Chapter 3

Fuzzy Hyperline Segment Neural Network\textsuperscript{1}

\textsuperscript{1} The results obtained from the proposed classifier in this chapter have been published \textit{in int. joint conference on neural networks: IEEE-IJCNN'01, held in Washington DC, USA, July 2001.}
3.1 INTRODUCTION

In this chapter we discuss proposed FHLSNN(66) that is used for invariant handwritten character recognition. The FHLSNN utilizes fuzzy sets as pattern classes in which each fuzzy set is an union of fuzzy set hyperline segments. The fuzzy set hyperline segment is an \( n \)-dimensional hyperline segment defined by two end points with a corresponding membership function. The performance of FHLSNN algorithm is compared with the FNN, MFNN and FMN algorithms.

This chapter is organized as follows. The proposed architecture and its learning algorithm are described in section 3.2 and 3.3, respectively. The experimental results derived after implementation are tabulated in section 3.4. In addition to this, section 3.5 describes an example of pattern classification in 2-D space to compare the FHLSNN with FMN algorithm. Conclusions are given in section 3.6.

3.2 TOPOLOGY OF THE FHLSNN

The architecture of FHLSNN consists of four layers as shown in Fig. 3.1. In this architecture first, second, third and fourth layer is denoted as \( F_R, F_E, F_D \) and \( F_C \), respectively. The \( F_R \) layer accepts an input pattern and consists of \( n \) processing elements, one for each dimension of the pattern. The \( F_E \) layer consists of \( m \) processing nodes that are constructed during training. There are two connections from each \( F_R \) to \( F_E \) node; one connection represents one end point for that dimension and the other connection represents another end point of that dimension, for a particular hyperline segment as shown in Fig. 3.2. One end point of fuzzy hyperline segment is stored in matrix \( V \) and the other end point is stored in matrix \( W \).
Each $F_E$ node represents hyperline segment fuzzy set and is characterized by the transfer function. Let $R_h = (r_{h1}, r_{h2}, \ldots, r_{hn})$ represents the $h$th input pattern, $V_j = (v_{j1}, v_{j2}, \ldots, v_{jn})$ is one end point of the hyperline segment $e_j$ and $W_j = (w_{j1}, w_{j2}, \ldots, w_{jn})$ is the other end point of $e_j$. The hyperline segment membership function of $j$th $F_E$ node is defined as,
\( e_j(R_h, V_j, W_j) = 1 - f^3(x, \gamma_1, l) \),

(3.1)
in which \( x = l_1 + l_2 \) and the distances \( l_1, l_2 \) and \( l \) are defined as,

\[
l_1 = \left( \sum_{i=1}^{n} (w_{ji} - r_{hi})^2 \right)^{1/2},
\]

(3.2)

\[
l_2 = \left( \sum_{i=1}^{n} (v_{ji} - r_{hi})^2 \right)^{1/2},
\]

(3.3)

\[
l = \left( \sum_{i=1}^{n} (w_{ji} - v_{ji})^2 \right)^{1/2},
\]

(3.4)

and \( f^3(\cdot) \) is a three-parameter ramp threshold function defined as,

\[
f^3(x, \gamma_1, l) = \begin{cases} 
0 & \text{if } x = l \\
1 & \text{otherwise}
\end{cases}
\]

\[
f^3(x, \gamma_1, l) = \begin{cases} 
xy_1 & \text{if } 0 \leq xy_1 \leq 1 \\
1 & \text{if } xy_1 > 1.
\end{cases}
\]

The fuzzy hyperline segment membership function shown in Fig. 3.3 returns highest membership value equal to one if the pattern \( R_h \) falls on the hyperline segment joined by two end points \( V_j \) and \( W_j \). The membership value is governed by the sensitivity parameter \( \gamma_1 \), which regulates how fast the membership value decreases when the distance between \( R_h \) and \( e_j \) increases. For the given input pattern \( R_h \), \( e_j \)'s output value is computed using (3.1).

