Speech Recognition In Neuro-Fuzzy System

Pranavkumar N. Chaudhari
Assistant Professor
Department of Electrical Engineering
Parul Institute of Engineering & Technology

Abstract

Neural networks are excellent classifiers, but performance is dependent on the quality and quantity of training samples presented to the network. In cases where training data is sparse or not fully representative of the range of values possible, incorporation of fuzzy techniques improves performance. That is, introducing fuzzy techniques allow the classification of imprecise data. The neuro-fuzzy system presented in this study is a neural network that processes fuzzy numbers. By incorporating this attribute, the system acquires the capacity to correctly classify imprecise input. Experimental results show that the neuro-fuzzy system’s performance is vastly improved over a standard neural network for speaker-independent speech recognition. Speaker independent speech recognition is a particularly difficult classification problem, due to differences in voice frequency (amongst speakers) and variations in pronunciation. The network developed in this study has an improvement of 45% over the original multi-layer perceptron used in a previous study. Keywords: Neural network, fuzzy system, Speech Recognition

I. INTRODUCTION

Neural network performance is directly related to the size and quality of training samples [1]. When the number of training pairs is small, or perhaps not representative of the possibility space, standard neural network results are predictably poor. Incorporation of fuzzy techniques can improve performance in these cases [2]. Even though standard neural networks are excellent classifiers, introducing fuzzy techniques allow them to classify imprecise data. The Neuro-fuzzy system is presented is a standard neural network modified to process fuzzy numbers [3, 4].

Processing fuzzy numbers can be accomplished in a variety of ways. One of the most elegant methods, because of its simplicity, is by using interval mathematics. The Neuro-fuzzy system, then, is a standard feed-forward neural network that has been modified to deal with fuzzy numbers via interval mathematics. The modifications that are entailed are basically generalizations in the learning rule and neuronal functions in order to accommodate the interval mathematics. The exact modifications are described in the next section, with the simulation results detailed in section 3 and conclusions discussed in section 4.

II. THE NEURO-FUZZY SYSTEM

A. Data Representation and Processing:

This Neuro-fuzzy system is based on the neural network described by Ishibushi et al. [5]. It was originally presented as a neural network that learned from fuzzy If-Then rules. This network configuration can be used in several ways, the key to which is taking α-cuts of the fuzzy number in question and utilizing interval mathematics. Specifically, α-cuts of the fuzzy numbers are represented by interval vectors [4]. That is, an α-cut of fuzzy input vector is represented by the interval vector $X_p = (X_{p1}, X_{p2}, ..., X_{pn})^T$ where

$$X_{pi} = [x_{piL}, x_{piU}]$$

indicate the lower and upper limits of the interval. Thus, the input and output vectors are interval vectors, and the neuronal operations are modified to deal with the interval numbers. Specifically, summation of weighted inputs is carried out as

$$\text{Net}_{pj}^L = \sum_{w_{ji} \geq 0} w_{ji} o_{piL}^j + \sum_{w_{ji} < 0} w_{ji} o_{piU}^j + \theta_j$$

and

$$\text{Net}_{pj}^U = \sum_{w_{ji} \geq 0} w_{ji} o_{piU}^j + \sum_{w_{ji} < 0} w_{ji} o_{piL}^j + \theta_j.$$ (3)

These calculations are consistent with the interval multiplication operation described in Alefeld [3]. The output equation can then be expressed as

$$o_{pj} = [o_{pjL}, o_{pjU}]$$

$$= [f(\text{Net}_{pj}^L), f(\text{Net}_{pj}^U)]$$ (4)
**B. Derivation of the Learning Rule**

The learning rule for this network must then be modified accordingly. Succinctly stated, the Generalized Delta (backpropagation with momentum) rule to update any weight is

$$\Delta w_{ji}(t+1) = \eta \left( \frac{\partial E_p}{\partial w_{ji}} \right) + \alpha \Delta w_{ji}(t).$$  \hspace{1cm} (5)

The error is computed as the difference between the target output, \( t_p \), and the actual output, \( o_p \):

$$E_p = \max \left\{ \frac{1}{2} (t_p - o_p)^2, \quad o_p \leq o_p \right\},$$  \hspace{1cm} (6)

where

$$\begin{align*}
(t_p - o_p), & \quad \text{if } t_p = 1, \text{ and} \\
(t_p - o_p), & \quad \text{if } t_p = 0.
\end{align*}$$  \hspace{1cm} (7)

For units in the output layer, calculation of \( \frac{\partial E_p}{\partial w_{ji}} \) is straightforward, and can be thought of as four cases based on the value of target output and weight. Note that in the four equations the value of \( j \) in the subscript is fixed (to the output neuron that had the maximum error). In the first case, the \( t_p = 1 \), and \( w_{ji} \geq 0 \):

