

Stuttered Speech Recognition

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology in **Electronics and Communication Engineering**

by

Anshuman S. Dhamoon

12BEC0111

Sarthak Khanna

12BEC0120

Under the guidance of

Prof. Sankar Ganesh S

**School of Electronics Engineering,
VIT University, Vellore.**



VIT[®]
UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)

VELLORE ■ CHENNAI

www.vit.ac.in

May, 2016

DECLARATION

I hereby declare that the thesis entitled “**Stuttered Speech Recognition**” submitted by Anshuman S. Dhamoon and Sarthak Khanna, for the award of the degree of *Bachelor of Technology in Communication Engineering* to VIT University is a record of bonafide work carried out by me under the supervision of Prof. Sankar Ganesh S.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date : May, 2016

Signature of the Candidate

CERTIFICATE

This is to certify that the Project work titled “**Stuttered Speech Recognition**” that is being submitted by Anshuman S. Dhamoon and Sarthak Khanna in partial fulfilment of the requirements for the award of Bachelor of Technology [Electronics and Communication Engineering], is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Guide

The Thesis is satisfactory/ unsatisfactory

Internal Examiner

External Examiner

Approved By

ACKNOWLEDGEMENT

I would like to the Management of VIT University, our Chancellor, Vice Chancellor and other esteemed members of the management of VIT University for making the final year project compulsory to be carried out by all it students. It is one of the most important learning source for the students both practically as well as theoretically.

I would also like to thank the Dean, Dr.Elizabeth Rufus and HoD of SENSE school, Prof. S.Renuga Devi for their support and guidance.

I would like to thank my guide Prof. Sankar S Ganesh for giving me this project. Without his guidance and constant motivation, this project would never have been completed.

I would also like to thank the Panel members of the reviews for their advice and assistance which constantly motivated me to carry out this project.

Also, thank you to my family and friends for tolerating my regular madness during this project.

Anshuman S. Dhamoon

12BEC0111

Sarthak Khanna

12BEC0120

ABSTRACT

Stuttering is a speech disorder in which the flow of speech is disrupted by involuntary repetitions and prolongations of sound, syllables, words or phrases as well as involuntary silent pauses and it affects millions of people in today's world. The aim of this project is to come up with a new algorithm to enhance speech recognition for people suffering from stuttering. Since the speech will be different for different people, we'll be using ANN to make the system learn for each individual separately. The basic idea is to first remove stuttering from the sample by using the amplitude threshold obtained from neural networks and then passing the clean sample through Google's Speech Recognition API. To obtain the amplitude threshold, we need to keep a large dataset so the technique can be generalised for everyone. The major problem until now is that a system stops recognizing after a pause is encountered in the speech and hence, the average accuracy of stuttered speech recognition is around 70%. With a new algorithm which will take into account the words or characters after the pause and then use that also for recognition, the accuracy can be improved.

INDEX

	LIST OF TABLES	1
	LIST OF FIGURES	2
1	INTRODUCTION	4
	1.1 Overview	4
	1.2 Thesis Outline	5
2	BACKGROUND THEORY	6
	2.1 Introduction to stuttering	6
	2.2 Detection of stuttered speech	8
	2.2.1 Artificial Neural Networks	9
	2.2.2 Hidden Markov Models	22
	2.2.3 Support Vector Machines	24
3	SPEECH RECOGNITION	33
	3.1 Introduction	33
	3.2 Existing Solutions	33
	3.2.1 Hidden Markov models	34
	3.2.2 Dynamic time warping	36
	3.2.3 Neural networks	37
	3.2.4 Deep Feedforward & Recurrent NN	38
	3.3 Our Solution	39
	3.3.1 Introduction	39
	3.3.2 Theory	39

3.3.3	Algorithm	46
3.3.4	Implementation	47
3.3.5	Results	49
4	STUTTERED SPEECH CORRECTION	50
4.1	Proposed Solution	50
4.2	Algorithm	51
4.3	Implementation	52
4.4	Simulated Results	54
4.4.1	Plots	54
4.4.2	Tabulated Results	56
5	CONCLUSION AND FUTURE SCOPE	57
8.1	Conclusion	57
8.2	Future Scope	58
	REFERENCES	59

LIST OF TABLES

Table No.	Table Name	Page No.
Table 3.1	Results obtained	49
Table 4.1	Results obtained	56

LIST OF FIGURES

Figure No.	Figure Name	Page No.
Figure 2.1	Basic speech detection	8
Figure 2.2	Basic ANN	9
Figure 2.3	McCulloch-Pitts model	11
Figure 2.4	Simple Perceptron	12
Figure 2.5	Perceptron updating its linear boundary as more training examples are added.	14
Figure 2.6	Back Propagation Network model	15
Figure 2.7	The Sigmoid function	16
Figure 2.8	The BPN notation	17
Figure 2.9	General architecture of HMM	23
Figure 2.10	SVM Plane	25
Figure 2.11	Basic SVM	26
Figure 2.12	Hyperplane scenario 1	27
Figure 2.13	Hyperplane scenario 2a	27
Figure 2.14	Hyperplane scenario 2b	28
Figure 2.15	Hyperplane scenario 3a	28
Figure 2.16	Hyperplane scenario 3b	29
Figure 2.17	Hyperplane scenario 4a	29
Figure 2.18	Hyperplane scenario 4b	30
Figure 2.19	Simple kernel trick	31
Figure 3.1	Linear regression sample	41
Figure 3.2	Gradient Decent	42
Figure 3.3	Filtering of speech sample	45
Figure 3.4	Inputting the speech samples with labels	47
Figure 3.5	Testing the program	48

Figure 4.1	Stuttered and Clean Sample A	54
Figure 4.2	Stuttered and Clean Sample B	55
Figure 4.3	Stuttered and Clean Sample C	

Chapter 1

Introduction

1.1 OVERVIEW

In today's world there are millions of persons suffering from various speech disorders like stuttering, lisp, etc., this often renders them unable to utilize certain things that we take for granted, like speech recognition systems. Stuttering is one such speech disorder affecting the fluency of speech. It begins during childhood and, in some cases, lasts throughout life. The disorder is characterized by disruptions in the production of speech sounds, also called "disfluencies." Most people produce brief disfluencies from time to time. For instance, some words are repeated and others are preceded by "um" or "uh." Disfluencies are not necessarily a problem; however, they can impede communication when a person produces too many of them.

In most cases, stuttering has an impact on at least some daily activities. The specific activities that a person finds challenging to perform vary across individuals. For some people, communication difficulties only happen during specific activities, for example, talking on the phone or talking before large groups, utilizing everyday tools that use speech as inputs.

Currently, the speech recognition systems have a great accuracy for fluent speech but are unable to recognise speech with repetitions or long involuntary pauses, i.e. stuttering. This is mainly because the systems are created to stop the identification process when a pause is encountered. Also, these systems are trained with proper words without any repetitions and so, when it encounters a stuttered speech, it is unable to identify the words, since it hasn't been trained to do so.

Many people have detected stuttering from speech samples, however they haven't corrected the sample. They have used ANN, HMM, SVM to name a few and advanced DSP to remove noise from the samples and correct them. Our project aims to detect as well as correct these stuttered speech samples on the fly and then give the corrected speech sample devoid of stuttering. We will be using neural networks and DSP to detect and correct the speech. This

system can then be integrated with phones and laptops and help people suffering from this speech impairment to control their devices with speech, the same way as most of the population in today's world does.

Thus our main aim is to help these people by making certain already accessible tools available to them, without worrying about their speech impairment.

1.2 THESIS OUTLINE

The project work aims at identifying and correcting stuttered speech. The speech sample is corrected using MATLAB and a new file is created which is without any stuttering. This new file is passed into the open sourced Google Speech API, which recognises the words and these words can now be input into any other application.

In order to get a better understanding of speech recognition we have also made a speech recognition algorithm using a back propagating neural network to recognise 10 words.

This thesis is divided into multiple sections consisting of the various possible methods to solve the problem and their background theories, proceeded by the detailed description of all the algorithms and their functioning. To conclude, the results are presented which have been obtained from the simulations performed in various software and conclusions are presented which have been deduced from the results acquired.

Chapter 2

Background Theory

2.1 INTRODUCTION TO STUTTERING

Stuttering (or Stammering) is a speech disorder in which the flow speech is disrupted by involuntary repetitions and prolongations of sounds, syllables, words or phrases as well as involuntary silent pauses or blocks in which the person who stutters is unable to produce sounds. The term stuttering is most commonly associated with involuntary sound repetition, but it also encompasses the abnormal hesitation or pausing before speech, referred to by people who stutter as blocks, and the prolongation of certain sounds, usually vowels or semivowels. For many people who stutter, repetition is the primary problem. The term "stuttering" covers a wide range of severity, encompassing barely perceptible impediments that are largely cosmetic to severe symptoms that effectively prevent oral communication. In the world, approximately 70 million people, or about 1% of the world's total population stutters. The impact of stuttering on a person's functioning and emotional state can be severe. This may include fears of having to enunciate specific vowels or consonants, fears of being caught stuttering in social situations, self-imposed isolation, anxiety, stress, shame, being a possible target of bullying (especially in children), having to use word substitution and rearrange words in a sentence to hide stuttering, or a feeling of "loss of control" during speech. Stuttering is sometimes popularly seen as a symptom of anxiety, but there is actually no direct correlation in that direction.

Some examples of stuttering include:

- "W- W- W- Where are you going?" (Part-word repetition: The person is having difficulty moving from the "w" in "where" to the remaining sounds in the word. On the fourth attempt, he successfully completes the word.)

- "SSSS ave me a seat." (Sound prolongation: The person is having difficulty moving from the "s" in "save" to the remaining sounds in the word. He continues to say the "s" sound until he is able to complete the word.)
- "I'll meet you - um um you know like - around six o'clock." (A series of interjections: The person expects to have difficulty smoothly joining the word "you" with the word "around." In response to the anticipated difficulty, he produces several interjections until he is able to say the word "around" smoothly.)

Stuttering is generally not a problem with the physical production of speech sounds or putting thoughts into words. Acute nervousness and stress do not cause stuttering, but they can trigger stuttering in people who have the speech disorder, and living with a highly stigmatized disability can result in anxiety and high allosteric stress load (i.e., chronic nervousness and stress) that reduce the amount of acute stress necessary to trigger stuttering in any given person who stutters, exacerbating the problem in the manner of a positive feedback system; the name 'stuttered speech syndrome' has been proposed for this condition. Neither acute nor chronic stress, however, itself creates any predisposition to stuttering.

The disorder is also variable, which means that in certain situations, such as talking on the telephone or in a large group, the stuttering might be more severe or less, depending on whether or not the stutterer is self-conscious about their stuttering. Stutterers often find that their stuttering fluctuates and that they have "good" days, "bad" days and "stutter-free" days. The times in which their stuttering fluctuates can be random. Although the exact aetiology of stuttering is unknown, both genetics and neurophysiology are thought to contribute. There are many treatments and speech therapy techniques available that may help increase fluency in some people who stutter to the point where an untrained ear cannot identify a problem; however, there is essentially no cure for the disorder at present. The severity of the person's stuttering would correspond to the amount of speech therapy needed to increase fluency. For severe stuttering, long-term therapy and hard work will be required to increase fluency.

2.2 DETECTION OF STUTTERING

Detection is one of the main steps in the recognition of any speech sample. There are various methods to detect stuttered speech like Artificial Neural Networks, Hidden Markov Models, Standard Vector Machines and Digital Signal Processing and all of these require some degree of DSP in order to pre-process the sound signal, that is, removing various noises from the sample and making the samples uniform so that they can be worked upon. The basic process of detection is shown in Fig.2.1.

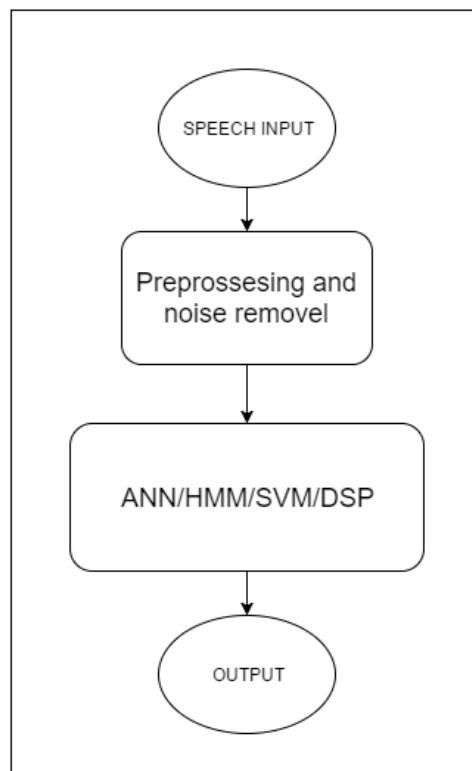


Figure 2.1: Basic speech detection

2.2.1 ARTIFICIAL NEURAL NETWORKS

In machine learning and cognitive science, artificial neural networks are a family of models inspired by biological neural networks (the central nervous systems of animals, in particular the brain) and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected "neurons" which exchange messages between each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.

