The Performance Evaluation of Speech Recognition by Comparative Approach

R.L.K. Venkateswarlu, R. Raviteja and R. Rajeev

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/50640

1. Introduction

For more than five decades, Automatic speech recognition has been the area of active research. This problem was thoroughly studied with the advent of digital computing and signal processing. Increased awareness of the advantages of conversational systems led to the development of this problem. Speech recognition has wide applications and includes voice-controlled appliances fully featured speech-to-text software, automation of operator-assisted services and voice recognition aids for the handicapped.

There are four main approaches in speech recognition:

The acoustic –phonetic approach, The pattern recognition approach, The artificial intelligence approach and Neural network approach. Hidden Markov Model(HMM) and Gaussian Mixture models are also adopted in ASR.

Speech recognition has a big potential in becoming an important factor of interaction between human and computer in the near future. A successful speech recognition system has to determine features not only present in the input pattern at one point in time but also features of the input pattern changing over time (Berthold, M.R, Benyettou). In the speech recognition domain, the first model used by weibel is based on multilayer perceptron using Time Delay Neural network. But this model was the hard time processing. For new applications, the adjustment of parameters became a laborious stain.

RBF networks don't require a special adjustment and with regard to the Time delay Neural network the training time becomes shorter. But RBF problem is the shift invariant in time [Berthold,M.R].

The NN approach for SR can be divided into two main categories: Conventional neural networks and recurrent neural networks. The main rival to the multilayer perceptron is RBF which is becoming an increasingly popular neural network with diverse applications. Traditional statistical pattern classification techniques became inspiration for RBF networks.

In RBF network, process is performed in the hidden layer which is its unique feature. The patterns in the input space form clusters. The distance from the cluster center can be measured if the centers of these clusters are known. Further this distance measure is made non-linear so that it gives a value close to 1 if a pattern is in an area that is close to a cluster center. Serious rivals to MLP are RBF networks which are statistical feed-forward networks. The learning mechanisms in these networks are not biologically plausible- they have not been taken by some researchers who insist on biological anologies.

2. Present work

In the dependent mode, the sequence of sound units from the speech signal can be determined so that the linguistic message in the form of text can be decoded from the speech signal.

The steps used in the present speech recognition system are discussed below:

2.1. Speech dataset design

To design the system, five different speech datasets were used that consists of speech of both male and female. The recordings which are used in this work, were made with 8 bits and 11 KHz via a headset microphone in a closed room. The following speech dataset is chosen for speech recognition system.

The basic unit for sentence parsing and understanding is word. In order to identify the sentences, the words must be identified properly. The following dataset which consists of words that are made up of both consonants and vowels is chosen. The dataset composed of 6 words is tabulated in Table 1.

Words	passion	galaxy	marvellous
Pronunciation	pas·sion	gal·ax·y	mar·vel·ous
Words	manifestation	almighty	pardon
Pronunciation	man·i·fes·ta·tion	al·might·y	par·don

Table 1. Words with pronunciation

2.2. Speech database design

Speech Database Six different speakers (2 male and 4 female) are allowed to utter 6 words of speech dataset 4. So the speech database consists of 216 .wav files.

2.3. Preprocessing

Using good quality recording equipment, the speech signals are recorded in a low noise environment. The signals are sampled at 11KHz frequency. When the input data is surrounded by silence, reasonable results can be achieved in isolated word recognition.

2.4. Speech processing

The speech signal also contains data that is unnecessary like noise and non speech, which need to be removed before feature extraction. The resulting speech signals will be passed through an endpoint detector to determine the beginning and the ending of a speech data.

2.5. Sampling rate

Samples at sampling rate 11KHz are chosen to represent all speech sounds.

2.6. Windowing

A window length of 0.015 is selected for the given 12Mel frequency coefficients for time 0.005 seconds by the Praat object software tool.

2.7. Soft signal

The analog signal captured is converted into a smoothened analog signal by speech preprocessing which can be readily accessed by machine.

2.8. Front – End analysis

The speech signal exists as pressure variations in air. These pressure variations are converted into an electric current related to pressure by using microphone. In humans, the pressure variations are converted by ear into a series of nerve impulses which are then transmitted to the brain. For speech recognition task, selection of features is very important. For achieving good results, good features are required. Proper identification of features for speech recognition task and a strategy to extract these features from speech signal is the basic problem in speech recognition.

2.9. Feature extractor

The object of feature extractor is to transform an input in the signals space to an output in a feature space by using priori knowledge in order to achieve some desired criteria (Rabiner, 1993). The feature extractor (FE) block is designed in speech recognition in order to reduce the complexity of the problem before the next stage starts to work with data. Its aim is to use a priori knowledge to transform an input in the signal space to an output in a feature space to achieve some desired criteria. If lots of clusters present in a high dimensional space are to be classified, then FE transforms that space such that classification becomes easier.

2.9.1. LPCC analysis

Linear predictive coding (LPC) is a tool used in audio signal processing and also for speech processing which represents the spectral envelope of a digital speech signal in compressed form with the information of a linear predictive mode. The speech signal is analyzed by LPC by estimating the formants, their effects are removed and intensity and frequency of the remaining buzz are estimated. The process of removing the formants is known as inverse filtering. After the subtraction of the filtered modeled signal, the remaining signal is called the residue. This residue signal, the formants and the number which represents the intensity and frequency of the buzz are stored or transmitted elsewhere. The speech signal is synthesized by LPC by reversing the process using the buzz parameters and the residue in order to produce a source signal. This process is done on short chunks of the speech signal, as it varies with time, which are called frames. 30 to 50 frames per second are required for intelligible speech with good compression. LPC is one of the most powerful speech analysis techniques. LPC is the most useful method for encoding good quality speech at a low bit rate. It provides extremely accurate estimates of speech parameters. Since a very small error can distort the whole spectrum, LPC has to be tolerant of transmission errors. A small error might make the prediction filter unstable.

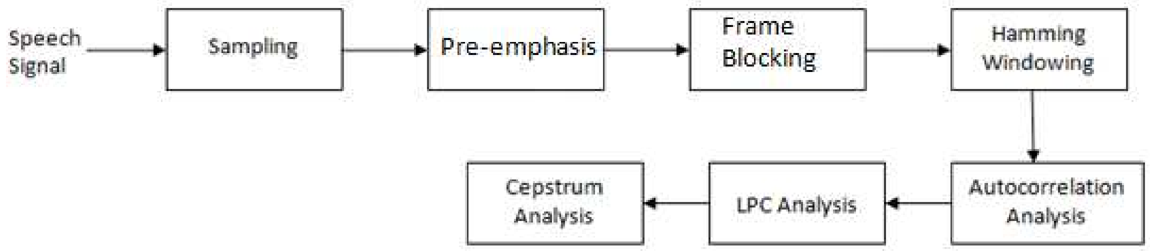


Figure 1. Feature Extractor schematic diagram for LPCC analysis

2.9.1.1. Speech sampling

Higher sampling frequency or more sampling precision is needed to achieve higher recognition accuracy. Commercial speech recognizers typically use comparable parameter values and achieve impressive results. It is not necessary to increase the sampling rate beyond 11,025 Hz or the sampling precision higher than 8 bits.

2.9.1.2. Pre-emphasis

The speech waveform which is digitized has a high dynamic range and it suffers from additive noise. So pre-emphasis is applied to spectrally flatten the signal so as to make it less susceptible to finite precision effects in the processing of speech.The most widely used pre-emphasis is the fixed first-order system. The calculation of pre-emphasis is shown as follows.

$$H(z) = 1 - az^{-1} \quad 0.9 \le a \le 1.0$$

The most common value for a is 0.95 (Deller et at; 1993). A Pre-Emphasis can be expressed as

$$\hat{s}(n) = s(n) - 0.95\, s\,(n-1)$$

2.9.1.3. Frame blocking

The speech signal is dynamic or time-variant in the nature. According to Rabiner (1993), the speech signal is assumed to be stationary when it is examined over a short period of time. In order to analyze the speech signal, it has to be blocked into frames of N samples, with adjacent frames being separated by M samples. If $M \leq N$, then LPC spectral estimates from frame to frame will be quite smooth. On the other hand if $M > N$ there will be no overlap between adjacent frames.

2.9.1.4. Windowing

Each frame is windowed in order to minimize the signal discontinuities or the signal is tapered to zero at the starting and ending of each frame.If window is defined as w(n), then the windowed signal is

$$\tilde{x}(n) = x(n)w\,(n), 0 \leq n \leq N-1$$

A typical window used is the Hamming window, which has the form

$$w(n) = 0.54 - 0.46\cos\left(\frac{2n}{N-1}\right), 0 \leq n \leq N-1$$

The value of the analysis frame length N must be long enough so that tapering effects of the window do not seriously affect the result.

