Stony Brook University

ESE 588 - Pattern Recognition

Project 2

Gautham Botlaguduru
112608440
Devesh Murugan
112874016

Contents

1 Introduction

Machine learning algorithms are rapidly being used in the domain for various applications such as speech recognition, speech to text systems, medical imaging diagnosis, smart health records etc. Analysis and application of machine learning algorithms to Speech Processing implies a strong analytical system capable of quickly interpreting audio waveforms with high fidelity.
The proposed project explores the domain of speech procesing using techniques we have learnt in class. The project involves the implementation of various speech processing techniques, dirichlet process clustering and the concept of slotted ALHOA from Wireless Networks.

Audio and speech analysis has been a subject of major interest amongst signal processing engineers for a long time. For the creative component we took up the task clustering voiced speech segments.
Voiced and unvoiced speech are defined as follows. Speech is composed of phonemes, which are produced by the vocal cords and the vocal tract (which includes the mouth and the lips). Voiced signals are produced when the vocal cords vibrate during the pronounciation of a phoneme. Unvoiced signals, by contrast, do not entail the use of the vocal cords. For example, the only difference between the phonemes /s/ and /z/ or /f/ and /v/ is the vibration of the vocal cords.

Voiced signals tend to be louder like the vowels /a/, /e/, /i/, /u/, /o/. Unvoiced signals, on the other hand, tend to be more abrupt like the stop consonants /p/, /t/, /k/. In this project, our goal was to build a model to cluster speech fragments. To achieve this, the following steps were followed:

- Built a vocabulary of only two words(bus and train).
- Recorded hours of audio samples.
- Segmenting samples to 1 second frames for speech processing.
- Feature Extraction in three domains.
- Dataset creation based on the features extracted.
- Clustering the data by implementing Dirichlet process clustering.
- Using the labels to predict to new data samples.
- Finally, we slot a new speech waveform built with the vocabulary to count the number of utterances of each word. We find their locations. Such a process is used for context extraction in speech.

The problem statement is as follows:

2 Problem Statement

In short, the problem is to build a speech to text system. For the sake of simplicity, a small vocabulary is considered. Raw speech is recorded with the words bus and train forming the vocabulary. A speech to text system is to built using traditional speech processing techniques and basic pattern recognition techniques. It is also required to analyze the text output i.e., count the number of occurences of each word, their locations of occurences.

3 Plan of Action

To tackle the above problem, we take the steps listed below and discussed in detail in later sections:

- Capture real data by recording audio and building a vocabulary. For this case, the vocabulary would have two words (bus and train).
- From the recorded audio extract utterances of each word.
- For each audio sample, extract features (will involve speech processing concepts)
- Build a dataset.
- For this dataset, cluster it using an implementation of Dirichlet Process Clustering.
- Assign a label to each cluster.
- Validate the model.
- Given a new speech waveform formed only from words from the vocabulary, find the number of utterances of each word, their locations and convert the speech waveform to text.

One of the most difficult problems is identifying an object location in image and a word utterance in speech. We simplify the problem here to slotted speech for which we propose an algorithm using concepts of slotted aloha. In future we plan to extend the algorithm to continuous striding windows which would use concepts of the softmax function and maximum likelihood estimation.

In the coming sections, we discuss each point above in detail.

4 Speech Samples Source

To obtain reliable data, we have recorded ourselves saying the words 'bus' and 'train' for over 10 hours. From the raw speech data we performed gaussian blurring to denoise the signal.
To reduce computation complexity, we resampled the audio files to 8bit, 8000 Hz, mono wav formats. We then segmented the audio files to 1 second frames.

5 Feature Extraction

Feature extraction is an essential part of any learning algorithm. It is a dimensionality reduction technique which makes traversing through datasets fairly quick and easy.
Feature extraction is the task extracting vectors from raw data points to form an N-dimensional feature space where linear combinations of these vectors give back the raw data points. If these features are independent of each other, then they form the bases vectors for the N-dimensional feature space(and this is the ideal case). Practically, it is observed that there is an interdependence. Representation of raw points using the extracted features often bring out hidden information from the data. For example, the periodicity of a speech waveform is not obvious but is a feature that must be extracted in the time domain. Similarly, the Fourier transform (and its variants like the short time Fourier transform and the wavelet transform) provide insights into the frequency and time-frequency domains of the signal, which are not directly visible in its raw time format. Therefore, feature extraction is a crucial step that helps us represent signals(or data) in a compact manner while bringing out new information. Extracted features can then be used to model various systems (e.g, a speech model). These features can also be used in the domain of data analytics to build intelligent systems capable of taking decisions.
In a nutshell, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations.

5.1 Sparse Filtering

The sparse filtering operation creates a nonlinear transformation of input features to output features. The transformation is based on optimizing an objective function that encourages the representation of each example by as few output features as possible while at the same time keeping the output features equally active across examples.

The sparse filtering algorithm does not try to model the data's distribution but rather to learn features which are sparsely activated, in the sense that

- for each example, only a small subset of features is activated ("Population Sparsity").
- each feature is only activated on a small subset of he examples ("Lifetime Sparsity")
- features are roughly activated equally often ("High Dispersal")

This sparsity is encoded as an objective function and L-BFGS is used to minimize this function.

6 Feature Domains

Feature extraction was done in three domains:

- Time
- Frequency
- Cepstral

6.1 Time Domain

Time domain is the domain in which the signal exists and is perceived. A number of useful features can be extracted from the time domain.
The signal (1 second long) is strided over with a window size of 50 ms and an overlap of 20 ms. The following time domain (or temporal domain) features are of usual interest:

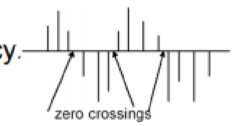
- Energy of the segment

Short-time energy of weighted signal around *n* is defined as

$$E_n = \sum_{m=-\infty}^{\infty} [x(m)w(n-m)]^2$$

- Zero crossing Rate

  - A zero-crossing occurs if successive samples have different algebraic signs.
  - It is a measure of the frequency.
  - Definition



$$Z_n = \sum_{m=-\infty}^{\infty} |\operatorname{sgn}[x(m)] - \operatorname{sgn}[x(m-1)]|\, w(n-m)$$

where

$$\operatorname{sgn}[x(n)] = \begin{cases} 1 & x(n) \geq 0 \\ -1 & x(n) < 0 \end{cases}$$

and

$$w(n) = \begin{cases} \frac{1}{2N} & 0 \leq n \leq N-1 \\ 0 & otherwise \end{cases}$$

- Maximum, Minimum Amplitude
- Periodicity for voiced audio segments.

6.2 Frequency Domain

A domain transformation always gives new insights into the data. Some features are not directly extractable in the temporal domain and thus a fourier transformation is applied to get better insights into the frequency domain of the signal. Again, useful features from the frequency domain were extracted.

- Fundamental frequency.
- power spectrum
- Frequency of max amplitude
- frequency of min amplitude
- frequency standard deviation

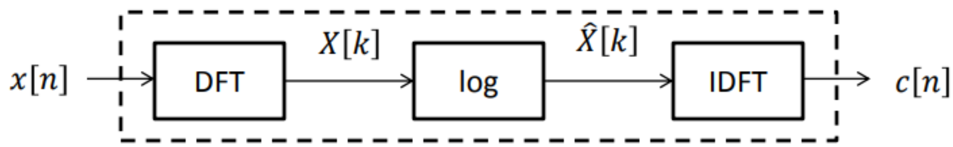
amongst others.