For example, a neural network for handwriting recognition is defined by a set of input neurons which may be activated by the pixels of an input image. After being weighted and transformed by a function (determined by the network's designer), the activations of these neurons are then passed on to other neurons. This process is repeated until finally; an output neuron is activated. This determines which character was read.

ANN also have various types of learnings, namely – supervised and unsupervised, which have been explained later.

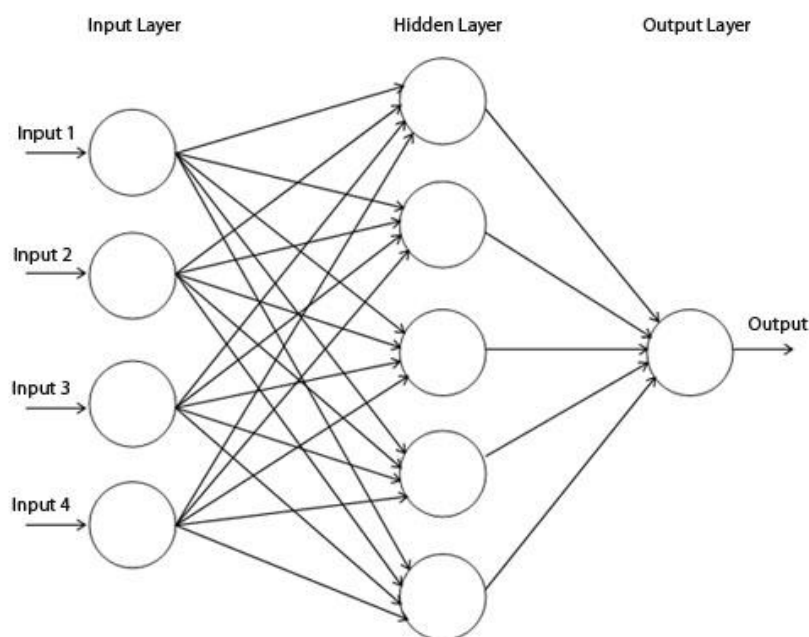


Figure 2.2: Basic ANN

People have been using ANNs to detect speech since a long time, and they are used widely as mathematical models or computational models that try to develop intelligent systems. Researchers from many scientific disciplines are designing ANNs to solve problems in pattern recognition, prediction, optimization, associative memory and control. Neural networks play an important role in the speech recognition and are an irreplaceable tool in distinguishing between similar signals. In recent years, ANNs are widely used in many ways in stuttering recognition, such as recognition of prolongation and repetition in stuttered speech, classification of fluent and dysfluent in stuttered speech.

1. History of ANN

Warren McCulloch and Walter Pitts created a computational model for neural networks based on mathematics and algorithms called threshold logic. This model led to the way for neural network research to split into two distinct approaches. One focused on biological processes in the human brain and the other focused on the application of neural networks to artificial intelligence and prediction. Since then there have been many advancements in the field of neural networks and various models have been created, the major ones are explained as follows.

1.1 McCulloch-Pitts Model

The artificial neuron receives one or more inputs (representing dendrites) and sums them to produce an output (representing a neuron's axon). Usually the sums of each node are weighted, and the sum is passed through a non-linear function known as an activation function or transfer function. The transfer functions usually have a sigmoid shape, but they may also take the form of other non-linear functions, piecewise linear functions, or step functions. They are also often monotonically increasing, continuous, differentiable and bounded.

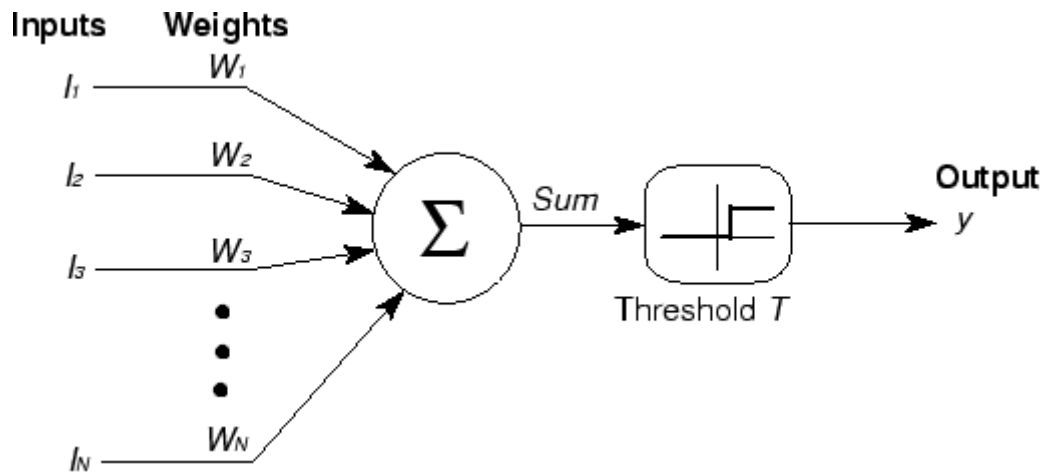


Figure 2.3: McCulloch-Pitts model

W_1, W_2, \dots, W_n are the weight values normalized in the range of either 0 or 1 and associated with each input line, **Sum** is the weighted sum, and **T** is a threshold constant. The function is a linear step function at threshold **T**. The symbolic representation of the linear threshold gate is shown in figure above.

$$y = \begin{cases} 1 & \text{if } u \geq \theta \\ 0 & \text{if } u < \theta \end{cases}$$

Equation 2.1: Step function

The transfer/step function of a neuron is chosen to have a number of properties which either enhance or simplify the network containing the neuron. There are various functions that can be used based on the need in the problem. Some of the other major transfer functions are:

The weights are updated using the formula:

$$\text{Sum} = I_1W_1 + I_2W_2 + \dots + I_nW_n$$

The main problem with this model was that it could not solve the more complex Exclusive OR, and thus a newer model was developed – the Perceptron

1.2 Perceptron

In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers: functions that can decide whether an input (represented by a vector of numbers) belongs to one class or another. It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector. The algorithm allows for online learning, in that it processes elements in the training set one at a time. The perceptron algorithm dates back to the late 1950s; its first implementation, in custom hardware, was one of the first artificial neural networks to be produced. The thesis will not be going into much detail of perceptron, just the basic concepts will be explained.

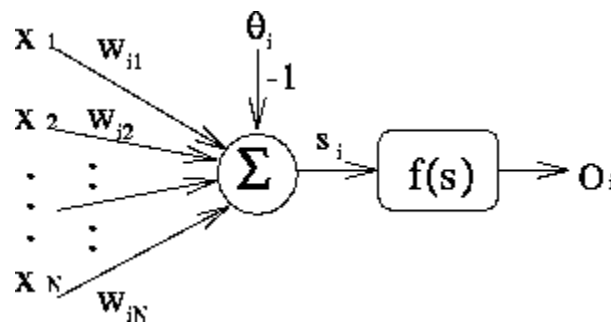


Figure 2.4: Simple Perceptron

The perceptron learning rule is a method for finding the weights in a network. We consider the problem of supervised learning for classification although other types of problems can also be solved. A nice feature of the perceptron learning rule is that if there exist a set of weights that solve the problem, then the perceptron will find these weights. This is true for either binary or bipolar representations.

There are certain assumptions in the perceptron, we have single layer network whose output is, **output = f(net) = f(W X)**

where f is a binary step function f whose values are (± 1) .

We also assume that the bias treated as just an extra input whose value is 1

p = number of training examples (x,t) where $t = \pm 1$

Now, since we know the basics of the perceptron, we move on to the algorithm.

1. Initialize the weights (either to zero or to a small random value)
2. Pick a learning rate μ (this is a number between 0 and 1)
3. Until stopping condition is satisfied (e.g. weights don't change):

3.1 For each training pattern (x, t):

- 3.1.1. compute output activation $y = f(w \cdot x)$
- 3.1.2. If y equals t , don't change weights
- 3.1.3. If y does not equal t , update the weights:
- 3.1.4. $w(\text{new}) = w(\text{old}) + 2 \mu t x$
- 3.1.5. or
- 3.1.6. $w(\text{new}) = w(\text{old}) + \mu (t - y) x$, for all t

The choice of learning rate μ does not matter because it just changes the scaling of w . The decision surface (for 2 inputs and one bias) has equation:

$$x_2 = - (w_1/w_2) x_1 - w_3 / w_2$$

where we have defined w_3 to be the bias: $\mathbf{W} = (w_1, w_2, b) = (w_1, w_2, w_3)$

From this we see that the equation remains the same if \mathbf{W} is scaled by a constant. The perceptron is guaranteed to converge in a finite number of steps if the problem is separable. May be unstable if the problem is not separable.

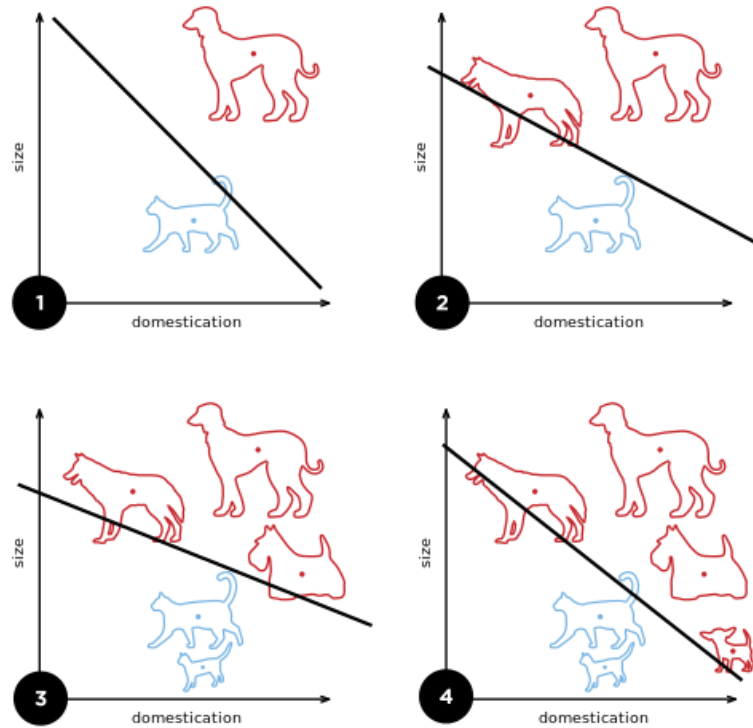


Figure 2.5: A diagram showing a perceptron updating its linear boundary as more training examples are added.

The example above, of a learning algorithm for a (single-layer) perceptron. For multilayer perceptron's, where a hidden layer exists, more sophisticated algorithms such as backpropagation must be used.

1.3 Back Propagation Network

BPN is a common method of training artificial neural networks. The method calculates the gradient of a loss function with respect to all the weights in the network. The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function.

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.

Backpropagation requires that the activation function used by the artificial neurons be continuous and differentiable.

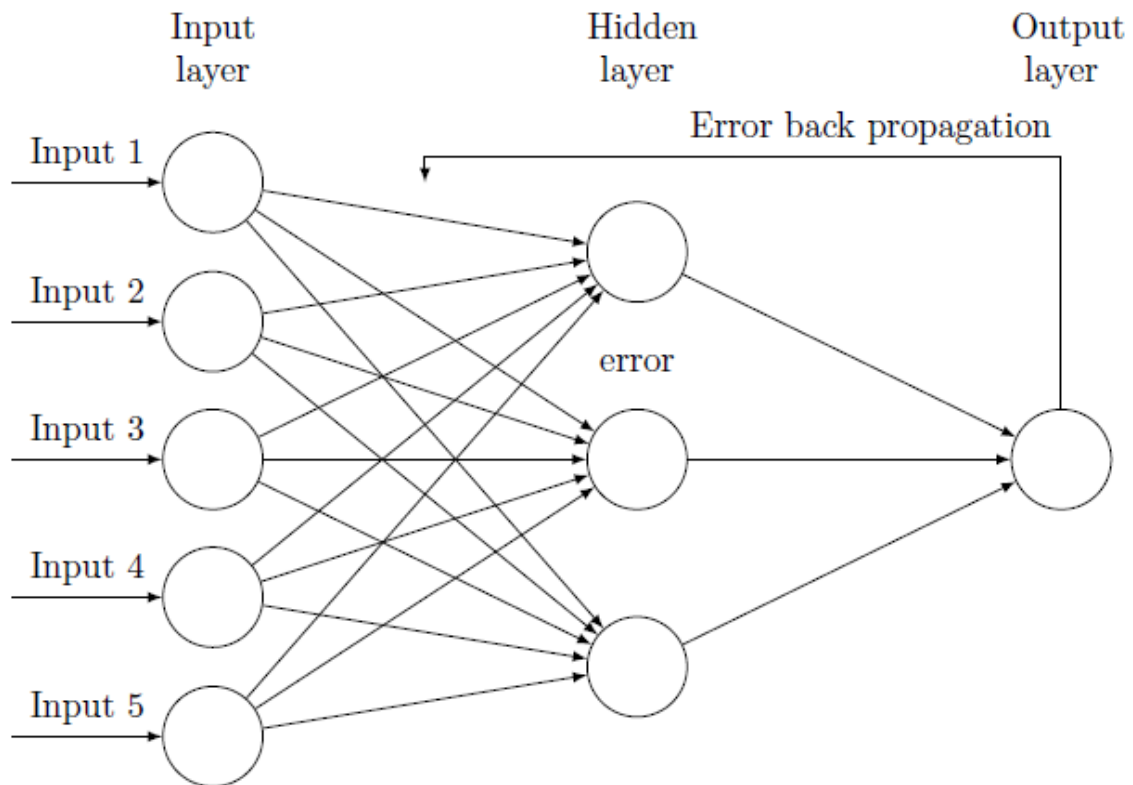


Figure 2.6: Back Propagation Network model

The main difference between BPN and the previously mentioned neural networks is the addition of the hidden layer, which results in better training results. As we have used BPN in our project, it will be explained in detail as follows.