Each node of \( F_C \) and \( F_D \) layer represents a class. The \( F_D \) layer gives soft decision and output of \( k \)th \( F_D \) node represents the degree to which the input pattern belongs to the class \( d_k \). The binary weight values assigned to the connections between \( F_E \) and \( F_D \) layers are stored in matrix \( U \) and defined as,
for \( k = 1 \) to \( p \).

\[
0.5 \ldots \quad \text{where} \quad 1 = \max\{d_k\}, \text{for} \quad k = 1,2,\ldots,m
\]

where \( e_j \) is the \( j \)th \( F_E \) node and \( d_k \) is the \( k \)th \( F_D \) node.

The transfer function of each \( F_D \) node performs the union of appropriate (of same class) hyperline segment fuzzy values, which is described as,

\[
d_k = \max_{j=1}^{m} e_j u_{jk} \quad \text{for} \quad k = 1,2,\ldots,p.
\]  \hspace{1cm} (3.6)

Each \( F_C \) node delivers nonfuzzy output described as,

\[
c_k = \begin{cases} 
0 & \text{if} \ d_k < T_i \\
1 & \text{if} \ d_k = T_i 
\end{cases} \quad \text{where} \quad T_i = \max(d_k), \text{for} \quad k = 1 \text{ to } p.
\]  \hspace{1cm} (3.7)

for \( k = 1 \) to \( p \).

Figure 3.3: The plot of fuzzy hyperline segment membership function for \( \gamma_1 = 1 \) with end points \( w = [0.5 \ 0.3] \) and \( v = [0.5 \ 0.7] \).
3.3 THE FHLSNN LEARNING ALGORITHM

The supervised FHLSNN learning algorithm for creating fuzzy hyperline segments in the hyperspace consists of three steps as stated below.

(a) Creation of hyperline segments

(b) Intersection test

(c) Removing intersection

Above three steps are described below in detail.

(a) Creation of hyperline segments: In the learning process the user defined parameter $\theta_2$ puts limit on the length of hyperline segments. The hyperline segments are extended only if the length after extension is less than or equal to $\theta_2$. Given the $h$th training pair $(R_h, d_h)$, find all the hyperline segments belonging to the class $d_h$. After this following four steps are carried sequentially for possible inclusion of the input pattern $R_h$.

Step 1: Determine whether the pattern $R_h$ falls on any one of the hyperline segments. This can be verified by using fuzzy hyperline segment membership function described in (3.1). If $R_h$ falls on any one of the hyperline segment then it is included, therefore in the training process all the remaining steps are skipped and training is continued with the next training pair.

Step 2: If the pattern $R_h$ falls on any one of the hyperline passing through two end points of the hyperline segment, then extend the hyperline segment to include the pattern. Suppose $e_j$ is that hyperline segment with end points $V_j$ and $W_j$ then $l_1$, $l_2$ and $l$ are calculated using (3.2), (3.3) and (3.4), respectively.

Where $l_1$ is the distance of $R_h$ from end point $W_j$, $l_2$ is the distance of $R_h$ from end point $V_j$ and $l$ is the length of hyperline segment.
2(a): If \( l_1 > l_2 \) then test whether the point \( V_j \) falls on the hyperline segment formed by the points \( W_j \) and \( R_h \). This condition can be verified using (3.1) i.e. if \( e_j(W_j, R_h, W_j) = 1 \), then the hyperline segment is extended by replacing end point \( V_j \) with \( R_h \) to include \( R_h \). Hence

\[
V_{j_{\text{new}}} = R_h \quad \text{and} \quad W_{j_{\text{new}}} = W_j.
\]

(3.8)

2(b): If \( l_2 > l_1 \) then test whether the point \( W_j \) falls on the hyperline segment formed by the points \( V_j \) and \( R_h \). If \( e_j(W_j, V_j, R_h) = 1 \), hyperline segment is extended by replacing end point \( W_j \) with \( R_h \) to include the pattern \( R_h \). Hence

\[
W_{j_{\text{new}}} = R_h \quad \text{and} \quad V_{j_{\text{new}}} = V_j.
\]

(3.9)

Step 3: If hyperline segment is a point then extend it to include the pattern \( R_h \) as described by (3.8).

