5.1 INTRODUCTION

Neural Networks have been used in many aspects of speech recognition. They have been used as phoneme classifier [95, 98], isolated word recognizers [45] and probability estimators of speech recognizers [12, 82, 85, 107]. In this chapter the multi-layer perceptron have been used to construct a speech recognizer to recognize the phonemes of Assamese and Bodo languages.

The most obvious way to use multi-layer perceptrons for speech recognition is to present all acoustic vectors of a speech unit (phoneme or word) at once at the input and detect the most probable speech unit at the output by determining the output neuron with highest activation. The problem associated with this approach is that a huge number of input units have to be used. This implies an evenly larger number of parameters are to be determined by learning and consequently the necessity to disposing of a large database.

In the present study, to reduce the volume of input data, clustering techniques have been used. The clustering of data advocates how the related data are categories into different classes. Two popular clustering techniques namely k-means clustering and Kohonen Self Organized Map (SOM) have been used in the present study and their performances have been compared. It has been observed that, when the
recognition accuracy is considered, SOM outperforms k-means. But k-means convergence time is many times smaller than SOM convergence time.

In the first section of this chapter, system description has been presented. The k-means algorithm, which is widely used for clustering of data has been described. Kohonen self organized map is then presented as a clustering technique.

In the second section of the chapter the theoretical background of Multi-layer perceptron has been presented and MLP based recognizer for the recognition of Assamese and Bodo phonemes has been constructed. Multi-layer perceptron with 108 input nodes, variable number of hidden nodes and layers and 53 output nodes has been constructed. The 53 output nodes corresponds to the 53 phonemes of Assamese and Bodo languages and 108 input nodes corresponds to 108 input taken from six clusters, which is the output of any one of the k-means or Kohonen algorithm. Feature vector clustered by both k-means and Kohonen has been considered as input to the recognizer.

Comparison between k-means and Kohonen approaches have been detailed in the third section. The same recognition platform and database has been used, fair comparison between the algorithms.

5.2 SYSTEM DESCRIPTION
5.2.1 K-MEANS CLUSTERING

K-means clustering is an algorithm to classify or to group objects based on their attributes (features) into k number of groups, where k is a positive integer. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid. Thus, the purpose of k-mean clustering is to classify
the data into \( k \) clusters. The \( k \)-mean clustering algorithm improves iteratively the codebook iteratively as follows:

1. Starting with a decision on the value of \( k \), the number of clusters
2. Taking the first \( k \) training sample as single-element clusters
3. Assigning each of the remaining \( (N-k) \) training samples to the cluster with the nearest centroid. After each assignment, recomputed the centroid of the gaining cluster.
4. Taking each sample in sequence and compute its distance from the centroid of each of the clusters. If a sample is not currently in the cluster with the closest centroid, switch this sample to that cluster and update the centroid of the cluster gaining the new sample and the corresponding cluster losing the sample.
5. Repeat step 4 until convergence is achieved, that is, until a pass through the training sample causes no new assignment.

5.2.2 SELF-ORGANIZED MAP (SOM)

Kohonen Self Organized Map (SOM) were initially introduced with the purpose of producing a special mapping from a high dimensional input space to a very low dimensional output space while conserving the topological information of the input space. SOM is a neural network trained by following a non-supervised algorithm. As illustrated on Fig – 5.1, the neural network is made up of two layers of neurons. The input (sensory) layer just distributes the inputs to the output layer. The output layer has \( M^p \) neurons, arranged on a \( p \)-dimensional lattice or map. Neurons on the output layer are characterized by:
1. a map coordinate \( k = (k_1, k_2, k_3, \ldots, k_p) \) with \( 1 \leq k_1, k_2, k_3, \ldots, k_p \leq M \), that locates the neurons on the map.

2. a synaptic weight vector \( w_k = (w_k^1, w_k^2, \ldots, w_k^m) \), where \( m \) is the neurons input dimension.

![Kohonen Self-Organized Map](image)

Fig (5.1): Kohonen Self-Organized Map

At a time \( n \), an \( m \)-dimensional input vector \( X = [x_1, x_2, x_3, \ldots, x_m]^T \) is presented to the input of the network. The following sequence of operations will have to be performed:

### 5.2.2.1 Competitive Process

The synaptic weight vector of each neuron in the network has the same dimension as the input space. The synaptic weight vector of neuron \( j \) is denoted by:

\[
W_j = [w_{j1}, w_{j2}, w_{j3}, \ldots, w_{jm}]^T, j=1,2,\ldots,l
\]

where \( l \) is the total number of neurons in the network. To find the best match of the input vector \( X \) with the synaptic weight vectors \( W_j \), the inner product \( W_j^TX \) for \( j=1,2,3, \ldots, l \) have been computed and neuron with highest inner product \( W_j^TX \) has been selected as the best match. Maximization of the inner product is taken as
mathematically equivalent to the minimization of the Euclidean distance between the vectors $X$ and $W_j$. Thus, if we use the index $i(x)$ to identify the neuron that best matches the input vector $X$, the following conditions is applied to identify $i(x)$:

$$i(x) = \arg \min_j \|x - w_j\|, \quad j = 1, 2, 3 \ldots, J$$

..... (5.2)

which sums up the essence of the competition process among the neurons. The particular neuron $i$, that satisfies this condition is called the best match neuron for a typical input vector $X$.

5.2.2.2 Cooperative Process

The winning neuron locates the centers of a topological neighbourhood. The key question at this point is — "how to define topological neighbourhood which is neurobiologically correct?" There is a neurobiological evidence of lateral interaction among a set of excited neurons. The probability that a neuron that is firing tends to excite the neurons in its immediate neighbourhood more than those farther away from it. This observation leads to make the topological neighbourhood around the winning neuron $i$ decaying smoothly with lateral distance. To be specific, let $h_{j,t}$ denote the topological neighbourhood centered on winning neuron $i$, and encompassing a set of excited (cooperative) neurons, a typical one denoted by $j$. Then the topological neighbourhood $h_{j,t}$ is a unimodel function of lateral distance $d_{j,t}$ such that it satisfies two distinct requirements:

i) The topological neighbourhood $h_{j,t}$ is symmetric about the maximum point defined by $d_{j,t}=0$. In other words, it attains its maximum value at the winning neuron $i$ for which the distance $d_{j,i}$ is zero.
ii) The amplitude of the topological neighbourhood $h_{j,i}$ decreases monotonically with increasing lateral distance $d_{j,i}$, decaying to zero for $d_{j,i} \to \infty$, which is a necessary condition for convergence. A typical choice of $h_{j,i}$ is a Gaussian Function as described by:

$$h_{j,i} = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2}\right)$$  \hspace{1cm} (5.3)

where parameter $\sigma$ is the "effective width" of the topological neighbourhood. It measures the degree to which excited neurons in the vicinity of the winning neuron participates in the learning process.

For cooperation among the neighbourhood neurons to be hold, it is necessary that topological neighbourhood $h_{j,i}$ must be dependent on lateral distance $d_{j,i}$ between winning neuron $i$ and excited neuron $j$ in the output space rather than an some distance measure on the original input space. This has been achieved through the Equation (5.3). In the case of one dimensional lattice, $d_{j,i}$ is an integer equal to $i - j$. On the other hand, in the case of two dimensional lattice, it is defined by

$$d_{j,i}^2 = \| r_j - r_i \|^2$$  \hspace{1cm} (5.4)

where the discrete vector $r_j$ defines the position of the excited neuron $j$ and $r_i$ defines the discrete position of the winning neuron, both measuring in discrete output space.

Another unique feature of the SOM algorithm is that the size of the topological neighbourhood shrinks with time. This requirement is satisfied by making the width $\sigma$ of the topological neighbourhood function $h_{j,i}$ decreases with time. A popular choice for the dependence of $\sigma$ on discrete time $n$ is the exponential delay described by equation (5.5)
\[ \sigma(n) = \sigma_0 \exp\left( -\frac{n}{\tau_1} \right), n = 0,1,2,\ldots \quad \ldots (5.5) \]

where \( \sigma_0 \) is the value of \( \sigma \) at the initiation of the SOM algorithm, and \( \tau_1 \) is the time constant. Correspondingly, the topological neighbourhood assumes a time-varying form of its own, as shown by equation (5.6)

\[ h_{j,i}(n) = \exp\left( -\frac{d_{ji}^2}{2\sigma^2(n)} \right), n = 0,1,2,\ldots \quad \ldots (5.6) \]

where \( \sigma(n) \) is defined by equation (5.5). Thus, as time \( n \) (i.e., the number of iteration) increases, the width \( \sigma(n) \) decreases at an exponential rate, and the topological neighbourhood shrinks in corresponding manner. Hence, \( h_{j,i}(n) \) is referred to as the neighbourhood function.