The BPN learning algorithm can be divided into two major parts: propagation and weight update.

I. Propagation:

Each propagation involves the following steps:

A. Forward propagation of a training pattern's input through the neural network in order to generate the propagation's output activations.

B. Backward propagation of the propagation's output activations through the neural network using the training pattern target (or output) in order to generate the deltas (the difference between the targeted and actual output values) of all output and hidden neurons.

II. Weight Update:

Each weight-synapse follows the following steps:

- A. Multiply its output delta and input activation to get the gradient of the weight.
- B. Subtract a ratio (percentage) of the gradient from 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; 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. Repeat parts 1 and 2 until the performance of the network is satisfactory.

The one condition for BPN, as mentioned earlier is that the activation function needs to be continuous and thus, differentiable, therefore the sigmoid function is used.

A sigmoid function is a mathematical function having an "S" shape (sigmoid curve). Often, sigmoid function refers to the special case of the logistic function. It is bound from 0 - 1.

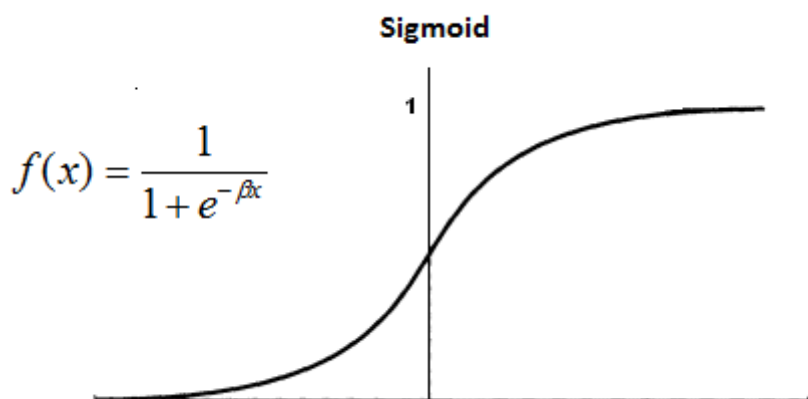


Figure 2.7: The Sigmoid function

The follow image makes it easier to understand the BPN algorithm steps.

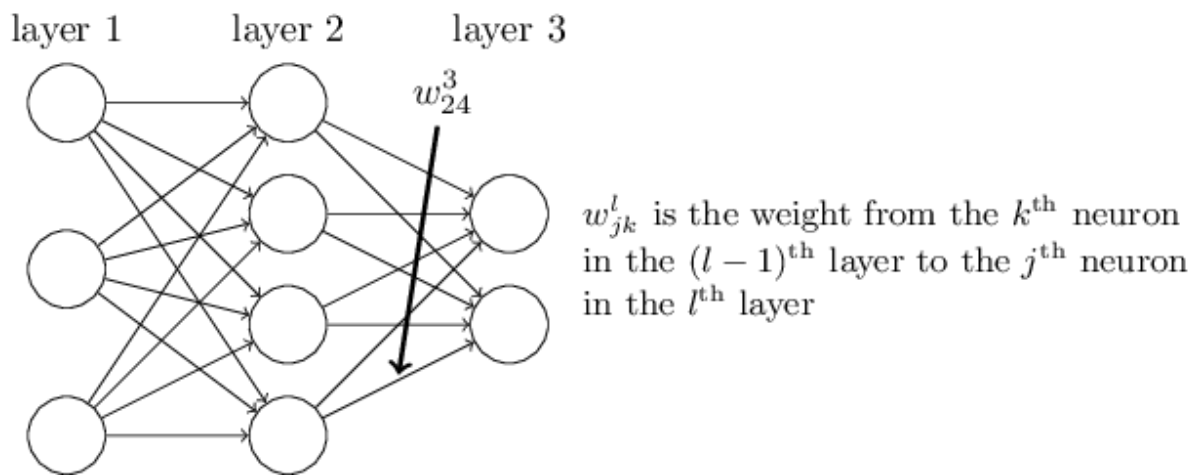


Figure 2.8: The BPN notation

The BPN Algorithm:

Here,

Input: Data set – D, Learning rate I

Output: Trained Neural Network


```

(1) Initialize all weights and biases in network;
(2) while terminating condition is not satisfied {
(3)   for each training tuple  $X$  in  $D$  {
(4)     // Propagate the inputs forward:
(5)     for each input layer unit  $j$  {
(6)        $O_j = I_j$ ; // output of an input unit is its actual input value
(7)     for each hidden or output layer unit  $j$  {
(8)        $I_j = \sum_i w_{ij} O_i + \theta_j$ ; // compute the net input of unit  $j$  with respect to the
        previous layer,  $i$ 
(9)        $O_j = \frac{1}{1+e^{-I_j}}$ ; } // compute the output of each unit  $j$ 
(10)    // Backpropagate the errors:
(11)    for each unit  $j$  in the output layer
(12)       $Err_j = O_j(1 - O_j)(T_j - O_j)$ ; // compute the error
(13)    for each unit  $j$  in the hidden layers, from the last to the first hidden layer
(14)       $Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$ ; // compute the error with respect to the
        next higher layer,  $k$ 
(15)    for each weight  $w_{ij}$  in network {
(16)       $\Delta w_{ij} = (l) Err_j O_i$ ; // weight increment
(17)       $w_{ij} = w_{ij} + \Delta w_{ij}$ ; } // weight update
(18)    for each bias  $\theta_j$  in network {
(19)       $\Delta \theta_j = (l) Err_j$ ; // bias increment
(20)       $\theta_j = \theta_j + \Delta \theta_j$ ; } // bias update
(21)  } }

```

Once the training is done the network has weights which can be used to predict the new test data. BPN has various applications, the major being Classification, Function Approximation and Time-series Prediction.

All the three major models that were discussed, the McCullough-Pitts model, Perceptron and Back Propagation are all part of supervised learning ANN. Now, a brief explanation of supervised and unsupervised learning is given and the algorithms which pertain to each and named.

1.4 Supervised Learning

It is fairly common in classification problems because the goal is often to get the computer to learn a classification system that we have created. Digit recognition, once again, is a common example of classification learning. More generally, classification learning is appropriate for any problem where deducing a classification is useful and the classification is easy to determine. In some cases, it might not even be necessary to give pre-determined

classifications to every instance of a problem if the agent can work out the classifications for itself. This would be an example of unsupervised learning in a classification context.

Supervised learning is the most common technique for training neural networks and decision trees. Both of these techniques are highly dependent on the information given by the pre-determined classifications. In the case of neural networks, the classification is used to determine the error of the network and then adjust the network to minimize it, and in decision trees, the classifications are used to determine what attributes provide the most information that can be used to solve the classification puzzle. We'll look at both of these in more detail, but for now, it should be sufficient to know that both of these examples thrive on having some "supervision" in the form of pre-determined classifications.

Speech recognition using hidden Markov models and Bayesian networks relies on some elements of supervision as well in order to adjust parameters to, as usual, minimize the error on the given inputs.

Notice something important here: in the classification problem, the goal of the learning algorithm is to minimize the error with respect to the given inputs. These inputs, often called the "training set", are the examples from which the agent tries to learn. But learning the training set well is not necessarily the best thing to do. For instance, if I tried to teach you exclusive-or, but only showed you combinations consisting of one true and one false, but never both false or both true, you might learn the rule that the answer is always true. Similarly, with machine learning algorithms, a common problem is over-fitting the data and essentially memorizing the training set rather than learning a more general classification technique.

As you might imagine, not all training sets have the inputs classified correctly. This can lead to problems if the algorithm used is powerful enough to memorize even the apparently "special cases" that don't fit the more general principles. This, too, can lead to overfitting, and it is a challenge to find algorithms that are both powerful enough to learn complex functions and robust enough to produce generalizable results.

Major supervised learning algorithms:

- A. BPN
- B. SVM
- C. Naive Bayes
- D. Classification Trees
- E. Discriminant Analysis (classification)
- F. k-Nearest Neighbours (classification)

1.5 Unsupervised Learning

Unsupervised learning seems much harder: the goal is to have the computer learn how to do something that we don't tell it how to do! There are actually two approaches to unsupervised learning. The first approach is to teach the agent not by giving explicit categorizations, but by using some sort of reward system to indicate success. Note that this type of training will generally fit into the decision problem framework because the goal is not to produce a classification but to make decisions that maximize rewards. This approach nicely generalizes to the real world, where agents might be rewarded for doing certain actions and punished for doing others.

Often, a form of reinforcement learning can be used for unsupervised learning, where the agent bases its actions on the previous rewards and punishments without necessarily even learning any information about the exact ways that its actions affect the world. In a way, all of this information is unnecessary because by learning a reward function, the agent simply knows what to do without any processing because it knows the exact reward it expects to achieve for each action it could take. This can be extremely beneficial in cases where calculating every possibility is very time consuming (even if all of the transition probabilities between world states were known). On the other hand, it can be very time consuming to learn by, essentially, trial and error.

But this kind of learning can be powerful because it assumes no pre-discovered classification of examples. In some cases, for example, our classifications may not be the best possible. One

striking example is that the conventional wisdom about the game of backgammon was turned on its head when a series of computer programs (neuro-gammon and TD-gammon) that learned through unsupervised learning became stronger than the best human chess players merely by playing themselves over and over. These programs discovered some principles that surprised the backgammon experts and performed better than backgammon programs trained on pre-classified examples.

A second type of unsupervised learning is called clustering. In this type of learning, the goal is not to maximize a utility function, but simply to find similarities in the training data. The assumption is often that the clusters discovered will match reasonably well with an intuitive classification. For instance, clustering individuals based on demographics might result in a clustering of the wealthy in one group and the poor in another.

Although the algorithm won't have names to assign to these clusters, it can produce them and then use those clusters to assign new examples into one or the other of the clusters. This is a data-driven approach that can work well when there is sufficient data; for instance, social information filtering algorithms, such as those that Amazon.com use to recommend books, are based on the principle of finding similar groups of people and then assigning new users to groups. In some cases, such as with social information filtering, the information about other members of a cluster (such as what books they read) can be sufficient for the algorithm to produce meaningful results. In other cases, it may be the case that the clusters are merely a useful tool for a human analyst. Unfortunately, even unsupervised learning suffers from the problem of overfitting the training data. There's no silver bullet to avoiding the problem because any algorithm that can learn from its inputs needs to be quite powerful.

Major unsupervised learning algorithms:

- A. Max net
- B. Hamming Net
- C. Mexican Net
- D. Kohonen Self Organising Map
- E. Linear Vector Quantisation
- F. Counter Propagation Network

G. Adaptive Resonance Theory

In the end, unsupervised learning has produced many successes, such as world-champion calibre backgammon programs and even machines capable of driving cars! It can be a powerful technique when there is an easy way to assign values to actions. Clustering can be useful when there is enough data to form clusters (though this turns out to be difficult at times) and especially when additional data about members of a cluster can be used to produce further results due to dependencies in the data.

Classification learning is powerful when the classifications are known to be correct (for instance, when dealing with diseases, it's generally straight-forward to determine the design after the fact by an autopsy), or when the classifications are simply arbitrary things that we would like the computer to be able to recognize for us. Classification learning is often necessary when the decisions made by the algorithm will be required as input somewhere else. Otherwise, it wouldn't be easy for whoever requires that input to figure out what it means.

Both techniques can be valuable and which one you choose should depend on the circumstances--what kind of problem is being solved, how much time is allotted to solving it (supervised learning or clustering is often faster than reinforcement learning techniques), and whether supervised learning is even possible.

2.2.2 HIDDEN MARKOV MODELS

The Hidden Markov Model is a very powerful statistical tool that is used to predict the outcome of a particular observation. Assume we have an input sequence $\mathbf{x}=\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ and a tag sequence $\mathbf{y}=\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_n$, then we can use the HMM to find out the probability of $P(\mathbf{y}_1|\mathbf{x}_1)$ or any other combination.

HMM is a stochastic model that captures the statistical properties of observed real world data. In speech recognition, speech signal could be viewed as a piecewise stationary signal or a short-time stationary signal. Thus, HMMs are widely used in speech recognition, especially in stuttering recognition to recognize speech dysfluency such as prolongation and repetition

Hidden Markov Models use only the immediate previous state to predict the outcome of the current state.