Step 4: If the pattern \( R_h \) is not included by any one of the above steps then new hyperline segment is created for that class, which is described as,

\[
W_{\text{new}} = R_h \quad \text{and} \quad V_{\text{new}} = R_h.
\]

(3.10)

(b) Intersection test: The learning algorithm allows intersection of hyperline segments from the same class and eliminates the intersection between hyperline segments from separate classes. Therefore, it is necessary to eliminate intersection between the hyperline segments that represent different classes. Intersection test is carried out as soon as the hyperline segment is extended either by step 2 or step 3 or created in step 4.

Let \( W_{\text{lst}} = [x_1, x_2, ..., x_n] \), \( V_{\text{lst}} = [y_1, y_2, ..., y_n] \) represent two end points of extended or created hyperline segment and \( W_n = [x'_1, x'_2, ..., x'_n] \), \( V_n = [y'_1, y'_2, ..., y'_n] \) are end points of the hyperline segment of other class.
First of all test whether the hyperlines passing through end points of two hyperline segments intersect. This is described by the following equations. The equation of hyperline passing through $W_{st}$ and $V_{st}$ is

$$\begin{bmatrix} a_i - x_i \\ y_i - x_i \end{bmatrix} = r_1 \text{ for } i = 1, 2, \ldots, n$$ (3.11)

where $[a_1, a_2, \ldots, a_n]$ represents a point on the hyperline and the equation of the hyperline passing through $W_n$ and $V_n$ is

$$\begin{bmatrix} b_i - x_i \\ y_i - x_i \end{bmatrix} = r_2 \text{ for } i = 1, 2, \ldots, n$$ (3.12)

where $[b_1, b_2, \ldots, b_n]$ represents a point on the hyperline and $r_1$, $r_2$ are the constants. These hyperlines intersect if and only if $a_i = b_i$. The equations (3.11) and (3.12) with the assumption $a_i = b_i$ leads to set of $n$ simultaneous equations which are described as

$$r_i (y_i - x_i) + x_i = r_2 (y_i' - x_i') + x_i'$$

for $i = 1, 2, \ldots, n$. (3.13)

The values of $r_1$ and $r_2$ can be calculated by solving any two simultaneous equations. If remaining $n-2$ equations are satisfied with calculated values then two hyperlines are intersecting and the point of intersection $p_i$ is calculated as,

$$p_i = (r_i (y_i - x_i) + x_i, \ldots, r_n (y_n - x_n) + x_n)$$ (3.14)

The point of intersection $p_i$, if falls on both hyperline segments then hyperline segments also intersect. This can be verified using (3.1) i.e. if $e_{st}(p_i, V_{st}, W_{st}) = 1$ and $e_{n}(p_i, V_n, W_n) = 1$ means two hyperline segments from separate class are intersecting. This intersection is eliminated by contraction of just extended hyperline segment.
(c) Removing intersection: If step 2(a) and step 3 has created intersection of hyperline segments from separate classes then intersection is removed by restoring the end point $V_j$ as $V_j^{\text{new}} = V_j$, if step 2(b) has created intersection then intersection is removed by restoring the end point $W_j$ as $W_j^{\text{new}} = W_j$ and new hyperline segment is created to include the input pattern $R_h$, which is described by (3.10). If step 4 creates intersection then it is removed by breaking earlier hyperline segment of other class into two hyperline segments described as,

$$W_{\text{new}+1} = V_{\text{new}+1} = V_n \quad \text{and} \quad V_n = W_n.$$  \hspace{1cm} (3.15)

### 3.4 EXPERIMENTAL RESULTS

We have implemented the proposed approach using MATLAB 5.1 and ran on Pentium-III, 733MHz PC. Several experiments are carried out to compare the proposed approach with FNN, MFNN and FMN algorithms. The best results provided by FNN and MFNN algorithms are carried forward from Table 2.2 in chapter 2 and listed in Table 3.3. The data sets that are described in chapter 2 are used in the experiments.

