as a function of the number of iterations in the 4-class problem. Competitive learning with multiple learning rates was used for learning the reference vectors. The reference vectors were computed separately for each class with the data samples belonging to the class. Ten reference vectors were defined for each class. The initialization of the various reference vectors were carried out by choosing the first ten training samples belonging to each class. Each of multiple learning rates \( C(t) \) was chosen as \( \frac{9}{t+8} \) for all the classes. The rejected vectors of a SNN were nonlinerly transformed with the shuffling and complementing method [ErHo90] [Hon90] and fed into the next SNN, as described in Section 4.1.2. RADPN and \( \text{RAD}^p \) \((p = 1, 2, ..., 6)\) were used in parallel as rejection schemes. Since the continuous number representation was used for the input vectors rather than binary representation, the number of neurons in the input layer was eight. We checked for every rejection scheme yielding no class with output 1 whether the classes of the three nearest rejection boundaries coincide. If so, we classified the input to that class. The vectors rejected in the last SNN were classified to the class of the nearest reference vector. Table 5.7 shows the number of
the testing vectors rejected in the last SNN and the testing classification accuracy when the rejected vectors were not classified. The best testing accuracy was 94.74%.

Table 5.8 is the summary of the experimental results with 4-class problem of the FLC1 data set. We observe that the FCSNN produces better classification performance than the CSNN, the backpropagation network and the PSHNN with the delta rule.
Table 5.1  The number of fuzzy sets derived for each attribute of the FLC1 data set.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>No. of Fuzzy Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Total</td>
<td>22</td>
</tr>
</tbody>
</table>
Table 5.2 The classification accuracy and the total number of SNN's of the FCSNN as a function of the number of iterations in the 4-class problem.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of SNN's</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Train (%)</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>100.00</td>
</tr>
</tbody>
</table>
Table 5.3 The classification accuracy and the total number of SNN's of the FCSNN as a function of the number of iterations in the 4-class problem when the vectors rejected in the last SNN are not classified.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of Rejected Vectors</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Train (%)</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>100.00</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>100.00</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>100.00</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>100.00</td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>100.00</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>100.00</td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td>100.00</td>
</tr>
<tr>
<td>8</td>
<td>18</td>
<td>100.00</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>100.00</td>
</tr>
<tr>
<td>10</td>
<td>25</td>
<td>100.00</td>
</tr>
</tbody>
</table>
Table 5.4 The classification accuracy of the backpropagation network (4NN) as a function of the number of iterations in the 4-class problem [ErHo90] [Hon90].

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train (%)</td>
</tr>
<tr>
<td>50</td>
<td>77.87</td>
</tr>
<tr>
<td>100</td>
<td>87.75</td>
</tr>
<tr>
<td>200</td>
<td>97.12</td>
</tr>
<tr>
<td>300</td>
<td>98.75</td>
</tr>
<tr>
<td>400</td>
<td>99.63</td>
</tr>
<tr>
<td>500</td>
<td>99.63</td>
</tr>
<tr>
<td>600</td>
<td>99.75</td>
</tr>
<tr>
<td>700</td>
<td>99.75</td>
</tr>
<tr>
<td>800</td>
<td>99.75</td>
</tr>
<tr>
<td>900</td>
<td>99.88</td>
</tr>
<tr>
<td>1000</td>
<td>99.88</td>
</tr>
</tbody>
</table>
Table 5.5 The classification accuracy and the total number of SNN's of the PSHNN with the delta rule as a function of the number of iterations in the 4-class problem [ErHo90] [Hon90].