The neighbouring function provides a useful method for reducing the computer resource requirement for SOM training. The purpose of a width \( h_{j,i}(n) \) is to correlate the directions of the weight updating a large number of excited neurons in the lattice. As the width of \( h_{j,i}(n) \) is decreased, the number of neurons whose update directions are correlated also decreases. It is wasteful of the computer resources moving a large number of degrees of freedom around a winning neuron in a correlated fashion, as in the standard SOM algorithm. In the present study, a renormalized SOM form of training has been used. The training process started with much smaller number of normalized degree of freedom. This operation is performed in discrete form by having a neighbourhood function \( h_{j,i}(n) \) of constant width, but gradually increasing the total number of neurons. The new neurons are inserted halfway between the old ones, and the smoothness properties of the SOM algorithm guarantee that the new ones join the synaptic adaptation in a graceful manner.
5.2.2.3 Adaptive Process

The last process in the self organized formation of a feature map is a synaptic adaptation process. For the network to be self-organized, the synaptic weight vector \( w_j \) of the neuron \( j \) in the network is required to be modified in relation to the input vector \( x \). The Hebbian hypothesis\[40\] in its basic form is unsatisfactory due to the following reasons: change in connection occurs in one direction only, which finally drives all the synaptic weights into saturation. The Hebbian hypothesis is modified with the inclusion of forgetting term \( g(y_j)w_j \), where \( w_j \) is the synaptic weight vector of neuron \( j \) and \( g(y_j) \) is some positive scalar function of response \( y_j \) and

\[
g(y_j) = \begin{cases} 0 & \text{for } y_j = 0. \end{cases} \tag{5.7} \]

The change to the weight vector of neuron \( j \) in the lattice can be expressed as:

\[
\Delta w_j = \eta y_j x - g(y_j)w_j \tag{5.8} \]

where \( \eta \) is the learning rate parameter of the algorithm. The first term on the right-hand side of Eq. (5.8) is the Hebbian term and the second term is the forgetting term.

To satisfy the requirement of Eq. (5.7), linear function \( g(y_j) \) has been expressed as:

\[
g(y_j) = \eta y_j \tag{5.9} \]

Eq. (5.8) may be further simplified by setting

\[
y_j = h_{j,i(x)} \tag{5.10} \]

Using Eq. (5.9) and Eq. (5.10), Eq. (5.8) is expressed as:

\[
\Delta w_j = \eta h_{j,i(x)}(x-w_j) \tag{5.11} \]

Finally, using discrete-time formalism and taking the synaptic weight vector \( w_j(n) \) of neuron \( j \) at time \( n \), the updated weight vector \( w_j(n+1) \) at time \( (n+1) \) is defined by [57]

\[
\Delta w_j(n+1) = \Delta w_j(n) + \eta(n)h_{j,i(x)}(n)(x-w_j(n)) \tag{5.12} \]
which is applied to all the neurons in the lattice that lie inside the topological 
neighbourhood of winning neuron $i$. Eq. (5.12) has the effect of maximizing the 
synaptic weight vector $w_i$ of winning neuron $i$ towards the input vector $x$. Upon 
repeated application of training data, the synaptic weight vectors tend to follow the 
distribution of input vectors due to the neighbourhood updation. The algorithm 
therefore, leads to the topological ordering of the feature map in the input space in the 
sense that neurons adjacent to the lattice will tend to have similar synaptic weight 
vectors.

5.3 MULTILAYER PERCEPTRON BASED SPEECH RECOGNIZER

In the present study Multilayer Perceptron (MLP) has been used to design the 
speech recognizer to recognize the phonemes of Assamese and Bodo languages. The 
MLP consist of 108 input nodes, variable number of hidden nodes as well as layers 
and 53 output layers. To train the MLP, a modified version of well known Back 
Propagation Algorithm [40] has been used. To avoid the oscillations at the local 
minima a momentum constant has been introduced which provides optimization in the 
weight updating process. The algorithm is detailed below:

1) **Initialization**

The weights of each layer have been initialized to random number lies between -1 
to 1.

2) **Forward computation**

In the forward pass the synaptic weight remain unaltered throughout the network 
and *functional signal* of the network is computed neuron-by-neuron basis. The
induced local field \( v_j^{(l)}(n) \) for neuron \( j \) in layer \( l \) which is due to the functional signal produced by neurons of layer \( l-1 \) is given by [136]

\[
v_j^{(l)}(n) = \sum_{i=0}^{m} w_{ji}^{(l)}(n) y_{i}^{l-1}(n)
\]

... (5.13)

where \( m \) is the total number of inputs, excluding bias applied to neuron \( j \). The synaptic weight \( w_{ji} \) corresponds to fixed input \( y_{i}^{0} = 1 \), equals the bias \( b_j \) applied to neuron \( j \). Hence the functional signal appearing at the output neuron \( j \) of layer \( l \) is expressed as

\[
y_j^{(l)} = \psi_j(v_j^{(l)}(n))
\]