2.9.1.5. Autocorrelation analysis

Auto correlation analysis can be used to find fundamental frequency or a pitch of the signal. It can also be used for finding repeating patterns in a signal or identifying the missing fundamental frequency. The technique relies on finding the co-relation between the signal and a delayed version of itself. Each frame of windowed signal is next auto correlated to give

$$R(n) = \sum_{x=0}^{N-1-m} \tilde{x}\,(n)\,\tilde{x}(n+m), m = 0,1,2,\dots,P.$$

Where, the highest autocorrelation value, P is the order of the LPC analysis. The selection of p depends primarily on the sampling rate

2.9.1.6. LPC analysis

The next processing step is the LPC analysis which converts each frame of autocorrelation coefficients R into the LPC parameters. The LPC parameters can be the LPC coefficients. This method of converting autocorrelation coefficients to LPC coefficients is known as Durbin's method. Levinson-Durbin recursive algorithm is used for LPC analysis.

$$E_0 = R(0)$$

$$k_i = \left[R(i) - \sum_{j-1}^{i-1} a_j^{i-1} R(i-j)\right] / E_{i-1,}\ 1 \leq i \leq p$$

$$a_i^i = k_i$$

$$a_i^i = a_i^{i-1} - k_i a_{i-j}^{i-1}, 1 \leq j \leq i-1$$

$$E_i = (1 - k_1^2)E_{i-1}$$

The above set of equations is solved recursively for i = 1,2, p, where p is the order of the LPC analysis. The k_i are the reflection or PARCOR coefficients. The a_j are the LPC coefficients. The final solution for the LPC coefficients is given as

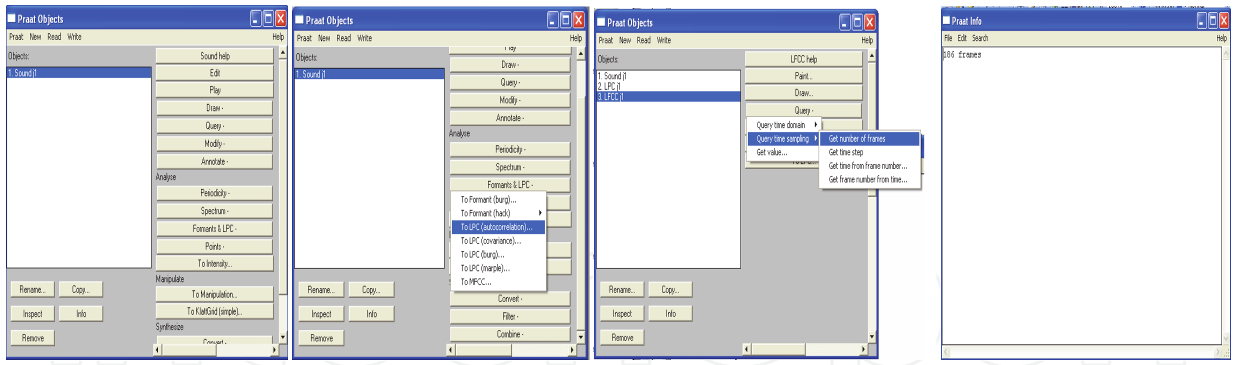
$$a_j = a_j^{(p)}, 1 \leq j \leq P$$

2.9.1.7. Cepstrum analysis

The parametric representations are divided into two groups- those based on the Fourier spectrum and those based on the linear prediction spectrum. The first group comprises the mel-frequency cepstrum coefficients (MFCC) and the linear frequency cepstrum coefficients (LFCC). The second group includes the linear prediction coefficients (LPC), Reflection coefficients (RC) and cepstrum coefficients derived from the linear prediction coefficients (LPCC). Linear Predictive Cepstral coefficients are extension to LPC.

2.9.1.8. Implementation of LPCC

After capturing the speech for the speech datasets with the microphone, the speech signals are captured in .wavfiles (sound objects), these sound objects are converted into mono to get mono sound objects. These mono objects are re-sampled at 11kHz with precision of 50 samples. These objects are subject to prediction order of 16, Window length is chosen as 0.025 with time steps 0.005 and pre-emphasis frequency is maintained at 50 kHz. The LPC (autocorrelation) co-efficients are converted into Linear frequency cepstral co-efficients by using Praat object software tool.



```
286 frames
```

Figure 2. LPCC Feature Extraction for Sound object

2.9.2. MFCC analysis

The Mel frequency Cepstral co-efficients (MFCC) are the most widely used features in speech recognition today. The Mel scale was developed by Stevens and Volkman [1940] as a result of a study of the human auditory perception. It was used by Mermelstein and Davis [1980], to extract features from the speech signal for improved recognition accuracy. MFCC'S are one of the more popular parameterization methods used by researchers in the speech technology. It is capable of capturing the phonetically important characteristics of speech. The coefficients are largely independent, allowing probability densities to be modeled with the diagonal co-variances matrices. Mel scaling has been shown to offer better discrimination between phones, which is an obvious help in recognition. It has good discriminating properties.

MFCC'S are based on the known variation of the human ears critical band widths with frequency to capture the phonetically important characteristics of speech, filters spaced linearly at low frequencies and logarithmically at high frequencies have been used. This is expressed in the mel-frequency scale. Mel –frequency scale is linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000Hz. The process of computing MFCC is shown in figure.

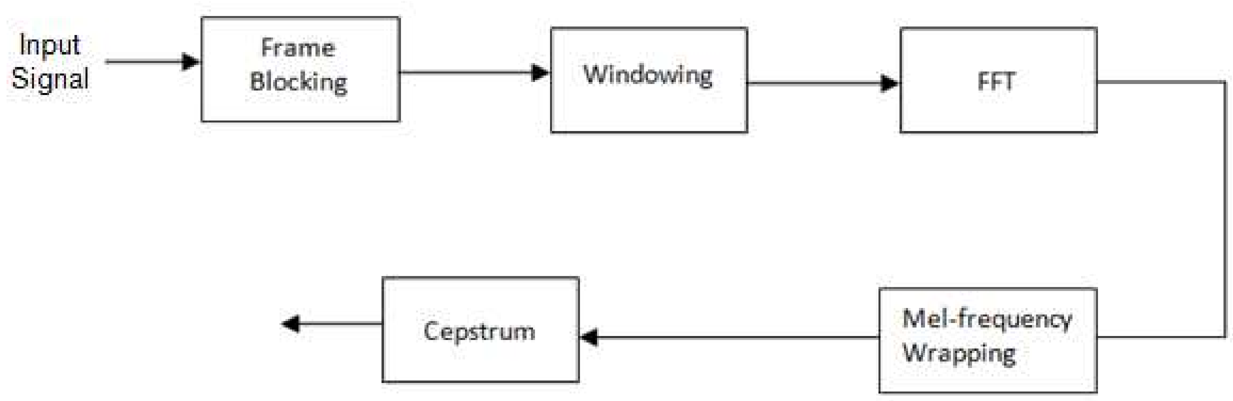


Figure 3. Feature Extractor schematic diagram for MFCC analysis

2.9.2.1. Blocking of frames

In this step, the continuous speech signal is blocked into frame of N samples with adjacent frames being separated by M (M<N). The first frame consists of the first N samples. The second frame begins M samples after the first frame and overlaps it by N-M samples. Similarly third frame begins 2M samples after the first frame and overlaps it by N-2M samples. This process continues until all the speech is accounted for within one of more frames.

2.9.2.2. Windowing of speech signal

The next step is to window each frame. Each frame is windowed in order to minimize the signal discontinuities at the starting and ending of the frame. Also spectral distortion is minimized by using the window for tapering the signal to zero at he beginning and ending of each frame. If window is defined as w(n), 0<n<N, where N is the number of samples in each frame then the result of windowing is

$$\mathrm{y}_1(\mathrm{n}) = \mathrm{x}_1(\mathrm{n})\mathrm{w}(\mathrm{n}), 0 < n < N - 1$$

Typically the hamming window has the form

$$\mathrm{w}(\mathrm{n}) = 0.54 - 0.46\cos\,(2\pi\mathrm{n}/\mathrm{N} - 1), 0 < N < N - 1$$

2.9.2.3. Fast Fourier transform (FFT)

Fast fourier transform converts each frame of N sample from the time domain into the frequency domain. The FFT is a fast algorithm to implement the discrete Fourier Transform (DFT) which is defined on the set of N samples (x_n).

2.9.2.4. Mel-frequency warping

It is shown that human perception of the frequency contents of sound for speech signals does not follow a linear scale. The actual frequency 'f' measured in Hz a subjective pitch is measured on a scale called the mel scale. The mel scale is a linear frequency spacing below 1000Hz and a logarithmic spacing above 1000Hz. The approximate formula to compute the mels for a given frequency f in kHz is mel (F) = 2595 x log 10 (1+ f/700).

The MFCC analysis is carried out in two steps.

2.9.2.5. Filter bank analysis on Mel frequency scale

Mel Filter is an object that represents an acoustic time- frequency representation of a sound. The power spectral density $P(f,t)$ expressed in dBs is sampled into a number of points around equally spaced times t_i and frequencies f_i on a mel frequency scale. A mel filter object is created for every selected sound object by band filtering in the frequency domain as follows:

Divide a sound into frames according to a certain window length and time step. A filter bank is simulated with N filters. Apply the following algorithm for every selected sound object with a bank of filters.