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of SNN's</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Train (%)</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>95.38</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>96.12</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>96.50</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>97.50</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
<td>97.87</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>96.88</td>
</tr>
<tr>
<td>60</td>
<td>2</td>
<td>98.63</td>
</tr>
<tr>
<td>70</td>
<td>2</td>
<td>98.75</td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>98.75</td>
</tr>
<tr>
<td>90</td>
<td>2</td>
<td>98.75</td>
</tr>
<tr>
<td>100</td>
<td>2</td>
<td>98.50</td>
</tr>
<tr>
<td>200</td>
<td>2</td>
<td>99.00</td>
</tr>
<tr>
<td>300</td>
<td>2</td>
<td>98.38</td>
</tr>
<tr>
<td>400</td>
<td>2</td>
<td>98.50</td>
</tr>
</tbody>
</table>
Table 5.6  The classification accuracy and the total number of SNN's of the CSNN as a function of the number of iterations in the 4-class problem.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of SNN's</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Train (%)</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>100.00</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>100.00</td>
</tr>
</tbody>
</table>
Table 5.7  The classification accuracy and the total number of SNN's of the CSNN as a function of the number of iterations in the 4-class problem when the vectors rejected in the last SNN are not classified.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of Rejected Vectors</th>
<th>Classification Train (%)</th>
<th>Accuracy Test (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>100.00</td>
<td>93.45</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>100.00</td>
<td>94.38</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
<td>100.00</td>
<td>94.31</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>100.00</td>
<td>94.34</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>100.00</td>
<td>94.68</td>
</tr>
<tr>
<td>6</td>
<td>16</td>
<td>100.00</td>
<td>94.74</td>
</tr>
<tr>
<td>7</td>
<td>17</td>
<td>100.00</td>
<td>94.47</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>100.00</td>
<td>94.37</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>100.00</td>
<td>94.50</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>100.00</td>
<td>94.42</td>
</tr>
</tbody>
</table>
Table 5.8 The experimental results with the FLC1 multispectral image data (4-class).

<table>
<thead>
<tr>
<th>Name of Neural Network</th>
<th>Training Data</th>
<th>Testing Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iterations</td>
<td>Accuracy(%)</td>
</tr>
<tr>
<td>FCSNN (classified)</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>FCSNN (unclassified)</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>CSNN (classified)</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>CSNN (unclassified)</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>Backpropagation</td>
<td>900</td>
<td>99.88</td>
</tr>
<tr>
<td>PSHNN with the delta rule</td>
<td>200</td>
<td>99.00</td>
</tr>
</tbody>
</table>
5.4.2 Experiments with the Colorado Data Set

In this section we describe the experimental result with a multispectral earth observation remotely sensed data covering a mountainous area in Colorado. The FCSNN is compared with the CSNN in terms of classification performance.

5.4.2.1 Data Set

In this experiment, we used the same data set as in the experiment described in Section 4.4.2. See Section 4.4.2.1 for the description of the data set.

5.4.2.2 Simulation

The shuffling and complementing method was used for nonlinear transformation of rejected vectors, as described in Section 4.4.1.2. RADPN and RADP \( p = 1, 2, \ldots, 6 \) were used in parallel as rejection schemes.

When the membership functions for fuzzy sets were derived, the Hamming distance was used as the similarity measure. The Hamming distance 3.5 was chosen as the threshold to determine whether two fuzzy sets were similar or not. Table 5.9 shows the number of fuzzy sets derived from the Colorado data set. After each fuzzy set was split into two, the total number of fuzzy set was 50. 40 fuzzy sets with the largest class separability values were used for the fuzzy input representation. Thus, the number of neurons in the input layer was 40.

Competitive learning with multiple learning rates was used for learning the reference vectors. The reference vectors were computed separately for each class with the data samples belonging to the class. Ten reference vectors were defined for each class. The initialization of the various reference vectors were carried out by choosing the first ten training samples belonging to each class. Each of the multiple learning rates \( C_i(t) \) was chosen as \( \frac{3}{t+2} \) for all the classes.
We checked for every rejection scheme yielding no class with output 1 whether the class of the three nearest rejection boundaries coincide. If so, we classified the input to that class. The vectors rejected in the last SNN were classified to the class of the nearest reference vector.

Table 5.10 shows the classification accuracy of the FCSNN and the total number of SNN's as a function of the number of iterations in the 10-class problem. The best training accuracy was 99.92 % with only one pass of the training vectors (no iteration) and the best testing accuracy was 62.94 % when the number of iterations was 4.