6.3 Cepstral domain

As it turns out, the time domain and frequency domain features are not distinct enough for speech recognition and detection because the concept of scale space is missing in these domains . A number of transforms such as the short time fourier transform and wavelet transforms are alternate domains where speech analysis is done. The most popular domain for speech analysis is the cepstral domain. The cepstral domain of a time domain signal is obtained by taking the inverse fourier transform of the log of it's absolute value of the fourier transform. Mathematically,

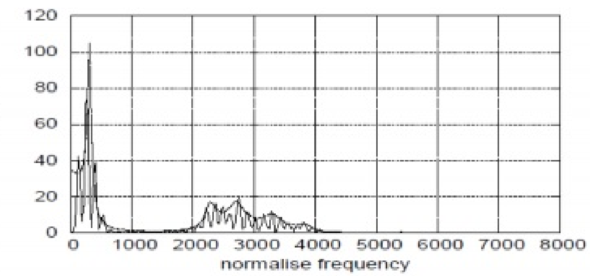
$$c[n] = FT^{-1}\left\{log\left|FT\left\{x[n]\right\}\right|\right\}$$

A block diagram of the implementation is as follows:

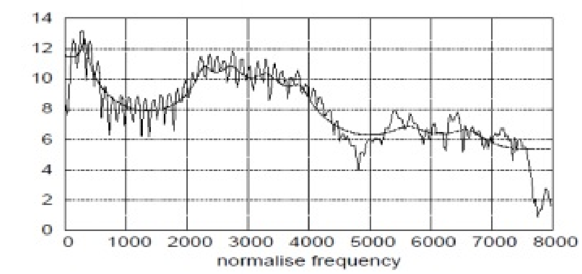


The illustrations of the fourier transform, log of the FT and the cepstral domain signal are shown below for reference:

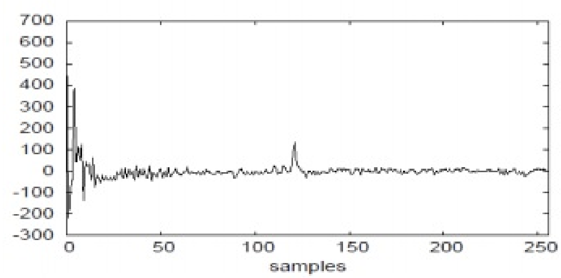
$\mathcal{F}\{x[n]\}$



$\log|\mathcal{F}\{x[n]\}|$



$\mathcal{F}^{-1}\{\log|\mathcal{F}\{x[n]\}|\}$



Note:

- Filtering in the cepstral domain is called liftering.
- Since it is not time or frequency domain, the independent variable is called quefrency.
- A very important application of the cepstral domain is that the convolution of two signals is computed as the sum of their cepstra.

Some important features of the cepstral domain are:

- Pitch (for voiced signals)
- Formant frequencies
- Mel frequency coffficients

Pitch of voiced signals

An important property of the cepstral coefficents is that they describe the periodicity present in the signal. As we know, music (Western and Indian) have periodicity as defined the rhythm and taal of the composition. It is therefore important for us to exploit this inherent periodicity. The pitch of the signal can be easily found in the cepstral domain by finding the number of the cofficient of the peak.

Formant frequency

Formant frequencies are the resonances of the acoustic scource. They can also be visualised as the local maxima in the spectrum of the signal. It can be computed using linear predictive coding. For this project, we have extracted three formant frequencies bandlimited at 8000 Hz by using the inbuilt lpc library in matlab.

Mel Frequency coefficients

Since the cepstral domain takes into consideration the log relationship between frequency and the spectral content at that frequency, it closely resembles the response of the human ear. To further improve on this modelling, the frequency axis is transformed to a different scale called the mel-scale. It is given as follows:

$$m = 2595 * log_{10}\left(1 + \frac{f}{700}\right)$$

The inverse exists and can be easily found.

We pick weighted samples at frequencies f_k as follows:

$$u_k = \sum_{f_{k-1}+1}^{f_{k+1}-1} w_{k,h} \left| x_h \right|^2$$

where $w_{k,h}$ are scalar weighting parameters. With this we get a robust estimate of the sampled signal and a perceptual frequency scale. The weights are usually triangular functions given as shown in the figure below:

$$w_{k,h} = \begin{cases} \dfrac{h - f_{k-1}}{f_k - f_{k-1}} & \text{for} \quad f_{k-1} < h \leq f_k, \\ \dfrac{f_{k+1} - h}{f_{k+1} - f_k} & \text{for} \quad f_k < h \leq f_{k+1}, \\ 0 & \text{otherwise.} \end{cases}$$

We finally take the DCT of u_k to get the Mel Frequency Coefficients (MFCC). We ignore the first coefficient which is the bias component while building the model.

7 Dirichlet Process

The Dirichlet distribution Dir(α) is a family of continuous multivariate probability distributions parameterized by a vector α of positive reals. It is a multivariate generalisation of the Beta distribution. Dirichlet distributions are commonly used as prior distributions in Bayesian statistics.

In Bayesian probability theory, if the posterior distribution p(θ—x) and the prior distribution p(θ) are from the same probability distribution family, then the prior and posterior are called conjugate distributions, and the prior is the conjugate prior for the likelihood function. If we think about the problem of inferring the parameter θ for a distribution from a given set of data x, then Bayes' theorem says that the posterior distribution is equal to the product of the likelihood function $\theta \rightarrow$ p(x—θ) and the prior p(θ), normalised by the probability of the data p(x):

$$p(\boldsymbol{\theta}|\mathbf{x}) = \frac{p(\mathbf{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathbf{x}|\boldsymbol{\theta}')p(\boldsymbol{\theta}')d\boldsymbol{\theta}'}$$

Since the likelihood function is usually defined from the data generating process, we can see that the difference choices of prior can make the integral more or less difficult to calculate. If the prior has the same algebraic form as the likelihood, then often we can obtain a closed-form expression for the posterior, avoiding the need of numerical integration.

The Dirichlet distribution defines a probability density for a vector valued input having the same characteristics as our multinomial parameter θ. It has support (the set of points where it has non-zero values) over

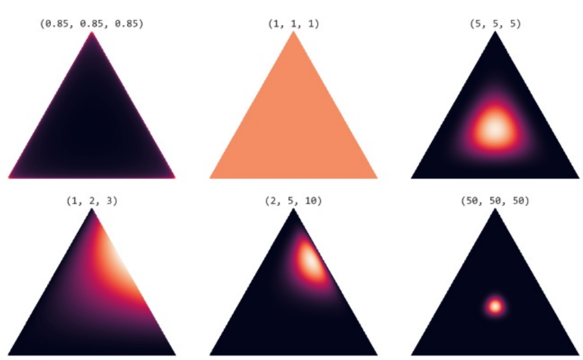
$$x_1, \ldots, x_K \text{ where } x_i \in (0,1) \text{ and } \sum_{i=1}^{K} x_i = 1$$

where K is the number of variables. Its probability density function has the following form:

$$\mathrm{Dir}(\boldsymbol{\theta}|\boldsymbol{\alpha}) = \frac{1}{\mathrm{Beta}(\boldsymbol{\alpha})} \prod_{i=1}^{K} \boldsymbol{\theta}_i^{\alpha_i - 1}, \text{ where } \mathrm{Beta}(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)} \text{ and } \boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k).$$

The Dirichlet distribution is parameterised by the vector α, which has the same number of elements K as our multinomial parameter θ. So you can interpret p(θ—α) as answering the question “what is the probability density associated with multinomial distribution θ, given that our Dirichlet distribution has parameter α.”

For the Dirichlet distribution Dir($alpha$) we generalise these shapes to a K simplex. For K=3, visualising the distribution requires us to do the following: 1. Generate a set of x-y coordinates over our triangle, 2. Map the x-y coordinates to the 2-simplex coordinate space, 3. Compute Dir(α) for each point.