Step 1. Apply a Gaussian window to the sound frame
Step 2. Convert the windowed frame into a Spectrum object
Step 3. Convert the spectral amplitudes to energy values by squaring the real and imaginary parts and multiplying by df, the frequency distance between two successive frequency points in the spectrum. Since the Spectrum object only contains positive frequencies, multiply all energy values, except the first and the last frequency, by another factor of 2 to compensate for negative frequencies
Step 4. For each of the N filters in the filter bank: determine the inner product of its filter function with the energies as determined in the previous step. The result of each inner product is the energy in the corresponding filter
Step 5. Convert the energies in each filter to power by dividing by the window length
Step 6. Correct the power, due to the windowing of the frame, by dividing by the integral of the squared windowing function
Step 7. Convert all power values to dB's according to 10 * log10 (power / 4 10-10)

Filter functions which are in triangular shape are used on a linear frequency scale. These functions depends on three parameters the lower frequency f_l, the central frequency f_c and the higher frequency f_h.The distances $f_c - f_l$ and $f_h - f_c$ are the same for each filter and are equal to the distance between the f_c's of successive filters on mel scale.

$$H(f) = 0 \text{ for } f \le f_l \text{ and } f \ge f_h$$

$$H(f) = \frac{(f - f_i)}{(f_c - f_i)} \text{for } f_l \le f \le f_c$$

$$H(f) = (f_h - f)/(f_h - f_c) \text{ for } f_c \le f \le f_h$$

2.9.2.6. Cepstrum

The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given frame analysis. The Mel filter values are converted as Mel frequency cepstral coefficients in this stage.

This transformation was first used by Davis and Mermelstein (1980). From the discrete cosine transform of the filter bank spectrum (in dB), μ frequency cepstral coefficients are obtained by using the following relation

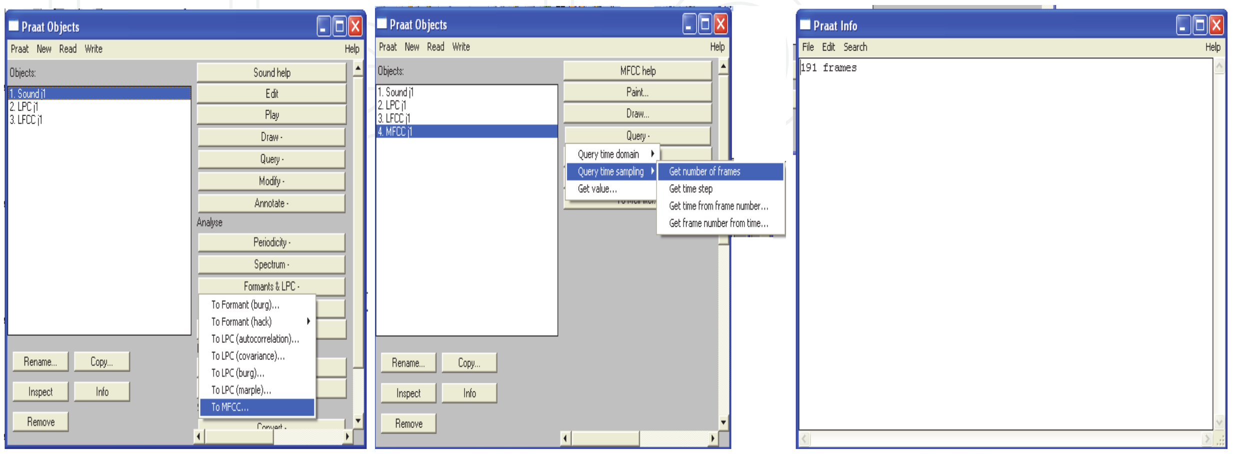
$$c_i = \sum_{j=1}^{N} P_j \cos(i\pi/N(j - 0.5))),$$

Where N represents the number of filters and P_j the power in dB in the j^{th} filter.

2.9.2.7. Implementation of MFCC

After capturing the speech for the speech dataset with the microphone, the speech signals are captured in .wav files (sound objects). These sound objects are converted into mono to get mono sound objects. These objects are applied to MFCC with 12 coefficients, window length 0.015, time step 0.005, the filter – bank parameters are chosen as follow:

The coefficient of 1st filter is taken as 100 mels the distance between filters is maintained as 100 mels.



```
191 frames
```

Figure 4. MFCC Feature Extraction for Sound object

3. Recognition methodology

In multi-class mode, each classifier identifies whether the set of input features which are derived from the current signal, belongs to a specific class of numbers or not. Also it tries to identify exactly to which class it belongs.

3.1. Classifiers used

3.1.1. Multilayer perceptron architecture

The demonstration of the limitations of single-layer neural neural networks was a significant factor in the decline of interest in neural networks in the 1970's. The discovery (by several researches independently) and wide spread dissemination of an effective general method of training a multilayer neural network [Rumelhart, Hinton, & Williams, 1986a, 1986b; McClelland & Rumelhart, 1988] played a major role in the reemergence of neural networks as a tool for solving a wide variety of problems. In this, we shall discuss this training method, known as backpropagation (of errors) or the generalized delta rule. It is simply a gradient descent method to minimize the total squared error of the output computed by the net. According to Laurene Fausett, the training of a network by a backpropagation involves three stages: the feed forward of the input training pattern, the calculation and backpropagation of the associated error, and the adjustment of the weights. After training, application of the net involves only the computations of the feed forward phase. Even if training is slow, a trained net can produce its output very rapidly. Numerous variations of backpropagation have been developed to improve the speed of the training process. A multilayer neural network with one layer of hidden units (the Z units) is shown in figure. The output units (the Y units) and the hidden units also may have biases (as shown). The biases on a typical output unit Y_k is denoted by $w0_k$; the bias on a typical hidden unit Zj is denoted v0j. These bias terms act like weights on connections from units whose output is always 1. (These units are shown in Figure but are usually not displayed explicitly.) Only the direction of information flow for the feed forward phase of operation is shown. During the backpropagation phase of learning, signals are sent in the reverse direction.

3.1.1.1 Backpropagation algorithm

According to Laurene Fausett, during feed forward, each input unit (X_i) receives an input signal and broadcasts this signal to the each of the hidden units Z_1,....,Z_p. Each hidden unit then computes its activation and sends its signal (z_j) to each output unit. Each output unit (Y_k) computes its activation (y_k) to form the response of the net for the given input pattern.

During training, each output unit compares its computed activation y_k with its target value t_k to determine the associated error for that pattern with that unit. Based on this error, the factor δ_k (k = 1 ,...., m) is computed. δ_k is used to distribute the error at output unit Y_k back to all units in the previous layer (the hidden units that are connected to Y_k). It is also used (later) to update the weights between the output and the hidden layer. In a similar manner, the factor δ_j (j= 1,....., p) is computed for each hidden unit Z_j. It is not necessary to propagate the error back to the input layer, but δj is used to update the weights between the hidden layer and the input layer.

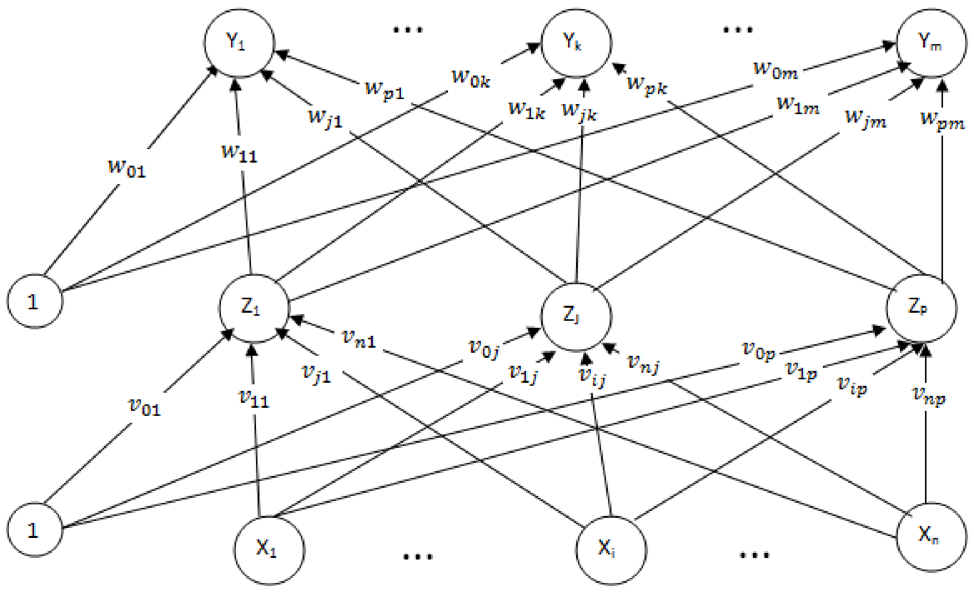


Figure 5. Backpropagation neural network with one hidden layer.

After all of the δ factors have been determined, the weights for all layers are adjusted simultaneously. The adjustment to the weight w_{jk} (from hidden unit Z_j to output unit Y_k) is based on the factor δ_k and the activation z_j of the hidden unit Z_j. The adjustment to the weight v_{ij} (from input unit X_i to hidden unit Z_j) is based on the factor δ_j and the activation xi of the input unit.

Training Algorithm

The training algorithm is as follows:

Step 0. Initialize weights.

(Set to small random values)

Step 1. While stopping condition is false, do Steps 2-9.
Step 2. For each training pair, do Steps 3-8.