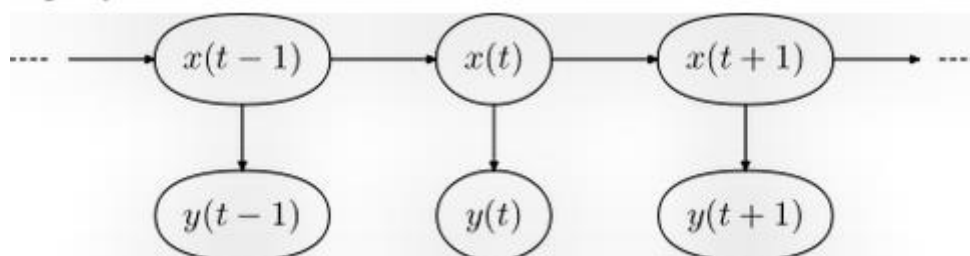


Fig 2.9 – General architecture of HMM

The diagram above shows the general architecture of HMM. All ‘x’ states represent the hidden states at different times whereas ‘y’ represents the observations. It is also showing conditional dependencies amongst different states (via the arrows). It is clear that the probability distribution of a hidden state like $x(t+1)$ is only dependent on $x(t)$ and not on $x(t-1)$, which has absolutely no effect. Also, the observation of a hidden state is only dependent on the value of the hidden variable at that time, like for $x(t)$, the observation is $y(t)$, and $x(t)$ is the only contributing factor.

For eg. Suppose a blind man wants to predict the weather of his city based on the work his friend does. The weather can be either sunny or rainy and the blind man’s friend is involved in only 3 activities – walking, shopping and cleaning, and this is based solely on the weather of the day. The blind man understands that his goal of predicting the weather can be accomplished by using the Hidden Markov Model, where the 2 states that are hidden from him are – sunny and rainy. Everyday his friend tells him what he did that day, so, those activities, walking, shopping and cleaning, become observations for the blind man. The blind man also considers transitional probabilities (conditional dependencies) that is, he finds out the probability of the weather of his current day, based on the weather the previous day. This shows how his problem can be solved by the Hidden Markov Model.

One of the biggest areas where HMM is used is speech recognition.

For speech recognition, the sentences are first divided in to words and words into phonemes. Phonemes are the small unit of a word. Then, we use HMM and assign probabilities to

determine which phoneme will be present after the current phoneme. This is extended to obtain a word, which can further be extended to predict an entire sentence (Speech Recognition).

For eg. For a sentence like “The dog laughs”, we have $n=3$, $x=x_1, x_2, x_3$ and $y=y_1, y_2, y_3, y_4$ equal to tag sequence D N V Stop, where D – determiner, N – Noun, V- Variable and Stop – Special tag that terminates the sequence. We need to calculate the probability of the word sequence (x) with tag sequence (y). With probability, we can determine that for a sentence to make sense, the first word needs to be a determiner and noun is always preceded by a verb and never the other way round. So, once, by stringing the phonemes together, the word sequence can be obtained by using grammatical rules such as these. This is an easy and simplistic example which shows how Hidden Markov Models can be used for speech recognition.

2.2.3 SUPPORT VECTOR MACHINES

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyse data used for classification and regression analysis. Given a set of training examples, each marked for belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well.

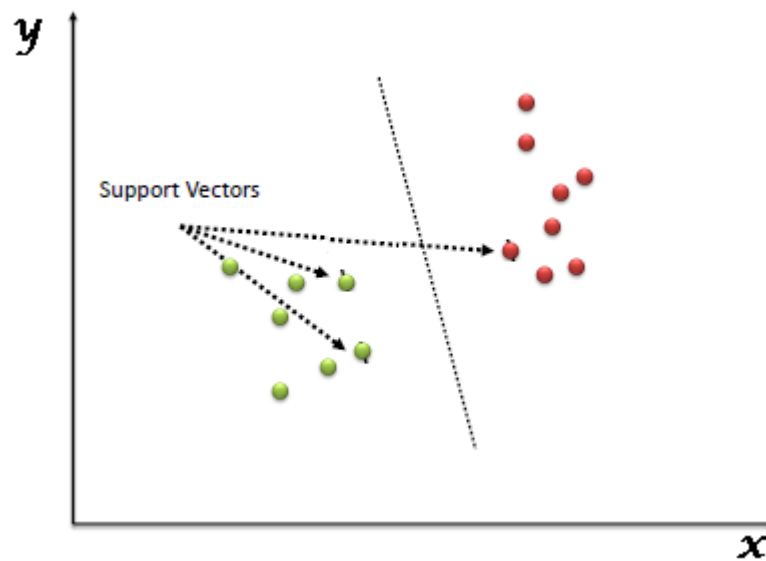


Figure 2.10: SVM Plane

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

Basic idea of support vector machines is to have an Optimal hyperplane for linearly separable patterns and to extend to patterns that are not linearly separable by transformations of original data to map into new space, that is, Kernel function.

SVMs belong to a family of generalized linear classifiers. They can also be considered a special case of Tikhonov regularization. A special property is that they simultaneously minimize the empirical classification error and maximize the geometric margin; hence they are also known as maximum margin classifiers.

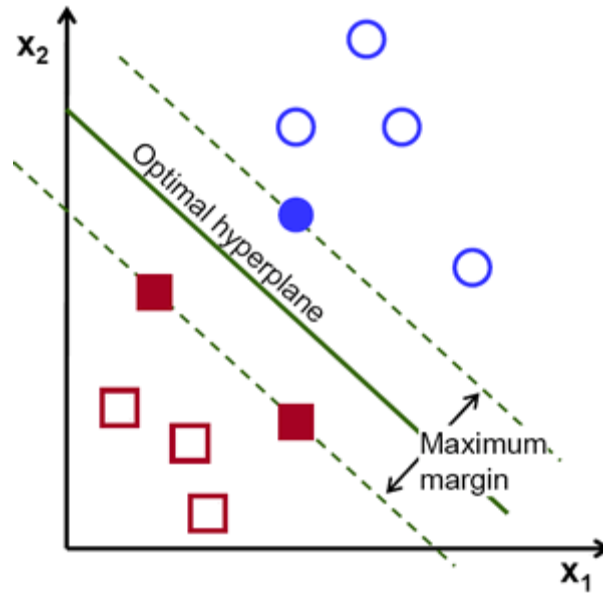


Figure 2.11: Basic SVM

The original optimal hyperplane algorithm proposed by Vladimir Vapnik in 1963 was a linear classifier. However, in 1992, Bernhard Boser, Isabelle Guyon and Vapnik suggested a way to create non-linear classifiers by applying the kernel (trick to maximum-margin hyperplanes). The resulting algorithm is formally similar, except that every dot product is replaced by a non-linear kernel (integral operator). This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be non-linear and the transformed space high dimensional; thus though the classifier is a hyperplane in the high-dimensional feature space, it may be non-linear in the original input space. The effectiveness of SVM depends on the selection of kernel, kernel's parameters and soft margin parameter

1.1 Identifying the correct hyperplane

Detecting the correct hyperplane is the essence of SVM, we will now look at various scenarios where the hyper plane is identified.

A. Identify the right hyper-plane (Scenario-1)

Here, we have three hyper-planes (A, B and C). Now, we identify the right hyper-plane to classify star and circle.

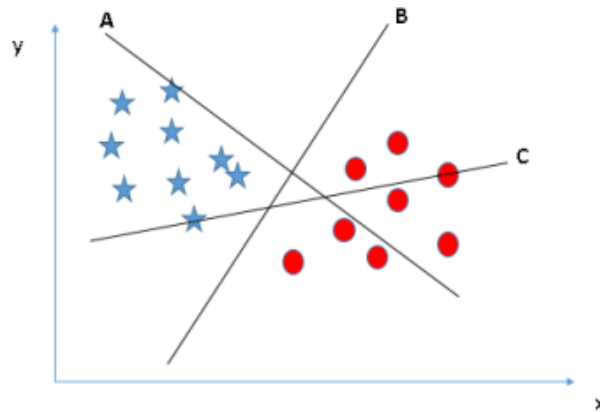


Figure 2.12: Hyperplane scenario 1

As a rule of thumb we need to remember to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

B. Identify the right hyper-plane (Scenario-2)

Here, we have three hyper-planes (A, B and C) and all are segregating the classes well.

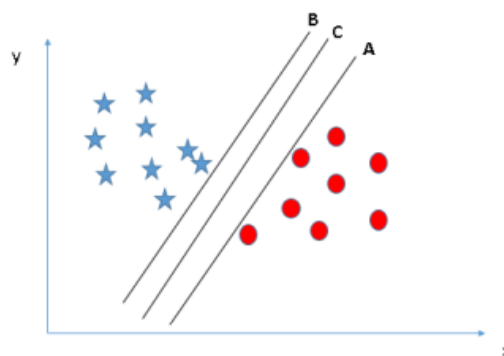


Figure 2.13: Hyperplane scenario 2a

Here, maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as Margin. Let's look at the image below:

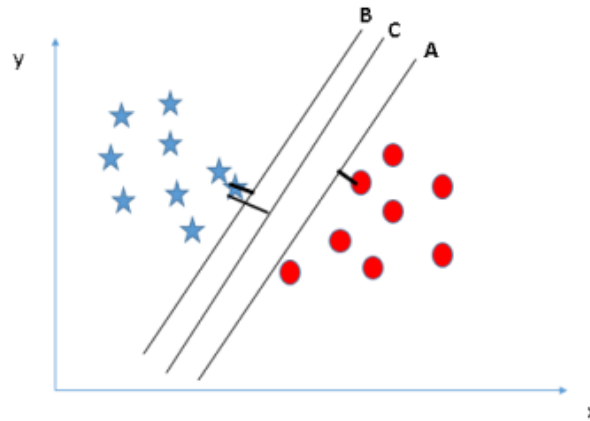


Figure 2.14: Hyperplane scenario 2b

We can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another great reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin, then there is high chance of miss-classification. Thus to obtain best the classification the correct margin has to be obtained.

C. Classification two classes (Scenario-3)

Below, we are unable to segregate the two classes using a straight line, as one of star lies in the territory of other(circle) class as an outlier.



Figure 2.15: Hyperplane scenario 3a

As we have already mentioned, one star at other end is like an outlier for star class. SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin. Hence, we can say, SVM is robust to outliers.

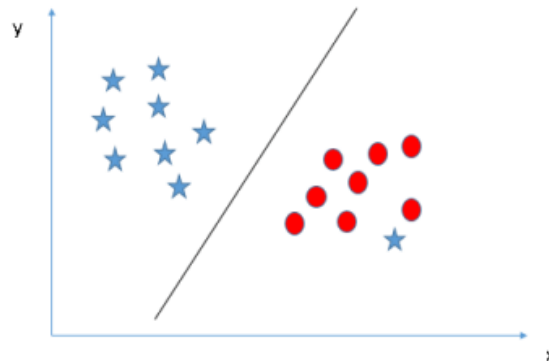


Figure 2.16: Hyperplane scenario 3b

D. Classification to segregate to classes (Scenario-4):

In the scenario below, we can't have linear hyper-plane between the two classes.

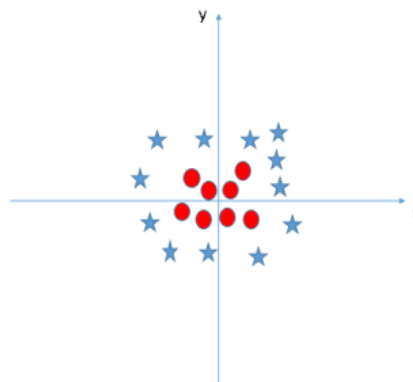


Figure 2.17: Hyperplane scenario 4a:

However, SVM can easily this problem by introducing additional feature. Here, we add a new feature $z = x^2 + y^2$. Plotting we get.

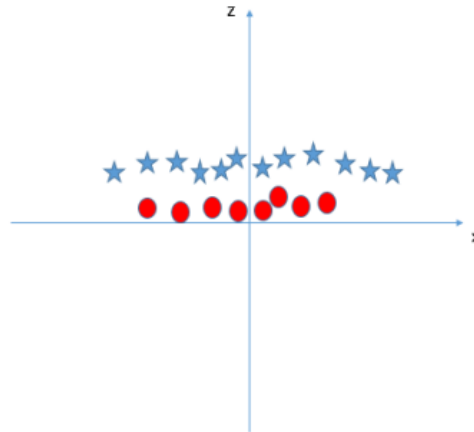


Figure 2.18: Hyperplane scenario 4b

In above plot, we can see that all values for z would be positive always because z is the squared sum of both x and y and in the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z .

In SVM, it is easy to have a linear hyper-plane between these two classes. But, should we need to add this feature manually to have a hyper-plane. The answer to that is no, SVM has a technique called the kernel trick.

1.2 Kernel Trick

Some classifications seem very difficult on a linear plane and a good enough hyperplane cannot be established. Thus, the data is transformed in to a higher dimension using squares, cubic, etc. This results in the data being easily classified, is becomes clear in Fig. 2.19 given below. Thus kernel trick is when data in a lower dimension is converted in to a higher dimension to better separate the data, it is known as a kernel trick.