8 Dirichlet Process Clustering

The dimensionality of a clustering model is equal to the number of clusters in the model. Bayesian clustering algorithms often rely of the Dirichlet Distribution to encode prior information about these cluster assignments. The Dirichlet distribution (DD) can be considered a distribution of distributions. Each sample from the DD is a categorial distribution over K categories. It is parameterized G0, a distribution over K categories and α, a scale factor.

The expected value of the DD is G0. The variance of the DD is a function of the scale factor. When α is large, samples from DD(αG0) will be very close to G0. When α is small, samples will vary more widely.

Most clustering algorithms require the number of clusters to be given in advance. If we don't know the number of species in the data, how would we be able to classify these flowers?

The Dirichlet Process is a stochastic process used in Bayesian nonparametrics to cluster data without specifying the number of clusters in advance. The Dirichlet Process Prior is used to mathematically descirbe prior information about the number of clusters in the data.

The Dirichlet Process Prior is nonparametric because its dimensionality is infinite. This characteristic is advantageous in clustering because it allows us to recognize new clusters when we observe new data.

The Dirichlet Process can be considered a way to generalize the Dirichlet distribution. While the Dirichlet distribution is parameterized by a discrete distribution G0 and generates samples that are similar discrete distributions, the Dirichlet process is parameterized by a generic distribution H0 and generates samples which are distributions similar to H0. The Dirichlet process also has a parameter α that determines how similar how widely samples will vary from H0.

When learning a clustering using a Dirichlet Process Prior, observations are probabilistically assigned to clusters based on the number of observations in that cluster nk.

$$P(\text{cluster=k}) = \frac{n_k}{\alpha + n - 1}$$

Probability of new clusters are paramteterized by α

$$P(\text{cluster=new}) = \frac{\alpha}{\alpha + n - 1}$$

In other words, these clusters exhibit a rich get richer property.

The expected number of clusters in a dataset clustered with the Dirichlet Process is O(αlog(N))

The expected number of clusters with m observations is $\frac{\alpha}{m}$.

We now look at how we can generate samples from a dirichlet process. We focus on the chinese restaurant process and the stick breaking process.

Chinese Restaurant Process

We will define a distribution on the space of partitions of the positive integers, N. This would induce a distribution on the partitions of the first n integers, for every n ϵ N. Imagine a restaurant with countably infinitely many tables, labelled 1, 2, ... Customers walk in and sit down at some table. The tables are chosen according to the following random process.

- The first customer always chooses the first table.
- The nth customer chooses the first unoccupied table with probability $\frac{\alpha}{n-1+\alpha}$ and an occupied table with probability $\frac{c}{n-1+\alpha}$ where c is the number of people sitting at that table.

In the above, α is a scalar parameter of the process. One might check that the above does define a probability distribution. Let us denote by kn the number of different tables occupied after n customers have walked in. Then $1 \leq$ kn $\leq$ n and it follows from the above description that precisely tables 1, . . . , kn are occupied.

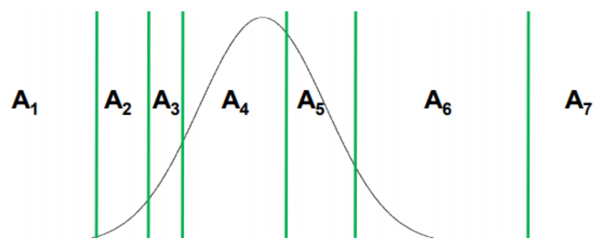
Stick Breaking Process

This method, developed by Sethuraman in 1994 is very briefly given below. The content for this process has been borrowed and has been listed in the references. Draws from a Dirichlet process are distributions over a set S. The distribution drawn is discrete with probability 1. In the stick-breaking process view, we explicitly use the discreteness and give the probability mass function of this (random) discrete distribution.

Formal Definition (not constructive)

- Let α be a positive, real-valued scalar
- Let G_0 be a non-atomic probability distribution over support set A
- If G ~ DP(α, G_0), then for any finite set of partitions
  $A_1 \cup A_2 \cup \ldots \cup A_k$ of A:

$$(G(A_1), \ldots, G(A_k)) \sim \text{Dirichlet}(\alpha G_0(A_1), \ldots, \alpha G_0(A_k))$$



1. Draw X_1^* from G_0
2. Draw v_1 from Beta(1, α)
3. $\pi_1 = v_1$
4. Draw X_2^* from G_0
5. Draw v_2 from Beta(1, α)
6. $\pi_2 = v_2(1 - v_1)$

...

9 Experiment Setup and Assumptions

In this section, we discuss the experiment setup and assumptions. This is important to understand future possibilities.

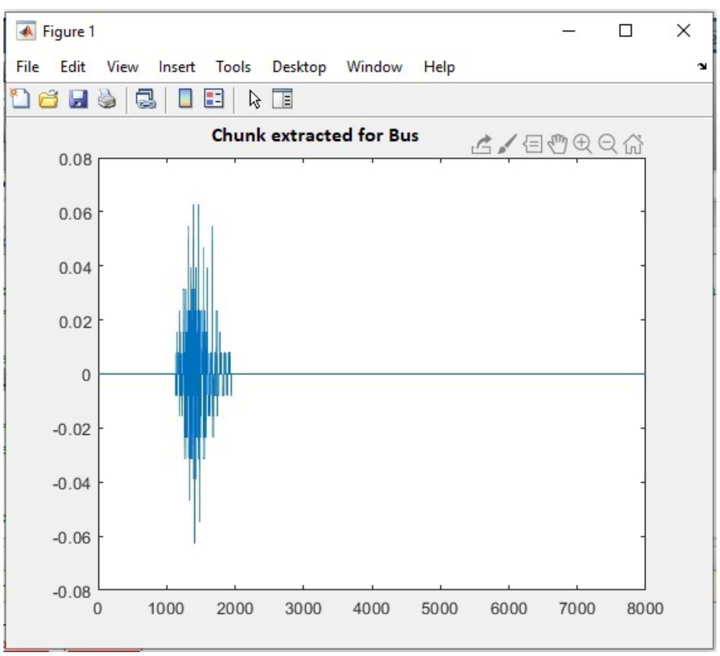
- Audio is recorded by saying either 'Bus' or 'Train' every second.
- The Audio is 8-bit, 8000Hz mono signal.
- An important assumption is that one word is spoken every second. This gives us flexibility for processing the signal.
- The vocabulary, V = (Bus, Train)

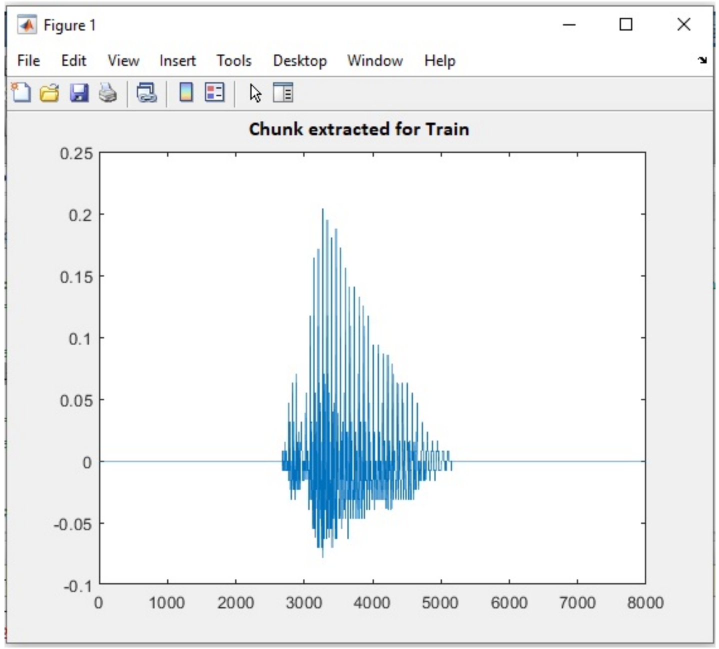
10 Results

In this section, we go through a detailed analysis of results obtained at each step.