Feed forward:

Step 3. Each input unit (X_i, i = 1,……,n) receives input signal xi and broadcasts this signal to all units in the layer above (the hidden units).
Step 4. Each hidden unit (Zj, j = 1,…….,p) sums its weighted input signals,

$$z\_in_j = v_{oj} + \sum_{i=1}^{n} x_i v_{ij},$$

applies its activation function to compute its output signal,

$$z_j = f(z\_in_j),$$

and sends this signal to all units in the layer above (output units).

Step 5. Each output unit (Y_k, k = 1,…….,m) sums its weighted input signals,

$$y\_in_k = w_{ok} + \sum_{j=1}^{p} z_j w_{jk},$$

applies its activation function to compute its output signal,

$$y_k = f(y\_in_k),$$

Backpropagation of error:

Step 6. Each output unit (Y_k, k = 1,…….,m) receives a target pattern corresponding to the input training pattern, computes its error information term.

$$\delta_k = (t_k - y_k)f'(y\_in_k)$$

calculates its weights correction term (used to update w_{jk} later),

$$\Delta w_{jk} = \alpha \delta_k z_j,$$

calculates its bias correction term (used to update w_{ok} later),

$$w_{ok} = \alpha \delta_k,$$

Step 7. Each hidden unit (Z_j, j = 1,…….,p) sums its delta inputs (from units in the layer above),

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk},$$

multiples by the derivative of its activation function to calculate its error information term,

$$\delta_j = \delta\_in_j f'(z\_in_j),$$

calculates its weight correlation term (used to update v_{ij} later),

$$\Delta v_{ij} = \alpha \delta_j x_i,$$

and calculates its bias correlation term (used to update v_{oj} later),

$$\Delta v_{oj} = \alpha \delta_j.$$

Update weights and biases:

Step 8. Each output unit (Y_k, k = 1,…,m) updates its bias and weights (j = 0, …,p):

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$

Each hidden unit (Z_j, j = 1,…,p) updates its bias and weights (i = 0,…,n):

$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$

Step 9. Test stopping condition.

3.1.1.2. Quick propagation

Quick propagation is an extension to Newton's method. This is used as an optimization to backpropagation. This is useful for decreasing the magnitude of the gradient and for changing the sign in between two steps. In order to determine the weight for the next step a parabolic estimate of the MSE is used. We can compute the derivatives in the direction of each weight by using Quick propagation. A direct step of the error minimum is attempted after computing the first gradient with regular backpropagation.

After computing the first gradient with regular back-propagation, a direct step of the error minimum is attempted by

$$\Delta x(t) = \frac{f'\big((t)\big)}{f'\big((t-1)\big) - f'\big((t)\big)}\,\Delta x(t-1)$$

3.1.1.3. Delta-Bar-Delta algorithm

Delta-Bar-Delta algorithm can be used to control the learning rates. This is possible with the possible sign changes of an exponential average gradient. The learning rates can be increased by adding a constant value rather than multiplying it.

Hence,

Choose some small initial value for every $\eta_{ji}(0)$.

Adapt the learning rates:

$$\eta_{ji}(n) = \eta_{ji}(n-1) + u, \text{if } \frac{\partial \xi}{\partial w_{ji}}(n) * \frac{\partial \xi}{\partial w_{ji}}(n-1) \geq 0$$

$$\eta_{ji}(n) = \eta_{ji}(n-1) * d, \text{if } \frac{\partial \xi}{\partial w_{ji}}(n) * \frac{\partial \xi}{\partial w_{ji}}(n-1) \leq 0$$

$$\eta_{ji}(n) = \eta_{ji}(n-1), \text{else.}$$

In particular it is difficult to find a proper u. Small values may result in slow adaptations while big ones endanger the learning process. As small values result in slow adaptations while big ones are dangerous for learning process. Vivid values are chosen for u and d. They are respectively u(5.0, 0.095, 0.085, 0.035) and d(0.9, 0.85, 0.666).

3.1.1.4. Conjugate gradient method

A se of Vectors {sj} is conjugate with respect to a positive definite matrix (e.g., the Hessaian) if sTj Hsi=0 where j ≠i. What this expression says is that the rotation by Hof the vector sj. In the n-dimensional Euclidean space Rn there are an infinite number of conjugate vector sets. It is easy to show that the eigenvectors of the Hessian form a conjugate set and can then be used to search the performance surface. The problem is that understanding the Hessian, which is not a practical assumption. However, there is a way to find a conjugate set of vectors that does not require knowledge of the Hessian. The idea is to express the conditions for a conjugate vector set as a function of difference in consecutive gradient directions as

$$(\nabla j(i) - \nabla j(i-1))T\ s(j) = 0\ i \neq j$$

For this expression to be true, the minimum of the gradient of J(i) in the direction s(j) is needed, so the algorithm works as follows.

Start with the gradient-descent direction, $s(0) = -\nabla J(0)$. Search the minimum along this direction. Then construct a vector s(j) that is orthogonal to the set of vectors $\{\nabla J(0), \nabla J(1), \ldots\ldots \nabla J(j\ 1)\}$, which can be accomplished by

$$S(j) = -\nabla J(j) + \alpha\ s(j-1)$$

There are basically three well-known ways to find α namely, the Fletcher-Reeves, the Polak-Ribiere, or the Hestenes-Steifel formulas, which are equivalent for quadratic performance surfaces and are given respectively by

$$\alpha j = \frac{\nabla JT(j)\nabla J(j)}{\nabla JT(j-1)\nabla J(j-1)} \quad \alpha j = \frac{[\nabla J(j) - \nabla J(j-1)]T\ \nabla J(j)}{\nabla JT(j-1)\nabla J(j-1)}$$

$$\alpha j = \frac{[\nabla J(j) - \nabla J(j-1)]T\ \nabla J(j)}{\nabla JT(j-1)s(j-1)}$$

In quadratic performance surfaces, the above value can be find the minimum in n iterations, where n is the size of the search space. The minimization along the line can be accomplished for quadratic performance surfaces. The problem is that, for nonquadratic performance surfaces such as the one found in neurocomputing, quadratic termination is not guaranteed and the line search does not have an analytic solution.

The lack of quadratic termination can be overcome by executing the algorithm for n iterations and then resetting it to the current gradient direction. The problem of the lines search is more difficult to solve. There are two basic approaches: direct search or the scaled conjugate method [Shepherd 1997]. The first involves multiple cost-function evaluations and estimations to find the minimum, which complicates the mechanics of the algorithm. The scaled conjugate is more appropriate for neural network implementations. It uses Eq. 4B.1 and avoids the problem of nonquadratic surfaces by messaging the Hessian so as to guarantee positive definiteness, which is accomplished by H+λI, where I is the identity matrix. We get

$$\mu j = -\nabla J\ Tj\ sj/\ sTjHjsj + \lambda\ l\left|sj\right|/2$$

Any one may think that this method is more computationally expensive than search, because of the Hessian matrix. But in fact, this is not the case, since there are fast methods to estimate the product of a vector by the Hessian (the product has only n components). The perturbation method can be used to estimate the product [LeCun, Simard, and Pearlmutter 1993]:

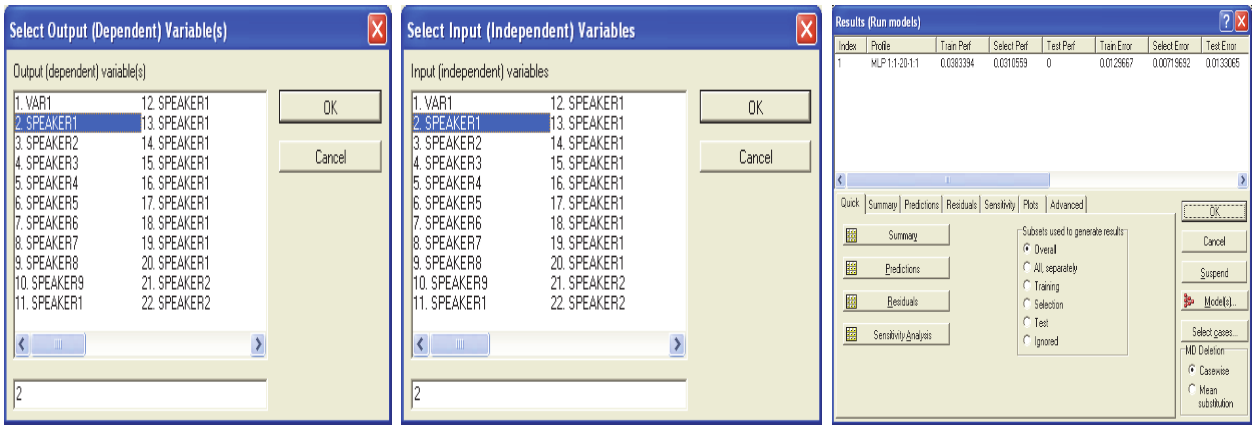
$$sT\ (\nabla J) = \frac{[\nabla J(w + \varepsilon s) - \nabla J(w)}{\varepsilon} + 0(\varepsilon)$$

Or use an analytic approach due to Pearlmutter [1994]. Both methods are compatible with backpropagation.

λ can be solved by using trial and error method. Notice that for large λ the denominator becomes approximately λ ||s||2. In this case one can use gradient descent, which is known to be convergent to the minimum (albeit slowly). When the error decreases, then λ should again be decreased to fully exploit the potential of the local quadratic information of the performance surface.