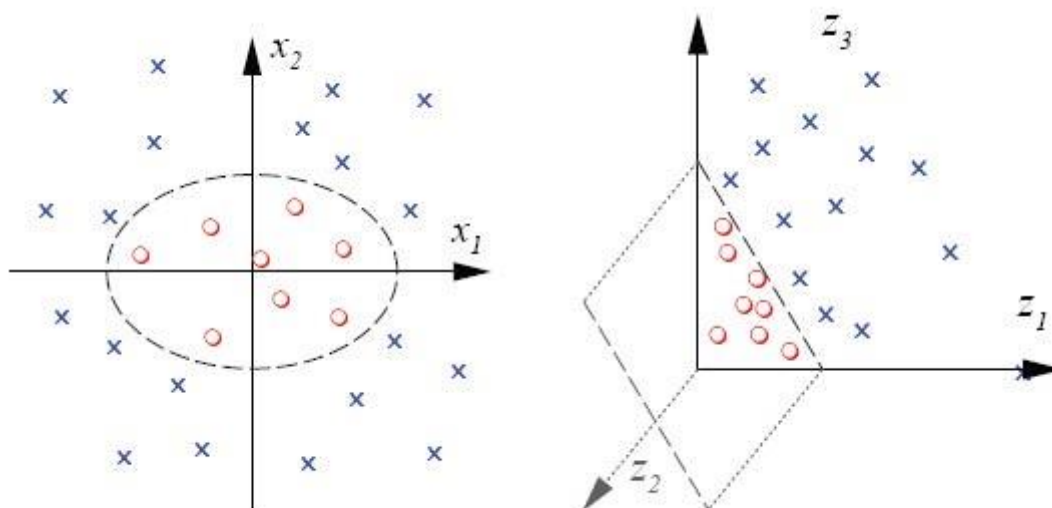


Figure 2.19: Simple kernel trick

These functions take low dimensional input space and transform it to a higher dimensional space i.e. it converts not separable problem to separable problem, these functions are called kernels. It is mostly useful in non-linear separation problem. Simply put, it does some extremely complex data transformations, then find out the process to separate the data based on the labels or outputs defined.

SVM has been used in addressing many real life problems, some of them are:

1. SVMs are helpful in text and hypertext categorization as their application can significantly reduce the need for labelled training instances in both the standard inductive and transductive settings.
2. Classification of images can also be performed using SVMs. Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback
3. Bioinformatics (Protein classification, Cancer classification), SVMs are useful in medical science to classify proteins with up to 90% of the compounds classified correctly.
4. Hand-written character recognition which are useful in various areas.

Some basic advantages of using SVMs:

1. It works really well with clear margin of separation
2. It is effective in high dimensional spaces.
3. It is effective in cases where number of dimensions is greater than the number of samples.
4. It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Some disadvantages of SVM:

1. It doesn't perform well when we have large data set because the required training time is high
2. It also doesn't perform very well when the data set has more noise i.e. target classes are overlapping.
3. SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. Which makes it computationally costly.

Support Vector Machines are very powerful classification algorithm. When used in conjunction with random forest and other machine learning tools, they give a very different dimension to ensemble models. Hence, they become very crucial for cases where very high predictive power is required. Such algorithms are slightly harder to visualize because of the complexity in formulation.

Chapter 3

Speech Recognition

3.1 INTRODUCTION

Speech recognition is the ability of a machine or program to identify words and phrases in spoken language and convert them to a machine-readable format. Rudimentary speech recognition software has a limited vocabulary of words and phrases and may only identify these if they are spoken very clearly. More sophisticated software has the ability to accept natural speech. People with disabilities that prevent them from typing have also adopted speech-recognition systems. If a user has lost the use of his hands, or for visually impaired users when it is not possible or convenient to use a Braille keyboard, the systems allow personal expression through dictation as well as control of many computer tasks. Some programs save users' speech data after every session, allowing people with progressive speech deterioration to continue to dictate to their computer

3.2 EXISTING SOLUTIONS

The common way to recognize speech is the following: we take waveform, split it on utterances by silences then try to recognize what's being said in each utterance. To do that we want to take all possible combinations of words and try to match them with the audio. We choose the best matching combination. There are few important things in this match.

First of all, it's a concept of features. Since number of parameters is large, we are trying to optimize it. Numbers that are calculated from speech usually by dividing speech on frames. Then for each frame of length typically 10 milliseconds we extract 39 numbers that represent

the speech. That's called feature vector. The way to generate numbers is a subject of active investigation, but in simple case it's a derivative from spectrum.

Second it's a concept of the model. Model describes some mathematical object that gathers common attributes of the spoken word. In practice, for audio model of Sen one is Gaussian mixture of its three states - to put it simple, it's a most probable feature vector. From concept of the model the following issues raised - how good does model fits practice, can model be made better of its internal model problems, how adaptive model is to the changed conditions.

The model of speech is called Hidden Markov Model or HMM, it's a generic model that describes black-box communication channel. In this model process is described as a sequence of states which change each other with certain probability. This model is intended to describe any sequential process like speech. It has been proven to be really practical for speech decoding.

Third, it's a matching process itself. Since it would take a huge time more than universe existed to compare all feature vectors with all models, the search is often optimized by many tricks. At any points we maintain best matching variants and extend them as time goes producing best matching variants for the next frame.

Some of the existing models are explained in brief as follows.

3.2.1 Hidden Markov model

Modern general-purpose speech recognition systems are based on Hidden Markov Models. These are statistical models that output a sequence of symbols or quantities. HMMs are used in speech recognition because a speech signal can be viewed as a piecewise stationary signal or a short-time stationary signal. In a short time-scale (e.g., 10 milliseconds), speech can be approximated as a stationary process. Speech can be thought of as a Markov model for many stochastic purposes.

Another reason why HMMs are popular is because they can be trained automatically and are simple and computationally feasible to use. In speech recognition, the hidden Markov model would output a sequence of n -dimensional real-valued vectors (with n being a small integer, such as 10), outputting one of every 10 milliseconds. The vectors would consist of cepstral coefficients, which are obtained by taking a Fourier transform of a short time window of speech and de correlating the spectrum using a cosine transform, then taking the first (most significant) coefficients. The hidden Markov model will tend to have in each state a statistical distribution that is a mixture of diagonal covariance Gaussians, which will give a likelihood for each observed vector. Each word, or (for more general speech recognition systems), each phoneme, will have a different output distribution; a hidden Markov model for a sequence of words or phonemes is made by concatenating the individual trained hidden Markov models for the separate words and phonemes.

Described above are the core elements of the most common, HMM-based approach to speech recognition. Modern speech recognition systems use various combinations of a number of standard techniques in order to improve results over the basic approach described above. A typical large-vocabulary system would need context dependency for the phonemes (so phonemes with different left and right context have different realizations as HMM states); it would use cepstral normalization to normalize for different speaker and recording conditions; for further speaker normalization it might use vocal tract length normalization (VTLN) for male-female normalization and maximum likelihood linear regression (MLLR) for more general speaker adaptation. The features would have so-called delta and delta-delta coefficients to capture speech dynamics and in addition might use heteroscedastic linear discriminant analysis (HLDA); or might skip the delta and delta-delta coefficients and use splicing and an LDA-based projection followed perhaps by heteroscedastic linear discriminant analysis or a global semi-tied co variance transform (also known as maximum likelihood linear transform, or MLLT). Many systems use so-called discriminative training techniques that dispense with a purely statistical approach to HMM parameter estimation and instead optimize some classification-related measure of the training data. Examples are maximum mutual information (MMI), minimum classification error (MCE) and minimum phone error (MPE).

Decoding of the speech (the term for what happens when the system is presented with a new utterance and must compute the most likely source sentence) would probably use the Viterbi

algorithm to find the best path, and here there is a choice between dynamically creating a combination hidden Markov model, which includes both the acoustic and language model information, and combining it statically beforehand (the finite state transducer, or FST, approach).

A possible improvement to decoding is to keep a set of good candidates instead of just keeping the best candidate, and to use a better scoring function (re scoring) to rate these good candidates so that we may pick the best one according to this refined score. The set of candidates can be kept either as a list (the N-best list approach) or as a subset of the models (a lattice). Re scoring is usually done by trying to minimize the Bayes risk (or an approximation thereof): Instead of taking the source sentence with maximal probability, we try to take the sentence that minimizes the expectancy of a given loss function with regards to all possible transcriptions (i.e., we take the sentence that minimizes the average distance to other possible sentences weighted by their estimated probability). Efficient algorithms have been devised to re score lattices represented as weighted finite state transducers with edit distances represented themselves as a finite state transducer verifying certain assumptions

3.2.2 Dynamic time warping

In time series analysis, dynamic time warping (DTW) is an algorithm for measuring similarity between two temporal sequences which may vary in time or speed. For instance, similarities in walking patterns could be detected using DTW, even if one person was walking faster than the other, or if there were accelerations and decelerations during the course of an observation. DTW has been applied to temporal sequences of video, audio, and graphics data — indeed, any data which can be turned into a linear sequence can be analysed with DTW. A well-known application has been automatic speech recognition, to cope with different speaking speeds. Other applications include speaker recognition and online signature recognition. Also it is seen that it can be used in partial shape matching application.

In general, DTW is a method that calculates an optimal match between two given sequences (e.g. time series) with certain restrictions. The sequences are "warped" non-linearly in the time dimension to determine a measure of their similarity independent of certain non-linear

variations in the time dimension. This sequence alignment method is often used in time series classification. Although DTW measures a distance-like quantity between two given sequences, it doesn't guarantee the triangle inequality to hold.

Dynamic time warping is an approach that was historically used for speech recognition but has now largely been displaced by the more successful HMM-based approach.

3.2.3 Neural networks

Neural networks emerged as an attractive acoustic modelling approach in ASR in the late 1980s. Since then, neural networks have been used in many aspects of speech recognition such as phoneme classification, isolated word recognition, and speaker adaptation.

In contrast to HMMs, neural networks make no assumptions about feature statistical properties and have several qualities making them attractive recognition models for speech recognition. When used to estimate the probabilities of a speech feature segment, neural networks allow discriminative training in a natural and efficient manner. Few assumptions on the statistics of input features are made with neural networks. However, in spite of their effectiveness in classifying short-time units such as individual phones and isolated words, neural networks are rarely successful for continuous recognition tasks, largely because of their lack of ability to model temporal dependencies.

However, recently LSTM Recurrent Neural Networks (RNNs) and Time Delay Neural Networks(TDNN's) have been used which have been shown to be able to identify latent temporal dependencies and use this information to perform the task of speech recognition.

.

Due to the inability of feedforward Neural Networks to model temporal dependencies, an alternative approach is to use neural networks as a pre-processing e.g. feature transformation, dimensionality reduction, for the HMM based recognition.

3.2.4 Deep Feedforward and Recurrent Neural Networks

A deep feedforward neural network (DNN) is an artificial neural network with multiple hidden layers of units between the input and output layers. Similar to shallow neural networks, DNNs can model complex non-linear relationships. DNN architectures generate compositional models, where extra layers enable composition of features from lower layers, giving a huge learning capacity and thus the potential of modelling complex patterns of speech data.

A success of DNNs in large vocabulary speech recognition occurred in 2010 by industrial researchers, in collaboration with academic researchers, where large output layers of the DNN based on context dependent HMM states constructed by decision trees were

One fundamental principle of deep learning is to do away with hand-crafted feature engineering and to use raw features. This principle was first explored successfully in the architecture of deep auto encoder on the "raw" spectrogram or linear filter-bank features showing its superiority over the Mel-Cepstral features which contain a few stages of fixed transformation from spectrograms. The true "raw" features of speech, waveforms, have more recently been shown to produce excellent larger-scale speech recognition results.

Large-scale automatic speech recognition is the first and the most convincing successful case of deep learning in the recent history, embraced by both industry and academic across the board. Between 2010 and 2014, the two major conferences on signal processing and speech recognition, IEEE-ICASSP and Interspeech, have seen near exponential growth in the numbers of accepted papers in their respective annual conference papers on the topic of deep learning for speech recognition. More importantly, all major commercial speech recognition systems (e.g., Microsoft Cortana, Xbox, Skype Translator, Google Now, Apple Siri, Baidu and iFlyTek voice search, and a range of Nuance speech products, etc.) nowadays are based on deep learning methods.

3.3 OUR SOLUTION

The main aim of the project was to remove and correct stuttered speech, however, in order to better learn how speech recognition works, we decided to implement a small speech recognition system of 10 words on our own. Our small program learns 10 words and recognises them when spoken to a great degree of accuracy.

3.3.1 Introduction

Our speech recognition program is based on Linear Predictive Coding[LPC] and a simple implementation of gradient decent and linear regression, both are integral part of neural networks.

We have trained the neural network to recognise 10 simple words, they are “One, Two, Three, Four, Five, Six, Seven, Eight, Nine and Zero”. The training set contains 120 samples; each word has been spoken 12 times. The samples were created by both of us and include various ways in which the word is spoken.

The results we obtained were good enough based on such a small learning data set. However, in order to make our main project robust, we will be using Googles Speech API in detection of the corrected stuttered speech because it is not feasible to create the whole speech recognition. Thus Googles API will be used to make the program robust.