10.1 Chunk extraction

The first step after recording the audio is to extract 1 second chunks from it. A small python script was written for this purpose. The following figures show the same.

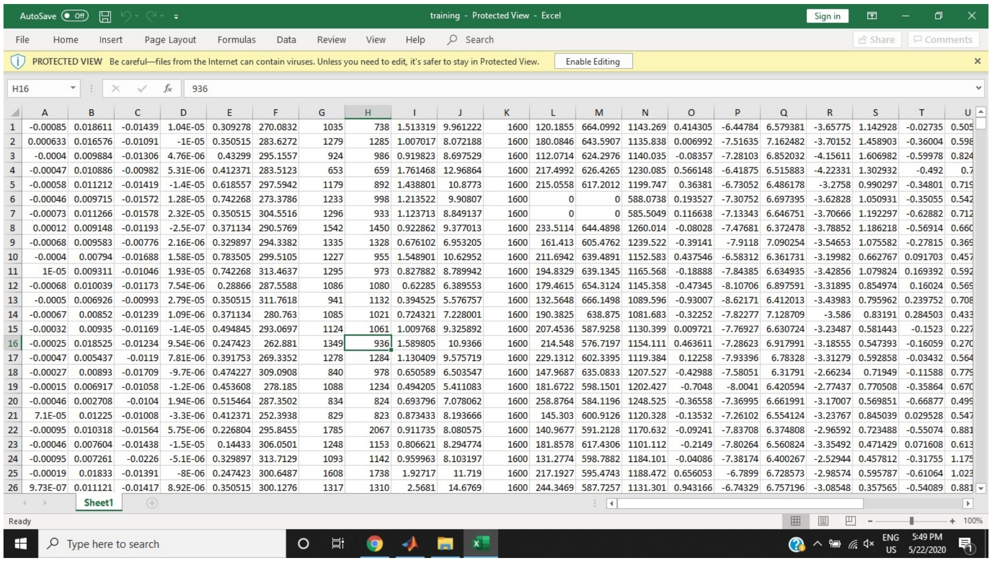




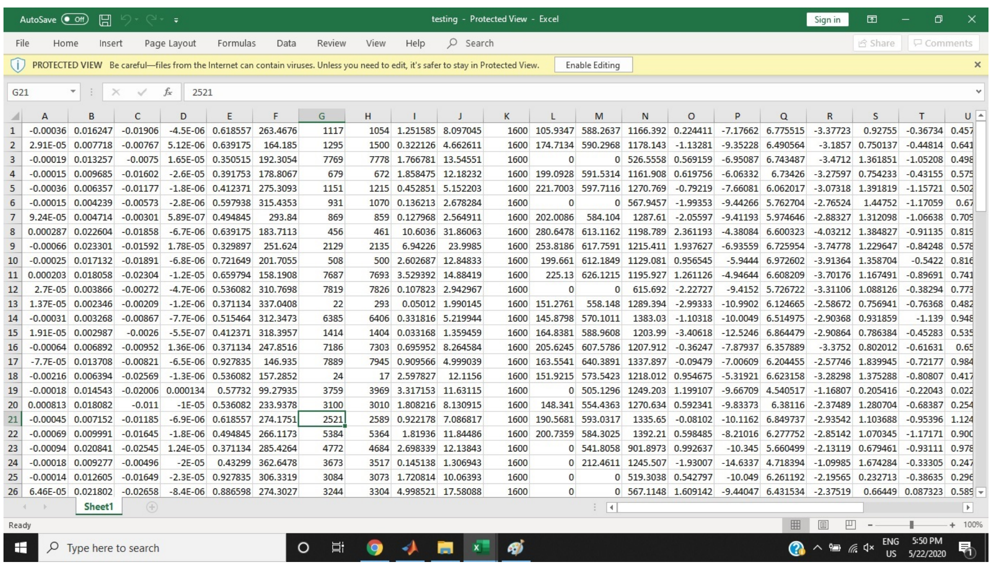
Around 15000 chunks were extracted for this project but to save on computational complexity we have used 6000 chunks only.

10.2 Feature Extraction and Dataset Generation

For each chunk, we perform feature extraction in the three domains i.e., time, frequency and cepstral domains. The signal is first is denoised using a gaussian blur. A sliding window is used to extract the corresponding features.
A very important step in generating the datatset is to split it into testing and training. This is done by randomly assigning a feature descriptor as test or train keeping in mind that their desired ratio is 80/20. The features are tabulated. A screenshot of the training and testing data is shown below.