3.1.1.5.Implementing multilayer perceptron with Trajan 6.0 demonstrator

Select the speaker (word) whose training and testing recognition rate is required. Create custom neural network training with output (dependent) variable as desired one and input (independent) as target one. Choose the network type as MLP. Train MLP with quick backpropagation in phase one with 100 number of epochs and with learning rate of 0.01. Train MLP with Conjugate gradient learning algorithm in phase two with conjugate gradient learning with 500 number of epochs.


Select Output (Dependent) Variable(s)

Output (dependent) variable(s)

1. VAR1, 2. SPEAKER1, 3. SPEAKER2, 4. SPEAKER3, 5. SPEAKER4, 6. SPEAKER5, 7. SPEAKER6, 8. SPEAKER7, 9. SPEAKER8, 10. SPEAKER9, 11. SPEAKER1, 12. SPEAKER1, 13. SPEAKER1, 14. SPEAKER1, 15. SPEAKER1, 16. SPEAKER1, 17. SPEAKER1, 18. SPEAKER1, 19. SPEAKER1, 20. SPEAKER1, 21. SPEAKER2, 22. SPEAKER2

2

Select Input (Independent) Variables

Input (independent) variables

1. VAR1, 2. SPEAKER1, 3. SPEAKER2, 4. SPEAKER3, 5. SPEAKER4, 6. SPEAKER5, 7. SPEAKER6, 8. SPEAKER7, 9. SPEAKER8, 10. SPEAKER9, 11. SPEAKER1, 12. SPEAKER1, 13. SPEAKER1, 14. SPEAKER1, 15. SPEAKER1, 16. SPEAKER1, 17. SPEAKER1, 18. SPEAKER1, 19. SPEAKER1, 20. SPEAKER1, 21. SPEAKER2, 22. SPEAKER2

2

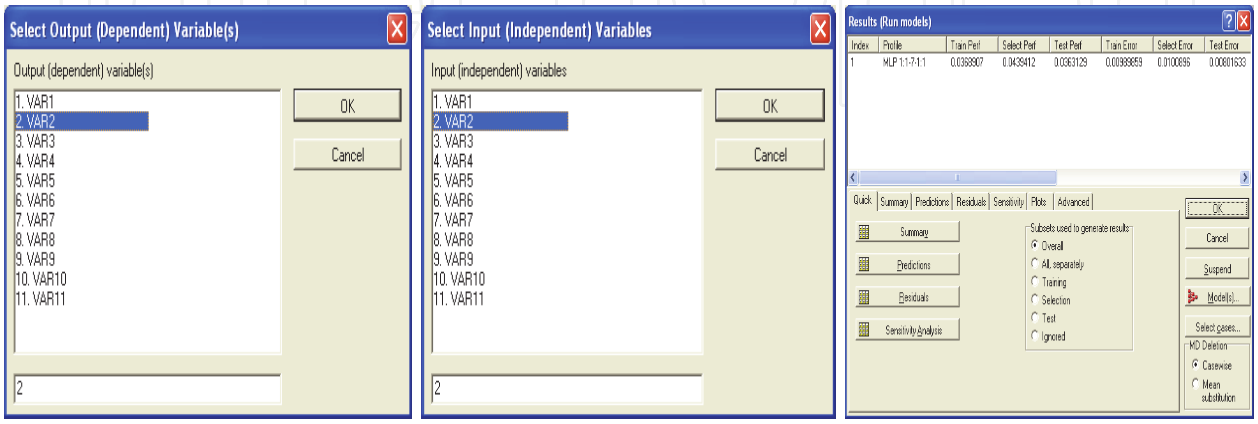
Results (Run models)

Index	Profile	Train Perf	Select Perf	Test Perf	Train Error	Select Error	Test Error
1	MLP 1:1-20-1:1	0.0383394	0.0310559	0	0.0129667	0.00719692	0.0133065

Subsets used to generate results: [✓ Overall] All, separately / Training / Selection / Test / Ignored

MD Deletion: [✓ Casewise] Mean substitution

Figure 6. Estimating MLP speaker recognition rate with Trajan


Select Output (Dependent) Variable(s)

Output (dependent) variable(s)

1. VAR1, 2. VAR2, 3. VAR3, 4. VAR4, 5. VAR5, 6. VAR6, 7. VAR7, 8. VAR8, 9. VAR9, 10. VAR10, 11. VAR11

2

Select Input (Independent) Variables

Input (independent) variables

1. VAR1, 2. VAR2, 3. VAR3, 4. VAR4, 5. VAR5, 6. VAR6, 7. VAR7, 8. VAR8, 9. VAR9, 10. VAR10, 11. VAR11

2

Results (Run models)

Index	Profile	Train Perf	Select Perf	Test Perf	Train Error	Select Error	Test Error
1	MLP 1:1-7-1:1	0.0368907	0.0439412	0.0363129	0.00989859	0.0100896	0.00801633

Subsets used to generate results: [✓ Overall] All, separately / Training / Selection / Test / Ignored

MD Deletion: [✓ Casewise] Mean substitution

Figure 7. Estimating MLP word recognition rate with Trajan

3.2. Radial basis function architecture

Radial basis function networks are a special case of multilayer perceptrons. The cells compute their activation not through a sigmoid function but according to a Euclidean similarity measure between patterns. The network uses radial basis functions as activation functions. These networks are used in function approximation, time series prediction and control.

Radial basis function networks typically has a two-layer feed forward architecture. The first layer maps an input feature vector to a set of hidden nodes or centers that constitute a basis for the input pattern space. The basis functions of this hidden layer produce a localized response to an input pattern. The second layer implements a linear mapping from the activation of the centers to the output nodes corresponding to different pattern classes. Hence the network carries out a non linear transformation by forming a linear combination of the basic functions. The basic function taken is Gaussian function. For each input pattern, the RBF network computes nodal outputs that are estimates of Bayesian Probabilities. Organization of RBF is shown in figure.

The training of an RBF network is a hybrid phase i.e. it is made up of an unsupervised phase followed by a supervised phase. As MLPs, RBF networks can be used for classification and function approximation tasks. They keep the pattern classification performances of other ANNs with a much lower training computation cost. These networks also feature good generalization capabilities.

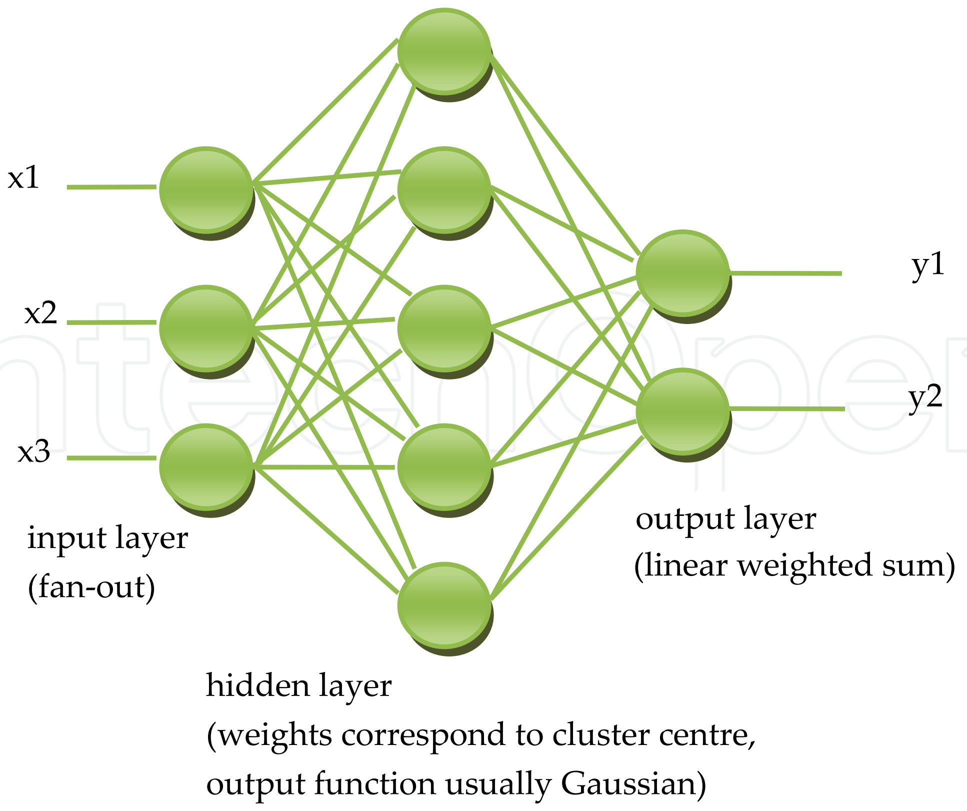


Figure 8. Radial Basis Function Neural Network Architecture

3.2.1. Training algorithm for an RBFN with fixed centers

The training algorithm for the radial basis function network is given below. The important aspect of the radial bais function network is the usage of activation function for computing the output.