The training set, set-1 is used to train the FMN algorithm with sensitivity parameter $\gamma = 1$ by varying the parameter $\lambda$ i.e. the maximum size of the hyperbox. The results obtained are listed in Table 3.1, which shows that the FMN creates 851 hyperboxes with $\lambda = 0.017$ and yields 100% recognition rate for the training set. After training, the performance of FMN algorithm is tested with different sets of ring data features. These results are listed in the third row of Table 3.3.

Similarly, the FHLSNN algorithm is also trained by varying the parameter $\theta_2$ with $\gamma_1$ equal to one. The results obtained are listed in Table 3.2. Table 3.2 shows that the FHLSNN algorithm creates 801 hyperline segments with $\theta_2 = 0.11$ and yields 100% recognition rate for the training set. Though the
FHLSNN algorithm can give 100% recognition rate for the training set with just 500 hyperline segments i.e. half of the training patterns, we have adjusted parameters such that number of hyperboxes and hyperline segments become comparable and the performance of these algorithms can be compared.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Hyperboxes created</th>
<th>Percentage recognition rates with set-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.02</td>
<td>773</td>
<td>99.1</td>
</tr>
<tr>
<td>0.019</td>
<td>795</td>
<td>99.7</td>
</tr>
<tr>
<td>0.018</td>
<td>818</td>
<td>99.8</td>
</tr>
<tr>
<td>0.017</td>
<td>851</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 3.1: Hyperboxes created and recognition rates obtained with training set of ring features

<table>
<thead>
<tr>
<th>$\theta_2$</th>
<th>Hyperline segments created</th>
<th>Percentage recognition rates with set-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.11</td>
<td>801</td>
<td>100</td>
</tr>
<tr>
<td>0.095</td>
<td>866</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 3.2: Hyperline segments created and recognition rates obtained with training set of ring features

The performance of FHLSNN algorithm is then verified for the different sets of ring data features with $\theta_2 = 0.11$ and 801 hyperline segments. The results obtained are listed in the last row of Table 3.3. This table reveals that the FHLSNN algorithm performs better for all sets compared to the FNN, MFNN algorithms and the recognition rates are comparable with that of FMN algorithm.

The performance of FHLSNN algorithm is compared with other classifiers for training time and recall time per pattern. The results obtained are tabulated in Table 3.4. These algorithms are trained by adjusting training parameters to get 100% recognition rate for the training set of ring data features. The neurons in output layer of the FNN, YAGER-MIN FNs of the MFNN, hyperboxes of
the FMN and hyperline segments of the FHLSNN are also listed in this table. The probability of intersection of hyperline segments is very less if dimension of feature space is moderately large, therefore when the FHLSNN is trained without intersection test, same recognition rates are obtained with drastic reduction in training time. These results are tabulated in the last row of Table 3.4. Hence, Table 3.4 indicates that the FHLSNN algorithm outperforms than all other classifiers as far as training time and recall time per pattern is concerned.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Set-1</th>
<th>Set-2</th>
<th>Set-3</th>
<th>Set-4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNN</td>
<td>100</td>
<td>78.7</td>
<td>33.1</td>
<td>29.6</td>
<td>60.35</td>
</tr>
<tr>
<td>MFNN</td>
<td>100</td>
<td>81.1</td>
<td>34.1</td>
<td>29.9</td>
<td>61.2</td>
</tr>
<tr>
<td>FMN</td>
<td>100</td>
<td>99.3</td>
<td>47.3</td>
<td>45.5</td>
<td>73.025</td>
</tr>
<tr>
<td>FHLSNN</td>
<td>100</td>
<td>99.7</td>
<td>45.8</td>
<td>44.7</td>
<td>72.55</td>
</tr>
</tbody>
</table>