3.3.2 Theory

The various techniques and algorithms will be discussed now, namely, Linear Predictive Coding, Linear Regression and Gradient Decent.

1. Linear Predictive Coding

Linear predictive coding (LPC) is a tool used mostly in audio signal processing and speech processing for representing the spectral envelope of a digital signal of speech in compressed form, using the information of a linear predictive mode. It is one of the most powerful speech analysis techniques, and one of the most useful methods for encoding good quality speech at a low bit rate and provides extremely accurate estimates of speech parameters.

LPC starts with the assumption that a speech signal is produced by a buzzer at the end of a tube (voiced sounds), with occasional added hissing and popping sounds (sibilants and plosive sounds). Although apparently crude, this model is actually a close approximation of the reality of speech production. The glottis (the space between the vocal folds) produces the buzz, which is characterized by its intensity (loudness) and frequency (pitch). The vocal tract (the throat and mouth) forms the tube, which is characterized by its resonances, which give rise to formants, or enhanced frequency bands in the sound produced. Hisses and pops are generated by the action of the tongue, lips and throat during sibilants and plosives.

LPC analyses the speech signal by estimating the formants, removing their effects from the speech signal, and estimating the intensity and frequency of the remaining buzz. The process of removing the formants is called inverse filtering, and the remaining signal after the subtraction of the filtered modelled signal is called the residue.

The numbers which describe the intensity and frequency of the buzz, the formants, and the residue signal, can be stored or transmitted somewhere else. LPC synthesizes the speech signal by reversing the process: use the buzz parameters and the residue to create a source signal, use the formants to create a filter (which represents the tube), and run the source through the filter, resulting in speech.

Because speech signals vary with time, this process is done on short chunks of the speech signal, which are called frames; generally, 30 to 50 frames per second give intelligible speech with good compression.

2. Linear Regression

In statistics, linear regression is an approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X . The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression.

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of y given the value of X is assumed to be an affine function of X ; less commonly, the median or some other quantile of the conditional distribution of y given X is expressed as a linear function of X . Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of y given X , rather than on the joint probability distribution of y and X , which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

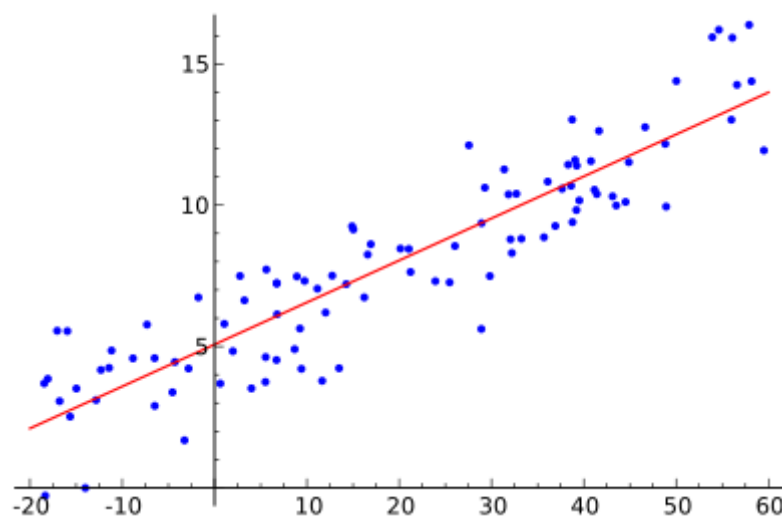


Figure 3.1: Linear regression sample

In simple linear regression a single independent variable is used to predict the value of a dependent variable. In multiple linear regression two or more independent variables are used to predict the value of a dependent variable. The difference between the two is the number of independent variables

Another way to understand is that Linear Regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable. For example, a modeller might want to relate the weights of individuals to their heights using a linear regression model.

3. Gradient Decent

Gradient descent is a first-order optimization algorithm. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point. It is basically used to optimize the cost function

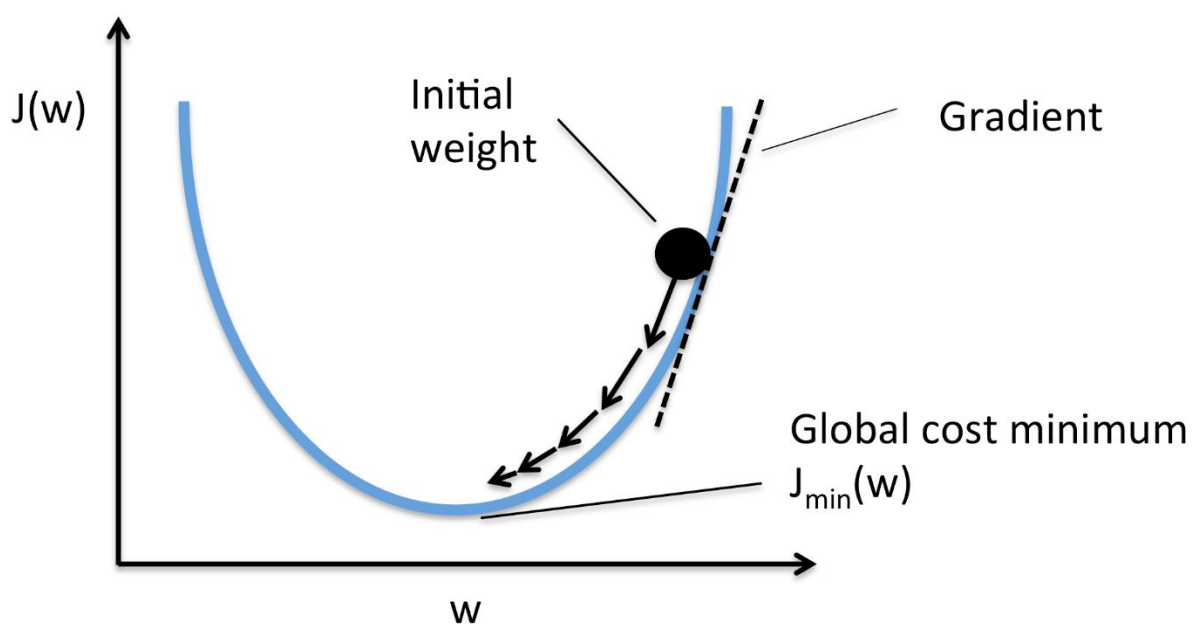


Figure 3.2: Gradient Decent

We use gradient decent along with linear regression to create classification models. The major terms used in gradient decent are as follows.

Just like any neural network, gradient decent uses a hypothesis function, $h_{\theta}(x)$, which is the sigmoid function (Fig. 2.8) where x_n is the input and θ_n are the parameters.

$$h_{\theta}(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

Equation 3.1: Hypothesis function

The hypothesis function is how we predict the unknown variable.

$$\textit{Parameters} : \theta_0, \theta_1, \dots, \theta_n$$

Equation 3.2: Parameters

The cost function returns a number representing how well the neural network performed to map training examples to correct output. The following is the cost function used in gradient decent

Cost Function:

$$J(\theta_0, \theta_1, \dots, \theta_n) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Equation 3.3: Cost Function

Our main aim in gradient decent now becomes to minimize the cost function, which is done by updating the parameters simultaneously, this can be seen in the formula given below.

Gradient Decent:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta} J(\theta_0, \theta_1, \dots, \theta_n)$$

} (simultaneously update for every $j = 0, 1, \dots, n$)

Equation 3.4: Parameter update A

We now replace $J(\theta_n)$ with the actual expression of cost function and substitute it in equation given in Equation 3.3. Thus, we finally obtain the update rule for gradient decent that minimizes the cost function and helps us predict with greater accuracy.

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

} (simultaneously update θ_j for $j = 0, 1, \dots, n$)

Equation 3.5: Parameter update B

Where,

- | | |
|-----------------|---|
| $J(\theta_n)$ | - Cost function |
| α | - learning rate |
| m | - training set size |
| $h_{\theta}(x)$ | - hypothesis function (activation function) |
| x^i | - input |
| y^i | - output corresponding to input |
| θ_j | - j^{th} parameter |

3.3.3 Algorithm

The flowchart of the method that we have used to filter the training set samples, which will be used to train the neural network.

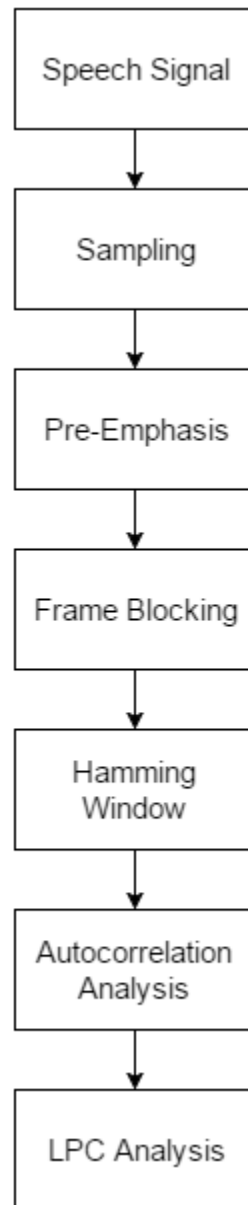


Figure 3.3: Filtering the sound sample

The data that we get from this is then used to train a back propagating network using linear regression, which finally recognises speech.

The detailed algorithm:

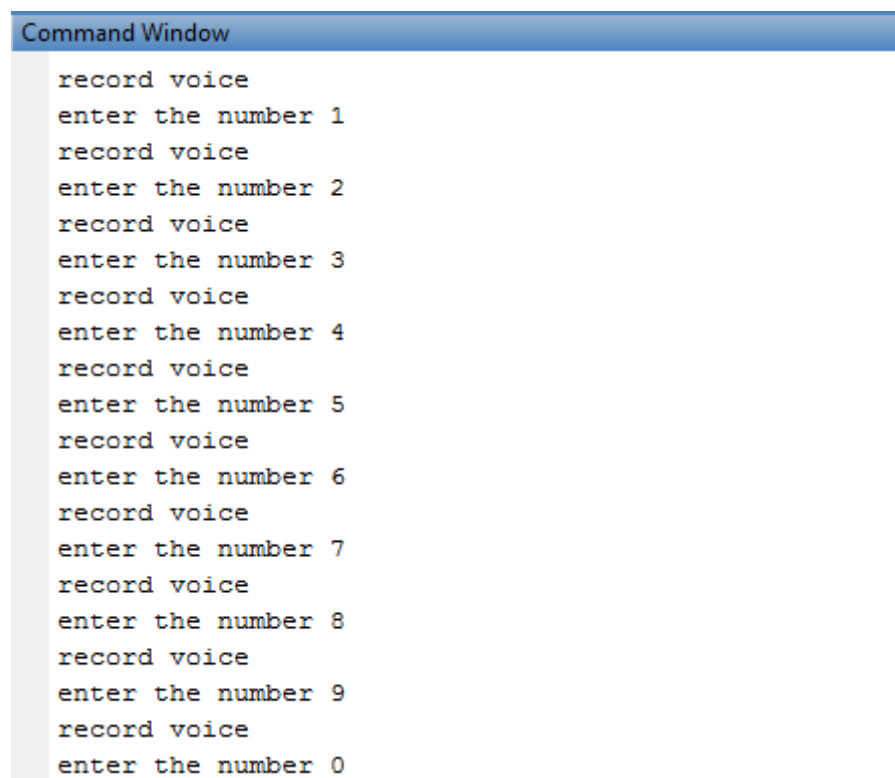
1. Record speech sample of length 1 second at sampling frequency 8000 Hz, also write down the label associated with the sample
2. Removal of noise using high pass filter and threshold
3. Creating overlapping frames for efficient analysis, called Frame Blocking
4. Each frame multiplied by hamming window to smoothen the signal
5. Auto correlation matrix is made and the LPC coefficients are obtained
6. Repeat steps 1-5 120 times, 12 times for each word.
7. We now have a matrix with the label and its corresponding data, this becomes our training data set.
8. The matrix obtained is now passed in to a neural network which trains weights based on the training samples.
9. Once weights are obtained, we test the network by recording a Test speech sample of length 1 second at 8000hz
10. The network tests the sample with its weights and predicts the word.

The algorithm is pretty straight forward, and we used LPC because of the low sampling frequency that we had because LPC is preferred when dealing with low sampling rates and also it was much easier to implement the speech recognition using out already existing knowledge of neutral networks.

3.3.4 Implementation

The algorithm was implemented in Matlab, and the sound samples were recorded using the on board mic of a laptop.

The following images will run through the actual running of the program in Matlab.

