MAEER's

BE Project Report

ON

# CLASSIFICATION OF BACKGROUND NOISE FOR SPEECH DENOISING APPLICATION

SUBMITTED BY,

**Pranav Gundewar (B120023049)**

**Hrishikesh Hippalgaonkar (B120023054)**

**Vihang Khare (B1200230154)**

Project Guide:

**Prof. Alwin D. Anuse (Internal Guide)**

**Sponsored by**

**College**

**Year: 2016-2017**

Department of Electronics and Telecommunication

Maharashtra Institute of Technology, Pune - 38.

MAEER's

MAHARASHTRA INSTITUTE OF TECHNOLOGY, PUNE.

## CERTIFICATE

This is to certify that the Project Phase-II entitled

**CLASSIFICATION OF BACKGROUND NOISE FOR SPEECH DENOISING APPLICATION**

has been carried out successfully by

**Pranav Gundewar (B120023049)**

**Hrishikesh Hippalgaonkar (B120023054)**

**Vihang Khare (B1200230154)**

during the Academic Year **2016-2017**
in partial fulfillment of their
course of study for Bachelor's Degree in
**Electronics and Telecommunication** as per the syllabus prescribed by the
**SPPU.**

**Prof. Alwin D. Anuse**
Internal Guide

Head of Department
(Electronics & Telecommunication)

## DECLARATION

We the undersigned, declare that the work carried under

Project Phase-II entitled

# CLASSIFICATION OF BACKGROUND NOISE FOR SPEECH DENOISING APPLICATION

has been carried out by us and it is being not implemented by any external agency/company that sells projects. We further declare that work submitted in the form of report is not been copied from any paper/thesis/site as it is. However existing methods/approaches from any paper/thesis/site are being cited and are been acknowledged in the reference section of this report.

We are aware that our failure to adhere the above, the Institute/University/Examiners can take strict action against us. In such a case, whatever action is taken, it would be binding on us.

| Roll no | Name of student | Signature with date |
|---|---|---|
| B120023049 | Pranav Gundewar | |
| B120023054 | Hrishikesh Hippalgaonkar | |
| B120023154 | Vihang Khare | |

Academic Year **2016-2017**

MIT, Pune

# ACKNOWLEDGEMENT

A project is an opportunity for the student to practically implement theoretical concepts. It proves to be a learning platform for the students so that they can compete successfully in their professional life. However, in this entire journey of completing the project, we need proper guidance so as to avoid obvious mistakes.

We would like to thank our principal Prof. Dr. L. K. Kshirsagar for his constant encouragement. We would also like to thank out head of E&TC department Prof. Dr. B. S Choudhary.

We would thank our internal guide Prof. Alwin Anuse for his invaluable guidance and support for making this project a success. We would also like to thank Prof. P. Mahajani for her profound help.

Lastly, our sincere thanks to all staff members and friends who helped us in all possible ways to make this project a success.

# ABSTRACT

Although research in audio processing has traditionally focused on speech recognition, study of environmental sound recognition (ESR) has gained impetus over recent years. Environmental sound recognition is a process of identifying background sound and ignoring speech and other foreground sounds. Once the class of the background environmental sound is determined, it can be suppressed by passing it through filters appropriate for that class of environment sound. ESR can also be used to locate a person by analyzing the background sounds.

Sound recognition involves the collection of audio data, extraction of significant features and finding common features between them, thus leading to grouping of the data and training it to classify new audio samples. Further, this trained system can be used to classify various background sounds into different categories and predict the background environment in the audio samples viz. Airport, Railway, Restaurants, and Highway etc. Efficiency of entire system is calculated using various methods like signal to noise ratio (SNR), Itakura-Saito distance etc.

# CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# Chapter 1 INTRODUCTION

Speech recognition has important problems when the main speaker is embedded in noisy environments. These problems are related to the correct detection of speech: there are false alarms (provoked by strong noises), and speech losses (when this speech is confused with noise). These factors degrade speech recognition rates producing an unsatisfactory experience for the user. If there are too many recognition mistakes, the user is forced to correct the system which takes too long, is a nuisance, and the user will finally reject the system. A high error rate is not acceptable for critical tasks, such as in Air Traffic Control (ATC) environments, which is probably the main reason for the low use of speech interfaces in ATC.

Owing to numerous problems associated with speech recognition & speech denoising, working on background noises instead of foreground speech proves to be more helpful for speech denoising. Environmental noise can be a rich source of information about the current situation. For example, we often infer the situation of a respondent in a mobile phone conversation by identifying the background noise and adjusting our response accordingly. The noise classification system described here classifies the audio context in the typical environments of our daily life, such as the airport, restaurant and railway station to identify the scenes in order to predict the user's needs and adjust the mode of operation accordingly. This work is part of a larger investigation into the integration.

The proposed system is based on noise feature extraction & machine learning. Noise features include but are not limited to different descriptors such as Mel Frequency Cepstral Coefficients (MFCCs), frequency roll-off, spectral centroid, spectral flux, spectral entropy, zero-crossing, energy, Linear Predictive Cepstral Coefficients (LPCCs). Machine learning is a branch of artificial intelligence, which deals with the study and construction of systems which are able to learn from the data. Artificial Intelligence concepts like neural networks are used to perform the work as human mind can do. This explores the idea of how humans recognize background from background noise in general and are used to develop machines that simulated this process. A Neural Network acquires knowledge as human brain acquires knowledge from learning process. For recognition of background, learning process in which back propagation/KNN/multiclass logistic regression algorithm takes input from the user. In this learning process, training and testing of different background noise is done. A library is made in which different noises that we come across in different scenes are stored which are used for future comparisons, this library helps in acceptance and rejection of noise models.

Once the class of noise is identified, corresponding input is passed through Least Mean Square(LMS)/ Recursive Least Square(RLS) filter to remove the noise in the input samples. After filtering the noise, system efficiency is found out using signal to noise ratio (SNR), Itakura-Saito(IS) distance. Nowadays, there is an increasing interest for developing robust Environmental Sound Recognition (ESR) for real-time applications in adverse conditions.

## 1.1 SCOPE OF PROJECT

The main objective is to identify and classify background noise.

Feature extraction includes MFCC & LPC algorithms. The proposed system includes classification noise inputs into 5 classes like restaurants, airport etc. Different classifiers to be implemented to classify backgrounds include KNN, multi-class logistic regression, back propagation. To remove background noise, different adaptive filters to be implemented include Recursive Least Square (RLS), Least Mean Square (LMS). The enhanced noise-less output signal and the original noisy signal is compared to find out the efficiency of a particular filter bank using subjective and objective measures. The objective measures like Signal to Noise Ratio (SNR), Itakura-Saito (IS) distance will be used to compute efficiency.

**The scope of the project is restricted to five noise classes and previously recorded noise samples.**

## 1.2 ORGANIZATION OF THE REPORT

- Chapter 1: Introduction and scope
  This chapter provides an overview of the basic functionality of the system and describes its scope of expansion
- Chapter 2: Literature Survey and present scenario
  This chapter enlightens the literature survey of the work done in this field so far as well as the present scenario.
- Chapter 3: System Block Diagram and Flow Chart
  Explains in detail the design and development process of the system. Includes system specifications block diagram, description of each block.
- Chapter 4: System Design
  It explains the algorithms and various methods used for feature extraction and classification and filtering and measures to calculate system efficiency.
- Chapter 5: System Implementation
  It includes all the algorithms used for feature extraction and classification and filtering and measures to calculate system efficiency.
- Chapter 6: Result and Conclusion
  It includes results and conclusions about feature extraction, classification, filtering and system efficiency measures
- Chapter 7: References
  Includes important reference papers referred for the system.

# Chapter 2 LITERATURE SURVEY

While researching about how to detect, classify background noise and filter it from different environmental scenes, we came across several research papers regarding environmental activity recognition, environment dependent noise tracking, speech denoising.
Many pieces of work have been done towards ESR. Among the various methods known to us are feature extraction using MFCC and LPC, classification of noise using classifiers like KNN, SVM, Backpropagation are widely used. Various adaptive filters are used to remove noise. Also, various methods to calculate system efficiency like SNR, IS distance are used.
The summary of research papers we referred is given as follows-

Nitish Krishnamurthy John H.L. Hansen explained use of environment dependent noise tracking for speech enhancement. The concept of noise tracking and noise update rate estimation. The proposed algorithm has successfully reduced the complexity of algorithm in sense of speed and time constraint. The environmental properties were also used to for determining update rate of noise. This algorithm provides as effective foundation for addressing noise in speech.

Ling Ma, Dan Smith, and Ben Milner presented Environmental Noise classification for Context Aware Applications. The hidden Markov model (HMM) based noise classifier is explained here. The system is able to capture, recognise and track environmental noise changes as a user changes locations. The accuracy of system is maximum up to 92.27%.

Stéphane Bressan and Boon Tiang Tan proposed Environmental Noise Classification for Multimedia Libraries. The system uses KNN, SVM & HMM. The accuracy of HMM algorithm starts decreasing as time duration of segments It is found out that system obtains best result from SVM compared to other.

Mohanapriya. S. P, E. P. Sumesh, R. Karthika explained Environmental Sound Recognition using Gaussian Mixture Model and Neural Network Classifier. Noise recognition involves the collection of audio data, extraction of important features, clustering of similar features and their classification. The Mel frequency cepstrum co-efficients are extracted. These features are used for clustering by a Gaussian mixture model which is a probabilistic model. KNN classifier is used for classification of the features and to identify the environmental audio scene.

Souli Sameh  Zied Lachiri presented Multiclass support vector machines for environmental sounds. This paper presents an approach aimed at recognizing environmental sounds for surveillance and security applications. In the proposed method, the spectrograms are passed through an appropriate log-Gabor filter banks and the outputs are averaged and underwent an optimal feature selection procedure based on a mutual information criteria.The classification results based on Multiclass Support Vector Machines show that this method is the most efficient with an average classification accuracy of 89.62 %.

Oscar Varela, Ruben San-Segundo & Luis A. Hernandez proposed Robust Speech Detection for Noisy Environments. This presents robust voice activity detector based on HMM in stationary and non-stationary environments. This presents an improved VAD for robust detection in noisy environments with different SNRs without the need of tuning. This improvement is based on three main contributions: an improved front-end including a selection of the most discriminative MFCCs, an improved reference level estimator for log energy normalization, and finally, the HMMs training considering the log energy normalization process.

Jyoti Dhiman, Shadab Ahmad, Kuldeep Gulia proposed the paper on comparison between adaptive filter algorithms. The paper describes the comparison between LMS, NLMS, TVLMS, RLS, FTRLS. The computational complexity and SNR of these algorithms are examined. Three performance criteria i.e minimum mean square error, execution time and required filter order are used to study these algorithms. Paper concludes that best adaptive algorithm is RLS based on SNR improvement.

## 2.1 PRESENT SCENARIO

In the real world, human speech recognition nearly always involves listening in background noise. The impact of such noise on speech signals and on intelligibility performance increases with the separation of the listener from the speaker.

Today, many researches have been done to in speech recognition and processing. But the problem of separating speech from noise or background noise is a challenging one. Even today, many algorithms have been proposed by many researchers so that speech recognition can be easily carried out. But the efficiency of these algorithms is not satisfactory. On the other side, background noise recognition and classification of background noise has gained impetus over traditional system involving speech recognition.

We are currently living in digital era where we frequently come across high speed internet and most important 4G. The important feature of 4G voice calling is noise free communication. Various algorithm and noise suppressing filters are used to remove noise during communication. The proposed system can be used in cell phones which can remove background noise and will provide noise free communication.

# Chapter 3 SYSTEM DEVELOPMENT

## 3.1 PROJECT SPECIFICATIONS

The main aim is to develop a robust system which can classify input audio samples into various noise classes namely restaurant, airport etc. The classifier used for classification is based on KNN, multi-class logistic regression, backpropagation algorithm. After classification, noise is then passed through appropriate adaptive filter like RLS, LMS to give noise free output. This enhanced noise free output and input is compared using SNR, IS distance.

*Software:*

- ✓ PC/ Laptop with Windows 7/10 OS

*Platform / Language used:*

- ✓ MATLAB R2015b, Audacity

*Audio Samples of 5 classes:*

- ✓ *Restaurant*
- ✓ *Airport*
- ✓ *Railway station*
- ✓ *Rain*
- ✓ *Highway Traffic*

## 3.2 SYSTEM BLOCK DIAGRAM

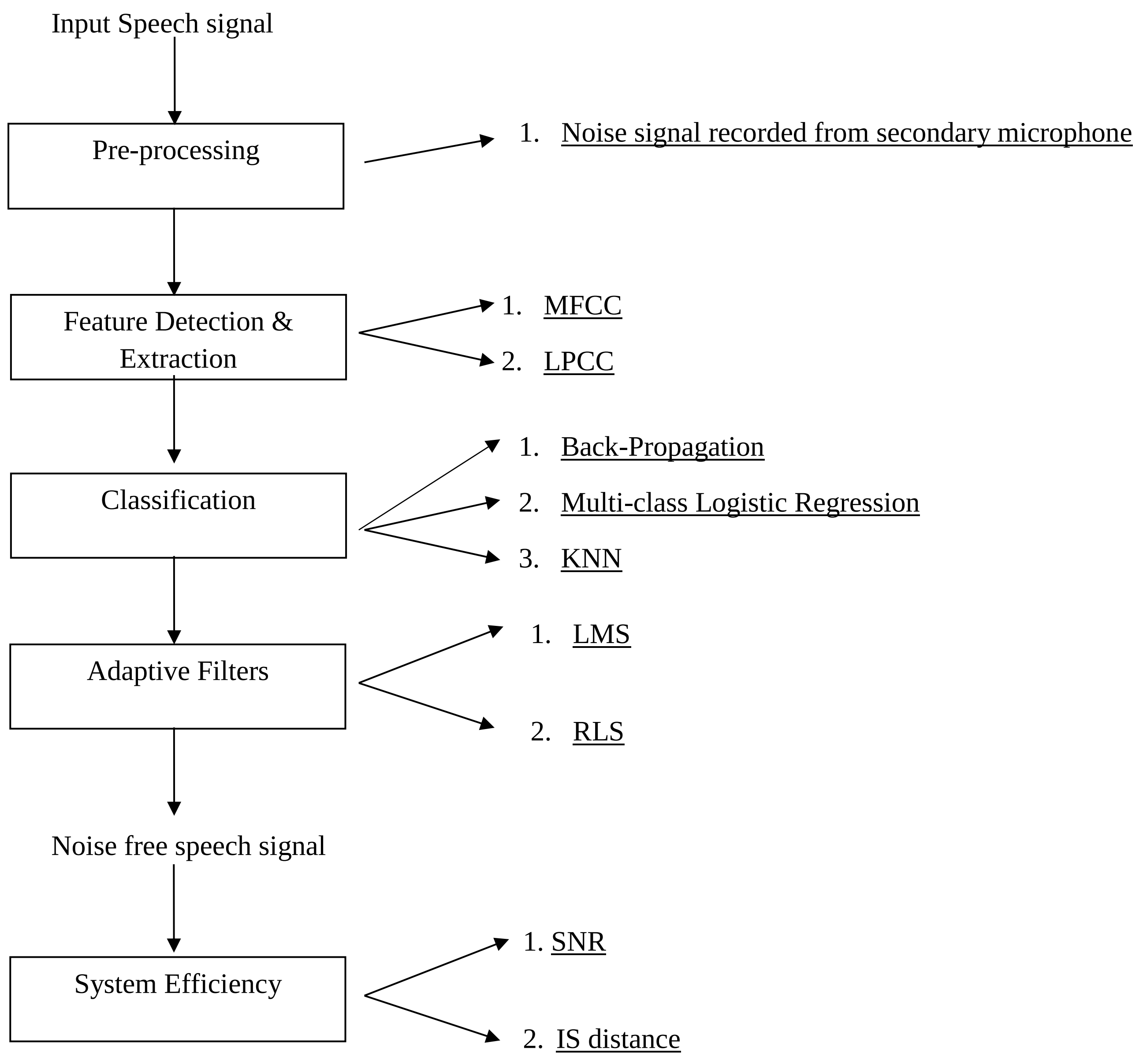


**Fig 3.2: System block diagram**

Speech signal recorded in different environments like railway station, airport, restaurant etc. is used as input dataset. In pre-processing, pauses in speech are detected manually and background noise is extracted. Thus, the extracted audio will contain background noise only. This background noise is given to feature extraction block. Features of interest from background noise are extracted in this block using MFCC (Mel Frequency Cepstral Coefficients) or Linear Predictive Cepstral Coefficients (LPCC). Total of 13 features are extracted for each frame for better efficiency and ease. These features are given to the classifier which uses multiclass classification algorithms to identify the environment. Various algorithms like KNN, backpropagation, multi-class logistic regression are used in classifier to classify background noise into different environments namely airport, restaurant, railway etc. According to the identified environment, input signal is passed through class specific filter to achieve noise free speech signal. Then enhanced noise free signal and input signal are compared for efficiency using SNR, IS distance measures etc.