Table 5.11 shows the number of the testing vectors rejected in the last SNN and the testing classification accuracy when the rejected vectors were not classified. The best training and testing accuracy were 100.00 % and 63.27 %, respectively.

5.4.2.3 Comparison to Other Neural Networks

The experimental results with the Colorado data set are summarized in Table 5.12. We observe that the FCSNN has better classification performance than the backpropagation network, the PSHNN with the delta rule, and the CSNN.
Table 5.9  The number of fuzzy sets derived for each attribute of the Colorado data set.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>No. of Fuzzy Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Total</td>
<td>25</td>
</tr>
</tbody>
</table>
Table 5.10  The classification accuracy and the total number of SNN’s of the FCSNN as a function of the number of iterations in the 10-class problem.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of SNN’s</th>
<th>Classification Train (%)</th>
<th>Accuracy Test (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
<td>99.92</td>
<td>59.81</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>99.75</td>
<td>61.49</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>99.75</td>
<td>61.97</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>99.75</td>
<td>62.94</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>99.58</td>
<td>61.25</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>99.58</td>
<td>62.21</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>99.92</td>
<td>60.89</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>99.92</td>
<td>61.25</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>99.92</td>
<td>60.28</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>99.83</td>
<td>62.45</td>
</tr>
</tbody>
</table>
Table 5.11  The classification accuracy and the total number of SNN's of the FCSNN as a function of the number of iterations in the 10-class problem when the vectors rejected in the last SNN are not classified.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of Rejected Vectors</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Train (%)</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>100.00</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>100.00</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>100.00</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>100.00</td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td>100.00</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>100.00</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>100.00</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td>100.00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>100.00</td>
</tr>
</tbody>
</table>
Table 5.12 The experimental results with the Colorado data set.

<table>
<thead>
<tr>
<th>Name of Neural Network System</th>
<th>Training Data</th>
<th>Testing Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iterations</td>
<td>Accuracy(%)</td>
</tr>
<tr>
<td>FCSNN (classified)</td>
<td>1</td>
<td>99.92</td>
</tr>
<tr>
<td>FCSNN (unclassified)</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>CSNN (classified)</td>
<td>1</td>
<td>98.14</td>
</tr>
<tr>
<td>CSNN (unclassified)</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>Backpropagation</td>
<td>2500</td>
<td>94.11</td>
</tr>
<tr>
<td>PSHNN with the delta rule</td>
<td>20</td>
<td>95.79</td>
</tr>
</tbody>
</table>
5.4.3 Experiments with the ATV Survey Data Set

In this section the experimental results are described when the ATV survey data are used [LeFo89] [Fol90]. The classification accuracy of the FCSNN is compared with the classification performances of the CSNN and the PSHNN with the delta rule.

5.4.3.1 Data Set [LeFo89] [Fol90]

An All-Terrain Vehicle (ATV) is a small, single, off-road vehicle designed to travel on three or four low pressure tires (see Figure 5.6). The ATV data set consists of 6 attributes and 5 classes which represents the relationship between 6 attributes and the self-reported frequency of helmet use. Table 5.13 shows the attribute names of the ATV data. They are selected from the original ATV survey data with 121 attributes. The class names of the ATV data are shown in Table 5.14.

The survey was completed in the summers of 1988 and 1989. Seven sites in six states (Indiana, Oregon, Wisconsin, Kentucky, California, and Michigan) were included in the survey in an effort to obtain a good cross section of the riding population (see Table 5.15). In Indiana, the surveys were conducted on private property; at all the other sites the surveys were administered on public lands-usually at a park specifically permitting the use of ATVs (and other off-highway vehicles). At each of the seven sites, the frequency of helmet use and other safety-related behaviors of randomly selected riders were interviewed and recorded.

136 samples were selected from 1988 data and 1989 data randomly for training and 130 samples for testing. The number of samples for each class are shown in Table 5.16.

5.4.3.2 Simulation

In this experiment, the rejected vectors of a SNN were non-linearly transformed
Figure 5.6  All-Terrain Vehicle (ATV).
Table 5.13  6 dominant attributes of the ATV survey data set.