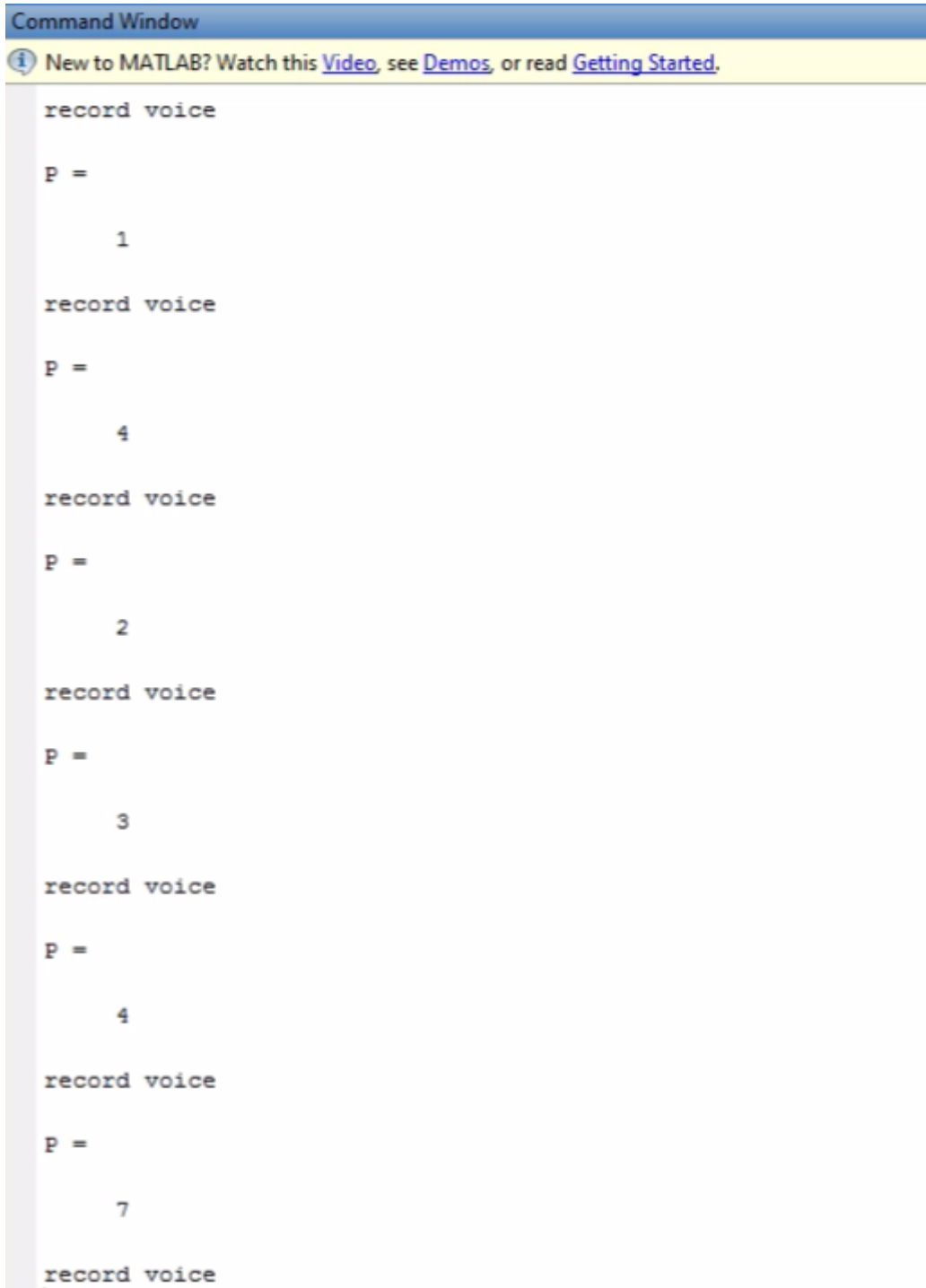


```
Command Window
record voice
enter the number 1
record voice
enter the number 2
record voice
enter the number 3
record voice
enter the number 4
record voice
enter the number 5
record voice
enter the number 6
record voice
enter the number 7
record voice
enter the number 8
record voice
enter the number 9
record voice
enter the number 0
```

Figure 3.4: Input the speech samples with labels

Fig. 3.9 shows how each word was recorded and a label associated with it, the image shows the recording of only 10 samples, however, this was done a total of 120 times to obtain more training data set.

We also recorded a video of us testing the program, which can also be seen.

The image shows a MATLAB Command Window interface. At the top, there is a blue header bar with the text "Command Window". Below the header, a yellow banner contains an information icon and the text "New to MATLAB? Watch this [Video](#), see [Demos](#), or read [Getting Started](#)." The main area of the window displays a series of commands and their outputs. The commands are "record voice" followed by "P =" and then a numerical prediction. The predictions are 1, 4, 2, 3, 4, and 7, corresponding to the six "record voice" commands shown. The text is in a monospaced font, typical of MATLAB's Command Window.

```
Command Window

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

record voice

P =

    1

record voice

P =

    4

record voice

P =

    2

record voice

P =

    3

record voice

P =

    4

record voice

P =

    7

record voice
```

Figure 3.5: Testing the program

Fig. 3.10 shows testing the program after the neural network had been trained. We pronounced a word from the list and our network predicted the said word and displayed it as P, that is, the prediction.

3.3.5 Results

We tested using a total of 84 test cases, with 17 of them being recorded in the video. A summary of the result obtained is shown in the table below.

Total # of test samples	Predicted correctly	Predicted incorrectly
84	73	11

Table 3.1: Results obtained

$$K = P/n$$

Where,

K - Accuracy

P - Correctly predicted test samples

n - Total samples

Therefore, putting values from Table 3.1 into the equation we get an accuracy of:

$$\text{Accuracy} = 86.9 \%$$

Chapter 4

Stuttered speech correction

4.1 PROPOSED SOLUTION

Not a lot of work has been done for stuttered speech recognition, but almost all papers have used neural networks as their primary technique for this problem by simply recording a lot of stuttered speech datasets and then feeding the system the correct speech, trying to make it learn.

Some papers are based only on stutter detection, which use MFCC and fuzzy logic to extract features from the given speech samples and use fuzzy logic along with neural networks, to detect stuttering.

Some have used HMM for stuttered speech recognition, by dividing the speech samples into phonemes and assigning probabilities for the different decisions the system is supposed to take, in case a pause is encountered, a phoneme is repeated or it is elongated.

A major drawback of all these approaches is the amount of computation power that will be required for the above techniques. When using just neural networks for stuttered speech recognition, it is impossible to come up with an efficient solution since there is no limited number of sentences than can be put into the system with all its permutations and combinations, to detect a sample every time.

Similarly, when using HMM with MFCC, a lot of different observations will have to predicted and the greater the dataset, more will be the number of observations for HMM to predict from and it might become less accurate.

So, we thought of a new, simple solution to overcome the enormous computation power and gigantic datasets, which is to use amplitude as a factor, to remove stuttering. We observed, that when a person stutters, the amplitude of his speech drops significantly, when compared to the rest, and hence, could be used to filter out the stutter from the sample.

So, we checked a lot of stuttered speech datasets for the above, and noticed that the threshold needed for removing the stutter from the sample largely depended on the maximum amplitude observed in the sample.

So, we created a neural network that would only take the maximum amplitude of the sample as an input and give the required threshold as an output. We initially selected the thresholds manually, by looking at the plots of the speech samples and after acquiring a large enough dataset, we used those values to train a neural network using back propagation algorithm. After the training was complete, the threshold output that was given by the network was accurate enough for us to clean the stuttered sample enough, that it could be easily recognised.

Since we eliminated the stuttering, we now simply had to pass a normal speech to an already existing system. Google is the leading company when it comes to speech recognition and they have spent years, perfecting it for different types of words, sequences, accents, etc. And the system that they have created has been open sourced for the use of general public, unlike any other advanced speech recognition platform.

So, after acquiring the clean speech, we created a python script to which we could pass this speech, from MATLAB. Then we call the services of Google's system, pass it our speech and the system does the rest of the work by analysing the speech and returning the result.

4.2 ALGORITHM

The algorithms for the correction and recognition of stuttered speech are as follows:

Stuttered Speech Recognition

1. *Take a speech sample from user (5 seconds with 8000Hz sampling frequency).*
2. *Insert a speech as a matrix in a variable (file).*
3. *Obtain the maximum amplitude of the speech.*
4. *Pass the maximum amplitude to the python script to compute a threshold value using neural networks*.*
5. *Divide the speech samples into short frames of equal length.*
6. *Analyse each frame and if the max value of the frame is greater than the threshold value, copy the frame onto a new signal variable.*
7. *Once all frames have been analysed, convert the variable into a .wav file.*
8. *Pass the .wav file to another python script, which will recognise the clean audio sample using Google's speech recognition API.*

***Neural Networks**

1. *Create a python script and import pybrain module to use the functions buildNetwork, SupervisedDataSet and BackpropTrainer.*
2. *Define a function which will take the maximum amplitude of the speech signal as an input.*
3. *Build a neural network specifying number of inputs, hidden layers and outputs, using buildNetwork function.*
4. *Define the type of dataset, again specifying the number of inputs and outputs using SupervisedDataSet function.*
5. *Add the training set to supervised dataset (at least 50), which comprises of maximum amplitude as input and the threshold for that amplitude as the output.*
6. *Call the BackpropTrainer function and pass the network and dataset to specify the algorithm being used to train the network (Back propogation)*
7. *Call the trainUntilConvergence function to train the data till it converges to a particular set of weights.*
8. *Now, use the activate function and pass the current input amplitude to it, to get the threshold value and return the given value to the MATLAB file.*

4.3 IMPLEMENTATION

We initially started off with recording voice samples on our laptop using the native recorded app on Windows 10. Then it was converted into .wav file since it is a format which is compatible with both, MATLAB and Python. After that, we put the file in its matrix format in MATLAB. Once, it was acquired on MATLAB, the filtering of stuttering could be started.

First, we plotted the speech in time against amplitude graph and visually looked for the stutter. After observing the stutter, the amplitude till which the stutter was noticed, is saved in a different variable. This variable will be used to filter the stutter out.

Now, the speech is divided into small frames. Each frame is checked for the maximum amplitude that it obtains, and if the value is greater than the threshold, the matrix values for that frame are copied to a new variable. This is done for the entire speech (or all samples) and after the process is complete, the new variable has a speech sample which is mostly free from stuttering.

Once this variable is obtained, we used MATLAB's in-built function 'audiowrite', to write this variable as a .wav audio file.

Then we wrote a python script and passed the new audio file to it. We called Google's services to use its speech recognition API. Once this was accomplished, we called the API and passed the audio to it. Google's servers compute the passed audio file online and returns its result.

Once all this was accomplished, it was important to automate the process to calculate the threshold value for removing the stutter. As mentioned above, we observed that the threshold value was largely dependent on the maximum amplitude that was present in the sample, though there were a few exceptions.

So, we created a python script to apply the back propagation algorithm to a neural network which had the maximum amplitude as its input and the threshold value as its output. To train the network, we recorded close to 50 samples, and manually checked their amplitudes and the required threshold, which could be used to train the network. We used simple sentences like, "I'm using a computer", "I'm reading a book" and "I like to play games" for this. After this, all that was left was to test the script by passing a sample's maximum amplitude value and checking the accuracy of the output threshold value, which was quite high.

The last step left was to integrate all the activities that were currently in different python and MATLAB files. We found out that since MATLAB 14, MATLAB had included a native support to call python scripts and functions from within its command window, with a few limitations to this ability. So, we changed the python scripts to fit the requirements of MATLAB, included the input recording of the voice samples directly into MATLAB and

merged everything into one file, which then became a standalone application and could be used for recognising any kind of stuttered speech sample.

4.4 SIMULATED RESULTS

4.4.1 Plots

The following plots show the stuttered speech and speech after the stutter has been removed for three different samples.