# Chapter 4 SYSTEM DESIGN

## 4.1 DATA COLLECTION

While implementing this project in real world systems like smartphones, primary and secondary microphones are used for data collection. The primary microphone will record user speech mixed with background noise while the secondary noise cancelling microphone will record only the background noise. For training our classifiers, we have obtained eight hours of audio for each of the aforementioned five noise classes from YouTube. The audio samples obtained are in format .m4a with sampling frequency of 44.1 kHz. We have also obtained two podcasts from YouTube which are used as ‘foreground speech’ for training and testing the filters. These podcasts are also of .m4a format with sampling frequency of 44.1 kHz

## 4.2 FEATURE EXTRACTION

Feature extraction is an important audio analysis stage. In general, feature extraction is an essential processing step in pattern recognition and machine learning tasks. The goal is to extract a set of features from the dataset of interest. These features must be informative with respect to the desired properties of the original data. Feature extraction can also be viewed as a data rate reduction procedure because we want our analysis algorithms to be based on a relatively small number of features. In our case, the original data, i.e. the audio signal, is voluminous and as such, it is hard to process directly in any analysis task. We therefore need to transform the initial data representation to a more suitable one, by extracting audio features that represent the properties of the original signals while reducing the volume of data. Some of the frequency domain audio features include but are not limited to spectral centroid and spread, spectral entropy, spectral flux, spectral roll off, MFCC, LPCC, chroma vector.

### 4.2.1 MFCC

- MFCC, also known as Mel Frequency Cepstral Coefficients, is the dominant format that is used to represent features extracted from speech and is widely used in speech recognition software’s.
- Empirical evidence shows that using MFCC vectors to represent noise segments improves recognition performance.
- Mel frequency Cepstrum is a representation of short term power spectrum of a sound, based on linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

- The difference between cepstrum and mel frequency cepstrum is that in MFC, the frequency bands are equally spaced on the "Mel scale" which approximates the human auditory system's response more closely than the linearly spaced frequency bands in normal cepstrum.
- The standard implementation of MFCC is shown in the following block diagram:

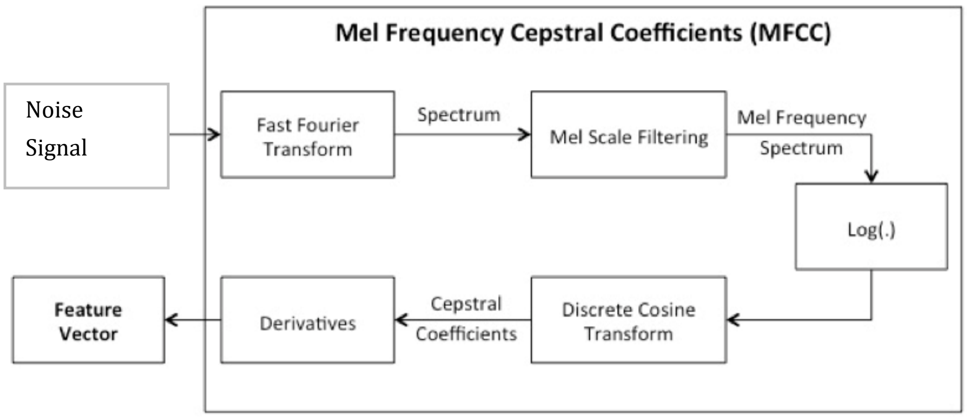


**Fig 4.2.1: MFCC block diagram**

### 4.2.1.1 IMPLEMENTATION

- The first processing step is framing the audio signal in short quasi stationary frames of 20 ms with an overlap of 10 ms. This is done because an audio signal is constantly changing, and so as to simplify things we assume that on short time scales the audio signal doesn't change much statistically. If the frame is much shorter we don't have enough samples to get a reliable spectral estimate, if it is longer the signal changes too much throughout the frame.
- The second step is the computation of the frequency domain representation of the input signal. This is achieved by computing the Discrete Fourier Transform.

$$S_i(k) = \sum_{n=1}^{N}\{\, s_i(n)h(n)e^{-2\pi kn/N}\,\} \qquad 1 \le k \le K \qquad (1)$$

Where h(n) is N sample long analysis window (e.g., Hamming window), and K is the length of the DFT. We take the absolute value of the complex Fourier transform, and square the result. We would generally perform a 512 point FFT and keep only the first 257 coefficients.

- The third step includes the computation of the mel-frequency spectrum. Therefore, the spectrum is filtered with $N_d$ different band-pass filters and the power of each frequency band is computed. This filtering mimics the human ear because the human auditory system uses the power over a frequency band as signal for further processing. This processing step can be described by

$$c_{\tau,j}^{(2)} = \sum_{k=0}^{N/2-1}\left\{ d_{j,k} c_{\tau,k}^{(1)} \right\} \qquad j = 0, 1, \dots, N_d \tag{2}$$

where d is the amplitude of the band-pass filter with the index j at the frequency k and C($_{r,k}$)=S$_i$(k). The filter bank with the band-pass filters cannot mimic the ear because the ear can use any frequency as center frequency. The mel-scale is a non-linear scale that is adapted to the non-linear pitch perception of the human auditory system. The number, the shape (triangular, trapezoidal rectangular) and the center frequency of the band-pass filters can be varied. Figure 2 shows a typical filter-bank with 25 triangular band-pass filters. Some research suggests that too few and too many band-pass filters have a negative impact on the classification performance and that overlapping rectangular shaped filters achieve a better performance compared to triangular shaped filters.

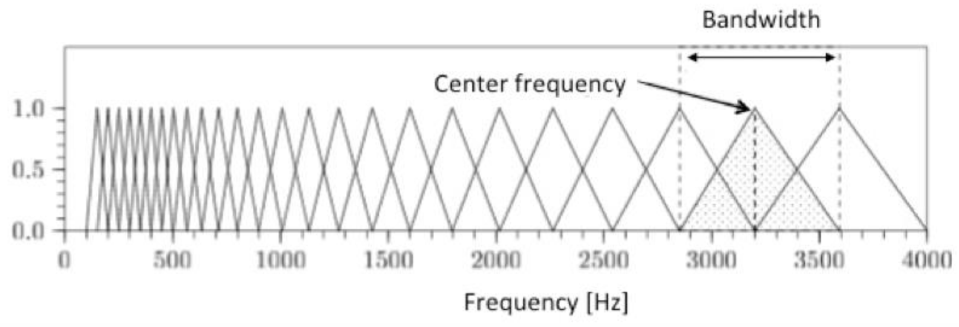


**Fig 4.2.1.1: Filterbank with 25 triangular bandpass filters to compute the mel frequency spectrum.**

- The fourth processing step computes the logarithm of the signal, to mimic the human perception of loudness because experiments showed that humans perceive loudness on a logarithmic scale.

$$c_{\tau,j}^{(3)} = \left\{ \log\left(c_{\tau,j}^{(2)}\right) \right\} \qquad j = 0, 1, \dots, N_d \tag{3}$$

- The fifth step computes the cepstral coefficients. Cepstral analysis is a special case within a general class of methods known as 'homomorphic' signal processing. The cepstral coefficients are computed by taking DCT of the mel log powers.

$$c_{\tau,j}^{(4)} = \sum_{j=1}^{N_d} \left\{ c_{\tau,j}^{(3)} \cos\left[\frac{k(2j-1)\pi}{2N_d}\right] \right\} \qquad k = 0, 1, \dots, N_{mc} < N_d \tag{4}$$

where N$_{mc}$ is the number of chosen cepstral coefficients for further processing. Typically, N$_{mc}$ is in the range of thirteen to twenty.

- This results in MFCC feature vector

## 4.3 CLASSIFICATION OF BACKGROUND NOISE:

- After extracting features from the audio input various algorithms like Back propagation(BP), K-Nearest Neighbor(K-NN), Support Vector Machines(SVM), Multiclass Logistic Regression(MLR) can be used to identify the environment wherein the audio sample has been recorded.
- This will give us some more insight and make our system more efficient at de-noising.
- Following is a brief description about the algorithms which we have planned to use.

### 4.3.1 K-Nearest Neighbor:

- This algorithm works on the principle: "tell me who your neighbors are, and I'll tell you who you are."
- The K indicates number of neighbors the algorithm evaluates to conclude about the class of a particular sample.
- It classifies an unknown example with the most common class among k closest examples.
- To demonstrate a *k*-nearest neighbor analysis, let's consider the task of classifying a new object (query point) among a number of known examples. This is shown in the figure below, which depicts the examples (instances) with the plus and minus signs and the query point with a red circle. Our task is to estimate (classify) the outcome of the query point based on a selected number of its nearest neighbors. In other words, we want to know whether the query point can be classified as a plus or a minus sign.

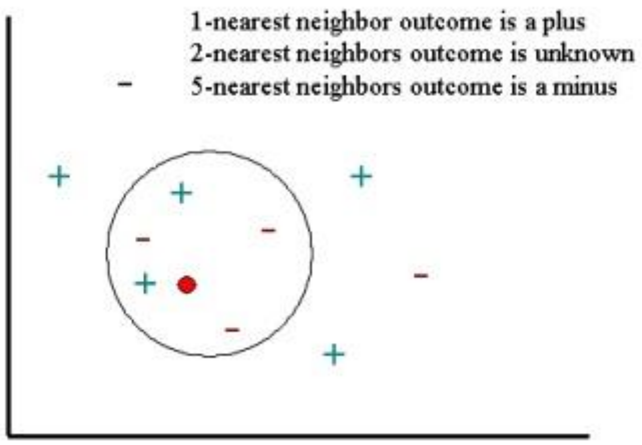

1-nearest neighbor outcome is a plus
2-nearest neighbors outcome is unknown
5-nearest neighbors outcome is a minus

**Fig 4.3.1: KNN diagram**

**Distance Metric**

- As mentioned before, given a query point, *KNN* makes predictions based on the outcome of the *K* neighbors closest to that point. Therefore, to make predictions with *KNN*, we need to define a metric for measuring the distance between the query point and cases from the examples sample. One of the most popular choices to measure this distance is known as Euclidean.
- We have tested 3 models viz. Euclidian, correlation and cosine distance model
- Distance Weighting
- Since *KNN* predictions are based on the intuitive assumption that objects close in distance are potentially similar, it makes good sense to discriminate between the *K* nearest neighbors when making predictions, i.e., let the closest points among the *K* nearest neighbors have more say in affecting the outcome of the query point. This can be achieved by introducing a set of weights *W*, one for each nearest neighbor, defined by the relative closeness of each neighbor with respect to the query point.

**Deciding Value of K:**

- In theory, if infinite number of samples available, the larger is k, the better is classification
- The caveat is that all k neighbors have to be close
- Possible when infinite samples available
- Impossible in practice since samples is finite
- Larger k gives smoother boundaries, better for generalization
- k = 1 is often used for efficiency, but sensitive to "noise"

Implementation of KNN:

- We took samples from 3 different environments viz. airport, railway station and restaurant.
- We took 3 samples of few seconds for each environment for different noise levels (0 db, 5 db ,10 db)
- Each sample had around 260 to 280 frames of 20 milliseconds each and an overlap of 10 milliseconds.
- We took K=16 (Efficiency was maximum for this value of K nearest neighbors).
- We got around 80 % accuracy of recognizing for each sample and for different distance metrics (Euclidian, cosine, correlation).

### 4.3.2 Logistic Regression

- Logistic regression is a method for classifying data into discrete outcomes. For example, we might use logistic regression to classify an email as spam or not spam.
- To attempt classification, one method is to use linear regression and map all predictions greater than 0.5 as a 1 and all less than 0.5 as a 0. However, this method doesn't work well because classification is not actually a linear function.
- The classification problem is just like the regression problem, except that the values y we now want to predict take on only a small number of discrete values. For now, we will focus on the binary classification problem in which y can take on only two values, 0 and 1. (Most of what we say here will also generalize to the multiple-class case.)
- For instance, if we are trying to build a spam classifier for email, then x(i) may be some features of a piece of email, and y may be 1 if it is a piece of spam mail, and 0 otherwise. Hence, $y \in \{0,1\}$. 0 is also called the negative class, and 1 the positive class, and they are sometimes also denoted by the symbols "-" and "+." Given x(i), the corresponding y(i) is also called the label for the training example.
- We use sigmoid function to find out the hypothesis:

$$h_\theta(x) = g(\theta^T x) \tag{5}$$

$$z = \theta^T x \tag{6}$$

$$g(z) = \frac{1}{(1+e^{-z})} \tag{7}$$

The sigmoid function can be shown graphically as follows:

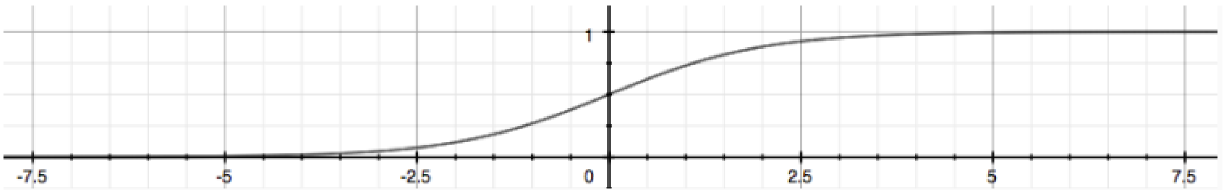


**Fig 4.3.2.1: Sigmoid function**

- hθ(x) will give us the probability that our output is 1. For example, hθ(x)=0.7 gives us a probability of 70% that our output is 1. Our probability that our prediction is 0 is just the complement of our probability that it is 1 (e.g. if probability that it is 1 is 70%, then the probability that it is 0 is 30%).

- The decision boundary is the line that separates the area where y = 0 and where y = 1. It is created by our hypothesis function.