Table 3.3: Percentage recognition rates with ring-data features

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Neurons in output layer/ YAGER-MIN FNs/ hyper-boxes/ Hyperline segments</th>
<th>Training time in seconds</th>
<th>Recall time per pattern in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNN</td>
<td>971</td>
<td>584.74</td>
<td>1.1685</td>
</tr>
<tr>
<td>MFNN</td>
<td>963</td>
<td>702.61</td>
<td>1.3848</td>
</tr>
<tr>
<td>FMN</td>
<td>851</td>
<td>384.25</td>
<td>1.5235</td>
</tr>
<tr>
<td>FHLSNN</td>
<td>801</td>
<td>42.461</td>
<td>0.593</td>
</tr>
</tbody>
</table>

Table 3.4: Timing analysis with ring-data features

The performance of FNN, MFNN, FMN and FHLSNN algorithms is also compared with Zernike features. The best results provided by FNN and MFNN algorithms are carried forward from Table 2.4 and listed in Table 3.6. The training set, set-1 is used to train the FMN algorithm with sensitivity parameter $\gamma = 1$ by varying the size of hyperbox. The results obtained are listed in Table 3.5, which shows that the FMN creates 957 hyperboxes with $\lambda = 0.003$ and
yields 100% recognition rate for the training set. After training, the performance of FMN algorithm is tested with different sets of Zernike features. These results are listed in the third row of Table 3.6.

Similarly, the FHLSNN algorithm is also trained by varying the parameter $\theta_2$ with $\gamma_1$ equal to one. It is observed that the FHLSNN algorithm creates 913 hyperline segments when trained with $\theta_2 = 0.02$. Since the number of hyperline segments is comparable to that of hyperboxes, we verified the performance of FHLSNN algorithm for testing sets. These results are listed in the last row of Table 3.6.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Hyperboxes created</th>
<th>Percentage recognition rates with set-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.005</td>
<td>767</td>
<td>98.9</td>
</tr>
<tr>
<td>0.004</td>
<td>872</td>
<td>99.9</td>
</tr>
<tr>
<td>0.003</td>
<td>957</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 3.5: Hyperboxes created and recognition rates obtained with training set of Zernike features

Table 3.6 indicates that the FHLSNN yields better results as compared to all other algorithms. The comparison of Table 3.3 and 3.6 shows that the percentage recognition rates obtained using all the classifiers are better for ring features than Zernike features.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Set-1</th>
<th>Set-2</th>
<th>Set-3</th>
<th>Set-4</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNN</td>
<td>100</td>
<td>25.6</td>
<td>20.8</td>
<td>13.9</td>
<td>40.075</td>
</tr>
<tr>
<td>MFNN</td>
<td>100</td>
<td>27.9</td>
<td>20.1</td>
<td>15.4</td>
<td>40.85</td>
</tr>
<tr>
<td>FMN</td>
<td>100</td>
<td>51.9</td>
<td>28.7</td>
<td>21.5</td>
<td>50.525</td>
</tr>
<tr>
<td>FHLSNN</td>
<td>100</td>
<td>53.5</td>
<td>29.5</td>
<td>24.5</td>
<td>51.875</td>
</tr>
</tbody>
</table>

Table 3.6: Percentage recognition rates with the Zernike features

53
The noise tolerance of FMN and FHLSNN algorithms is studied experimentally using ring features. The sets, set-5 to 11 that are described in chapter 2 are used in the experiments. The FMN and FHLSNN are trained using set-1 and the performance is tested with sets, set-5 to 11. These results are listed in Table 3.7. The results for FNN and MFNN algorithm are brought forward from Table 2.5. This table shows that the performance of FNN and MFNN degrades relatively as the noise density increases, whereas the FMN and FHLSNN perform well with moderate level of noise.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Set-5</th>
<th>Set-6</th>
<th>Set-7</th>
<th>Set-8</th>
<th>Set-9</th>
<th>Set-10</th>
<th>Set-11</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNN</td>
<td>79.1</td>
<td>65.0</td>
<td>58.7</td>
<td>45.7</td>
<td>33.4</td>
<td>25.1</td>
<td>22.0</td>
</tr>
<tr>
<td>MFNN</td>
<td>80.7</td>
<td>67.7</td>
<td>63.0</td>
<td>48.6</td>
<td>35.8</td>
<td>26.9</td>
<td>23.1</td>
</tr>
<tr>
<td>FMN</td>
<td>98.3</td>
<td>94.9</td>
<td>91.2</td>
<td>74.3</td>
<td>53.4</td>
<td>37.8</td>
<td>29.9</td>
</tr>
<tr>
<td>FHLSNN</td>
<td>98.6</td>
<td>94.9</td>
<td>91.1</td>
<td>73.6</td>
<td>52.6</td>
<td>37.7</td>
<td>31.0</td>
</tr>
</tbody>
</table>