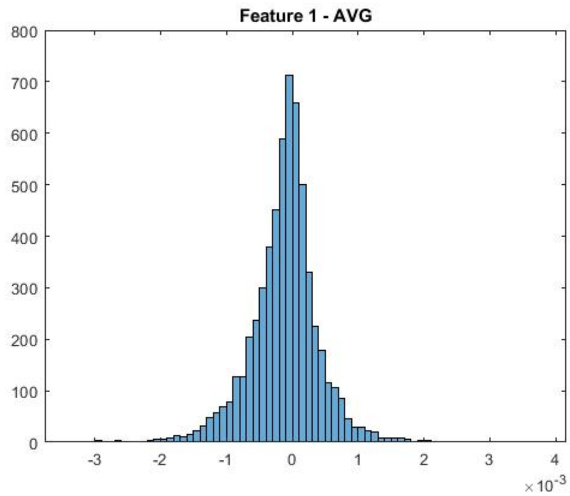
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	-0.00085	0.018611	-0.01439	1.04E-05	0.309278	270.0832	1035	738	1.513319	9.961222	1600	120.1855	664.0992	1143.269	0.414305	-6.44784	6.579381	-3.65775	1.142928	-0.02735	0.505
2	0.000633	0.016576	-0.01091	-1E-05	0.350515	283.6272	1279	1285	1.007017	8.072188	1600	180.0846	643.5907	1135.838	0.006992	-7.51635	7.162482	-3.70152	1.458903	-0.36004	0.598
3	-0.0004	0.009884	-0.01306	4.76E-06	0.43299	295.1557	924	986	0.919823	8.697529	1600	112.0714	624.2976	1140.035	-0.08357	-7.28103	6.852032	-4.15611	1.606982	-0.59978	0.824
4	-0.00047	0.010886	-0.00982	5.31E-06	0.412371	283.5123	653	659	1.761468	12.96864	1600	217.4992	626.4265	1230.085	0.566148	-6.41875	6.515883	-4.22331	1.302932	-0.492	0.7
5	-0.00058	0.011212	-0.01419	-1.4E-05	0.618557	297.5942	1179	892	1.438801	10.8773	1600	215.0558	617.2012	1199.747	0.36381	-6.73052	6.486178	-3.2758	0.990297	-0.34801	0.719
6	-0.00046	0.009715	-0.01572	1.28E-05	0.742268	273.3786	1233	998	1.213522	9.90807	1600	0	0	588.0738	0.193527	-7.30752	6.697395	-3.62828	1.050931	-0.35055	0.542
7	-0.00073	0.011266	-0.01578	2.32E-05	0.350515	304.5516	1296	933	1.123713	8.849137	1600	0	0	585.5049	0.116638	-7.13343	6.646751	-3.70666	1.192297	-0.62882	0.712
8	0.00012	0.009148	-0.01193	-2.5E-07	0.371134	290.5769	1542	1450	0.922862	9.377013	1600	233.5114	644.4898	1260.014	-0.08028	-7.47681	6.372478	-3.78852	1.186218	-0.56914	0.660
9	-0.00068	0.009583	-0.00776	2.16E-06	0.329897	294.3382	1335	1328	0.676102	6.953205	1600	161.413	605.4762	1239.522	-0.39141	-7.9118	7.090254	-3.54653	1.075582	-0.27815	0.369
10	-0.0004	0.00794	-0.01688	1.58E-05	0.783505	299.5105	1227	955	1.548901	10.62952	1600	211.6942	639.4891	1152.583	0.437546	-6.58312	6.361731	-3.19982	0.662767	0.091703	0.457
11	1E-05	0.009311	-0.01046	1.93E-05	0.742268	313.4637	1295	973	0.827882	8.789942	1600	194.8329	639.1345	1165.568	-0.18888	-7.84385	6.634935	-3.42856	1.079824	0.169392	0.592
12	-0.00068	0.010039	-0.01173	7.54E-06	0.28866	287.5588	1086	1080	0.62285	6.389553	1600	179.4615	654.3124	1145.358	-0.47345	-8.10706	6.897591	-3.31895	0.854974	0.16024	0.569
13	-0.0005	0.006926	-0.00993	2.79E-05	0.350515	311.7618	941	1132	0.394525	5.576757	1600	132.5648	666.1498	1089.596	-0.93007	-8.62171	6.412013	-3.43983	0.795962	0.239752	0.708
14	-0.00067	0.00852	-0.01239	1.09E-06	0.371134	280.763	1085	1021	0.724321	7.228001	1600	190.3825	638.875	1081.683	-0.32252	-7.82277	7.128709	-3.586	0.83191	0.284503	0.433
15	-0.00032	0.00935	-0.01169	-1.4E-05	0.494845	293.0697	1124	1061	1.009768	9.325892	1600	207.4536	587.9258	1130.399	0.009721	-7.76927	6.630724	-3.23487	0.581443	-0.1523	0.227
16	-0.00025	0.018525	-0.01234	9.54E-06	0.247423	262.881	1349	936	1.589805	10.9366	1600	214.548	576.7197	1154.111	0.463611	-7.28623	6.917991	-3.18555	0.547393	-0.16059	0.270
17	-0.00047	0.005437	-0.0119	7.81E-06	0.391753	269.3352	1278	1284	1.130409	9.575719	1600	229.1312	602.3395	1119.384	0.12258	-7.93396	6.78328	-3.31279	0.592858	-0.03432	0.564
18	-0.00027	0.00893	-0.01709	-9.7E-06	0.474227	309.0908	840	978	0.650589	6.503547	1600	147.9687	635.0833	1207.527	-0.42988	-7.58051	6.31791	-2.66234	0.71949	-0.11588	0.779
19	-0.00015	0.006917	-0.01058	-1.2E-06	0.453608	278.185	1088	1234	0.494205	5.411083	1600	181.6722	598.1501	1202.427	-0.7048	-8.0041	6.420594	-2.77437	0.770508	-0.35864	0.670
20	-0.00046	0.002708	-0.0104	1.94E-06	0.515464	287.3502	834	824	0.693796	7.078062	1600	258.8764	584.1196	1248.525	-0.36558	-7.36995	6.661991	-3.17007	0.569851	-0.66877	0.499
21	7.1E-05	0.01225	-0.01008	-3.3E-06	0.412371	252.3938	829	823	0.873433	8.193666	1600	145.303	600.9126	1120.328	-0.13532	-7.26102	6.554124	-3.23767	0.845039	0.029528	0.547
22	-0.00095	0.010318	-0.01564	5.75E-06	0.226804	295.8455	1785	2067	0.911735	8.080575	1600	140.9677	591.2128	1170.632	-0.09241	-7.83708	6.374808	-2.96592	0.723488	-0.55074	0.881
23	-0.00046	0.007604	-0.01438	-1.5E-05	0.14433	306.0501	1248	1153	0.806621	8.294774	1600	181.8578	617.4306	1101.112	-0.2149	-7.80264	6.560824	-3.35492	0.471429	0.071608	0.613
24	-0.00095	0.007261	-0.0226	-5.1E-06	0.329897	313.7129	1093	1142	0.959963	8.103197	1600	131.2774	598.7882	1184.101	-0.04086	-7.38174	6.400267	-2.52944	0.457812	-0.31755	1.175
25	-0.00019	0.01833	-0.01391	-8E-06	0.247423	300.6487	1608	1738	1.92717	11.719	1600	217.1927	595.4743	1188.472	0.656053	-6.7899	6.728573	-2.98574	0.595787	-0.61064	1.023
26	9.73E-07	0.011121	-0.01417	8.92E-06	0.350515	300.1276	1317	1310	2.5681	14.6769	1600	244.3469	587.7257	1131.301	0.943166	-6.74329	6.757196	-3.08548	0.357565	-0.54089	0.881