Activation Function

Radial basis function uses Gaussian activation function. The response of such function is non-negative in all value of x. The function is defined as

$$f(x) = \exp(-x^2)$$

Its derivative is given by

$$f'(x) = -2x\exp(-x^2) = -2xf(x)$$

The radial basis function is different from the back propagation network in the Gaussion function it uses. The training algorithm for the network is given as follows:

Step 1. Initialize the weights (set to small random values)
Step 2. While stopping is false do Step 3 – 10.
Step 3. For each input do Step 4 – 9.
Step 4. Each input unit (x_i ,i = 1,. . . . n) receives input signals to all units in the layer above(hidden unit).
Step 5. Calculate the radial basis function.
Step 6. Choose the centres for the radial basis functions. The centres are chosen from the set of input vectors. A sufficient number of centers have to be selected in order to ensure adequate sampling of the input vector space.
Step 7. The output of an unit $v_i(x_i)$in the hidden layer.

$$v_i(x_i) = e\left(-\sum_{j=1}^{r}\left[x_{ji} - \widehat{x_{ji}}\right]^2 / \sigma_1^2\right)$$

Where

x_{ji} is center of RBF unit for input variables.
σ_i is width of the RBF unit.
X_{ji} is jth variable of input pattern.

Step 8. Initialize the weights in the output layer of the network to some small random value.
Step 9. Calculate the output of the neural network

$$y_{net} = \sum_{i=1}^{H} w_{im} v_i\,(x_i) + w_0$$

Where,

H is number of hidden layer nodes (RBF function)
Y_{net} is output value of mth node in output layer for the nth incoming pattern.
W_{im} is weight between ith RBF unit and mth output node
W_o is biasing term at nth output node.
Step 10. Calculate error and test stopping condition.

3.2.1.1. Isotropic

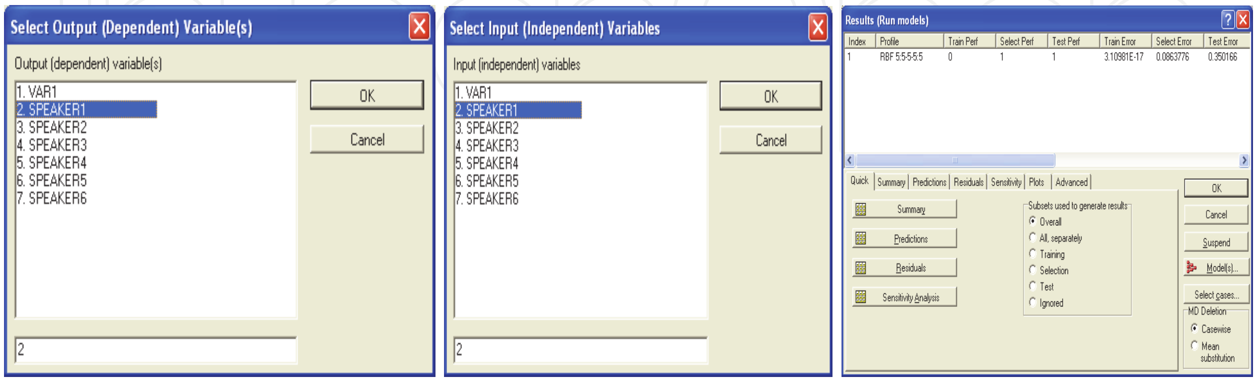
Select the deviation same for all units. Select these deviations heuristically in order to reflect the number of the centres and the space of the volume that they occupy(Haykin, 1994).

3.2.1.2. K-Nearest neighbor

Set each unit's deviation individually to the mean distance to its K-nearest neighbors (Bishop, 1995). Deviations are kept smaller in the space whose area is tightly packed whereas deviations are kept higher for sparse areas by interpolation. The output layer can be optimized after centres and deviations are set. The output layer can be optimized by using Pseudo-inverse (Singular-value decomposition) algorithm (Haykin, 1994; Golub and Kahan, 1965).

3.2.1.3. Implementing the radial basis function with Trajan 6.0 demonstrator

The Trajan 6.0 demonstrator accepts column formatted ASCII files. Use the Browse. Create custom neural network training with input data file as independent variables. Choose the network type as RBF. Choose the output (dependent) variable as the desired one. Select input (independent) variable as target one. Train RBF with the algorithm 3.2.1 with quick propagation. Training cases are sampled at Radial assignment. The Radial spread is set to the value 1, Isotropic is scaled by 1. K- nearest neighbour chosen as 10.



1. VAR1
2. SPEAKER1
3. SPEAKER2
4. SPEAKER3
5. SPEAKER4
6. SPEAKER5
7. SPEAKER6

2

1. VAR1
2. SPEAKER1
3. SPEAKER2
4. SPEAKER3
5. SPEAKER4
6. SPEAKER5
7. SPEAKER6

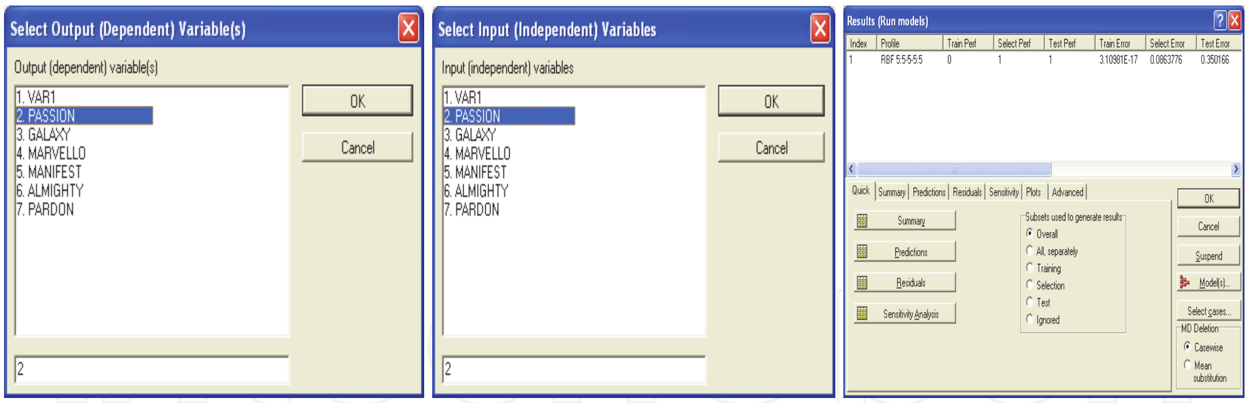
2

Index	Profile	Train Perf	Select Perf	Test Perf	Train Error	Select Error	Test Error
1	RBF 5:5-5-5:5	0	1	1	3.10981E-17	0.0863776	0.350166

Subsets used to generate results: [✓ Overall] All, separately Training Selection Test Ignored

MD Deletion: [✓ Casewise] Mean substitution

Figure 9. Estimating RBF speaker performance with Trajan



Output (dependent) variable(s)

1. VAR1
2. PASSION
3. GALAXY
4. MARVELLO
5. MANIFEST
6. ALMIGHTY
7. PARDON

2

Input (independent) variables

1. VAR1
2. PASSION
3. GALAXY
4. MARVELLO
5. MANIFEST
6. ALMIGHTY
7. PARDON

2

Index	Profile	Train Perf	Select Perf	Test Perf	Train Error	Select Error	Test Error
1	RBF 5:5-5-5:5	0	1	1	3.10981E-17	0.0863776	0.350166

Subsets used to generate results: [✓ Overall] All, separately Training Selection Test Ignored

MD Deletion: [✓ Casewise] Mean substitution

Figure 10. Estimating RBF word performance with Trajan

4. Results and discussions

The speaker and word identification rates for each speaker with the features LPCC and MFCC are obtained for both conventional and proposed systems in order to find maximum speaker and word identification rates for both training and testing the classifiers MLP and RBF. The results are presented in tables 2-3 and its analysis is presented in graph.

4.1. Estimating speaker identification rates

The experiment is carried out with 6 speakers (2 male,4 female)taking continuous speech and sound proof closed room. The system is trained continuously using high quality microphone with vocabulary size of 216 .wav files in constrained spoken input format.

Speakers	LPCC Training (%)	LPCC Testing (%)	MFCC Training (%)	MFCC Testing (%)
Speaker 1	90	85	92	90
Speaker 2	89	89	94	92
Speaker 3	93	90	90	88
Speaker 4	88	85	93	89
Speaker 5	91	90	94	91
Speaker 6	92	91	91	85

Table 2. Speaker identification rate with Multilayer perceptron

Speakers	LPCC Training (%)	LPCC Testing (%)	MFCC Training (%)	MFCC Testing (%)
Speaker 1	96	93	97	95
Speaker 2	94	91	99	98
Speaker 3	98	94	99	99
Speaker 4	97	95	97	98
Speaker 5	99	93	99	97
Speaker 6	95	92	98	96