Cost Function:

- We cannot use the same cost function that we use for linear regression because the Logistic Function will cause the output to be wavy, causing many local optima. In other words, it will not be a convex function.
- Instead, our cost function for logistic regression looks like:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\{\, Cost(h_\theta(x^{(i)}), y^{(i)})\,\} \tag{8}$$

$$Cost(h_\theta(x), y) = -\log(h_\theta(x)) \qquad if\ y = 1 \tag{9}$$

$$Cost(h_\theta(x), y) = -\log(1 - h_\theta(x)) \quad if\ y = 0 \tag{10}$$

Multiclass Classification: One-vs-all

- Now we will approach the classification of data when we have more than two categories. Instead of y = {0,1} we will expand our definition so that y = {0,1...n}.
- Since y = {0,1...n}, we divide our problem into n+1 (+1 because the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

$$y \,\epsilon\, \{\, 0, 1, \dots n \,\} \tag{11}$$

$$h_\theta^{(0)}(x) = P(y = 0 \mid x; \theta) \tag{12}$$

$$h_\theta^{(1)}(x) = P(y = 1 \mid x; \theta) \tag{13}$$

$$h_\theta^{(n)}(x) = P(y = n \mid x; \theta) \tag{14}$$

$$Prediction = \max_i(h_\theta^{(i)}(x)) \tag{15}$$

- We are basically choosing one class and then lumping all the others into a single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returned the highest value as our prediction.
- The following image shows how one could classify 3 classes:

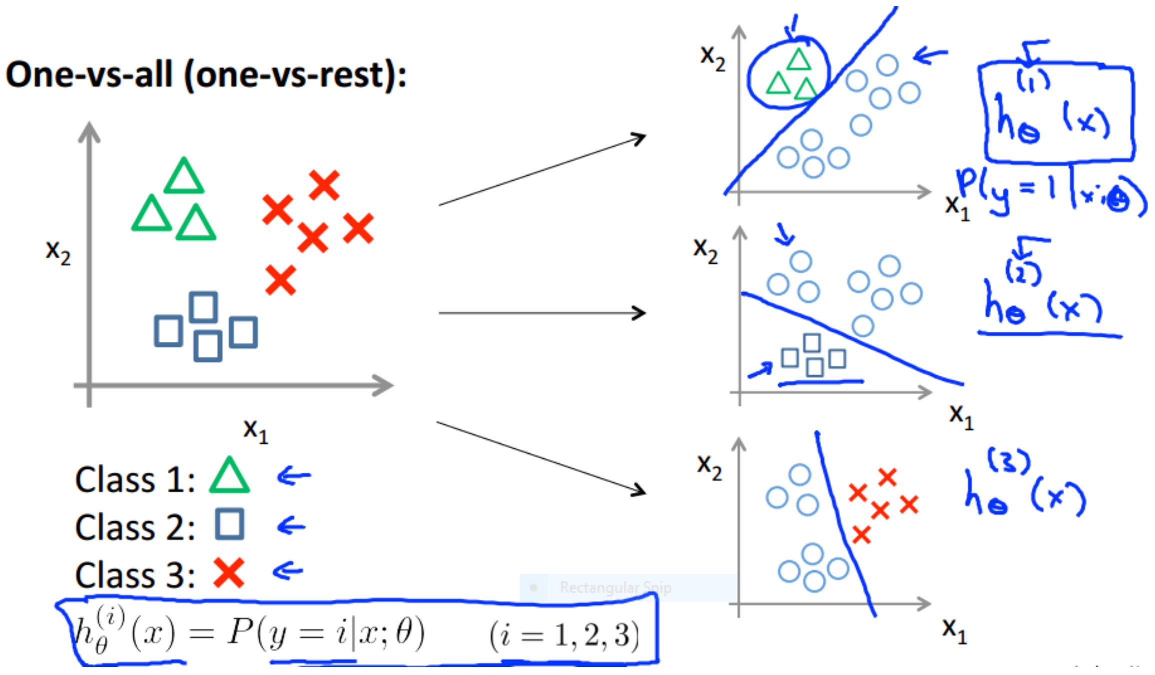


**Fig 4.3.2.2: Multiclass Classification**

### 4.3.3 Backpropagation

- Method of training artificial neural networks used in conjunction with an optimization method such as gradient descent
- Requires a known desired output for each input value in order to calculate the loss function gradient
- Backpropagation requires a known, desired output for each input value in order to calculate the loss function gradient.

- It is therefore usually considered to be a supervised learning method, although it is also used in some unsupervised networks such as autoencoders.
- It is a generalization of the delta rule to multi-layered feedforward networks, made possible by using the chain rule to iteratively compute gradients for each layer. Backpropagation requires that the activation function used by the artificial neurons (or "nodes") be differentiable.
- The number of hidden layers, no of neurons decide its efficiency
- The backpropagation learning algorithm can be divided into two phases: propagation and weight update.

  Phase 1: Propagation
- Each propagation involves the following steps:
- Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.
- Backward propagation of the propagation's output activations through the neural network using the training pattern target in order to generate the deltas (the difference between the targeted and actual output values) of all output and hidden neurons.

  Phase 2: Weight update

  For each weight-synapse follow the following steps:

  1. Multiply its output delta and input activation to get the gradient of the weight.
  2. Subtract a ratio (percentage) from the gradient of the weight.

  This ratio (percentage) influences the speed and quality of learning; it is called the *learning rate*. The greater the ratio, the faster the neuron trains, but the lower the ratio, the more accurate the training is. The sign of the gradient of a weight indicates where the error is increasing, this is why the weight must be updated in the opposite direction.

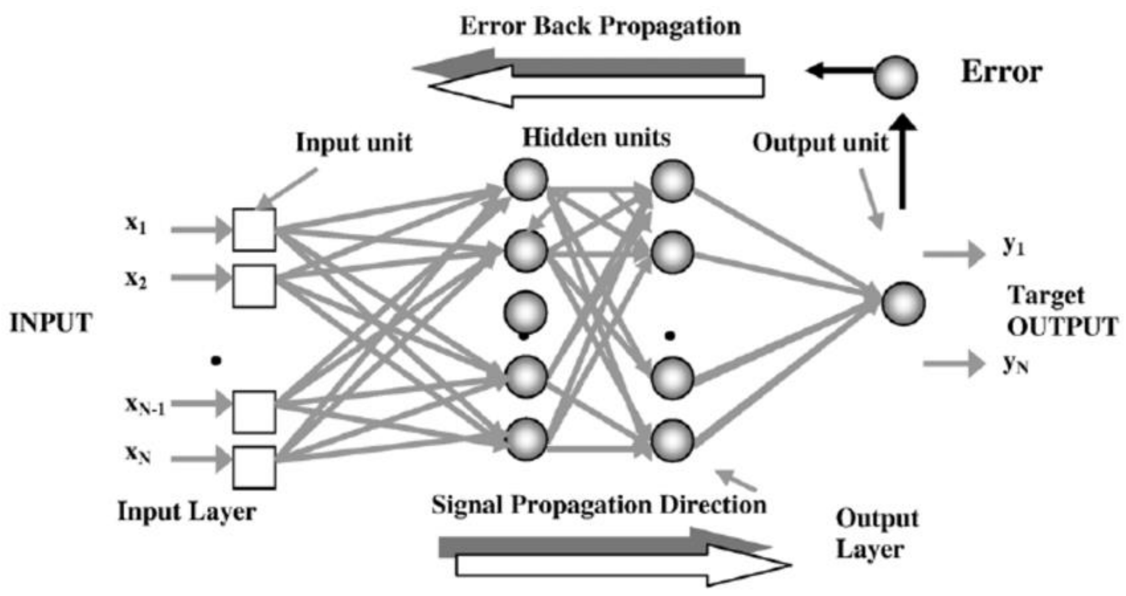


**Fig 4.3.3.1 Backpropagation diagram**

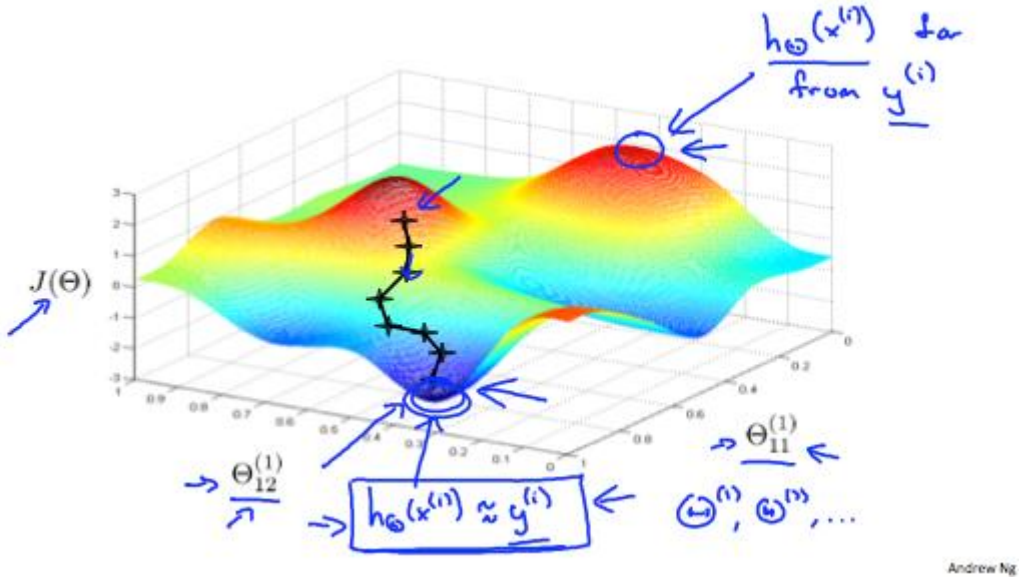


**Fig 4.3.3.2 Implementation of Backpropagation**

## 4.4 ADAPTIVE FILTER:

Input to classifier contains features extracted from noise. Classifier identifies the environment. Then according to the environment determined by the classifier, the noise is passed through particular denoising filter suitable for the environment, to get noise free speech signal

### 4.4.1 PREPROCESSING

Using the open source audio editing and mixing software 'Audacity', for each of the five classes, we have created 30 min of audio samples for feature extraction and training the classifiers, 5 minutes of audio samples for testing the classifiers and 2 minutes of audio samples for final filtering process. These 2 minutes of noise audio samples are then mixed with 2 minutes of foreground speech audio samples to create 'noisy speech' signal which used for training and testing the filters.

There are few types of filter available for different scenario:

### 4.4.2 WEINER FILTER

In signal processing, the Wiener filter is a filter used to produce an estimate of a desired or target random process by linear time-invariant (LTI) filtering of an observed noisy process, assuming known stationary signal and noise spectra, and additive noise. The Wiener filter minimizes the mean square error between the estimated random process and the desired process.

The goal of the Wiener filter is to compute a statistical estimate of an unknown signal using a related signal as an input and filtering that known signal to produce the estimate as an output. For example, the known signal might consist of an unknown signal of interest that has been corrupted by additive noise. The Wiener filter can be used to filter out the noise from the corrupted signal to provide an estimate of the underlying signal of interest. The Wiener filter is based on a statistical approach, and a more statistical account of the theory is given in the minimum mean square error (MMSE) estimator article.

Wiener filters are characterized by the following:

- Assumption: signal and (additive) noise are stationary linear stochastic processes with known spectral characteristics or known autocorrelation and cross-correlation
- Requirement: the filter must be physically realizable/causal (this requirement can be dropped, resulting in a non-causal solution)

- Performance criterion: minimum mean-square error (MMSE)

  This filter is frequently used in the process of deconvolution.

  $H(\omega)=Ps(\omega)/Ps(\omega)+Pv(\omega)$ (16)

  $Ps(\omega)$ and $Pv(\omega)$ are the power spectral densities of the clean and the noise signals

  The signal-to- noise ratio is defined by

  $SNR=Ps(\omega)/ Pv(\omega)$ (17)

  Wiener filter equation is

  $H(\omega) =[1+(1/SNR)]^{-1}$ (18)

### 4.4.3 LMS FILTER

**Least mean squares (LMS)** algorithms are a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean squares of the error signal (difference between the desired and the actual signal). It is a stochastic gradient descent method in that the filter is only adapted based on the error at the current time. This filter adapts (changes) to filter tap weights in order to minimize mean square error. The normalized version of LMS i.e. NLMS is generally used for better results.

#### 4.4.3.1 Standard LMS

The standard LMS algorithm performs the following operations to update the coefficients of an adaptive filter:

1. Calculates the output signal y(n) from the adaptive filter.
2. Calculates the error signal $e(n)$ by using the following equation:

   $e(n) = d(n)–y(n)$. (19)
3. Updates the filter coefficients by using the following equation:

$\overline{w}(n+1) = \overline{w}(n) + \mu \cdot e(n) \cdot \overline{u}(n)$ (21)

where $\mu$ is the step size of the adaptive filter, $\vec{w}(n)$ is the filter coefficients vector, and $\vec{u}(n)$ is the filter input vector.

### 4.4.3.2 Normalized LMS

The normalized LMS (NLMS) algorithm is a modified form of the standard LMS algorithm. The NLMS algorithm updates the coefficients of an adaptive filter by using the following equation:

$$\bar{w}(n+1) = \bar{w}(n) + \mu \cdot e(n) \cdot \frac{\bar{u}(n)}{||\bar{u}(n)||^2} \tag{22}$$

You also can rewrite the above equation to the following equation:

$$\bar{w}(n+1) = \bar{w}(n) + \mu(n) \cdot e(n) \cdot \bar{u}(n) \tag{23}$$

Where

$\mu(n) = \frac{\mu}{||\bar{u}(n)||^2}$.

In the previous equation, the NLMS algorithm becomes the same as the standard LMS algorithm except that the NLMS algorithm has a time-varying step size $\mu(n)$. This step size can improve the convergence speed of the adaptive filter.

## 4.4.4 RLS FILTER

The **Recursive least squares (RLS)** is an adaptive filter which recursively finds the coefficients that minimize a weighted linear least squares cost function relating to the input signals. This is in contrast to other algorithms such as the least mean squares (LMS) that aim to reduce the mean square error. In the derivation of the RLS, the input signals are considered deterministic, while for the LMS and similar algorithm they are considered stochastic. Compared to most of its competitors, the RLS exhibits extremely fast convergence. However, this benefit comes at the cost of high computational complexity.

The standard RLS algorithm performs the following operations to update the coefficients of an adaptive filter.

1. Calculates the output signal y(n) of the adaptive filter.
2. Calculates the error signal $e(n)$ by using the following equation: $e(n) = d(n) - y(n)$.
3. Updates the filter coefficients by using the following equation:

$$\bar{w}(n+1) = \bar{w}(n) + e(n) \cdot \bar{k}(n) \tag{24}$$

where $\vec{w}(n)$ is the filter coefficients vector and $\vec{K}(n)$ is the gain vector. $\vec{K}(n)$ is defined by the following equation:

$$\bar{k}(n) = \frac{P(n)\bar{u}(n)}{\lambda + \bar{u}^T(n) \cdot P(n) \cdot \bar{u}(n)} \tag{25}$$

where $\lambda$ is the forgetting factor and $P(n)$ is the inverse correlation matrix of the input signal.

$P(n)$ has the following initial value $P(0)$:

$$P(0) = \begin{bmatrix} \delta^{-1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \delta^{-1} \end{bmatrix}$$

where $\delta$ is the regularization factor. The standard RLS algorithm uses the following equation to update this inverse correlation matrix.

$$P(n+1) = \lambda^{-1}P(n) - \lambda^{-1}\bar{k}(n) \cdot \bar{u}^T(n) \cdot P(n) \tag{26}$$

## 4.5 ENHANCED/FILTERED OUTPUT CHECKING:

- Once the noise is filtered from the input signal, the enhanced noise-less output signal and the original noisy signal must be compared with each other to find out the efficiency of the particular filter bank.
- The signals can be compared either using objective (Mathematical) measures or subjective measures.
- Some of the objective measures are:
    - SNR (Signal to noise Ratio)
    - IS (Itakura- Saito) distance

### 4.5.1 SNR

- The SNR is the most widely used measure for analog and waveform coding systems and has also been used in the past for assessing enhancement algorithms for broadband noise distortions.