... (5.14)

If the neuron \( j \) is in the first hidden layer

\[
y_j^{(0)} = x_j(n)
\]

... (5.15)

where \( x_j(n) \) is the \( j^{th} \) element of the input vector. If on the other hand, network \( j \) is in the output layer of the network, and \( L \) the depth of the network, then

\[
y_j^{(L)} = o_j(n)
\]

... (5.16)

where \( o_j(n) \) is the \( j^{th} \) element of the output vector. The output is compared with the desired response \( d_j(n) \), obtain the error signal \( e_j(n) \) for the \( j^{th} \) output neuron

\[
e_j(n) = d_j(n) - o_j(n)
\]

... (5.17)

3) Backward computation

The backward pass starts at the output layer by passing the error signal leftward through the network, layer by layer, and recursively computing the \( \delta \) (i.e. the local gradient) for each neuron as follows:
\[ \delta_j^{(l)}(n) = \sum_k \delta_k^{(l+1)}(n)w_{jk}^{(l+1)}(n), \text{ for neuron } j \text{ in hidden layer } l \]

\[ \delta_j^{(l)}(n) = (v_j^{(l)}(n))X_{j} + \eta \delta_j^{(l)}(n)X^{(l-1)}(n) \text{, for neuron } j \text{ in output layer } L \]

The weight updation is taking place in accordance with the following rule -

\[ w_{jk}^{(l)}(n+1) = w_{jk}^{(l)}(n) + \alpha [w_{jk}^{(l)}(n-1)] + \eta \delta_j^{(l)}(n)X^{(l-1)}(n) \]

where \( \eta \) is the learning rate and \( \alpha \) is momentum constant.

It has been observed that MLP based speech recognizer work better if the input and output lies between 0 - 1. Therefore, the input vector has been normalized with respect to their maximum and minimum value.

A momentum constant \( \alpha \) has been used to avoid oscillation at the local minima.

The learning rate parameter has been changed gradually with each epoch number as expressed by equation (5.20)

\[ \eta(\text{epochNumber}) = \eta_0 \exp \left( - \frac{\text{epochNumber}}{100} \right) \]

where \( \eta_0 \) is the initial learning rate parameter.

### 5.4 EXPERIMENT

The feature vector, which is the output of the feature extraction block, has been normalized and then taken as the input for feature clustering block. The feature trajectory obtained from recorded speech of 1 second has been reduced into six clusters with 18 elements each, applying k-means and Kohonen SOM algorithm. The output of these clustering algorithms has been used as input to an MLP based speech recognizer. For the fair comparison of the algorithms, same architecture of the
recognizer is used for both the dataset. It has been observed that for the same dataset, the k-means algorithm convergence at 23 iterations whereas the Kohonen SOM required 513 iterations for convergence.

Initially an MLP based speech recognizer with one hidden layer has been constructed. The number of hidden nodes has been kept at 20 to provide 2160 connections between the input and hidden layer and 1080 connections between hidden and the output layer. The Enhance Back Propagation algorithm has been used to train the recognizer. Performance of the system is evaluated for both the dataset acquired from k-means and Kohonen SOM algorithm. Separate dataset is used for training and testing the system. For each phoneme, the performance of the system is evaluated with sample collected from 100 informants.

Gradually, the number of hidden nodes and layers has been increased. With the increasing number of hidden nodes and layers, better performance in terms of recognition accuracy has been obtained. But, with the increasing number of hidden layers and nodes, the time required for convergence is also increased. Considering all parameters, it has been noticed that MLP with 3-layers (two hidden and one output) give the optimal performance for the recognition of phoneme.

The feature vector which is the output of feature extractor block, for 1 sec utterance of the recorded Bodo vowel /a/ has been depicted in the Fig – (5.2). Fig – (5.3) presents the normalized feature vector for the utterance of the Bodo vowel /a/. The normalized feature vector has been clustered into six clusters using k-means and Kohonen SOM algorithm, which is shown in the Fig – (5.4) and Fig – (5.5) respectively. The architecture of the recognizers has been summarized in the Table – (5.1). Their convergence time has been shown in the Table – (5.2). The performance
of the recognizer in terms of the percentage of recognition accuracy has been shown in the Table (5.3) and Table – (5.4) for k-means input and Kohonen SOM input.