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \left[ \frac{(t_p - o_p)^2}{2} \right],$$  \hspace{1cm} (8)

The third line in the derivation assumes that the neuronal activation function, \( f \) (Net), is the binary sigmoid function, and thus substitutes the values accordingly. The other cases are evaluated with \( t_p = 1 \), and \( w_{ji} < 0 \), \( t_p = 0 \), and \( w_{ji} \geq 0 \), and \( t_p = 0 \), and \( w_{ji} < 0 \). This results in four output layer and eight hidden layer equations. The four output layer equations are summarized as follows:

$$\begin{align*}
\frac{\partial E_p}{\partial w_{ji}}, & \quad \text{for } t_p = 1, \text{ and } w_{ji} \geq 0, \\
\frac{\partial E_p}{\partial w_{ji}}, & \quad \text{for } t_p = 1, \text{ and } w_{ji} < 0.
\end{align*}$$  \hspace{1cm} (9)

where

$$\delta_{pj}^L = (t_p - o_p^L) \cdot o_p^L \cdot (1 - o_p^L) \text{ and } \delta_{pj}^U = (t_p - o_p^U) \cdot o_p^L \cdot (1 - o_p^U).$$

The calculation of the partial derivative \( \frac{\partial E_p}{\partial w_{ji}} \) for the hidden layers is based on back-propagating the error as measured at the output layer. The following discussion assumes one hidden layer, although subsequent terms could be derived in the same way for other hidden layers. Since this derivation involves a path of two neurons (output and hidden layer), there are eight cases. For the first case, \( t_p = 1 \), \( w_{kj} \geq 0 \) and \( w_{ji} \geq 0 \), where \( w_{kj} \) is the weight on the path from hidden layer neuron \( j \) to output layer neuron \( k \), \( k \) fixed:

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \left[ \frac{(t_{pk} - o_{pk})^2}{2} \right],$$  \hspace{1cm} (10)

The second case is a variation on the first, in which \( t_p = 1 \), \( w_{kj} \geq 0 \) and \( w_{ji} < 0 \), so the partial derivative becomes:
\[ \frac{\partial E_p}{\partial w_{ji}} = -\delta_{pk}^L \cdot w_{kj} \cdot o_{pj}^U \cdot (1 - o_{pj}^U) \cdot o_{pi}^U. \] 

The third case is characterized by \( t_p = 1, w_{kj} < 0 \) and \( w_{ji} \geq 0 \):

\[ \frac{\partial E_p}{\partial w_{ji}} = -\delta_{pk}^L \cdot w_{kj} \cdot o_{pj}^U \cdot (1 - o_{pj}^U) \cdot o_{pi}^U. \] 

\( t_p = 1, w_{kj} < 0 \) and \( w_{ji} < 0 \) for the fourth case:

\[ \frac{\partial E_p}{\partial w_{ji}} = -\delta_{pk}^L \cdot w_{kj} \cdot o_{pj}^U \cdot (1 - o_{pj}^U) \cdot o_{pi}^L. \] 

The fifth through eighth cases deal with a target value of 0. For the fifth case, \( t_p = 0, w_{kj} \geq 0 \) and \( w_{ji} \geq 0 \):

\[ \frac{\partial E_p}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} \left[ \frac{1}{2} (t_{pk} - o_{pk}^U)^2 \right] \]

\[ = \frac{\partial}{\partial o_{pk}^U} \left[ \frac{1}{2} (t_{pk} - o_{pk}^U)^2 \right] \cdot \frac{\partial o_{pk}^U}{\partial c_{Net_{pk}^U}} \cdot \frac{\partial c_{Net_{pk}^U}}{\partial w_{ji}} \]

\[ = -\delta_{pk}^U \cdot w_{kj} \cdot o_{pj}^U \cdot (1 - o_{pj}^U) \cdot o_{pi}^U. \] 

For the sixth case:

\[ t_p = 0, w_{kj} \geq 0 \) and \( w_{ji} < 0 \):

\[ \frac{\partial E_p}{\partial w_{ji}} = -\delta_{pk}^U \cdot w_{kj} \cdot o_{pj}^U \cdot (1 - o_{pj}^U) \cdot o_{pi}^L. \] 

In the seventh case, \( t_p = 0, w_{kj} \geq 0 \) and \( w_{ji} < 0 \):

\[ \frac{\partial E_p}{\partial w_{ji}} = -\delta_{pk}^U \cdot w_{kj} \cdot o_{pj}^U \cdot (1 - o_{pj}^U) \cdot o_{pi}^L. \] 

The eighth and final case has \( t_p = 0, w_{kj} < 0 \) and \( w_{ji} < 0 \):

\[ \frac{\partial E_p}{\partial w_{ji}} = -\delta_{pk}^U \cdot w_{kj} \cdot o_{pj}^U \cdot (1 - o_{pj}^U) \cdot o_{pi}^U. \]

Equations (9) through (17) are used to code the training function in the simulation. Simulation results are presented in the next section.