- Several variations exist, including classical SNR, segmental SNR, and frequency-weighted segmental SNR, to name a few. It is important to note that SNR-based measures are only appropriate for coding or enhancement systems that seek to reproduce the original input waveform.
- Let *y(n)* denote a noisy speech signal at time *n*, *s(n)* its noise-free equivalent, and *s(n)* the corresponding enhanced or processed signal. The time error signal can then be written as

  $$\varepsilon(n) = s(i) - \hat{s}(i) \tag{27}$$
- The error energy is then

  $$E_e = \sum_{n=-\infty}^{\infty}\{\varepsilon^2(i)\} = \sum_{n=-\infty}^{\infty}\{s(i) - \hat{s}(i)\}^2 \tag{28}$$
- The energy contained in the speech signal itself is

  $$E_s = \sum_{n=-\infty}^{\infty} s^2(n) \tag{29}$$
- The resulting SNR measure (in dB) is obtained as

  $$SNR = 10log_{10}\left(\frac{E_s}{E_e}\right) = 10log_{10}\left(\frac{\sum_n s^2(n)}{\sum_n [s(n) - \hat{s}(n)]^2}\right) \tag{30}$$

### 4.5.2 ITAKURA-SAITO DISTANCE

- The human auditory system is relatively insensitive to phase distortion. This is particularly important in many speech communication systems (e.g., telephone or radio) and most enhancement algorithms. Therefore, many enhancement and coding systems focus only on the magnitude of the speech spectrum.
- As a result, the coded or enhanced waveform can be quite different from the original, yet still be perceived similarly by the listener.
- Measures such as those based on SNR, which obtain a distortion measure based on sample-by-sample differences in the original and processed time waveforms, do not provide a meaningful measure of performance when the two waveforms differ in their phase spectra. Distance measures that are sensitive to variations in the speech spectrum are therefore needed.
- One of the more successful is the Itakura distance measure. This measure is based on the dissimilarity between all-pole models of the reference and enhanced or coded speech waveforms. The distance measure is computed between sets of LP parameters estimated over synchronous frames (typically every 15-30 ms) in the original and processed speech.
- Itakura distance is not a metric because it does not have the required property of symmetry. That is, if *dl ( · , · )* denotes the Itakura distance, and *â(m)* and *б(m')* two LP vectors between which we desire the distance, then

$$d_1(\hat{a}(m), \hat{b}(m')) \neq d_1\left(\hat{b}(m'), \hat{a}(m)\right)$$

- If a symmetric measure is desired, a combination can be formed such as

$$\tilde{d}_1(\hat{a}(m), \hat{b}(m')) = \frac{1}{2}\left(d_1(\hat{a}(m), \hat{b}(m)) + d_1\left(\hat{b}(m), \hat{a}(m)\right)\right)$$

### 4.5.3 KS Density

[F,XI]=ksdensity(X) computes a probability density estimate of the sample in the vector or two-column matrix X. ksdensity evaluates the density estimate at 100 points (900 points for bivariate data) covering the range of the data. XI is the set of 100 (or 900) points. For bivariate data, XI is created using MESHGRID from 30 equally spaced points in each dimension. F is the vector of density values. The estimate is based on the normal kernel function, using the window parameter (bandwidth) that is a function of the number of points and dimension in X.

Mean & variance calculates mean and variance parameters for noisy, filtered and pure speech. Euclidean distance between noisy and filtered also pure and filtered speech is calculated.

# Chapter 5 IMPLEMENTATION OF SYSTEM

## 5. SUB-MODULE ALGORITHMS

### 5.1 Feature Extraction

Feature extraction using MFCC:

**Input:** Pre-processed audio signal

**Output:** Features for Classification and Recognition

**Method Begins**

**Step 1:** Obtain magnitude spectrum of noise using FFT.

**Step 2:** Map powers of spectrum onto "Mel Scale" using mel-scale filterbanks.

**Step 3:** Take log of power of each mel frequency.

**Step 4:** Take DCT of mel log powers, as if it were a signal.

**Step 5:** Finally, **13** such features will be obtained for classification and recognition.

**Method Ends**

### 5.2 Classification

**Classifier using KNN**

**Input:** Features extracted from one of the feature extraction algorithms

**Output:** The background environment wherein the sound is recorded.

**Method Begins**

**Step 1:** Load the data set

**Step 2:** Use the fitcknn function to create a model for different distance metrics

**Step 3:** Load the background noise sound clip (of about 20 minutes)(test data)

**Step 4:** Use the predict function in matlab to predict the background environment

**Step 5:** Calculatethe accuracy of the test data.

**Step 6:** Repeat this procedure for all the test cases. (Back to step 1)

**Method Ends**

**Classification using Multiclass Logistic Regression (MLR)**

**Input:** Features extracted from one of the feature extraction algorithms

**Output:** The background environment wherein the sound is recorded.

**Step 1:** Divide the data into training and testing set

**Step 2:** Take the training set

**Step 3:** Find the cost using logarithmic cost function

**Step 4:** Initialize all weights equal to 0

**Step 5:** Use sigmoid function to find hypothesis

**Step 6:** Use cost optimization algorithms like gradient descent or convolutional neural network based algorithms like fminunc to minimize the cost function

**Step 7:** Vary regularization parameter lambda depending on weather its an underfitting problem or overfitting problem.

**Step 8:** The sample having highest probability in a particular class is assigned that respective class

**Step 9:** Check the training accuracy.

**Step 10:** Test the test set with the help of weights found out for optimized cost function.

**Step 11:** Repeat steps 4 to 10 (epochs) to get more accuracy.

**Method Ends**

**Classification using Back Propagation (BP)**

**Input:** Features extracted from one of the feature extraction algorithms

**Output:** The background environment wherein the sound is recorded.

**Step 1:** Randomly initialize the weights

**Step 2:** Implement forward propagation to get hΘ(x(i)) for any x(i)

**Step 3:** Implement the cost function

**Step 4:** Implement backpropagation to compute partial derivatives

**Step 5:** Use gradient checking to confirm that your backpropagation works. Then disable gradient checking.

**Step 6:** Use gradient descent or a built-in optimization function to minimize the cost function with the weights in theta.

**Step 7:** When we perform forward and back propagation, we loop on every training example:

For i=1:m

Perform forward propogation and backpropogation using example (x(i),y(i))

Get activations a(1) and delta terms d(1) for l=2,……L

**Step 8:** The capital-delta matrix D is used as an "accumulator" to add up our values(delta values) as we go along and eventually compute our partial derivative.

**Method Ends**

**5.3 Adaptive filter**

**LMS filter**

**Input:** Pre-processed audio signalhaving speech + noise

**Output:** The enhanced noise less audio

**Step 1:** Load the data set

**Step 2:** Calculates the error signal e(n) by using the following equation: e(n) = d(n)–y(n)

**Step 3:** Calculates the output signal y(n) from the adaptive filter.

**Step 4:** Updates the filter coefficients by$\vec{w}(n+1) = \vec{w}(n) + \mu \cdot e(n) \cdot \vec{u}(n)$

where μ is the step size of the adaptive filter, $\vec{w}(n)$ is the filter coefficients vector, and $\vec{u}(n)$ is the filter input vector

**Step 4:** Use the predict function in MATLAB to predict the background environment

**Step 5:** Calculatethe accuracy of the test data.

**Step 6:** Repeat this procedure for all the test cases. (Back to step 1)

**Method Ends**

**RLS filter**

**Input:** Pre-processed audio signalhaving speech+noise

**Output:** The enhanced noise less audio

The RLS algorithm for a *p*-th order RLS filter:

Parameters: *p* = filter order

$\lambda$ = forgetting factor

$\delta$ = value to initialize P(0)

**Step 1:** Initialize

w(n) = 0;

*x(k) = 0, k= -p,……, -1;*

*d(k) = 0, k= -p,……, -1;*

P(0) = $\delta^{-1} * I$ where *I* is the identity matrix of rank p+1;

**Step 2:** Compute: For *n*=1, 2, …….

$$x(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ \vdots \\ x(n-p) \end{bmatrix}$$

$\alpha(n) = d(n) - x^T(n)w(n-1);$

$g(n) = P(n-1)x(n)\{\lambda + x^T(n)P(n-1)x(n)\}^{-1}$ ;

$P(n) = \lambda^{-1}P(n-1) - g(n)x^T(n)\,\lambda^{-1}P(n-1)$ ;

$w(n) = w(n-1) + \alpha(n)g(n)$ ;

**Method Ends**

**5.4 System efficiency measures**

**Signal to noise ratio (SNR)**

**Input:** Pre-processed audio signal having speech + noise, filtered output containing speech and background noise

**Output:** SNR of input audio and enhanced noise less audio

**Step 1:**Compute FFT with a frequency resolution of ~ 10 - 20 Hz. With a sample rate of 48 kHz you would use an FFT length of sample rate/resolution = 4800 samples, which should the get rounded to the nearest power of 2, which is 4096

**Step 2:**Identify the necessary bins which hold the results from 300 - 3000 Hz. The bin index k holds the result for frequency k*sample_rate / FFT_length. For above 48 kHz input and FFT length 4096 this is k(300 Hz) = 300 * 4096 / 48000 ~= 25 and k(3000 Hz) = 3000 * 4096 / 48000 ~= 250.

**Step 3:**Calculate the energy in each necessary bin: E[k] = FFT[k].re ^2 + FFT[k].im ^2. It depends on your FFT algorithm "where" the real and imaginary parts are written.

**Step 4:**N = min{ E[k=25..250] } * number_of_bins (=250-25+1)

**Step 5:**S = sum{ E[k=25..250] }

**Step 6:**SNR = (S-N)/N. The level is 10*log10(SNR)

**Method Ends**

**Itakura-Saito distance (IS)**

**Input:** Pre-processed audio signal having speech + noise, filtered output containing speech and background noise

**Output:** IS distance between input audio and enhanced noise less audio

**Step 1:** Prediction error is distance matrix. Given the autocorrelation matrix R and prediction coefficient vector A, prediction error is

$E=A^TRA$

**Step 2:** If A is optimum predictor and B is any predictor then $\frac{B^TRB}{A^TRA}>=1$

with equality only for A=B

**Step 3:** Likelihood l for unknown sample is

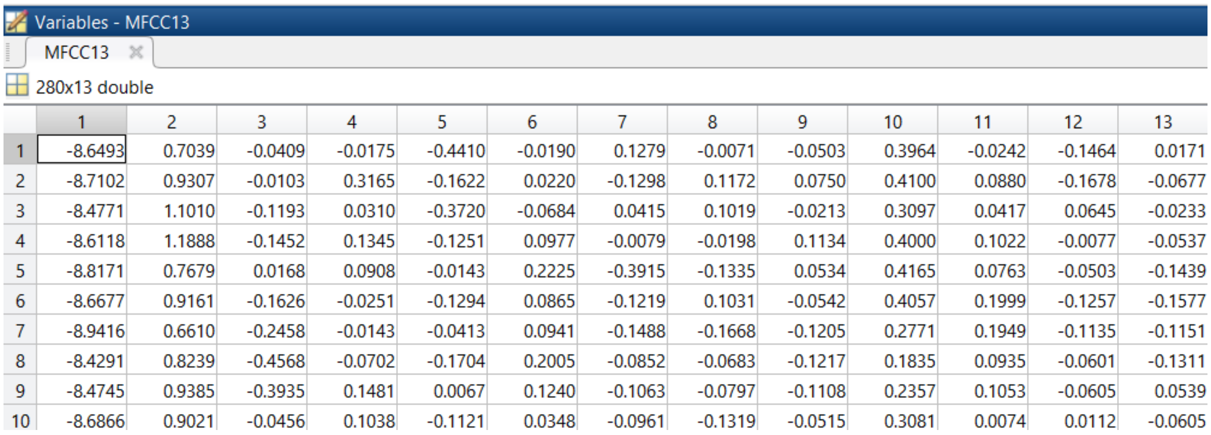
$L=\ln(B^TRB)-\ln(A^TRA)$

**Method Ends**

# Chapter 6 EXPERIMENTAL IMPLEMENTATION AND RESULTS

## 6.1 MFCC

Firstly, we implemented the MFCC algorithm using MATLAB. The program accepts window step time and window overlap time, in milliseconds, as the input arguments for framing the input audio sample. Thirteen MFCC are calculated for each frame of the input audio signal and stored in a single matrix. Rows indicate the frames and columns indicate the MFCC calculated



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -8.6493 | 0.7039 | -0.0409 | -0.0175 | -0.4410 | -0.0190 | 0.1279 | -0.0071 | -0.0503 | 0.3964 | -0.0242 | -0.1464 | 0.0171 |
| 2 | -8.7102 | 0.9307 | -0.0103 | 0.3165 | -0.1622 | 0.0220 | -0.1298 | 0.1172 | 0.0750 | 0.4100 | 0.0880 | -0.1678 | -0.0677 |
| 3 | -8.4771 | 1.1010 | -0.1193 | 0.0310 | -0.3720 | -0.0684 | 0.0415 | 0.1019 | -0.0213 | 0.3097 | 0.0417 | 0.0645 | -0.0233 |
| 4 | -8.6118 | 1.1888 | -0.1452 | 0.1345 | -0.1251 | 0.0977 | -0.0079 | -0.0198 | 0.1134 | 0.4000 | 0.1022 | -0.0077 | -0.0537 |
| 5 | -8.8171 | 0.7679 | 0.0168 | 0.0908 | -0.0143 | 0.2225 | -0.3915 | -0.1335 | 0.0534 | 0.4165 | 0.0763 | -0.0503 | -0.1439 |
| 6 | -8.6677 | 0.9161 | -0.1626 | -0.0251 | -0.1294 | 0.0865 | -0.1219 | 0.1031 | -0.0542 | 0.4057 | 0.1999 | -0.1257 | -0.1577 |
| 7 | -8.9416 | 0.6610 | -0.2458 | -0.0143 | -0.0413 | 0.0941 | -0.1488 | -0.1668 | -0.1205 | 0.2771 | 0.1949 | -0.1135 | -0.1151 |
| 8 | -8.4291 | 0.8239 | -0.4568 | -0.0702 | -0.1704 | 0.2005 | -0.0852 | -0.0683 | -0.1217 | 0.1835 | 0.0935 | -0.0601 | -0.1311 |
| 9 | -8.4745 | 0.9385 | -0.3935 | 0.1481 | 0.0067 | 0.1240 | -0.1063 | -0.0797 | -0.1108 | 0.2357 | 0.1053 | -0.0605 | 0.0539 |
| 10 | -8.6866 | 0.9021 | -0.0456 | 0.1038 | -0.1121 | 0.0348 | -0.0961 | -0.1319 | -0.0515 | 0.3081 | 0.0074 | 0.0112 | -0.0605 |

**Fig 6.1: Thirteen MFCC of first 10 frames of and audio signal**

- MFCC for a number of audio samples with having different environment noise are calculated and saved in .mat files.
- Next, these different .mat files are combined into a single .mat file ‘X’ and another .mat file ‘Y’ is created which contains the class name for each frame.