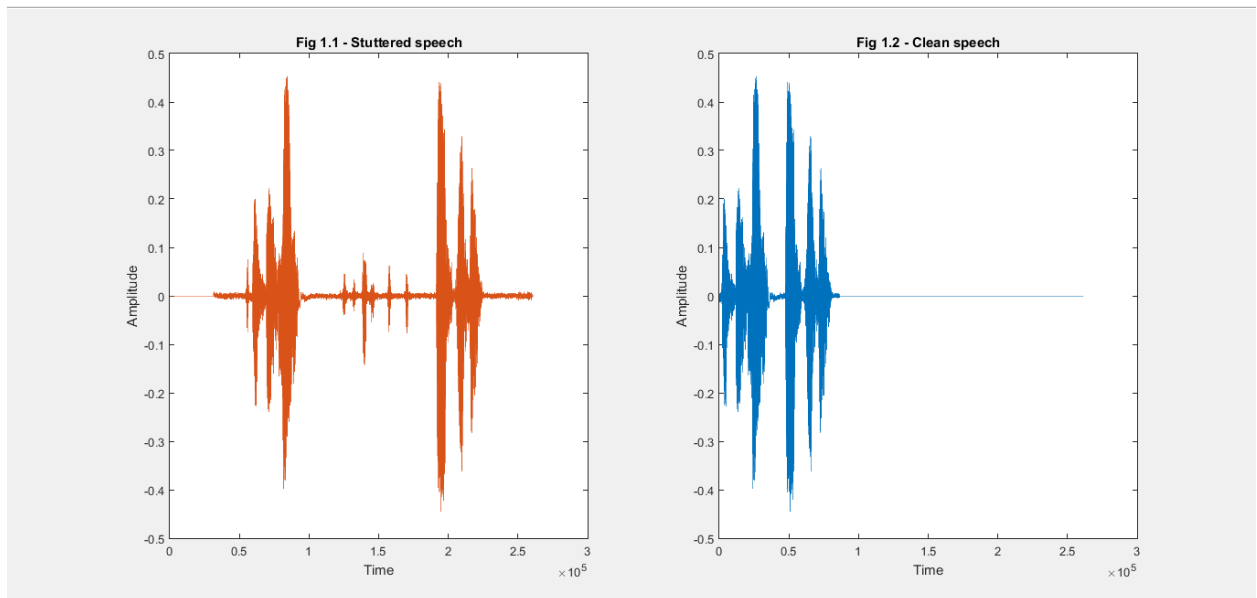


Figure 4.1: Stuttered and Clean Sample A

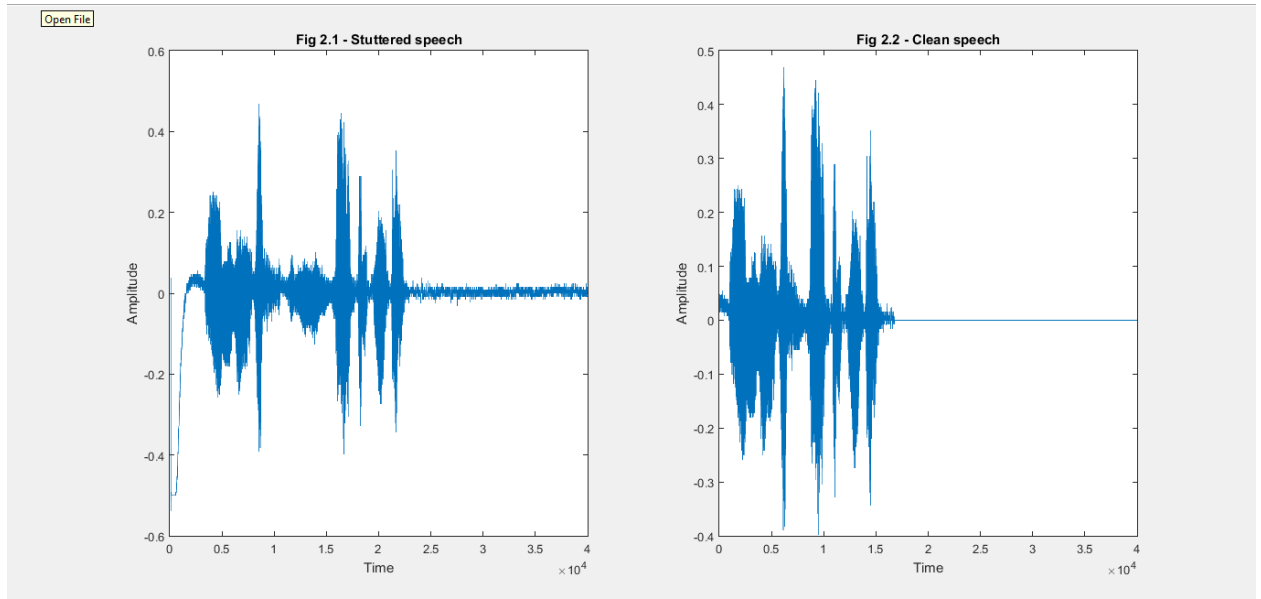


Figure 4.2: Stuttered and Clean Sample B

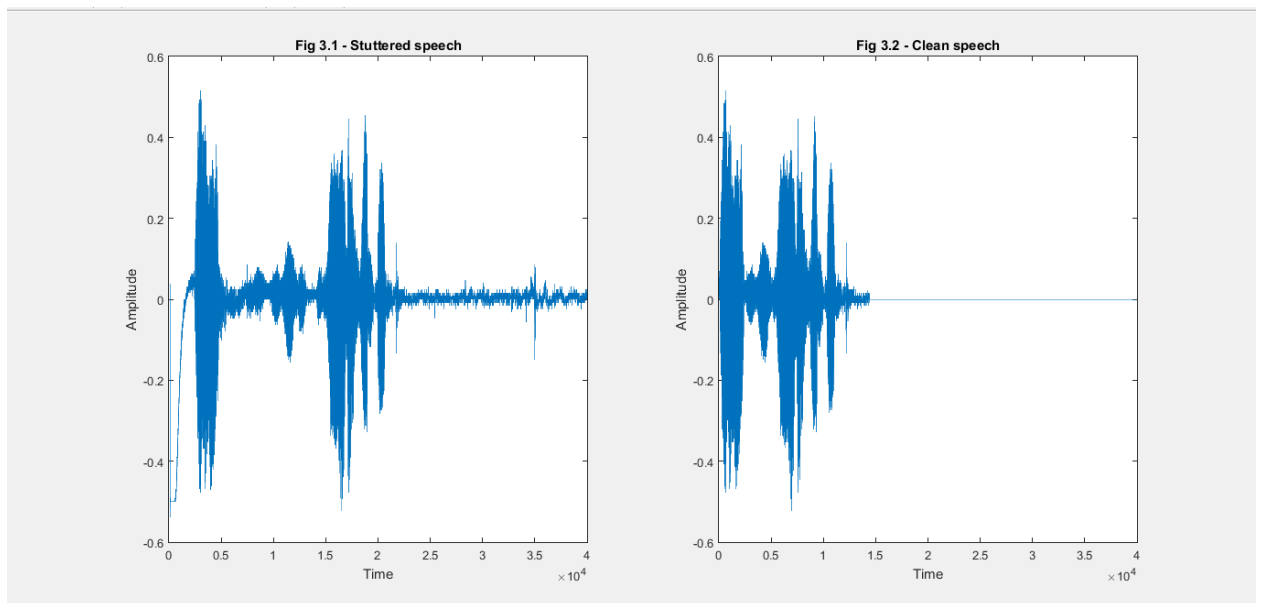


Figure 4.3: Stuttered and Clean Sample C

From the above plots, we can observe the difference in the amplitudes when stuttering occurs and how we used threshold values, which we obtained from the neural network that we trained, are used to clean the sample.

4.4.2 Tabulated Results

We tested using a total of 50 test cases, with 42 of them being detected correctly. A summary of the result obtained is shown in the table below.

Total # of test samples	Predicted correctly	Predicted incorrectly
50	42	8

Table 4.1: Results obtained

$$K = P/n$$

Where,

K - Accuracy

P - Correctly predicted test samples

n - Total samples

Therefore, putting values from Table 4.1 into the equation we get an accuracy of:

$$\text{Accuracy} = 84.0 \%$$

Chapter 5

Conclusion and Future Scope

5.1 CONCLUSION

The increasing usage of speech recognition systems by people has led to the ease of access in their day to day lives. People use personal assistants like Apple's Siri, Microsoft Cortana or Google Now and make their lives easier, however people with speech impairments like stuttering cannot benefit from these services because these companies have catered their speech recognition algorithms to the majority of people, that is, the people without any speech disorders even when about 70 million people in the world suffer from stuttering alone. These voice recognition systems are unable to detect when people afflicted by stuttering use it because when the person starts stuttering the service thinks that the person has completed speaking and doesn't process what comes subsequently.

In order to make these above mentioned applications more universal we worked a project that would actually help in solving a real world problem and we were successful in getting favourable outcomes. Not a lot of work has been done using amplitude as the primary technique of elimination of stutter, and this makes our project novel and research oriented.

Our speech recognition system consisting of 10 words has an accuracy of 86.9 % on a test set of 84 samples, we did not extend it to more words as Google and Microsoft have already created extensive open source speech recognition APIs that help in accurate recognition of normal speech.

The stuttered speech correcting program was able to achieve an accuracy of 84% on 50 test samples. This shows promise for our approach and the accuracy can further be improved by increasing the number of training samples that we used in the training set.

5.1 FUTURE SCOPE

1. Integration of our solution with already existing speech recognition services across all platforms like PC, mobile, etc. Which would enable the affected persons to use speech recognition tools and services even with their stutter.

2. We can further increase the accuracy and effectiveness of our technique by acquiring more data samples from affected individuals, which would result in a larger training set, thus making our neural network more robust.

3. We could also use another parameter along with amplitude to better detect and correct the stuttered speech.

4. Stuttering is only one of the common speech disorders, we could also implement the same with other speech impediments like lisp, etc.

REFERENCES

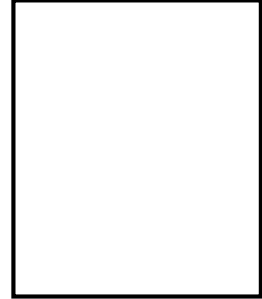
1. <http://www.asha.org/public/speech/disorders/stuttering/>
2. <https://en.wikipedia.org/wiki/Stuttering>
3. Overview of Automatic Stuttering Recognition System, Lim Sin Chee, Ooi Chia Ai, Sazali Yaacob, Universiti Malaysia Perlis Jalan Kangar-Arau 02600 Jejawi Perlis
4. <https://www.willamette.edu/~gorr/classes/cs449/Classification/perceptron.html>
5. <https://en.wikipedia.org/wiki/Backpropagation>
6. http://www.slideshare.net/Krish_ver2/25-backpropagation
7. http://www.aihorizon.com/essays/generalai/supervised_unsupervised_machine_learning.htm
8. <http://www.analyticsvidhya.com/blog/2015/10/understaing-support-vector-machine-example-code/>
9. Robert M. Gray, IEEE Signal Processing Society, Distinguished Lecturer Program
10. Cohen, J., Cohen P., West, S.G., & Aiken, L.S. (2003). Applied multiple regression/correlation analysis for the behavioral sciences. (2nd ed.) Hillsdale, NJ: Lawrence Erlbaum Associates
11. Draper, N.R.; Smith, H. (1998). Applied Regression Analysis (3rd ed.). John Wiley. ISBN 0-471-17082-8.
12. Automatic Speech Recognition for People with Speech Disorders, Manthiba Ramaboka, Jonas Manamela, Nelson Gasela, Telkom Centre for Excellence for Speech Technology, Department of Computer Science, University of Limpopo (Turfloop Campus)
13. Stuttered Speech Recognition for Robotic Control □ G. Manjula, Dr. M. Shiva Kumar
14. RECOGNITION OF REPETITIONS USING SUPPORT VECTOR MACHINES, Juraj P'alfy and Jiř'ı Posp'ıchal, Institute of Applied Informatics, Faculty of Informatics and Information Technologies
Slovak University of Technology, Ilkovičova 3, 842 16 Bratislava 4, Slovakia
15. Structure Analysis of Speech Signals and Filtering them from Noise, Anastasiya Andreevna Kuznetsova, Admiral Makarov State University of Maritime and Inland Shipping, Saint-Petersburg, Russia
16. Automatic Detection of Syllable Repetition in Read Speech for Objective Assessment of Stuttered Disfluencies K. M. Ravikumar, Balakrishna Reddy, R. Rajagopal, and H. C. Nagaraj

17. Neural Networks used for Speech Recognition Wouter Gevaert, Georgi Tsenov, Valeri Mladenov, Senior Member, IEEE
18. www.coursera.com/machine-learning - course by Andrew Ng
19. LPC and MFCC Performance Evaluation with Artificial Neural Network for Spoken Language Identification by Eslam Mansour mohammed, Mohammed Sharaf Sayed, Abdallaa Mohammed Moselh and Abdelaziz Alsayed Abdelnaiem
20. SPEECH RECOGNITION - Report of an Isolated Word experiment by Philip Felber, Illinois Institute of Technology
21. Pieraccini, Roberto. The Voice in the Machine. Building Computers That Understand Speech. The MIT Press. ISBN 978-0262016858.
22. Woelfel, Matthias; McDonough, John. Distant Speech Recognition. Wiley. ISBN 978-0470517048.

Curriculum vitae

Limited to One page

Name	:	Anshuman S. Dhamoon
Father's name	:	Amandeep S. Dhamoon
Date of Birth	:	09/08/1994
Nationality	:	Indian
Sex	:	Male
Company placed	:	Matrimony.com
Permanent Address	:	B-3/74, Janakpuri, New Delhi - 110058
Phone Number	:	+919999084042
Mobile	:	+919159869249
Email Id	:	anshumandhamoon@gmail.com



CGPA : 8.75

Examinations taken:

1. GRE: 317/340

Placement Details : Technical Consultant at Matrimon.com

Curriculum vitae

Limited to One page

Name	:	Sarthak Khanna
Father's name	:	Himanshu Khanna
Date of Birth	:	06/03/1994
Nationality	:	Indian
Sex	:	Male
Company placed	:	NA
Permanent Address	:	A-203, Neel Sidhi Splendor, Sector 15, CBD Belapur, Navi Mumbai, Maharashtra - 400614
Phone Number	:	+919323549829
Mobile	:	+917639050900
Email Id	:	khannasarthak.1994@gmail.com
CGPA	:	8.63

[Recent
Photo.]

Examinations taken:

1. GRE: 319/340
2. TOEFL: 113/120
3. WES Evaluated GPA: 4.0/4.0

Placement Details : Did not sit for placements. Will be pursuing **Masters** in **Computers Science at TU Delft, Netherlands.**