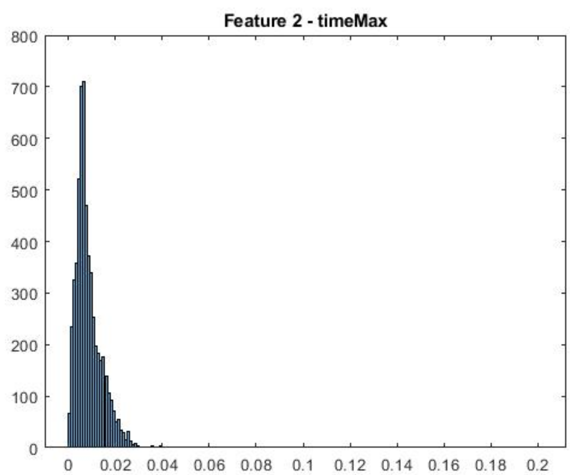


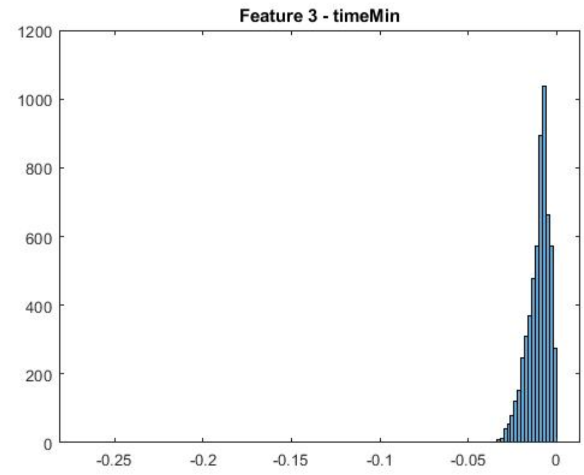
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	-0.00036	0.016247	-0.01906	-4.5E-06	0.618557	263.4676	1117	1054	1.251585	8.097045	1600	105.9347	588.2637	1166.392	0.224411	-7.17662	6.775515	-3.37723	0.92755	-0.36734	0.457
2	2.91E-05	0.007718	-0.00767	5.12E-06	0.639175	164.185	1295	1500	0.322126	4.662611	1600	174.7134	590.2968	1178.143	-1.13281	-9.35228	6.490564	-3.1857	0.750137	-0.44814	0.641
3	-0.00019	0.013257	-0.0075	1.65E-05	0.350515	192.3054	7769	7778	1.766781	13.54551	1600	0	0	526.5558	0.569159	-6.95087	6.743487	-3.4712	1.361851	-1.05208	0.498
4	-0.00015	0.009685	-0.01602	-2.6E-05	0.391753	178.8067	679	672	1.858475	12.18232	1600	199.0928	591.5314	1161.908	0.619756	-6.06332	6.73426	-3.27597	0.754233	-0.43155	0.575
5	-0.00036	0.006357	-0.01177	-1.8E-06	0.412371	275.3093	1151	1215	0.452851	5.152203	1600	221.7003	597.7116	1270.769	-0.79219	-7.66081	6.062017	-3.07318	1.391819	-1.15721	0.502
6	-0.00015	0.004239	-0.00573	-2.8E-06	0.597938	315.4353	931	1070	0.136213	2.678284	1600	0	0	567.9457	-1.99353	-9.44266	5.762704	-2.76524	1.44752	-1.17059	0.67
7	9.24E-05	0.004714	-0.00301	5.89E-07	0.494845	293.84	869	859	0.127968	2.564911	1600	202.0086	584.104	1287.61	-2.05597	-9.41193	5.974646	-2.88327	1.312098	-1.06638	0.709
8	0.000287	0.022604	-0.01858	-6.7E-06	0.639175	183.7113	456	461	10.6036	31.86063	1600	280.6478	613.1162	1198.789	2.361193	-4.38084	6.600323	-4.03212	1.384827	-0.91135	0.819
9	-0.00066	0.023301	-0.01592	1.78E-05	0.329897	251.624	2129	2135	6.94226	23.9985	1600	253.8186	617.7591	1215.411	1.937627	-6.93559	6.725954	-3.74778	1.229647	-0.84248	0.578
10	-0.00025	0.017132	-0.01891	-6.8E-06	0.721649	201.7055	508	500	2.602687	12.84833	1600	199.661	612.1849	1129.081	0.956545	-5.9444	6.972602	-3.91364	1.358704	-0.5422	0.816
11	0.000203	0.018058	-0.02304	-1.2E-05	0.659794	158.1908	7687	7693	3.529392	14.88419	1600	225.13	626.1215	1195.927	1.261126	-4.94644	6.608209	-3.70176	1.167491	-0.89691	0.741
12	2.7E-05	0.003866	-0.00272	-4.7E-06	0.536082	310.7698	7819	7826	0.107823	2.942967	1600	0	0	615.692	-2.22727	-9.4152	5.726722	-3.31106	1.088126	-0.38294	0.773
13	1.37E-05	0.002346	-0.00209	-1.2E-06	0.371134	337.0408	22	293	0.05012	1.990145	1600	151.2761	558.148	1289.394	-2.99333	-10.9902	6.124665	-2.58672	0.756941	-0.76368	0.482
14	-0.00031	0.003268	-0.00867	-7.7E-06	0.515464	312.3473	6385	6406	0.331816	5.219944	1600	145.8798	570.1011	1383.03	-1.10318	-10.0049	6.514975	-2.90368	0.931859	-1.139	0.948
15	1.91E-05	0.002987	-0.0026	-5.5E-07	0.412371	318.3957	1414	1404	0.033168	1.359459	1600	164.8381	588.9608	1203.99	-3.40618	-12.5246	6.864479	-2.90864	0.786384	-0.45283	0.535
16	-0.00064	0.006892	-0.00952	1.36E-06	0.371134	247.8516	7186	7303	0.695952	8.264584	1600	205.6245	607.5786	1207.912	-0.36247	-7.87937	6.357889	-3.3752	0.802012	-0.61631	0.65
17	-7.7E-05	0.013708	-0.00821	-6.5E-06	0.927835	146.935	7889	7945	0.909566	4.999039	1600	163.5541	640.3891	1337.897	-0.09479	-7.00609	6.204455	-2.57746	1.839945	-0.72177	0.984
18	-0.00216	0.006394	-0.02569	-1.3E-06	0.536082	157.2852	24	17	2.597827	12.1156	1600	151.9215	573.5423	1218.012	0.954675	-5.31921	6.623158	-3.28298	1.375288	-0.80807	0.417
19	-0.00018	0.014543	-0.02006	0.000134	0.57732	99.27935	3759	3969	3.317153	11.63115	1600	0	505.1296	1249.203	1.199107	-9.66709	4.540517	-1.16807	0.205416	-0.22043	0.022
20	0.000813	0.018082	-0.011	-1E-05	0.536082	233.9378	3100	3010	1.808216	8.130915	1600	148.341	554.4363	1270.634	0.592341	-9.83373	6.38116	-2.37489	1.280704	-0.68387	0.254
21	-0.00045	0.007152	-0.01185	-6.9E-06	0.618557	274.1751	2521	2589	0.922178	7.086817	1600	190.5681	593.0317	1335.65	-0.08102	-10.1162	6.849737	-2.93542	1.103688	-0.95396	1.124
22	-0.00069	0.009991	-0.01645	-1.8E-06	0.494845	266.1173	5384	5364	1.81936	11.84486	1600	200.7359	584.3025	1392.21	0.598485	-8.21016	6.277752	-2.85142	1.070345	-1.17171	0.900
23	-0.00094	0.020841	-0.02545	1.24E-05	0.371134	285.4264	4772	4684	2.698339	12.13843	1600	0	541.8058	901.8973	0.992637	-10.345	5.660499	-2.13119	0.679461	-0.93111	0.978
24	-0.00018	0.009277	-0.00496	-2E-05	0.43299	362.6478	3673	3517	0.145138	1.306943	1600	0	212.4611	1245.507	-1.93007	-14.6337	4.718394	-1.09985	1.674284	-0.33305	0.247
25	-0.00014	0.012605	-0.01649	-2.3E-05	0.927835	306.3319	3084	3073	1.720814	10.06393	1600	0	0	519.3038	0.542797	-10.049	6.261192	-2.19565	0.232713	-0.38635	0.296
26	6.46E-05	0.021802	-0.02658	-8.4E-06	0.886598	274.3027	3244	3304	4.998521	17.58088	1600	0	0	567.1148	1.609142	-9.44047	6.431534	-2.37519	0.66449	0.087323	0.589

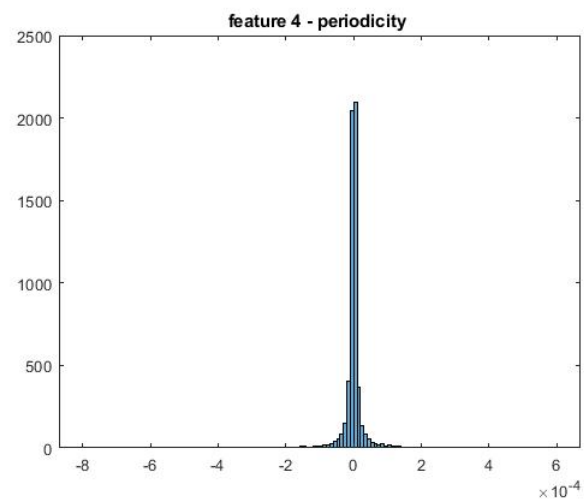
10.3 Feature Visualisation

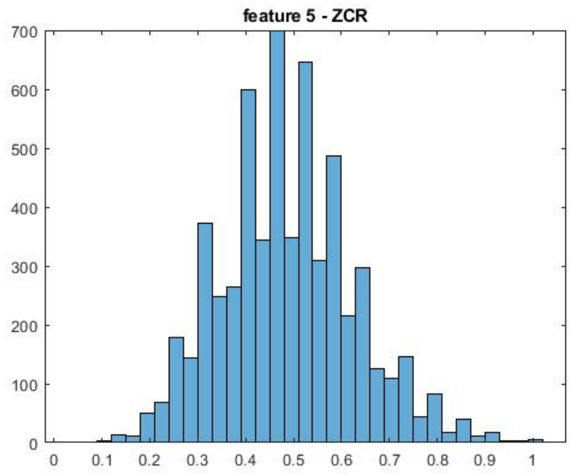
The next step is to visualise the features extracted to unearth some hidden patterns. These patterns give us an idea of what exactly where are dealing with. The following figures reveal exaclty this and give us some important insights into the data.