## 6.2 K-Nearest Neighbor:

- The X.mat and Y.mat files are passed as input arguments to the *fitcknn* in MATLAB to train the system.
- *fitcknn (X, Y)* returns a KNN classification model for predictors X and response Y. X contains the predictor variables. Y array of class labels. Y can be a categorical array, logical vector, numeric vector, or cell array of strings.
- We trained a classification model for two environment classes, namely airport and restaurant. We trained the system using 20 minutes of audio sample for each class, each having 1,19,993 frames.
- The confusion matrices for different values of k neighbors and **Euclidean distance metric** were as follows:

**1. K=4**

| Input Class | Number of frames identified as belonging to Class 1 | Number of frames identified as belonging to Class 2 | Number of frames identified as belonging to Class 3 | Number of frames identified as belonging to Class 4 | Number of frames identified as belonging to Class 5 |
|---|---|---|---|---|---|
| Restaurant (Class 1) | 25709 (85.69%) | 664 (2.21%) | 702 (2.34%) | 399 (1.33%) | 2525 (8.41%) |
| Train Station (Class 2) | 3051 (10.17%) | 23949 (79.83%) | 425 (1.41%) | 44 (0.14%) | 2530 (8.43%) |
| Airport (Class 3) | 834 (2.78%) | 118 (0.39%) | 26523 (88.41%) | 2191 (7.3%) | 333 (1.11%) |
| Rain (Class 4) | 184 (0.61%) | 3 (0.01%) | 2096 (6.98%) | 27257 (90.85%) | 459 (1.53%) |
| Highway Traffic (Class 5) | 1903 (6.34%) | 2877 (9.59%) | 147 (0.49%) | 1214 (4.04%) | 23858 (79.52%) |

**Table 1: Confusion Matrix for k=4 and Euclidean distance metric**

**2. K=8**

| Input Class | Number of frames identified as belonging to Class 1 | Number of frames identified as belonging to Class 2 | Number of frames identified as belonging to Class 3 | Number of frames identified as belonging to Class 4 | Number of frames identified as belonging to Class 5 |
|---|---|---|---|---|---|
| Restaurant (Class 1) | 25442 (84.8%) | 596 (1.98%) | 718 (2.39%) | 414 (1.38%) | 2829 (9.43%) |
| Train Station (Class 2) | 2830 (9.43%) | 23800 (79.33%) | 511 (1.7%) | 50 (0.16%) | 2808 (9.36%) |
| Airport (Class 3) | 589 (1.96%) | 57 (0.19%) | 26474 (88.24%) | 2525 (8.4%) | 359 (1.19%) |
| Rain (Class 4) | 103 (0.34%) | 0 (0.00%) | 1495 (4.98%) | 27950 (93.16%) | 451 (1.5%) |
| Highway Traffic (Class 5) | 1433 (4.77%) | 2674 (8.91%) | 122 (0.40%) | 1215 (4.05%) | 24555 (81.85%) |

**Table 2: Confusion Matrix for k=8 and Euclidean distance metric**

### 3. K=16

| Input Class | Number of frames identified as belonging to Class 1 | Number of frames identified as belonging to Class 2 | Number of frames identified as belonging to Class 3 | Number of frames identified as belonging to Class 4 | Number of frames identified as belonging to Class 5 |
|---|---|---|---|---|---|
| Restaurant (Class 1) | 25227 (84.09%) | 588 (1.96%) | 699 (2.33%) | 456 (1.52%) | 3029 (10.09%) |
| Train Station (Class 2) | 2646 (8.82%) | 23765 (79.21%) | 581 (1.93%) | 54 (0.18%) | 2953 (9.84%) |
| Airport (Class 3) | 480 (1.60%) | 34 (0.11%) | 26324 (87.74%) | 2781 (9.27%) | 380 (1.26%) |
| Rain (Class 4) | 76 (0.25%) | 0 (0.00%) | 1255 (4.18%) | 28207 (94.02%) | 461 (1.53%) |
| Highway Traffic (Class 5) | 1233 (4.11%) | 2636 (8.78%) | 135 (0.45%) | 1278 (4.26%) | 24717 (82.39%) |

**Table 3: Confusion Matrix for k=16 and Euclidean distance metric**

- The confusion matrices for different values of k neighbors and **Chebyshev distance metric** were as follows:

  **1. K=4**

| Input Class | Number of frames identified as belonging to Class 1 | Number of frames identified as belonging to Class 2 | Number of frames identified as belonging to Class 3 | Number of frames identified as belonging to Class 4 | Number of frames identified as belonging to Class 5 |
|---|---|---|---|---|---|
| Restaurant (Class 1) | 25549 (85.16%) | 680 (2.26%) | 715 (2.38%) | 420 (1.4%) | 2635 (8.78%) |
| Train Station (Class 2) | 3139 (10.46%) | 23867 (79.55%) | 445 (1.48%) | 49 (0.16%) | 2499 (8.33%) |
| Airport (Class 3) | 845 (2.81%) | 105 (0.35%) | 26398 (87.99%) | 2309 (7.69%) | 342 (1.14%) |
| Rain (Class 4) | 216 (0.72%) | 1 (0.01%) | 2078 (6.92%) | 27238 (90.79%) | 466 (1.55%) |
| Highway Traffic (Class 5) | 1930 (6.43%) | 2952 (9.84%) | 150 (0.5%) | 1290 (4.3%) | 23677 (78.92%) |

**Table 4: Confusion Matrix for k=4 and Chebyshev distance metric**

2. **K=8**

| Input Class | Number of frames identified as belonging to Class 1 | Number of frames identified as belonging to Class 2 | Number of frames identified as belonging to Class 3 | Number of frames identified as belonging to Class 4 | Number of frames identified as belonging to Class 5 |
|---|---|---|---|---|---|
| Restaurant (Class 1) | 25325 (84.41%) | 597 (1.99%) | 732 (2.44%) | 447 (1.49%) | 2898 (9.66%) |
| Train Station (Class 2) | 2858 (9.52%) | 23736 (79.12%) | 524 (1.74%) | 58 (0.19%) | 2823 (9.41%) |
| Airport (Class 3) | 602 (2.00%) | 44 (0.14%) | 26315 (87.71%) | 2679 (8.93%) | 359 (1.19%) |
| Rain (Class 4) | 132 (0.44%) | 0 (0.00%) | 1465 (4.88%) | 27936 (93.12%) | 466 (1.55%) |
| Highway Traffic (Class 5) | 1436 (4.78%) | 2821 (9.40%) | 134 (0.44%) | 1291 (4.3%) | 24317 (81.05%) |

**Table 5: Confusion Matrix for k=8 and Chebyshev distance metric**

3. **K=16**

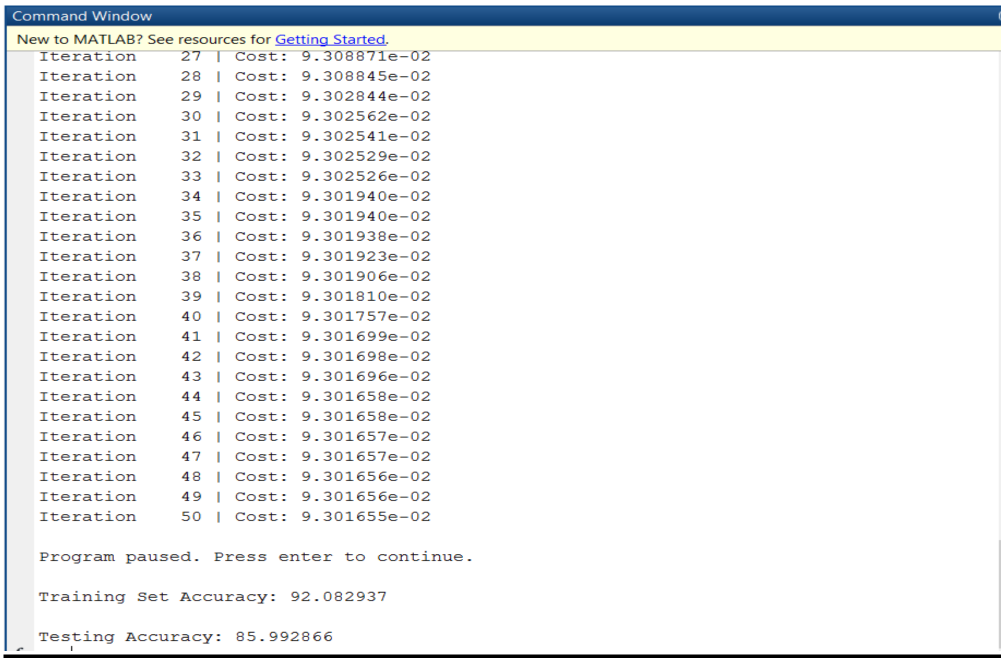
| Input Class | Number of frames identified as belonging to Class 1 | Number of frames identified as belonging to Class 2 | Number of frames identified as belonging to Class 3 | Number of frames identified as belonging to Class 4 | Number of frames identified as belonging to Class 5 |
|---|---|---|---|---|---|
| Restaurant (Class 1) | 25106 (83.68%) | 591 (1.97%) | 720 (2.4%) | 473 (1.57%) | 3109 (10.36%) |
| Train Station (Class 2) | 2700 (9%) | 23581 (78.6%) | 600 (2%) | 58 (0.19%) | 3060 (10.2%) |
| Airport (Class 3) | 490 (1.63%) | 29 (0.09%) | 26178 (87.26%) | 2922 (9.74%) | 380 (1.26%) |
| Rain (Class 4) | 77 (0.25%) | 0 (0.00%) | 1278 (4.26%) | 28166 (93.88%) | 478 (1.59%) |
| Highway Traffic (Class 5) | 1195 (3.98%) | 2737 (9.12%) | 143 (0.47%) | 1355 (4.51%) | 24569 (81.89%) |

**Table 6: Confusion Matrix for k=16 and Chebyshev distance metric**

- **Time required to train the classifier: 2.29 seconds**
- **Average time required to test a 2-minute input audio sample:181.17 seconds**

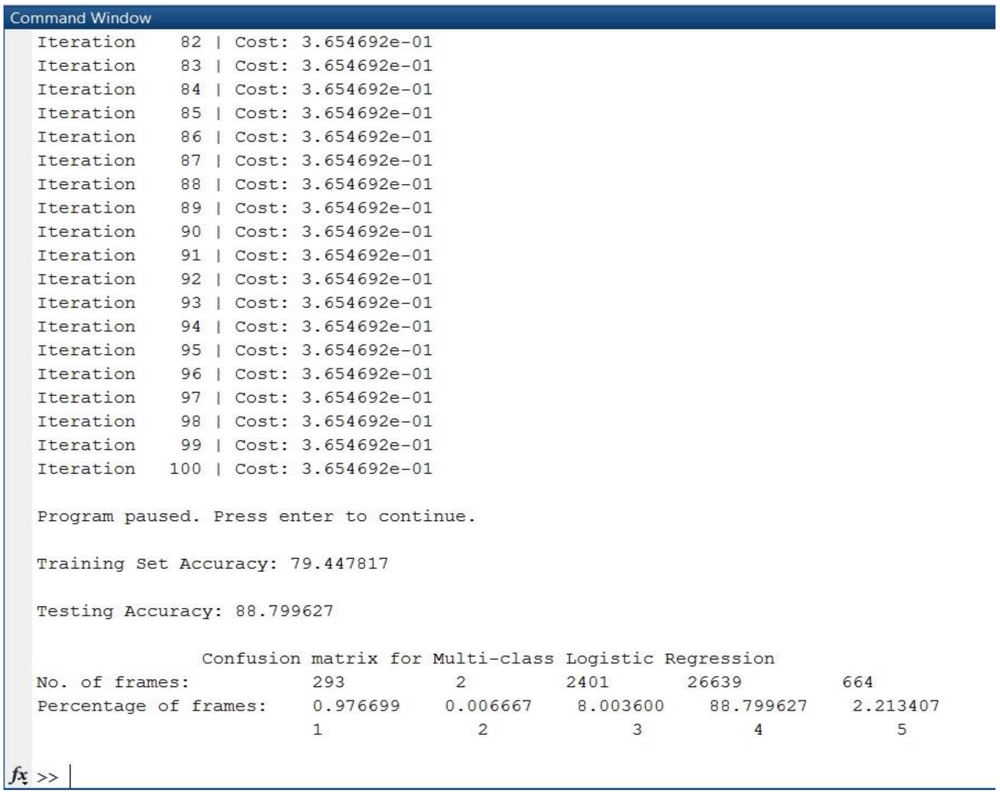
## 6.3 Multiclass Logistic Regression

- We trained a classification model for five environment classes. We trained the system using 30 minutes of audio sample for each class, each having 1,79,990 frames.
- For testing purpose, we have used 2 minutes of audio sample which contains 29,999 frames.



```
Iteration    27 | Cost: 9.308871e-02
Iteration    28 | Cost: 9.308845e-02
Iteration    29 | Cost: 9.302844e-02
Iteration    30 | Cost: 9.302562e-02
Iteration    31 | Cost: 9.302541e-02
Iteration    32 | Cost: 9.302529e-02
Iteration    33 | Cost: 9.302526e-02
Iteration    34 | Cost: 9.301940e-02
Iteration    35 | Cost: 9.301940e-02
Iteration    36 | Cost: 9.301938e-02
Iteration    37 | Cost: 9.301923e-02
Iteration    38 | Cost: 9.301906e-02
Iteration    39 | Cost: 9.301810e-02
Iteration    40 | Cost: 9.301757e-02
Iteration    41 | Cost: 9.301699e-02
Iteration    42 | Cost: 9.301698e-02
Iteration    43 | Cost: 9.301696e-02
Iteration    44 | Cost: 9.301658e-02
Iteration    45 | Cost: 9.301658e-02
Iteration    46 | Cost: 9.301657e-02
Iteration    47 | Cost: 9.301657e-02
Iteration    48 | Cost: 9.301656e-02
Iteration    49 | Cost: 9.301656e-02
Iteration    50 | Cost: 9.301655e-02

Program paused. Press enter to continue.

Training Set Accuracy: 92.082937

Testing Accuracy: 85.992866
```

**Fig 6.3.1: Multiclass Logistic Regression for 3 classes**



```
Iteration    82 | Cost: 3.654692e-01
Iteration    83 | Cost: 3.654692e-01
Iteration    84 | Cost: 3.654692e-01
Iteration    85 | Cost: 3.654692e-01
Iteration    86 | Cost: 3.654692e-01
Iteration    87 | Cost: 3.654692e-01
Iteration    88 | Cost: 3.654692e-01
Iteration    89 | Cost: 3.654692e-01
Iteration    90 | Cost: 3.654692e-01
Iteration    91 | Cost: 3.654692e-01
Iteration    92 | Cost: 3.654692e-01
Iteration    93 | Cost: 3.654692e-01
Iteration    94 | Cost: 3.654692e-01
Iteration    95 | Cost: 3.654692e-01
Iteration    96 | Cost: 3.654692e-01
Iteration    97 | Cost: 3.654692e-01
Iteration    98 | Cost: 3.654692e-01
Iteration    99 | Cost: 3.654692e-01
Iteration   100 | Cost: 3.654692e-01

Program paused. Press enter to continue.

Training Set Accuracy: 79.447817

Testing Accuracy: 88.799627

               Confusion matrix for Multi-class Logistic Regression
No. of frames:            293          2          2401        26639          664
Percentage of frames:     0.976699     0.006667     8.003600     88.799627     2.213407
                          1              2              3            4              5

fx >>
```