![Feature vector extracted from each frame](image)

**Fig 5.2:** 18-element Feature vector extracted from the all the frames of the utterance of the Bodo vowel /a/ which is the output of Feature extraction block
Normalized feature vector

Fig (5.3): Normalized feature vector extracted from all frames due to the utterance of the Bodo vowel /a/
Fig (5.4): Normalized feature vector extracted from all the frames of the utterance of the Bodo vowel /a/ has been clustered into 6 clusters of 18 using k-means clustering algorithm.
Fig (5.5): Normalized feature vector extracted from all the frames of the utterance of the Bodo vowel /a/ has been clustered into 6 clusters of 18 using Kohonen SOM algorithm
Table (5.1): Configuration of MLP based phoneme recognizer

<table>
<thead>
<tr>
<th>Type</th>
<th>Input Node</th>
<th>Output Node</th>
<th>Hidden node in the first layer</th>
<th>Hidden node in the second layer</th>
<th>Hidden node in the third layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>108</td>
<td>53</td>
<td>20</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>2</td>
<td>108</td>
<td>53</td>
<td>40</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>53</td>
<td>22</td>
<td>11</td>
<td>N/A</td>
</tr>
<tr>
<td>4</td>
<td>108</td>
<td>53</td>
<td>40</td>
<td>20</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>108</td>
<td>53</td>
<td>21</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>108</td>
<td>53</td>
<td>42</td>
<td>24</td>
<td>24</td>
</tr>
</tbody>
</table>

Table (5.2) : Convergence time of the recognizer in terms of number of iterations

<table>
<thead>
<tr>
<th>Recognizer Type</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convergence Time (in terms of iteration)</td>
<td>54</td>
<td>108</td>
<td>653</td>
<td>2160</td>
<td>8165</td>
<td>65319</td>
<td>8</td>
</tr>
</tbody>
</table>

Table (5.3): Result of the experiment for the recognition of Assamese and Bodo Phonemes with different types of MLP (100 experiment done for recognition of each phoneme). The dataset used for this experiment is the output of k-mean algorithm

<table>
<thead>
<tr>
<th>Recognizer Type -&gt; Language ↓</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assamese</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vowel</td>
<td>38.88</td>
<td>54.75</td>
<td>60.00</td>
<td>68.75</td>
<td>87.25</td>
<td>88.50</td>
<td></td>
</tr>
<tr>
<td>Consonant</td>
<td>26.88</td>
<td>30.25</td>
<td>32.38</td>
<td>55.50</td>
<td>62.75</td>
<td>67.50</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>32.88</td>
<td>42.50</td>
<td>46.19</td>
<td>62.125</td>
<td>75.00</td>
<td>78.00</td>
<td></td>
</tr>
<tr>
<td>Bodo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vowel</td>
<td>39.30</td>
<td>56.70</td>
<td>59.42</td>
<td>68.52</td>
<td>86.91</td>
<td>89.60</td>
<td></td>
</tr>
<tr>
<td>Consonant</td>
<td>25.89</td>
<td>31.53</td>
<td>33.41</td>
<td>56.90</td>
<td>61.67</td>
<td>67.21</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>32.60</td>
<td>44.12</td>
<td>46.415</td>
<td>62.71</td>
<td>74.29</td>
<td>78.41</td>
<td></td>
</tr>
</tbody>
</table>
Table (5.4): Result of the experiment for the recognition of Assamese and Bodo Phonemes with different types of MLP (100 experiment done for recognition of each phoneme). The dataset used for this experiment is the output of the Kohonen SOM algorithm.

<table>
<thead>
<tr>
<th>Recognizer Type → Language</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assamese</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vowel</td>
<td>40.12</td>
<td>55.37</td>
<td>61.24</td>
<td>71.70</td>
<td>87.66</td>
<td>89.21</td>
</tr>
<tr>
<td>Consonant</td>
<td>28.78</td>
<td>31.90</td>
<td>35.60</td>
<td>59.82</td>
<td>62.45</td>
<td>67.20</td>
</tr>
<tr>
<td>Average</td>
<td>34.45</td>
<td>43.64</td>
<td>48.42</td>
<td>65.76</td>
<td>75.06</td>
<td>78.21</td>
</tr>
<tr>
<td>Bodo</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vowel</td>
<td>40.55</td>
<td>57.34</td>
<td>65.64</td>
<td>71.46</td>
<td>87.32</td>
<td>90.32</td>
</tr>
<tr>
<td>Consonant</td>
<td>27.72</td>
<td>33.25</td>
<td>38.73</td>
<td>61.33</td>
<td>67.38</td>
<td>69.91</td>
</tr>
<tr>
<td>Average</td>
<td>34.14</td>
<td>45.30</td>
<td>52.19</td>
<td>66.39</td>
<td>77.35</td>
<td>80.12</td>
</tr>
</tbody>
</table>