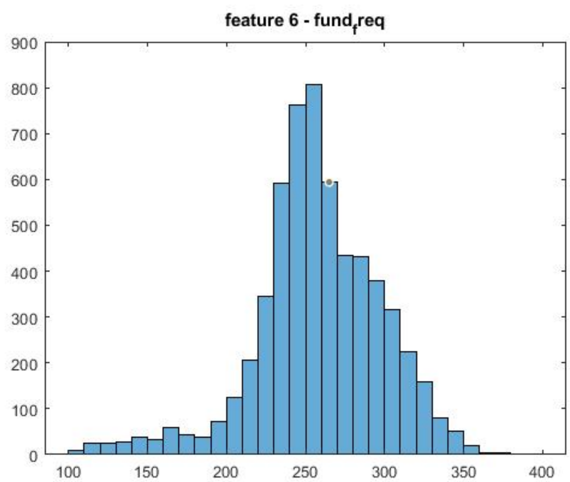

Feature 1 - AVG
800
700
600
500
400
300
200
100
0
-3
-2
-1
0
1
2
3
4
× 10^-3

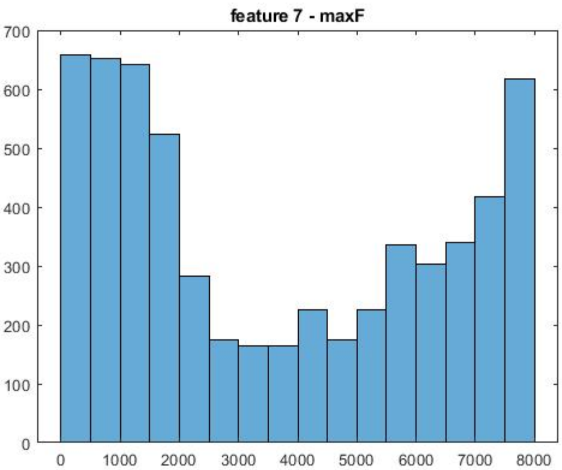

Feature 2 - timeMax
800
700
600
500
400
300
200
100
0
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2

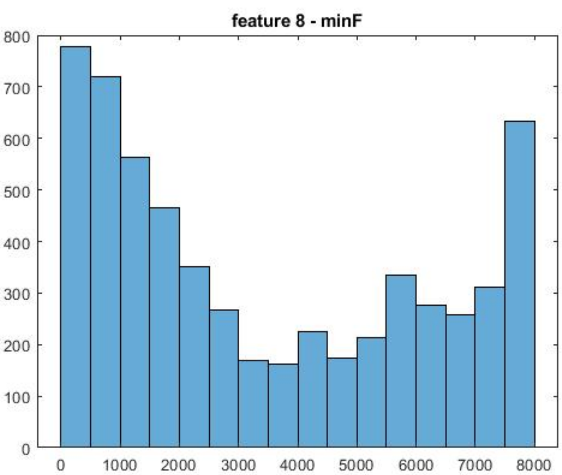

Feature 3 - timeMin
1200
1000
800
600
400
200
0
-0.25
-0.2
-0.15
-0.1
-0.05
0

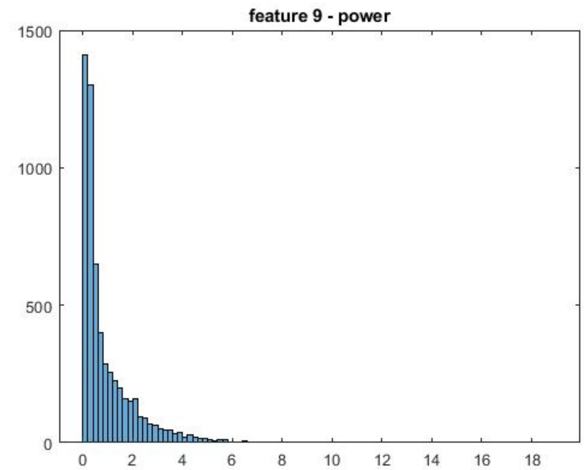

feature 4 - periodicity
2500
2000
1500
1000
500
0
-8
-6
-4
-2
0
2
4
6
$\times 10^{-4}$

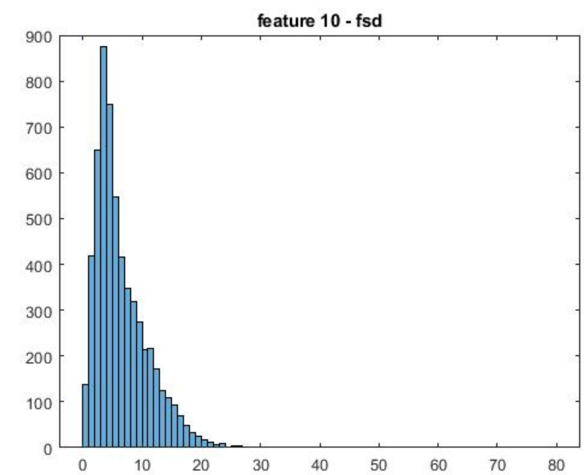

feature 5 - ZCR
700
600
500
400
300
200
100
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

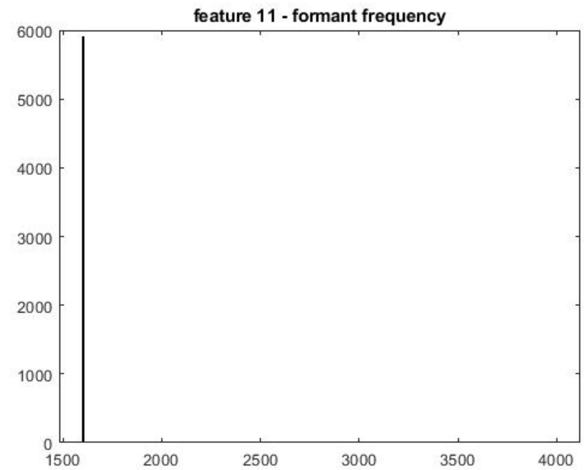

feature 6 - fund$_f$req
900
800
700
600
500
400
300
200
100
0
100 150 200 250 300 350 400

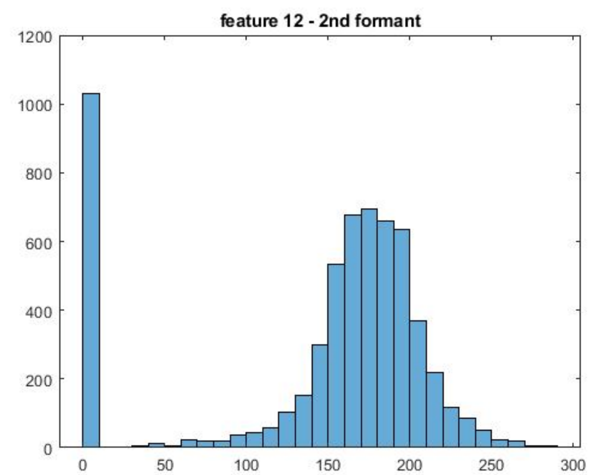

feature 7 - maxF
0
100
200
300
400
500
600
700
0
1000
2000
3000
4000
5000
6000
7000
8000