<table>
<thead>
<tr>
<th>Attribute Number</th>
<th>Attribute Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Riding Experience (years)</td>
</tr>
<tr>
<td>2</td>
<td>Riding Frequency per a Week</td>
</tr>
<tr>
<td>3</td>
<td>Comfort of Helmet</td>
</tr>
<tr>
<td>4</td>
<td>Age</td>
</tr>
<tr>
<td>5</td>
<td>Law</td>
</tr>
<tr>
<td>6</td>
<td>Number of ATV Accidents</td>
</tr>
</tbody>
</table>
Table 5.14  6 dominant classes of the ATV survey data set.

<table>
<thead>
<tr>
<th>Class</th>
<th>Frequency of Helmet Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Never/Very rarely</td>
</tr>
<tr>
<td>2</td>
<td>Rarely</td>
</tr>
<tr>
<td>3</td>
<td>About Half</td>
</tr>
<tr>
<td>4</td>
<td>Often</td>
</tr>
<tr>
<td>5</td>
<td>Very often/Always</td>
</tr>
</tbody>
</table>
Table 5.15  Survey sites of the ATV data set.

<table>
<thead>
<tr>
<th>State</th>
<th>Location</th>
<th>Helmet Law</th>
<th>Number of Surveys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indiana</td>
<td>North-Central</td>
<td>No</td>
<td>38</td>
</tr>
<tr>
<td>Oregon</td>
<td>Oregon Dunes National Park</td>
<td>No</td>
<td>56</td>
</tr>
<tr>
<td>Wisconsin</td>
<td>Bong Recreational Area</td>
<td>No</td>
<td>30</td>
</tr>
<tr>
<td>Kentucky</td>
<td>Daniel Boone National Forest</td>
<td>No</td>
<td>9</td>
</tr>
<tr>
<td>Kentucky</td>
<td>Land Between the Lakes OHV Area</td>
<td>Yes</td>
<td>27</td>
</tr>
<tr>
<td>California</td>
<td>Pismo Beach OHV Area</td>
<td>Yes</td>
<td>60</td>
</tr>
<tr>
<td>Michigan</td>
<td>Silver Lake Park</td>
<td>Yes</td>
<td>45</td>
</tr>
</tbody>
</table>
Table 5.16  The number of samples of each class of the ATV survey data set.

<table>
<thead>
<tr>
<th></th>
<th>Class</th>
<th>Train (136)</th>
<th>Test (130)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>95</td>
<td>93</td>
<td></td>
</tr>
</tbody>
</table>
with the shuffling and complementing method [ErHo90] [Hon90] and fed into the next SNN, as described in Section 4.1.2. RADPN and RAD\(^p\) (p = 1, 2, ..., 6) were used in parallel as rejection schemes.

When the membership functions for fuzzy sets were derived, the Hamming distance was used as the similarity measure. The Hamming distance 3.0 was chosen as the threshold to determine whether two fuzzy sets were similar or not. Table 5.17 shows the number of fuzzy sets derived from the ATV data set. After each fuzzy set was split into two, the total number of fuzzy set was 24. All the fuzzy sets were used. Thus, the number of neurons in the input layer was 24.

Competitive learning with multiple learning rates was used for learning the reference vectors. The reference vectors were computed separately for each class with the data samples belonging to the class. Four reference vectors were defined for each class. The initialization of the various reference vectors were carried out by choosing the first ten training samples belonging to each class. Each of the multiple learning rates \( C^i(t) \) was chosen as \( \frac{9}{t+8} \) for all the classes.

We checked for every rejection scheme yielding no class with output 1 whether the classes of the three nearest rejection boundaries coincide. If so, we classified the input vector to that class. The vectors rejected in the last SNN were classified to the class of the nearest reference vector.

Table 5.18 shows the classification accuracy of the FCSNN and the total number of SNN's as a function of the number of iterations. The training accuracy was always 100.00% and the best testing accuracy was 68.46% when the number of iterations was 4 and the number of SNN's was 3.