Table 3. Speaker identification rate with Radial basis function

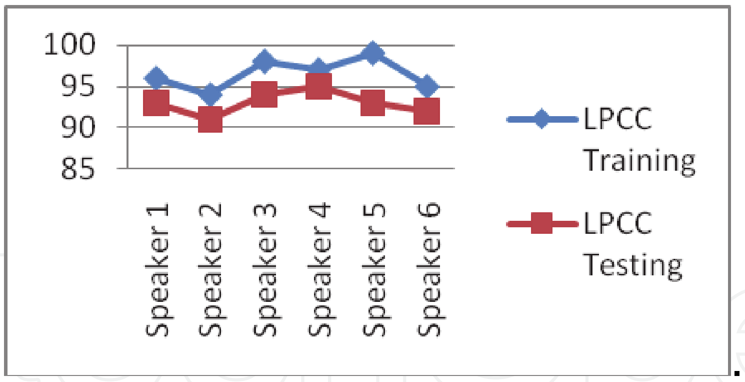


Figure 11. LPCC speaker identification rate with proposed system

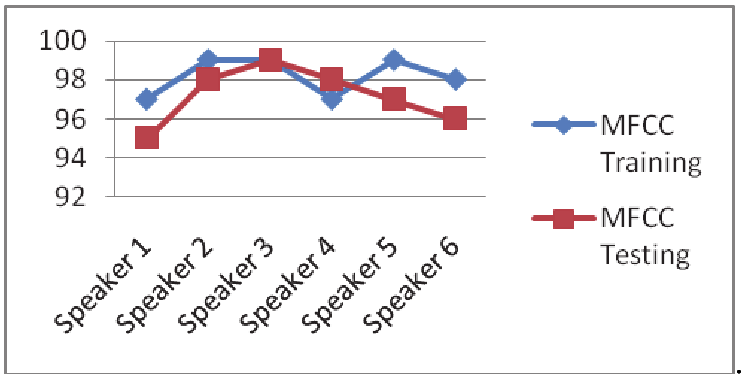


Figure 12. MFCC speaker identification rate with proposed system

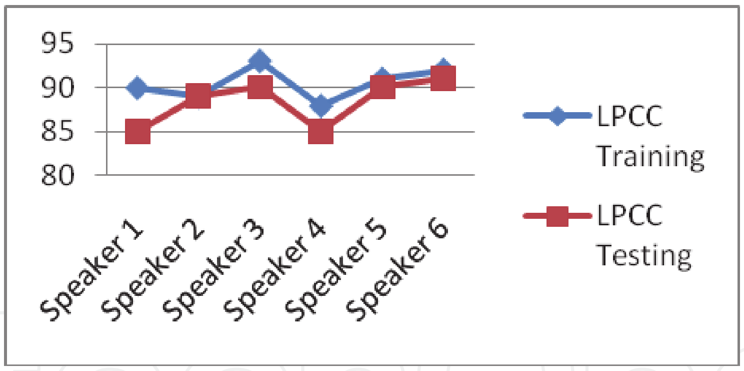


Figure 13. LPCC speaker identification rate with conventional system

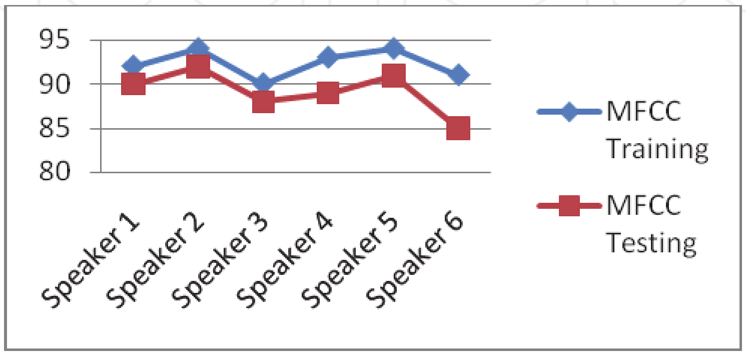


Figure 14. MFCC speaker identification rate with conventional system

4.1.1. Discussion

The LPCC training identification rate for speakers dominates the testing identification rate for all the speakers with proposed system. The MFCC training identification rate for speakers dominates the testing speaker identification rate for all the speakers except for speaker 4 and for the speaker 3 both the performances are same with proposed system. The LPCC training identification rate for speakers dominates the testing identification rate for all the speakers and for the speaker 2 both the performances are same with conventional system. The MFCC training identification rate for speakers dominates the testing speaker identification rate for all the speakers with conventional system. In the combination of MLP and RBF algorithms, the maximum identification of speakers is found in the range of 95% -99% for proposed system where as for the conventional system it is in the range of 89%-92%.

4.2. Estimating word identification rates

4.2.1. Discussion

For the proposed system, the highest recognition rate is obtained for Marvellous and Almighty with LPCC feature. For the conventional system, the highest recognition rate is obtained for Passion, Galaxy, Almighty and Pardon with LPCC feature. For the proposed system, the highest recognition rate is obtained for Galaxy, Almighty and Pardon with MFCC feature. For the conventional system, the highest recognition rate is obtained for Galaxy, Manifestaion, Almighty and Pardon respectively with MFCC feature. In the combination of MLP and RBF algorithms, the maximum identification of words is found in the range of 97%-99% for proposed system where as for the conventional system it is in the range of 94%-95%.

Feature Extraction	WIPS (%)		WICS (%)		WIPS-WICS (%)
	RBF Testing		MLP Testing		Difference
LPCC	Passion	94	Passion	91	3
	Galaxy	93	Galaxy	91	2
	Marvellous	95	Marvellous	90	5
	Manifestaion	91	Manifestaion	89	2
	Almighty	95	Almighty	91	4
	Pardon	94	Pardon	91	3

Table 4. Word identification rate for Multilayer perceptron and Radial basis function with LPCC

Feature Extraction	WIPS (%)		WICS (%)		WIPS-WICS (%)
	RBF Testing		MLP Testing		Difference
MFCC	Passion	97	Passion	94	3
	Galaxy	99	Galaxy	95	4
	Marvellous	98	Marvellous	94	4
	Manifestaion	97	Manifestaion	95	2
	Almighty	99	Almighty	95	4
	Pardon	99	Pardon	95	4

Table 5. Word identification rate for Multilayer perceptron and Radial basis function with MFCC

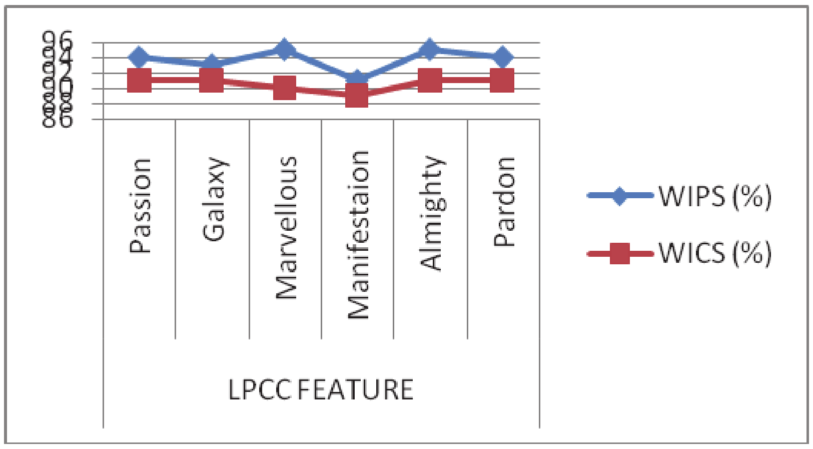


Figure 15. Word identification rate for conventional and proposed system with LPCC

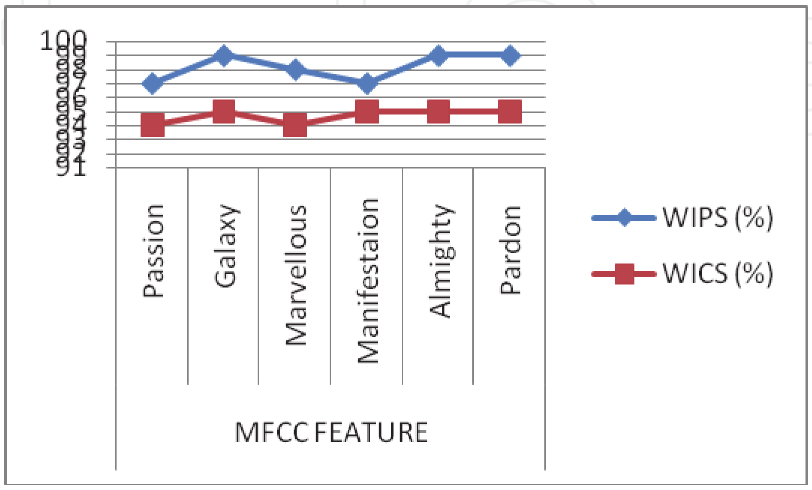


Figure 16. Word identification rate for conventional system and proposed system with MFCC

4.3. Testing results for conventional system

The recognition accuracy for training and testing set with the feature LPCC and MFCC for Letter recognition for conventional system (WRCS) is presented in Table 6 and its analysis is presented in graph form shown in Fig 17.