**Fig 6.3.2: Multiclass Logistic Regression for 5 classes**

**Confusion Matrix**

Total number of input frames for each class = 29999

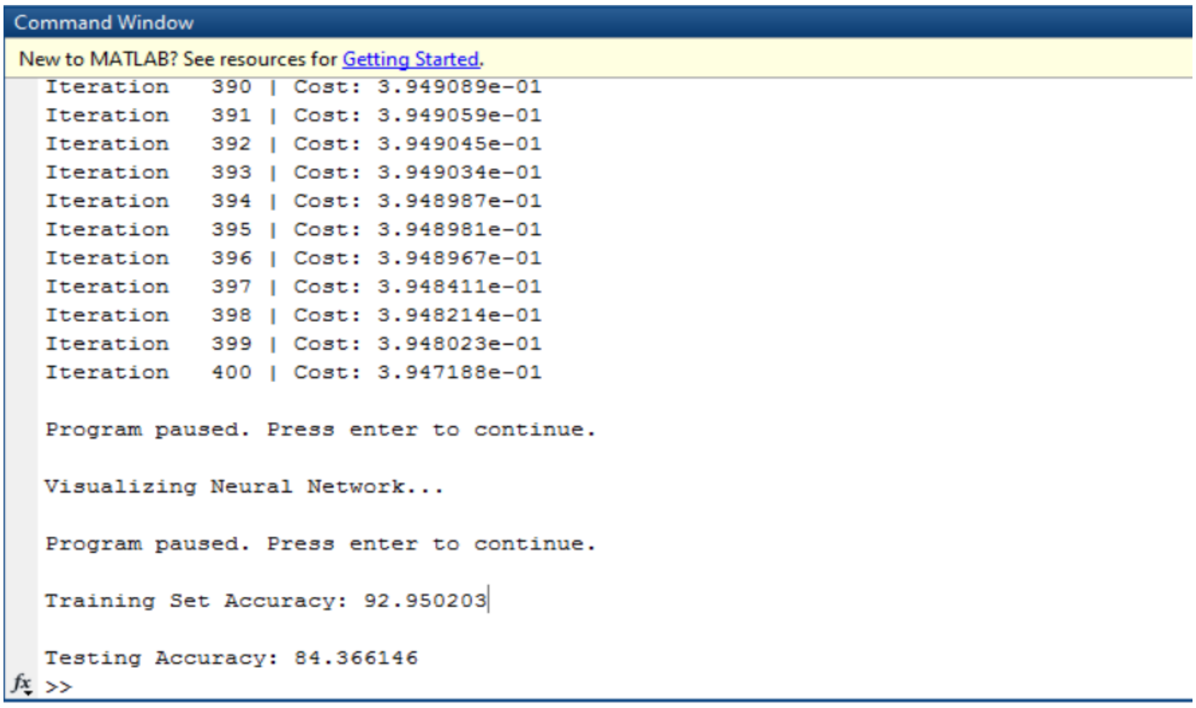
| Input Class | Number of frames identified as belonging to Class 1 | Number of frames identified as belonging to Class 2 | Number of frames identified as belonging to Class 3 | Number of frames identified as belonging to Class 4 | Number of frames identified as belonging to Class 5 |
|---|---|---|---|---|---|
| Restaurant (Class 1) | 24374 (81.24%) | 1396 (4.65%) | 620 (2.06%) | 613 (2.04%) | 2996 (9.98%) |
| Train Station (Class 2) | 2920 (9.73%) | 21929 (73.09%) | 1032 (3.44%) | 100 (0.33%) | 4018 (13.39%) |
| Airport (Class 3) | 1013 (3.37%) | 95 (0.31%) | 25890 (86.30%) | 2517 (8.39%) | 484 (1.61%) |
| Rain (Class 4) | 293 (0.97%) | 2 (0.006%) | 2401 (8%) | 26639 (88.79%) | 664 (2.21%) |
| Highway Traffic (Class 5) | 3236 (10.78%) | 5561 (18.53%) | 131 (0.43%) | 2930 (9.76%) | 18141 (60.47%) |

**Table 7: Confusion Matrix for Multiclass Logistic Regression for 5 classes**

- **Time required to train the classifier: 192.47 seconds**
- **Average time require to test a 2-minute input audio sample:0.1535 seconds**

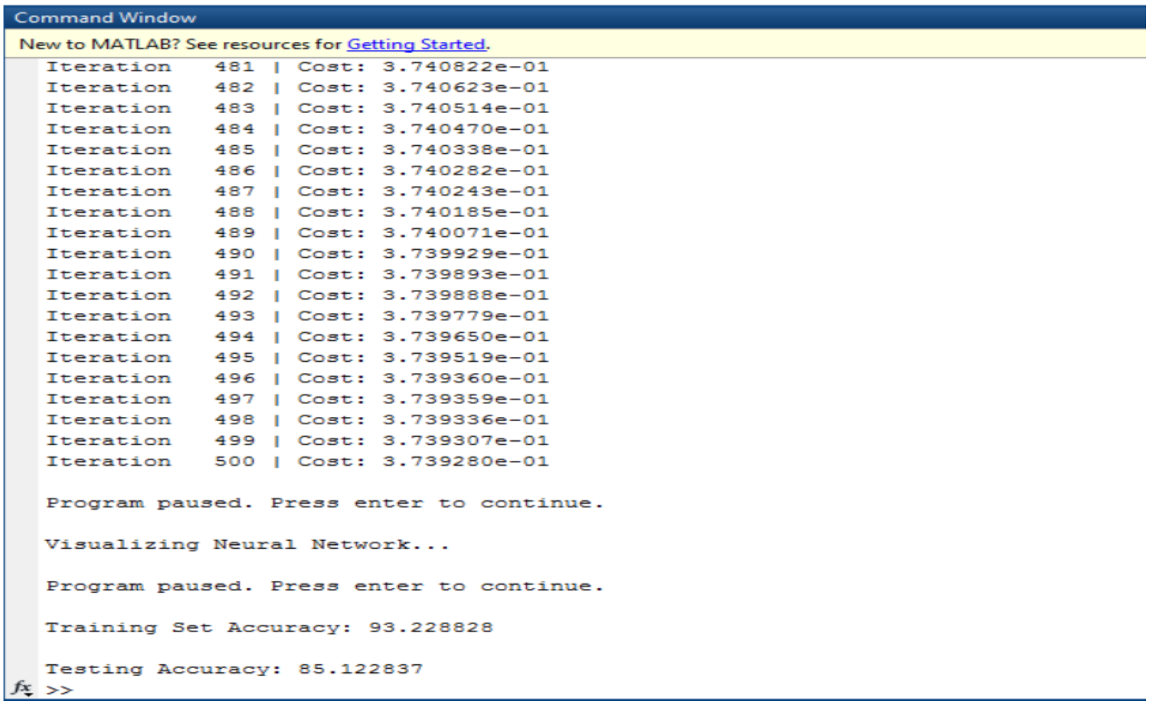
## 6.4 Backpropagation

**For 3 classes:**



```
Command Window
New to MATLAB? See resources for Getting Started.
  Iteration   390 | Cost: 3.949089e-01
  Iteration   391 | Cost: 3.949059e-01
  Iteration   392 | Cost: 3.949045e-01
  Iteration   393 | Cost: 3.949034e-01
  Iteration   394 | Cost: 3.948987e-01
  Iteration   395 | Cost: 3.948981e-01
  Iteration   396 | Cost: 3.948967e-01
  Iteration   397 | Cost: 3.948411e-01
  Iteration   398 | Cost: 3.948214e-01
  Iteration   399 | Cost: 3.948023e-01
  Iteration   400 | Cost: 3.947188e-01

  Program paused. Press enter to continue.

  Visualizing Neural Network...

  Program paused. Press enter to continue.

  Training Set Accuracy: 92.950203

  Testing Accuracy: 84.366146
fx >>
```

**Fig 6.4.1: Backpropagation for alpha=0.7**



```
Iteration    481 | Cost: 3.740822e-01
Iteration    482 | Cost: 3.740623e-01
Iteration    483 | Cost: 3.740514e-01
Iteration    484 | Cost: 3.740470e-01
Iteration    485 | Cost: 3.740338e-01
Iteration    486 | Cost: 3.740282e-01
Iteration    487 | Cost: 3.740243e-01
Iteration    488 | Cost: 3.740185e-01
Iteration    489 | Cost: 3.740071e-01
Iteration    490 | Cost: 3.739929e-01
Iteration    491 | Cost: 3.739893e-01
Iteration    492 | Cost: 3.739888e-01
Iteration    493 | Cost: 3.739779e-01
Iteration    494 | Cost: 3.739650e-01
Iteration    495 | Cost: 3.739519e-01
Iteration    496 | Cost: 3.739360e-01
Iteration    497 | Cost: 3.739359e-01
Iteration    498 | Cost: 3.739336e-01
Iteration    499 | Cost: 3.739307e-01
Iteration    500 | Cost: 3.739280e-01

Program paused. Press enter to continue.

Visualizing Neural Network...

Program paused. Press enter to continue.

Training Set Accuracy: 93.228828

Testing Accuracy: 85.122837
>>
```

**Fig6.4.2: Backpropagation for alpha=1**

**For 5 classes:**

**Confusion Matrix**

Total number of input frames for each class = 29999

| Input Class | Number of frames identified as belonging to Class 1 | Number of frames identified as belonging to Class 2 | Number of frames identified as belonging to Class 3 | Number of frames identified as belonging to Class 4 | Number of frames identified as belonging to Class 5 |
|---|---|---|---|---|---|
| Airport (Class 1) | 24288 (80.96%) | 1299 (4.33%) | 427 (1.42%) | 357 (1.19%) | 3628 (12.09%) |
| Train Station (Class 2) | 1106 (3.68%) | 25755 (85.85%) | 336 (1.12%) | 64 (0.21%) | 2738 (9.21%) |
| Restaurant (Class 3) | 585 (1.19%) | 57 (0.19%) | 27682 (92.27%) | 1365 (4.55%) | 310 (1.03%) |
| Rain (Class 4) | 23 (0.07%) | 0 (0.00%) | 623 (2.07%) | 29285 (97.61%) | 68 (0.22%) |
| Highway Traffic (Class 5) | 3181 (10.60%) | 7103 (23.67%) | 101 (0.33%) | 2283 (7.61%) | 17331 (57.77%) |

**Table 8: Confusion Matrix for Backpropagation for 5 classes**

## Scaled conjugate gradient backpropagation (using MATLAB pattern recognition tool):

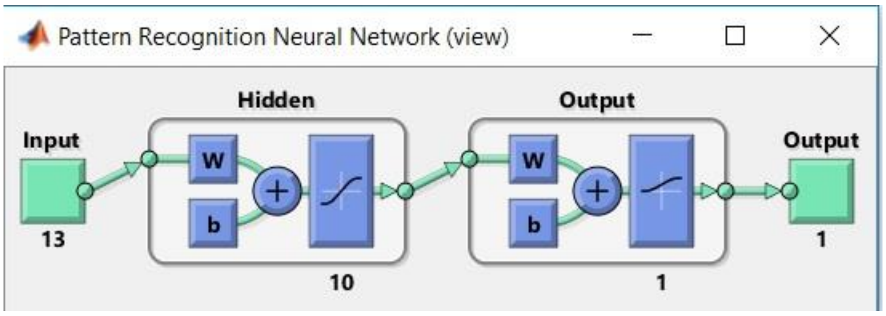
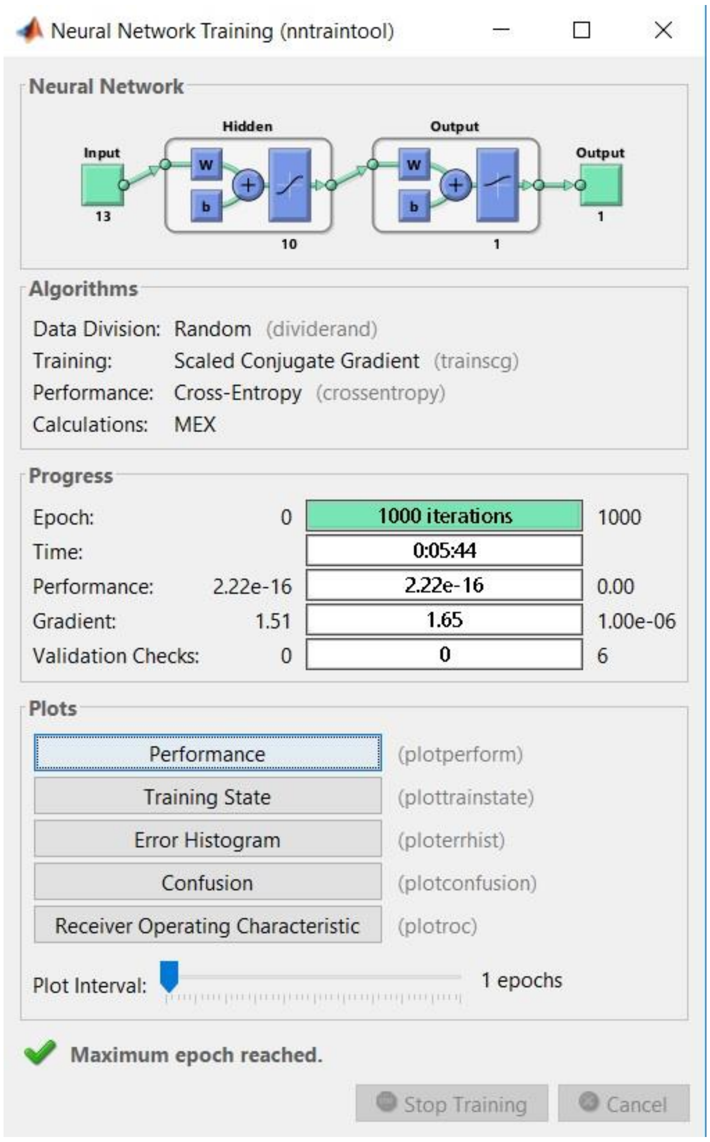


**Fig 6.4.3: Neural network for scaled conjugate gradient backpropagation**


Neural Network Training (nntraintool)
Neural Network
Hidden
Output
Input
W
b
13
10
W
b
1
Output
1

Algorithms

Data Division: Random (dividerand)
Training: Scaled Conjugate Gradient (trainscg)
Performance: Cross-Entropy (crossentropy)
Calculations: MEX

Progress

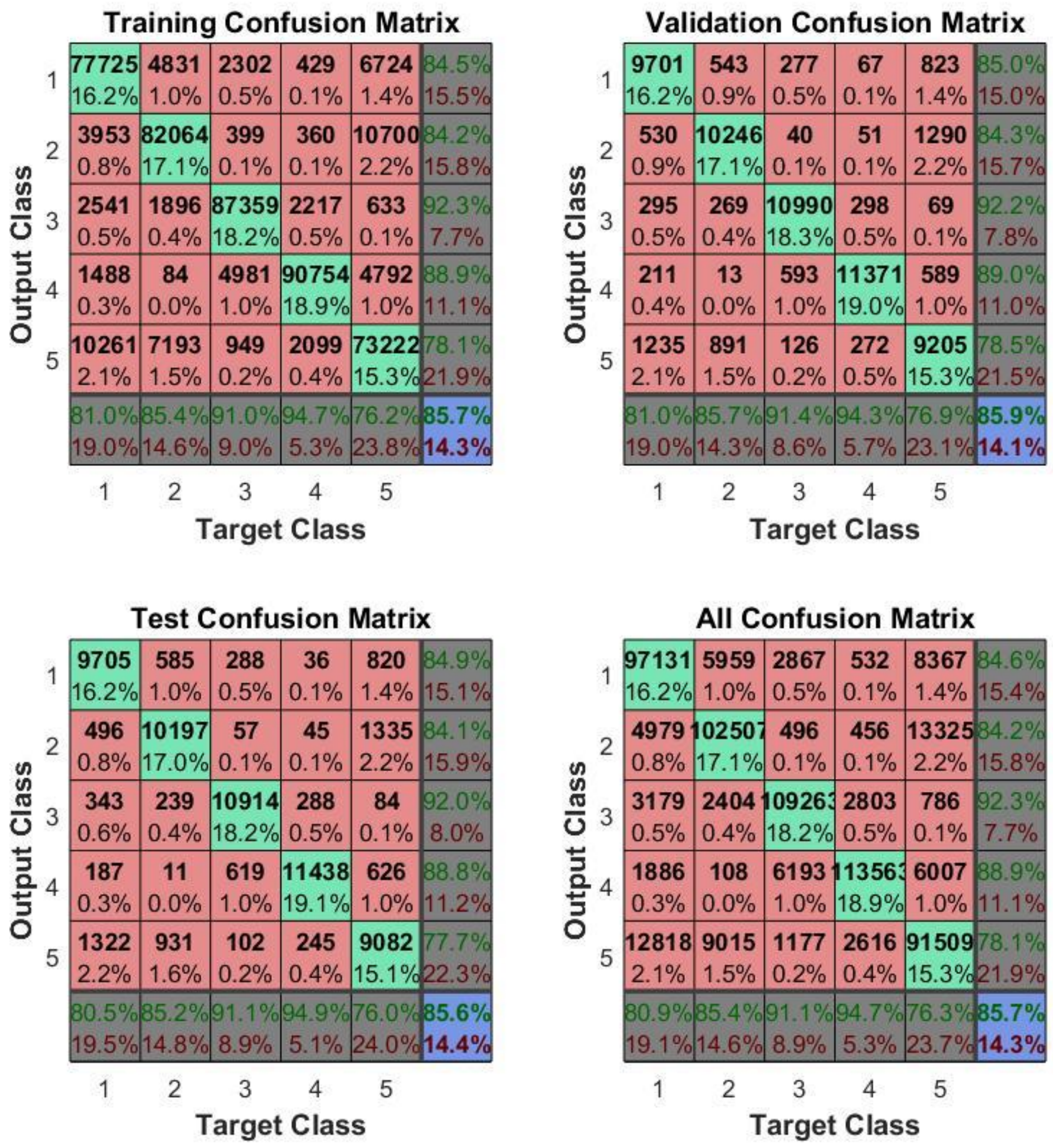
| | | | |
|---|---|---|---|
| Epoch: | 0 | 1000 iterations | 1000 |
| Time: | | 0:05:44 | |
| Performance: | 2.22e-16 | 2.22e-16 | 0.00 |
| Gradient: | 1.51 | 1.65 | 1.00e-06 |
| Validation Checks: | 0 | 0 | 6 |

Plots

Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Confusion (plotconfusion)
Receiver Operating Characteristic (plotroc)
Plot Interval: 1 epochs

✓ Maximum epoch reached.