Table 5.19 shows the number of the testing vectors rejected in the last SNN and the testing classification accuracy when the rejected vectors were not classified. The best testing accuracy was 68.99% with 6 iterations.
5.4.3.3 Comparison to Other Neural Networks

In the case of the PSHNN using the delta rule learning algorithm, the best training accuracy was 88.24 % when the allowed number of iterations was 150 and the number of SNN's was 5, and the best testing accuracy was 46.92 % when the maximum number of iterations was 100 and the number of SNN's was 3 (see Table 5.20). Gray code representation was used. In Gray code representation of the original decimal inputs, 8 bits were used to represent one attribute value. The input vector was generated by concatenating 6 corresponding attribute values together. Thus, 42 nodes were required in the input layer. The number of neurons in the output layer was 5. Each SNN was chosen as a single-layer network and was trained using the delta rule with a nonlinear activation function. Sigmoid function was used for the nonlinear activation function. Rejected input vectors of a SNN are nonlinerly transformed with the shuffling and complementing method and fed into the next SNN. The initial weights are randomly chosen in the range -0.5 to 0.5. The gain parameter was chosen as 0.05.

In the case of the CSNN, the best training accuracy was 100.00 % when the number of iterations was 1, and the best testing accuracy was 67.69 % when the number of iterations was 7. Table 5.21 shows the classification accuracy of the CSNN and the total number of SNN's as a function of the number of iterations with the ATV data set. Competitive learning with multiple learning rates was used for learning the reference vectors. The reference vectors were computed separately for each class with the data samples belonging to the class. Four reference vectors were defined for each class. The initialization of the various reference vectors were carried out by choosing the first ten training samples belonging to each class. Each of multiple learning rates $C_l(t)$ was chosen as $\frac{9}{t+8}$ for all the classes. The rejected vectors of a SNN were nonlinearly transformed with the shuffling and complementing method and fed into the next SNN, as described in Section 4.1.2. RADPN and $RAD^p$ ($p = 1, 2, ..., 6$) were used in parallel as rejection
schemes. Since the continuous number representation was used for the input vectors rather than binary representation, the number of neurons in the input layer was 6. We checked for every rejection scheme yielding no class with output 1 whether the classes of the three nearest rejection boundaries coincide. If so, we classified the input to that class. The vectors rejected in the last SNN were classified to the class of the nearest reference vector. Table 5.22 shows the number of the testing vectors rejected in the last SNN and the testing classification accuracy when the rejected vectors were not classified. The best testing accuracy was 67.69 %.

Table 5.23 is the summary of the experimental results with the ATV data set. We observe that the FCSNN produces higher classification accuracy than the CSNN and the PSHNN with the delta rule.
Table 5.17 The number of fuzzy sets derived for each attribute of the FLC1 data set.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>No. of Fuzzy Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>12</strong></td>
</tr>
</tbody>
</table>
Table 5.18  The classification accuracy and the total number of SNN's of the FCSNN as a function of the number of iterations with the ATV data set.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of SNN's</th>
<th>Classification Train (%)</th>
<th>Accuracy Test (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>100.00</td>
<td>63.07</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>100.00</td>
<td>67.69</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>100.00</td>
<td>66.15</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>100.00</td>
<td>68.46</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>100.00</td>
<td>67.69</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>100.00</td>
<td>68.46</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>100.00</td>
<td>68.46</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>100.00</td>
<td>68.46</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>100.00</td>
<td>67.69</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>100.00</td>
<td>67.69</td>
</tr>
</tbody>
</table>
Table 5.19 The classification accuracy and the total number of SNN's of the FCSNN as a function of the number of iterations with the ATV data set when the vectors rejected in the last SNN are not classified.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of Rejected Vectors</th>
<th>Classification Train (%)</th>
<th>Accuracy Test (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>100.00</td>
<td>63.07</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>100.00</td>
<td>68.75</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>100.00</td>
<td>67.19</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>100.00</td>
<td>68.22</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>100.00</td>
<td>67.69</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>100.00</td>
<td>68.99</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>100.00</td>
<td>68.99</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>100.00</td>
<td>68.99</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>100.00</td>
<td>68.22</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>100.00</td>
<td>68.22</td>
</tr>
</tbody>
</table>
Table 5.20  The classification accuracy and the total number of SNN's of the PSHNN with the delta rule as a function of the number of iterations with the ATV data set.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of SNN's</th>
<th>Classification Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Train (%)</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>84.56</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>83.09</td>
</tr>
<tr>
<td>150</td>
<td>5</td>
<td>88.24</td>
</tr>
</tbody>
</table>
Table 5.21 The classification accuracy and the total number of SNN's of the CSNN as a function of the number of iterations with the ATV data set.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of SNN's</th>
<th>Classification Train (%)</th>
<th>Accuracy Test (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>100.00</td>
<td>58.46</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>97.80</td>
<td>66.15</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>99.26</td>
<td>64.62</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>100.00</td>
<td>63.08</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>98.53</td>
<td>63.85</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>100.00</td>
<td>63.08</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>99.26</td>
<td>67.69</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>100.00</td>
<td>63.85</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>100.00</td>
<td>63.85</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>99.26</td>
<td>63.85</td>
</tr>
</tbody>
</table>
Table 5.22: The classification accuracy and the total number of SNN's of the CSNN as a function of the number of iterations with the ATV data set when the vectors rejected in the last SNN are not classified.