Table 3.7: Percentage recognition rates with different noise densities

3.5 TWO CLASS EXAMPLE IN 2-D SPACE

It is said that patterns of same class fall close to each other in the pattern space. But in the handwritten character recognition application, characters of different writers vary in style and shape. In addition to this, the features extracted are rotation invariant, which are not suitable for generalization because the character class boundaries defined on these features are heavily overlapped and ill defined. Due to this, it is expected that the distribution of the patterns in pattern space is random and it is not necessary that the patterns of same class shall fall close to each other.

In example described below, we chose twelve patterns such that patterns of same class are not close to each other and are distributed randomly in the pattern space as shown in Fig. 3.4 (a). With these patterns we have compared the performance of FHLSNN and FMN algorithms. It is observed that the FMN al-
algorithm creates twelve hyperboxes compared to seven hyperline segments of FHLSNN algorithm to obtain 100% recognition rate as shown in Fig. 3.4 (b) and (c), respectively. When we ran the FMN algorithm by varying size of hyperbox to get 100% recognition efficiency for the selected twelve patterns, it is observed that the hyperboxes created are points, i.e. same as the patterns in the pattern space.

Figure 3.4: Performance comparison between FMN and FHLSNN algorithms with two class 2-D example (a) Distribution of twelve patterns in the 2-D space, • represents class 1 and ○ represents class 2, (b) Hyperboxes created by FMN algorithm, (c) Hyperline segments created by FHLSNN algorithm.
3.6 CONCLUSIONS

A neural network classifier that utilizes hyperline segments as fuzzy sets that are aggregated into fuzzy set classes is introduced. The proposed learning algorithm has the ability to learn in few passes through the data and can be adapted for real time applications.

The recognition rates obtained using ring features with FHLSNN algorithm are superior compared to FNN and MFNN algorithms, whereas the performance of FHLSNN and FMN algorithms is comparable. However, the performance of FHLSNN algorithm is better as compared to all other classifiers with Zernike features.

The probability of intersection of hyperline segments is very less if dimension of the feature space is moderately large, therefore the FHLSNN can be trained without intersection test and its removal. Hence, the FHLSNN algorithm outperforms all other classifiers as far as training time and recall time per pattern is concerned.

The performance of FNN and MFNN degrades relatively as the noise density increases, whereas FMN and FHLSNN perform well with moderate level of noise.

It is mentioned in section 3.5 that the distribution of patterns is random in the pattern space because characters of different writers vary in style and shape and selected feature extraction techniques gave only rotation invariancy. This statement is supported by the results summarized in Table 2.2, 3.1 and 3.2, where the number of output classes is close to the number of 1000 training patterns as compared to the actual 10 distinct output classes, each representing a distinct numeral. This indicates that the selected features are not suitable for generalization.

In such situations, proposed FHLSNN creates fewer hyperline segments since the probability of intersection is less. This also helps to improve the over-
all performance and to have slight edge over other classifiers, which is supported by the results in Table 3.6.

The recognition rates obtained for set-1 and set-2 are acceptable, whereas for set-3 and set-4, the recognition rates are degraded drastically. However, the results obtained for the multi-writer character sets are comparable as that of Chiu and Tseng. They have also used ring-data features and moment normalization for invariant handwritten Chinese character recognition. These results are listed in the first row of Table 6 in their paper.

Hence, the results obtained indicate that the proposed FHLSNN may recover the invariant problem with slight edge in performance but cannot solve high shape variation of handwritten Devanagiri numerals. To recognize such unconstrained handwritten characters invariantly, more powerful invariant features or recognition techniques are needed.