**Figure 6.4.4: Pattern Recognition Tool**



**Training Confusion Matrix**

| Output Class \ Target Class | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| 1 | 77725<br>16.2% | 4831<br>1.0% | 2302<br>0.5% | 429<br>0.1% | 6724<br>1.4% | 84.5%<br>15.5% |
| 2 | 3953<br>0.8% | 82064<br>17.1% | 399<br>0.1% | 360<br>0.1% | 10700<br>2.2% | 84.2%<br>15.8% |
| 3 | 2541<br>0.5% | 1896<br>0.4% | 87359<br>18.2% | 2217<br>0.5% | 633<br>0.1% | 92.3%<br>7.7% |
| 4 | 1488<br>0.3% | 84<br>0.0% | 4981<br>1.0% | 90754<br>18.9% | 4792<br>1.0% | 88.9%<br>11.1% |
| 5 | 10261<br>2.1% | 7193<br>1.5% | 949<br>0.2% | 2099<br>0.4% | 73222<br>15.3% | 78.1%<br>21.9% |
| | 81.0%<br>19.0% | 85.4%<br>14.6% | 91.0%<br>9.0% | 94.7%<br>5.3% | 76.2%<br>23.8% | **85.7%**<br>**14.3%** |

**Validation Confusion Matrix**

| Output Class \ Target Class | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| 1 | 9701<br>16.2% | 543<br>0.9% | 277<br>0.5% | 67<br>0.1% | 823<br>1.4% | 85.0%<br>15.0% |
| 2 | 530<br>0.9% | 10246<br>17.1% | 40<br>0.1% | 51<br>0.1% | 1290<br>2.2% | 84.3%<br>15.7% |
| 3 | 295<br>0.5% | 269<br>0.4% | 10990<br>18.3% | 298<br>0.5% | 69<br>0.1% | 92.2%<br>7.8% |
| 4 | 211<br>0.4% | 13<br>0.0% | 593<br>1.0% | 11371<br>19.0% | 589<br>1.0% | 89.0%<br>11.0% |
| 5 | 1235<br>2.1% | 891<br>1.5% | 126<br>0.2% | 272<br>0.5% | 9205<br>15.3% | 78.5%<br>21.5% |
| | 81.0%<br>19.0% | 85.7%<br>14.3% | 91.4%<br>8.6% | 94.3%<br>5.7% | 76.9%<br>23.1% | **85.9%**<br>**14.1%** |

**Test Confusion Matrix**

| Output Class \ Target Class | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| 1 | 9705<br>16.2% | 585<br>1.0% | 288<br>0.5% | 36<br>0.1% | 820<br>1.4% | 84.9%<br>15.1% |
| 2 | 496<br>0.8% | 10197<br>17.0% | 57<br>0.1% | 45<br>0.1% | 1335<br>2.2% | 84.1%<br>15.9% |
| 3 | 343<br>0.6% | 239<br>0.4% | 10914<br>18.2% | 288<br>0.5% | 84<br>0.1% | 92.0%<br>8.0% |
| 4 | 187<br>0.3% | 11<br>0.0% | 619<br>1.0% | 11438<br>19.1% | 626<br>1.0% | 88.8%<br>11.2% |
| 5 | 1322<br>2.2% | 931<br>1.6% | 102<br>0.2% | 245<br>0.4% | 9082<br>15.1% | 77.7%<br>22.3% |
| | 80.5%<br>19.5% | 85.2%<br>14.8% | 91.1%<br>8.9% | 94.9%<br>5.1% | 76.0%<br>24.0% | **85.6%**<br>**14.4%** |

**All Confusion Matrix**

| Output Class \ Target Class | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| 1 | 97131<br>16.2% | 5959<br>1.0% | 2867<br>0.5% | 532<br>0.1% | 8367<br>1.4% | 84.6%<br>15.4% |
| 2 | 4979<br>0.8% | 102507<br>17.1% | 496<br>0.1% | 456<br>0.1% | 13325<br>2.2% | 84.2%<br>15.8% |
| 3 | 3179<br>0.5% | 2404<br>0.4% | 109263<br>18.2% | 2803<br>0.5% | 786<br>0.1% | 92.3%<br>7.7% |
| 4 | 1886<br>0.3% | 108<br>0.0% | 6193<br>1.0% | 113563<br>18.9% | 6007<br>1.0% | 88.9%<br>11.1% |
| 5 | 12818<br>2.1% | 9015<br>1.5% | 1177<br>0.2% | 2616<br>0.4% | 91509<br>15.3% | 78.1%<br>21.9% |
| | 80.9%<br>19.1% | 85.4%<br>14.6% | 91.1%<br>8.9% | 94.7%<br>5.3% | 76.3%<br>23.7% | **85.7%**<br>**14.3%** |

**Table 9: Confusion matrix for scaled conjugate gradient backpropagation**

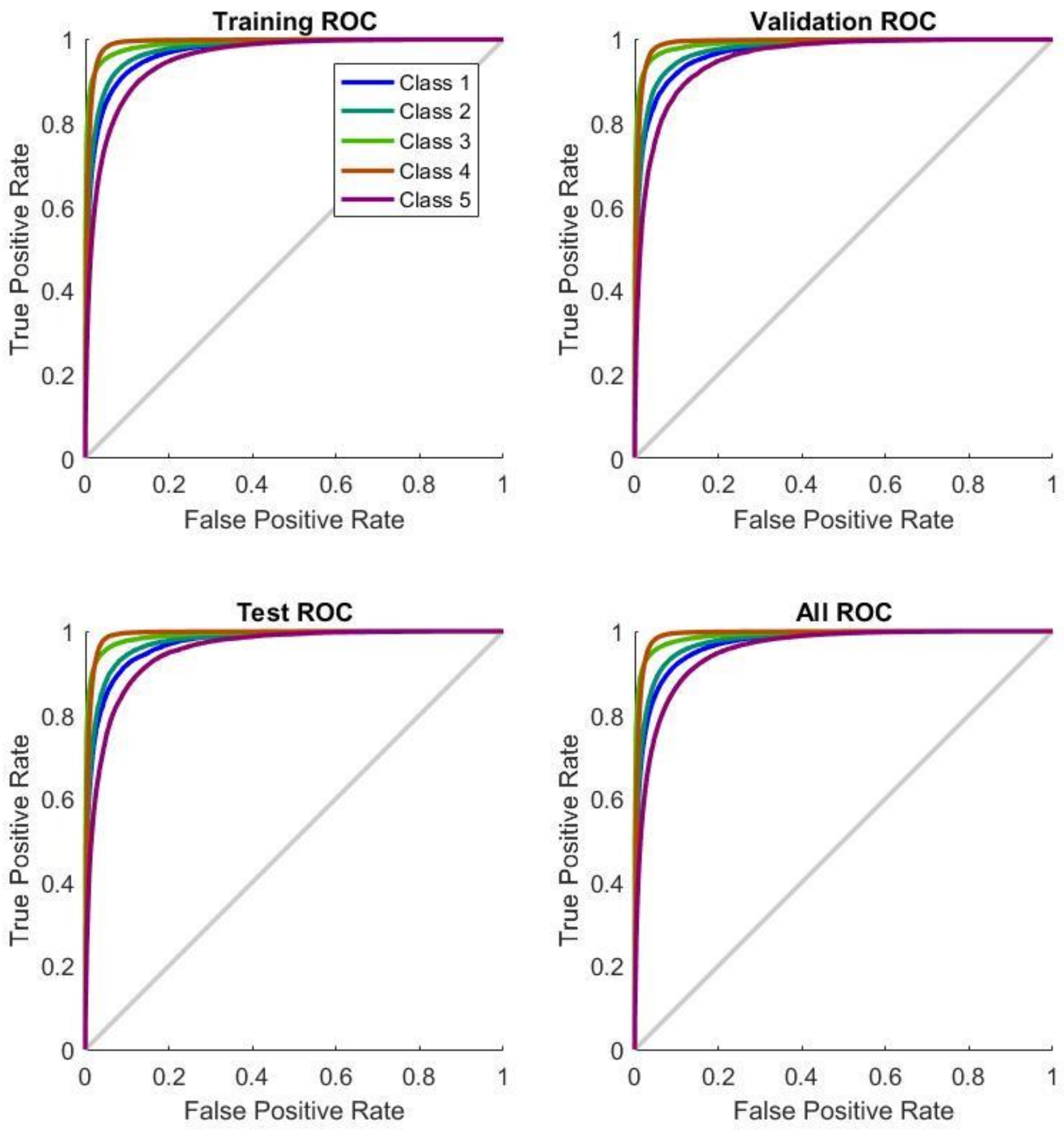


**Fig 6.4.5: Receiver operating characteristics (ROC) for scaled conjugate gradient backpropagation**

- **Time required to train the classifier: 354.02 seconds**
- **Average time require to test a 2-minute input audio sample:0.08 seconds**

## 6.5 LMS filter

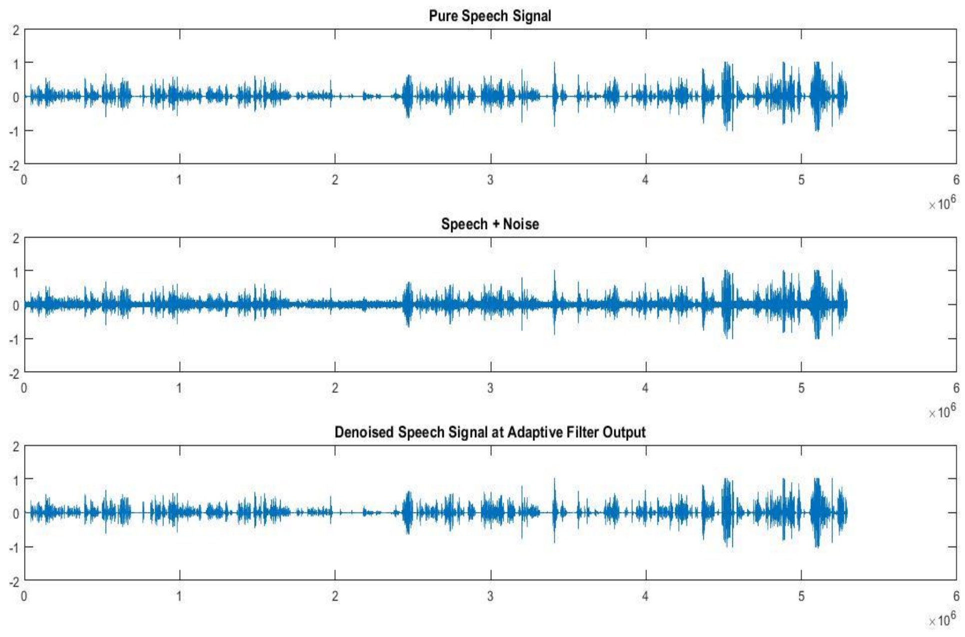


**Fig 6.5 LMS filter implementation**

- **Time required to train the filter: 2.94 seconds**
- **Average time require to test a 2-minute input audio sample:2.79 seconds**

## 6.6 RLS filter

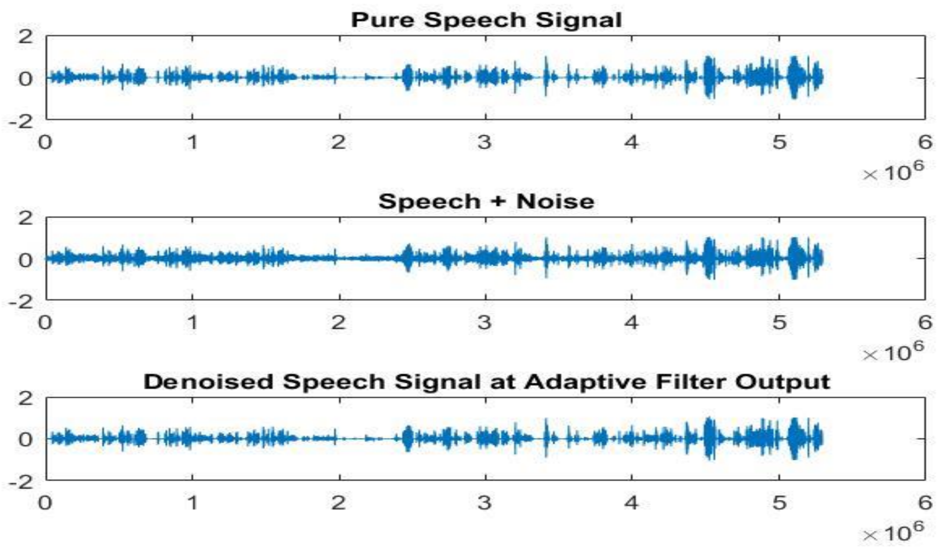


**Fig 6.6 RLS filter implementation**

- **Time required to train the filter: 191 seconds**
- **Average time require to test a 2-minute input audio sample:78 seconds**