<table>
<thead>
<tr>
<th>Number of Iterations</th>
<th>Number of Rejected Vectors</th>
<th>Classification Train (%)</th>
<th>Accuracy Test (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>100.00</td>
<td>61.47</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>100.00</td>
<td>66.15</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>100.00</td>
<td>64.62</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>100.00</td>
<td>63.57</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>100.00</td>
<td>66.94</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>100.00</td>
<td>63.57</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>100.00</td>
<td>67.69</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>100.00</td>
<td>63.85</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>100.00</td>
<td>63.85</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>100.00</td>
<td>63.85</td>
</tr>
</tbody>
</table>
Table 5.23 The experimental results with the ATV survey data.

<table>
<thead>
<tr>
<th>Name of Neural Network</th>
<th>Training Data</th>
<th>Testing Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iterations</td>
<td>Accuracy(%)</td>
</tr>
<tr>
<td>FCSNN (classified)</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>FCSNN (unclassified)</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>CSNN (classified)</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>CSNN (unclassified)</td>
<td>1</td>
<td>100.00</td>
</tr>
<tr>
<td>PSHNN with the delta rule</td>
<td>150</td>
<td>88.24</td>
</tr>
</tbody>
</table>
5.5 Conclusions

We presented parallel self-organizing, hierarchical neural networks with fuzzy input signal representation, competitive learning and safe rejection schemes. First a method to derive the grade of memberships for fuzzy sets from a set of observational samples was proposed. The derived fuzzy sets are modified so that the possibility of misclassification can be reduced. Then the selected fuzzy sets with the largest class separability and the computational scheme of the degree of partial match are used to convert the original decimal input into multidimensional values. The fuzzy input signals are input to the CSNN. The experimental results with the FCSNN in comparison to the CSNN show that the classification performance can be improved with the fuzzy input signal representation scheme.
CHAPTER 6
CONCLUSIONS

6.1 Conclusions

In this thesis the parallel self-organizing, hierarchical neural networks with fuzzy input signal representation, competitive learning and safe rejection schemes (FCSNN) were discussed.

The major contributions of this thesis can be considered to be the following: 1) a computational scheme of the partial degree of match (DM) in fuzzy systems was proposed; 2) the parallel self-organizing, hierarchical neural networks with competitive learning and safe rejection schemes (CSNN) was developed to get around disadvantages of the backpropagation network, the PSHNN with the delta rule, and competitive learning systems; 3) a method for the automatic derivation of the membership functions for fuzzy sets based on the distribution of samples in observational data set was presented; 4) the fuzzy input signal representation scheme was developed, not only for improving the classification accuracy but also for being able to classify objects imprecise attribute values.