### III. Implementation and Experimental Results

The implementation of the neuro-fuzzy network was carried out as a simulation. Using the equations derived in the previous section, the simulation was coded in the C programming language and tested with several classic data sets. These results proved favorable, with comparable or improved performance over a standard neural network [6].

#### A. Vowel Recognition Data

The application problem that will serve as a testbench for this study is Vowel, one of a collection of data sets used as a neural network benchmarks [7]. It is used for speaker-independent speech recognition of the eleven vowel sounds from multiple speakers. The Vowel data set used in this study was originally collected by Deterding [8], who recorded examples of the eleven steady state vowels of English spoken by fifteen speakers for a “non-connectionist” [7] speaker normalization study. Four male and four female speakers were used to create the training data, and the other four male and three female speakers were used to create the testing data. The actual composition of the data set consists of 10 inputs, which are obtained by sampling, filtering and carrying out linear predictive analysis.

Specifically, the speech signals were low pass filtered at 4.7 kHz and digitized to 12 bits with a 10 kHz sampling rate. Twelfth order linear predictive analysis was carried out on six 512 sample Hamming windowed segments from the steady part of the vowel. The reflection coefficients were used to calculate 10 log area parameters, giving a 10 dimensional input space [7]. Each speaker thus
yielded six frames of speech from eleven vowels. This results in 528 frames from the eight speakers used for the training set, and 462 frames from the seven speakers used to create the testing set.

528 samples is relatively small training set (in standard neural network applications), the values are diverse and thus Vowel is an excellent testbench for the neuro-fuzzy system.

B. Performance Results

As stated before, speaker-independent voice recognition is extremely difficult, even when confined to the eleven vowel sounds that Vowel consists of. Robinson carried out a study comparing performance of feed-forward networks with different structures using this data, which serve as a baseline for comparisons. The best recognition rate for a multi-layer perceptron reported in that study is 51% [9]. In spite of its difficulty, other studies have also used the Vowel data set. For various types of systems, the reported recognition rates (best case) are 51% to 59% [9, 10, 11]. The recognition rate obtained with the neuro-fuzzy system is 89%. Results of the simulation are summarized in Table 1, shown below.

A recognition rate of 89% surpassed expectations, especially with a data set as diverse as the speaker-independent speech (vowel recognition) problem.

Table 1: Best Recognition Rates for Standard Neural Networks and the Neuro-Fuzzy System

<table>
<thead>
<tr>
<th>Type of Network</th>
<th>Number of Hidden Neurons</th>
<th>Best Recognition Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Std. Neural Network</td>
<td>11</td>
<td>59.3</td>
</tr>
<tr>
<td>Std. Neural Network</td>
<td>22</td>
<td>58.6</td>
</tr>
<tr>
<td>Std. Neural Network</td>
<td>88</td>
<td>51.1</td>
</tr>
<tr>
<td>Neuro-Fuzzy System</td>
<td>11</td>
<td>88.7</td>
</tr>
</tbody>
</table>

IV. Conclusions

Fuzzy theory has been used successfully in many applications [12]. This study shows that it can be used to improve neural network performance. There are many advantages of fuzziness, one of which is the ability to handle imprecise data. Neural networks are known to be excellent classifiers, but their performance can be hampered by the size and quality of the training set. By combining some fuzzy techniques and neural networks, a more efficient network results, one which is extremely effective for a class of problems. This class is characterized by problems with its training set. Either the set is too small, or not representative of the possibility space, or very diverse. Thus, standard neural network solution is poor, because the training sequence does not converge upon an effective set of weights. With the incorporation of fuzzy techniques, the training converges and the results are vastly improved.

One example of the problem class which benefits from neuro-fuzzy techniques is that of speaker-independent speech recognition. The Vowel data set was used in simulation experiments, as reported in the previous section. This data set is well known and has been used in many studies, and has yielded poor results. This is true to such a degree that it caused one researcher to claim that “poor results seem to be inherent to the data” [10]. This difficulty with poor performance adds more credence to the effectiveness of the fuzzy techniques used in the neuro-fuzzy system.

In summation, the neuro-fuzzy system outperforms standard neural networks. Specifically, the simulations presented show that the neuro-fuzzy system in this study outperforms the original multi-layer perceptron study by 45% on the difficult task of vowel recognition.

REFERENCES