## 6.7 IS distance

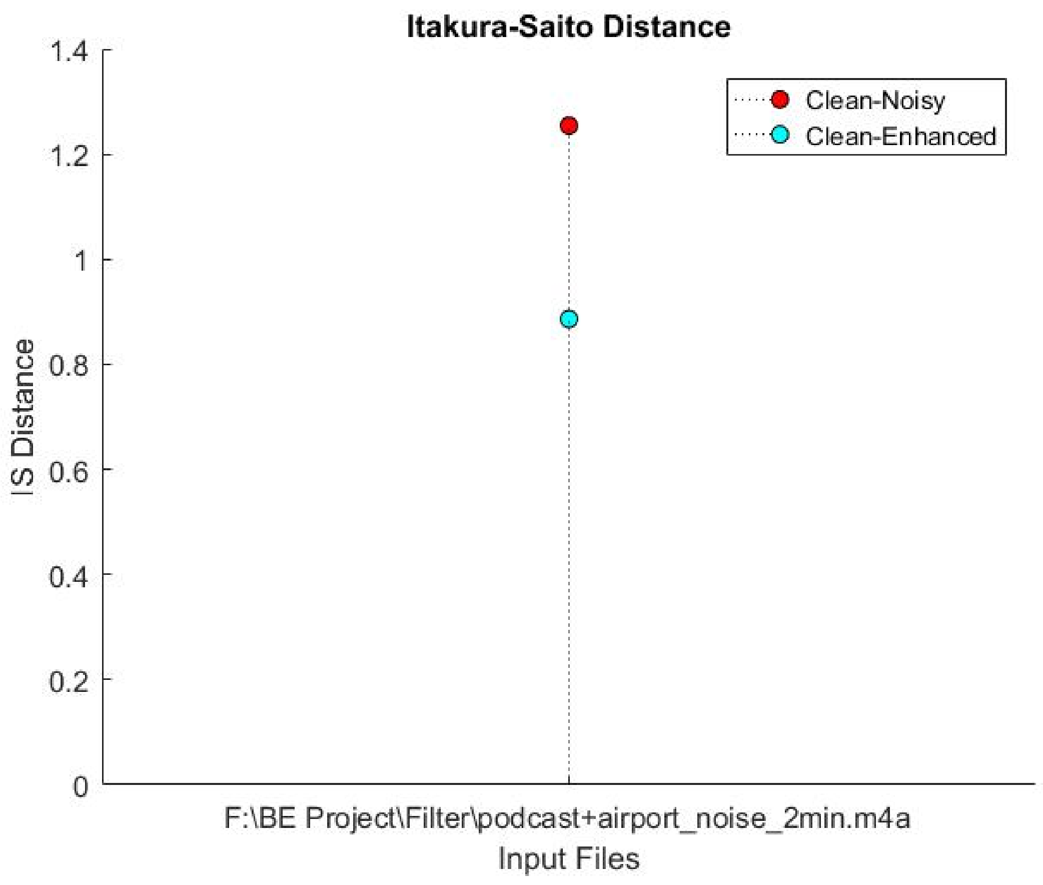


**Fig 6.7 IS distance**

## 6.8 Mean, Variance & Euclidean distance

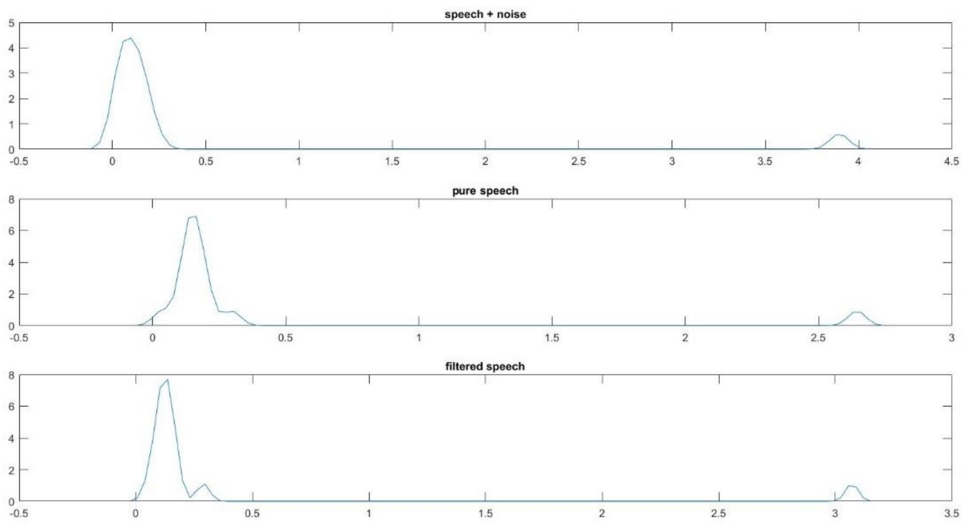


**Fig 6.8.1 KS Density for Airport**

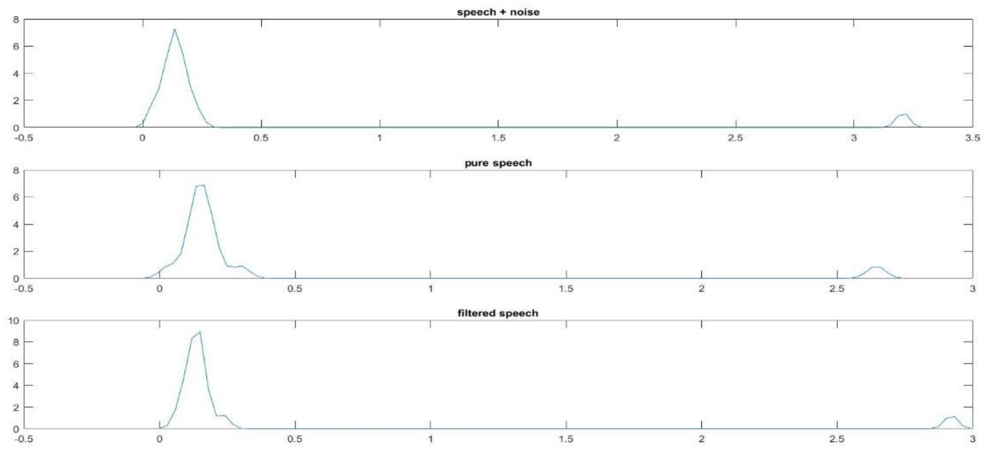


**Fig 6.8.2 KS Density for Train**

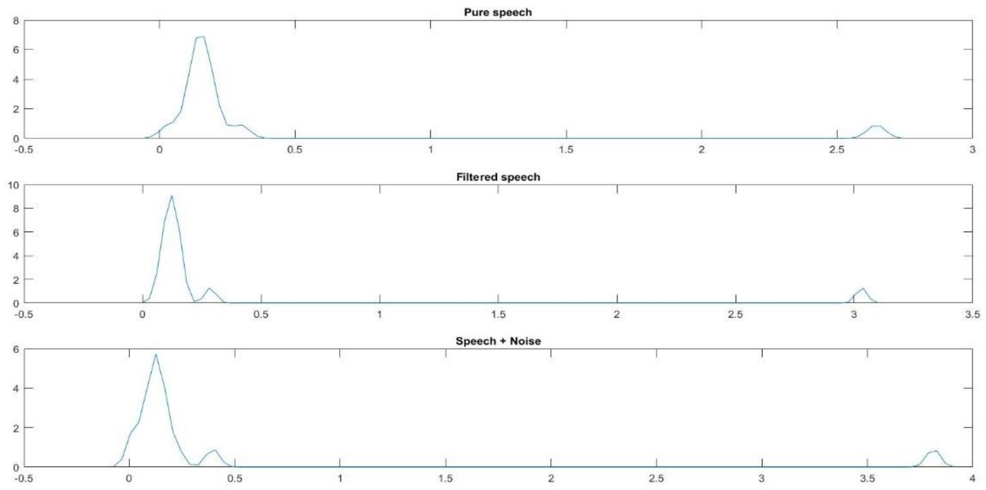


**Fig 6.8.3 KS Density for Restaurant**

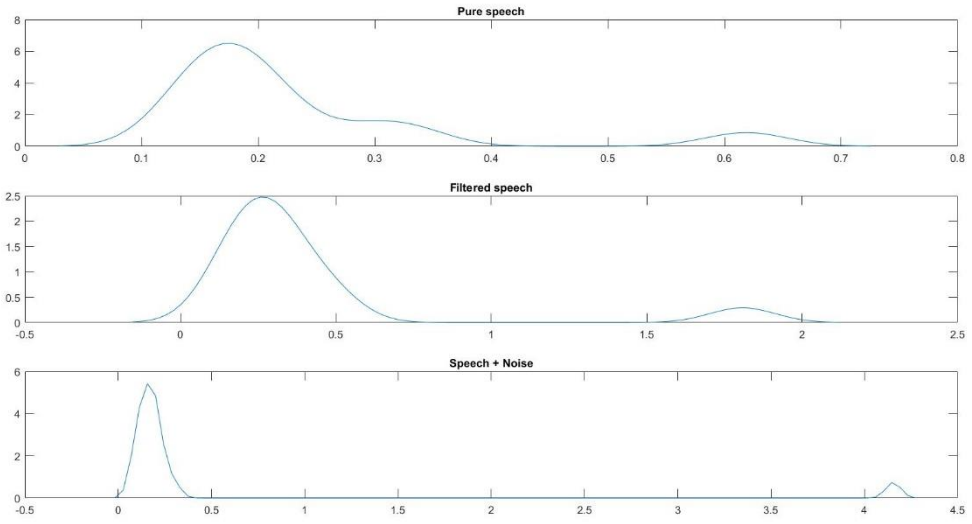


**Fig 6.8.4 KS Density for Rain**

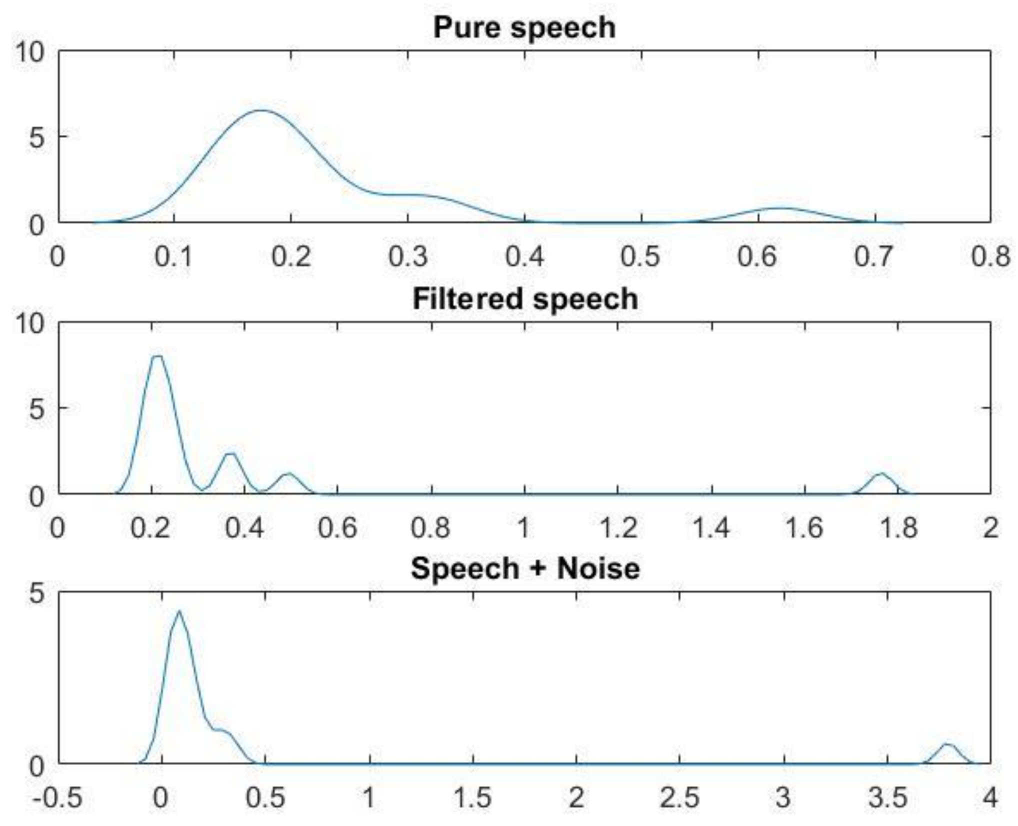


**Fig 6.8.5 KS Density for Highway**

| **Classes** | **Mean (Pure)** | **Mean (Filtered)** | **Mean (Noisy)** | **Variance (Pure)** | **Variance (Filtered)** | **Variance (Noisy)** |
|---|---|---|---|---|---|---|
| Airport | 0.3528 | 0.3117 | 0.2380 | 1.3807 | 1.4632 | 0.6868 |
| Restaurant | 0.3528 | 0.3307 | 0.2983 | 1.3807 | 1.8375 | 1.2600 |
| Railway | 0.3528 | 0.3191 | 0.2484 | 1.3807 | 1.7452 | 0.7540 |
| Highway | 1.4232 | 0.5755 | 0.2438 | 3.5988 | 2.3042 | 0.6007 |
| Rain | 1.4232 | 0.4337 | 0.2306 | 3.5988 | 0.5270 | 0.8002 |

**Table 10: Mean & Variance for different classes**

| **Classes** | **Euclidean Distance between filtered & noisy speech** | **Euclidean Distance between filtered & pure speech** |
|---|---|---|
| Airport | 51.0489 | 147.0487 |
| Restaurant | 25.6635 | 213.6928 |
| Railway | 79.5306 | 241.1127 |
| Highway | 215.1399 | 690.8681 |
| Rain | 315.2245 | 307.9755 |

**Table 11: Euclidean distance for different classes**

## 6.9 Signal to noise ratio (SNR)

| Classes | SNR on input side(dB) | SNR on output side(dB) |
|---|---|---|
| Restaurant | 14.2371 | 14.4242 |
| Railway | 1.7566 | 3.9946 |
| Airport | 7.2801 | 8.0727 |
| Rain | 10.6349 | 11.0269 |
| Highway | 6.3507 | 7.2725 |

**Table 12: SNR for different classes**

## 6.10 CONCLUSION

In this project, we have proposed classification of background noise for speech denoising. Pre-processing is manually done. The pre-processed audio contains background noise only. Feature extraction using MFCC is done. 13 coefficients for each frame have been taken for better accuracy and result. All these coefficients are passed through classifier block for various classification algorithms.

It is being found that:

KNN classifier:

- For 2 classes-
    1. 13 features each of 280 frames are extracted from noise.

    2. All these features are given to KNN classifier. Efficiency of KNN is greatest for value of k=8.
    3. Euclidean distance metric is more accurate than cosine or correlation distance metric.
- For 3 classes-
    1. As number of classes increases, efficiency of KNN decreases.
    2. SNR value of input sample plays vital role as recognition rate is better at 10,15 dB
    3. KNN failed to identify station environment
- For 5 classes-
    1. Classification efficiency is better at k=8 for both distances.
    2. Time required for testing is very high.
    3. As training data set is increased, efficiency increases.

Multiclass Logistic Regression:

1. Training and testing time for MLR is very low. Hence better time efficiency
2. Classification efficiency is very good.
3. Computationally efficient.

Backpropagation:

1. Very high training time.
2. Classification efficiency is very good.
3. As number of classes increases, system becomes more complex.

LMS adaptive filter:

- After classifying the class of the speech, input is passed through class specific LMS filter.
- At the output, noise was successfully removed. Also, quality of speech was deteriorated up to some extent.
- Stable and robust performance against all signal conditions
- Filter has low convergence rate due to eigen spread but minimizes mean square error.

RLS adaptive filter:

- High convergence rate.
- The background noise has been successfully removed.
- Testing time is high as computational complexity is high.

- High filter stability.

KS Density:

- Filtered speech is close to the pure speech than to noisy speech.
- Euclidean distance between filtered and noisy as well as between filtered and pure speech

## 6.11 FUTURE SCOPE

Our future work aims to implement this system for more number of classes. Also, we are going to improve the classifier efficiency to achieve better recognition rate. We are aiming to design real time system to classify and reject background noise. We would like to improve filter efficiency. Algorithms for system efficiency measures like IS distance, SNR will be optimized.

# CHAPTER 7 REFERENCES

[1] Nitish Krishnamurthy John H.L. Hansen, "Environment dependent noise tracking for speech enhancement", Int J Speech Technol (2013) 16:303–312 DOI 0.1007/s10772-012-9182-0.

[2] Ling Ma, Dan Smith, and Ben Milner, "Environmental Noise Classification for Context-Aware Applications", Springer-Verlag Berlin Heidelberg 2003.

[3] Stéphane Bressan and Boon Tiang, "Environmental Noise Classification for Multimedia Libraries", Springer-Verlag Berlin Heidelberg 2005.

[4] Mohanapriya. S. P, E. P. Sumesh, R. Karthika, "Environmental Sound Recognition Using Gaussian Mixture Model and Neural Network Classifier".

[5] Souli Sameh Zied Lachiri, "Multiclass support vector machines for environmental sounds classification in visual domain based on log-Gabor filters", Int J Speech Technol (2013) 16:203–213 DOI 10.1007/s10772-012-9174-0.

[6] Oscar Varela, Ruben San-Segundo & Luis A. Hernandez, "Robust Speech Detection for Noisy Environments"

[7] Jyoti Dhiman, Shadab Ahmad, Kuldeep Gulia, "Comparison between adaptive filter algorithms (LMS, NLMS, RLS)"