Feature	Recognition in accuracy (%)	
	Training	Testing
LPCC	97	94
MFCC	99	95

Table 6. Recognition accuracy for conventional System with MLP classifier

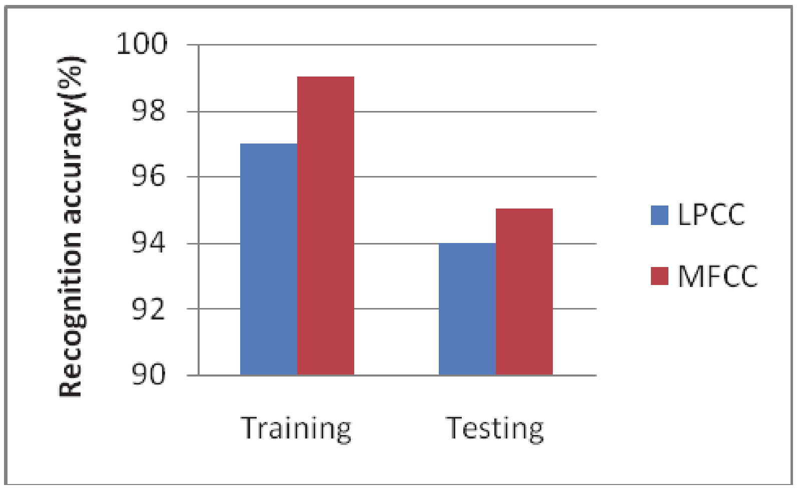


Figure 17. Recognition accuracy for conventional system with MLP classifier

From Table 6 and Fig 17, it is found that the highest recognition accuracy is obtained with MFCC feature both in training and testing. The training phase dominates the testing phase with both the features LPCC and MFCC.

4.4. Testing results for proposed system

From Table 7 and Figure 18, it is found that the highest recognition accuracy is obtained with MFCC feature both in training and testing. The training phase dominates testing phase with both the features LPCC and MFFC. The recognition accuracy for training and testing set with the features LPCC and MFCC for word recognition for proposed system(WRPS) is presented in Table 7 and its analysis is presented in graph form shown in Fig 18.

Feature	Recognition in accuracy (%)	
	Training	Testing
LPCC	99.9	98
MFCC	100	99

Table 7. Recognition accuracy for proposed System with RBF classifier

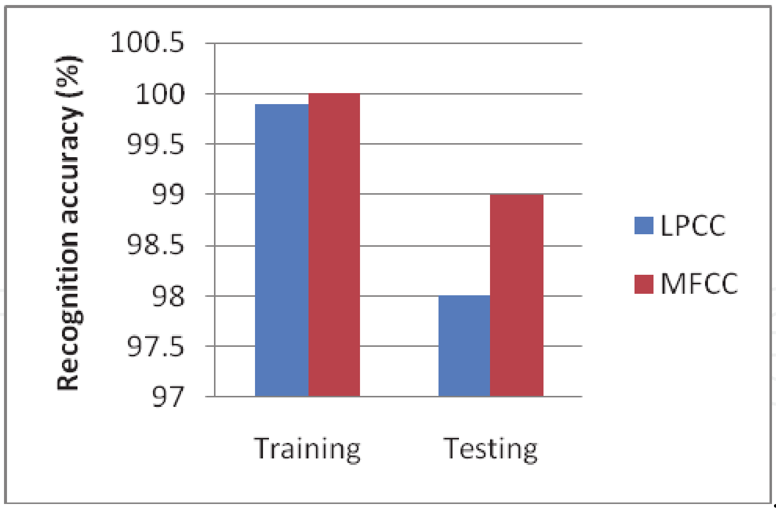


Figure 18. Recognition accuracy for conventional system with RBF classifier

4.5. Comparison of Performance for word recognition for proposed system (WRPS) and Word recognition for conventional system (WRCS) according to features

The comparison of performance for WRPS and WRCS according to feature is presented in Table 8 and analyzed in Fig 19. The comparison of performance is presented for testing the results.

Feature	Recognition in accuracy (%)		
	WRPS	WRCS	WRPS - WRCS
LPCC	98	94	4
MFCC	99	95	4

Table 8. Comparison of performance for WRPS and WRCS according to features.

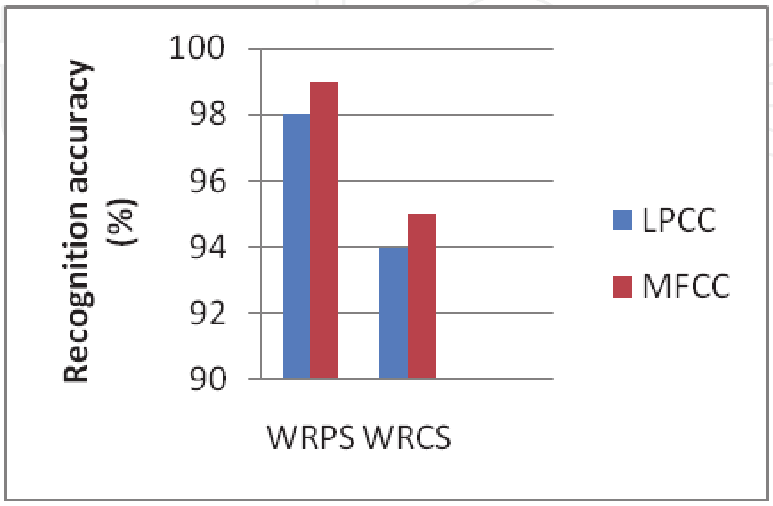


Figure 19. Comparison of performance for WRPS and WRCS according to features

From Table 8 and Fig 19, the following observations are made

1. For both WRCS and WRPS, the highest recognition accuracy is obtained with MFCC.
2. WRPS has the highest accuracy than WRCS for every test sets.

It is shown that WRPS out perform the WRCS in every test set according to features. From the tests, MFCC out performs LPCC.

The comparison of performance for speaker recognition for proposed system (SRPS) and speaker recognition for conventional system (SRCS) according to features is presented in Table 9. The comparison of performance is presented for testing the results.

Feature	Testing System	Mean (%)	Mode (%)	Median (%)	Min (%)	Max (%)	Standard deviation	Variance	Co-Efficient of variation	Skewness	Correlation coefficient
LPCC	SRCS	88.3	89.5	90	85	91	2.27	5.15	2.57	-0.75	-0.25
	SRPS	93	93	93	91	95	1.290	1.66	1.79	0.1	
MFCC	SRCS	93.59	88.35	93	90	92	1.2	1.44	2.3	0.23	-0.01
	SRPS	87.56	92	89.22	85	91	1.39	1.93	1.34	1.23	

Table 9. Comparative study of different feature recognition techniques of Multilayer perceptron and Radial basis function

In testing the speaker performance with both the features LPCC and MFCC, the proposed system is negatively correlated with the conventional system. The value of Co-efficient of variation for both the features LPCC and MFCC of the proposed system is less than the conventional system. Hence the proposed system is more efficient than the conventional system.

5. Conclusion

For the recognition of isolated words, it has been shown that the Radial basis function Neural network is suitable. Word recognition is carried out in speaker dependent mode. In this mode, trained data and tested date are chosen to be same. As the first 16 in the cepstrum represent most of the formant information 16 Linear Predictive cepstral coefficients and Mel frequency cepstral coefficients with 16 parameters are chosen. From the experimental results, it is found that RBF classifier performs better than MLP classifier. It is found that speaker 6 average performance is the best performance in training MLP classifier and speaker 2 average performance is the best performance in training RBF classifier. It is found that average speaker 4 performance is the best performance in testing MLP classifier and speaker 1 average performance is the best performance in testing RBF classifier. The real time factor for the proposed speech recognition system is found as less than 1.

6. Future scope

Besides improving the FE block and devising a more robust recognizer, the scope of the problem should be broadened to larger vocabularies, continuous speech and more speakers

Author details

R.L.K.Venkateswarlu*

Dean of Research, Research and Development Wing, Sasi Institute of Technology and Engineering Tadepalligudem, India

R. Raviteja and R. Rajeev

R.L.K. Institute of Mathematics, Rajahmundry, Andhra Pradesh, India

7. References

Al-Alaoui, M.A., Mouci, R., Mansour M.M., Ferzli, R., , 2002: A Cloning Approach to Classifier Training, IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans, vol.32, no.6, pp.746- 752).

Benyettou, A., 1995: Acoustic Phonetic Recognition in the Arabex System. Int. Work Shop on Robot and Human Communication, ATIP95.44, Japan.

Berthold, M.R., 1994: A Time Delay Radial Basis Function for Phoneme Recognition. Proc. Int. Conf. on Neural Network, Orlando, USA.

Gurney, K., 1997: An Introduction to Neural Networks, UCL Press, University of Sheffield, pp.no 234.

Kandil N, Sood V K, Khorasani K and Patel R V, 1992: Fault identification in an AC–DC transmission system using neural networks, IEEE Transaction on Power System, 7(2):812–9.

Laurene, Fausett., 2009: Fundamentals of Neural Networks Architectures, Algorithms and Applications, Pearson Education and Dorling Kindersly Publishing Inc., India, pp.no.467.

Morgan, D. and Scolfield, C., 1991: Neural Networks and Speech Processing, Kluwer Academic Publishers, pp.no.391.

Park D C, El-Sharakawi M A and Ri Marks II, 1991: Electric load forecasting using artificial neural networks, IEEE Trans Power System, 6(2), pp 442–449.

Picton, P. 2000: Neural Networks, Palgrave, NY .pp.no 195.

Rabiner, L. and Juang, B. -H., 1993: Fundamentals of Speech Recognition, PTR Prentice Hall, San Francisco, NJ. pp.no 507.

Tan Lee, P. C. Ching, L.W. Chan, 1998: Isolated Word Recognition Using Modular Recurrent Neural Networks, Pattern Recognition, vol. 31, no. 6, pp. 751-760.

* Corresponding Author