The proposed computational scheme of the degree of match (DM) can be used consistently to compute the degree of match for any type of combination between the antecedent part of a rule and an assertion. The computed degree of partial match between a precise assertion $(A^*)$ and an imprecise antecedent $(A)$ using the proposed scheme is equal to the grade of membership of $A^*$ in a fuzzy set $A$, which is a requirement for a computational scheme of the degree of match violated by the other methods.

The CSNN is characterized by two features, competitive learning and multiple safe rejection schemes. After reference vectors are computed using competitive learning, the
safe rejection schemes are constructed. There are two possible methods for the computation of reference vectors. When the reference vectors are computed separately for each class (Method II), higher classification accuracy was obtained and less number of SNN's were needed in comparison to the method that the reference vectors were computed together for all the classes (Method I). In addition, Method II can be performed in parallel, resulting in much faster learning. Competitive learning system with multiple learning rates converged to a stable equilibrium more rapidly and produced higher classification accuracy than the system with a single scalar learning rate when Method II was used.

The purpose of the safe rejection schemes is to reject the vectors in such a way that there are no misclassified training vectors. Two different kinds of safe rejection schemes, RADPN and RAD, were developed and used together. Experimental results indicated the superiority of the CSNN in comparison to the backpropagation network and the PSHNN with the delta rule, both in terms of classification accuracy and speed of processing.

The proposed method for the automatic derivation of the membership functions for fuzzy sets are useful in developing fuzzy systems when the memberships cannot be constructed by experts and/or users. Furthermore, it can be used for developing the fuzzy input signal representation scheme of neural networks.

The derived fuzzy sets are modified so that the possibility of the misclassification in neural networks can be reduced. Then the selected fuzzy sets with the largest class separability and the computational scheme of the degree of partial match constitute the fuzzy signal representation scheme. The original decimal input is converted into multidimensional DM values using the fuzzy input signal representation scheme. The DM vectors are passed to the CSNN. The experimental results with the FCSNN, the CSNN with the fuzzy input signal representation scheme, in comparison to the CSNN showed that the classification performance could be improved with the fuzzy input signal representation scheme.
6.2 Possible Future Research

The CSNN and the FCSNN need to be investigated further in order to improve the classification accuracy and the processing speed. The following is the outline of some of the major issues.

1. Development of safe rejection schemes:

The classification performance of the CSNN and the FCSNN is dependent on the used rejection schemes. Until now two kinds of safe rejection schemes, RADPN and RAD, are available. More safe rejection schemes need to be developed in order to improve the classification accuracy.

2. Improvement of competitive learning:

The CSNN and the FCSNN are affected by how well the reference vectors are learned. The development of better competitive learning algorithms is required in order to get higher classification accuracy.

3. Development of nonlinear transforms: In the CSNN and FCSNN, the input vectors rejected in the previous stage neural network are nonLinearly transformed before they are fed into the next stage neural network. The transformation method affects the classification accuracy of neural networks [ErHo90] [Hon90]. In the experiments described in this thesis, the shuffling and complementing method was used for the nonlinear transformations. It is a simple method for hardware and software implementations but is not necessarily the optimal transformation method. Additional studies on nonlinear transformation methods need to be conducted.
4. Learning strategy with the weighted data set:

Until now the samples in the data set are assumed to be equally weighted. In other words, each sample has the same importance. However, generally some samples can be more important than others when the data are collected from several data sources with different importance. Even one data source can give several samples with different weights. For the weighted data set, the learning strategy needs to be modified. How to weigh each sample or groups of data is also an important research topic.

5. The certainty value of the classification:

Our current system classifies one object into a class or rejects it in the all-or-nothing fashion without the indication of degree of certainty for the classification. In some classification systems, for example, medical diagnoses, the degree of the certainty of classification will be helpful to the system users. Even when an object is rejected without classification, the possible guess based on the degree of certainty of rejection will be better than just giving up the classification about the object. This issue needs to be investigated.
LIST OF REFERENCES


[EHB90] Ersoy, O.K. and Hong, D. and Busta, H.H., "Recent Study with the Parallel, Self-Organizing, Hierarchical Neural Networks", *Third Conference on Neural Networks and Distributed Processing*, Fort Wayne, Indiana, April 1990.