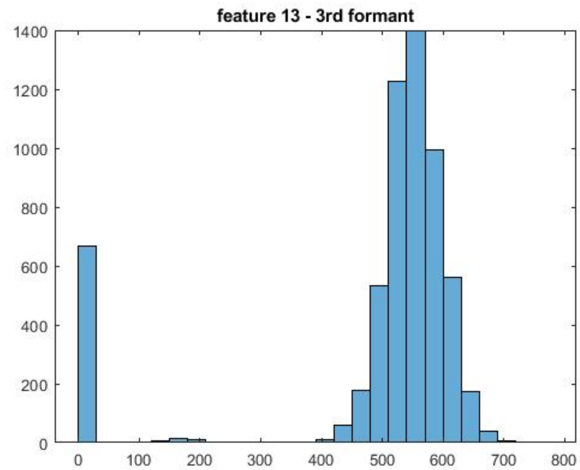

feature 8 - minF
0
100
200
300
400
500
600
700
800
0
1000
2000
3000
4000
5000
6000
7000
8000

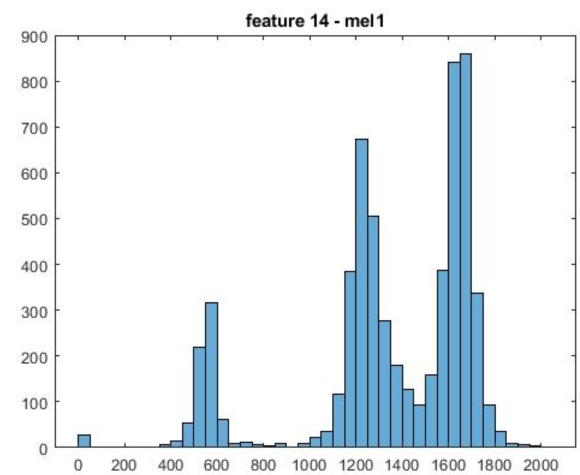

feature 9 - power
1500
1000
500
0
0 2 4 6 8 10 12 14 16 18

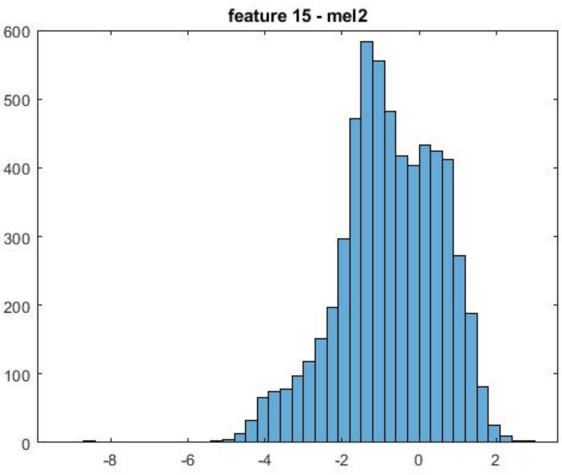

feature 10 - fsd
900
800
700
600
500
400
300
200
100
0
0 10 20 30 40 50 60 70 80

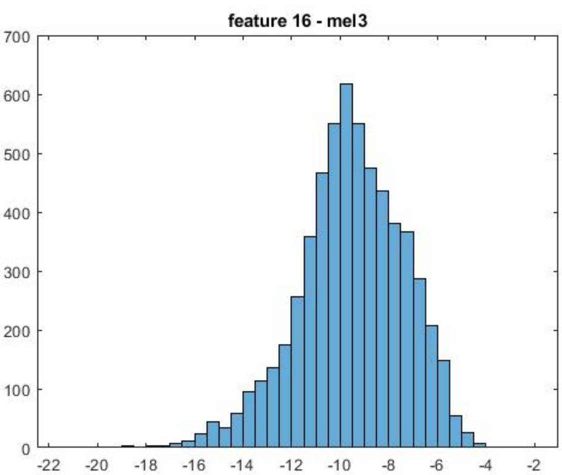

feature 11 - formant frequency
6000
5000
4000
3000
2000
1000
0
1500
2000
2500
3000
3500
4000

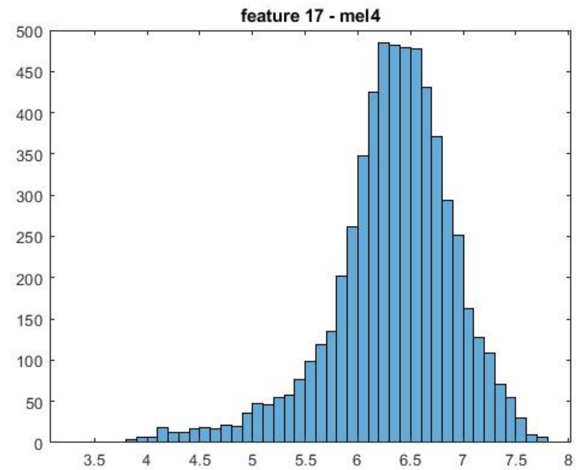

feature 12 - 2nd formant
1200
1000
800
600
400
200
0
0
50
100
150
200
250
300

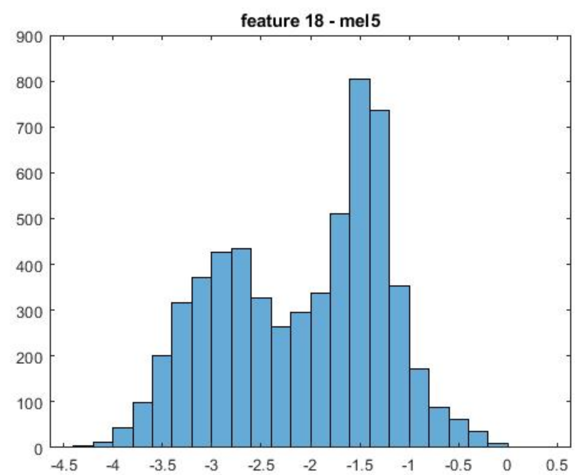

feature 13 - 3rd formant
1400
1200
1000
800
600
400
200
0
0 100 200 300 400 500 600 700 800

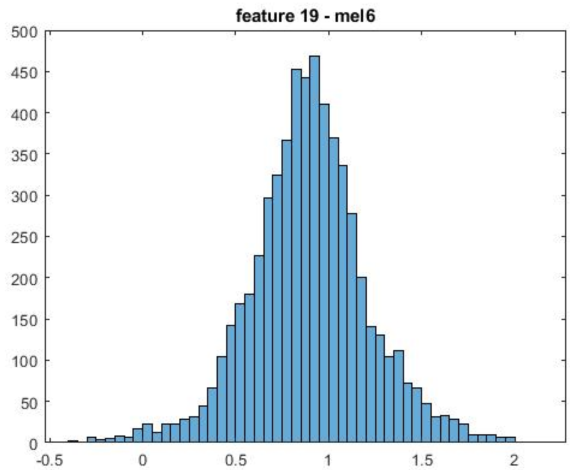

feature 14 - mel1
900
800
700
600
500
400
300
200
100
0
0 200 400 600 800 1000 1200 1400 1600 1800 2000

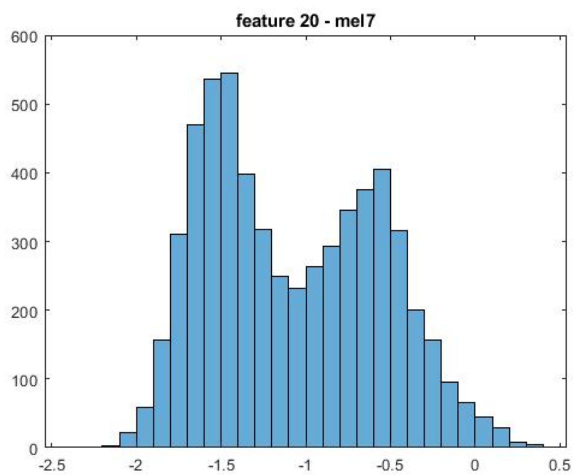

feature 15 - mel2
600
500
400
300
200
100
0
-8
-6
-4
-2
0
2

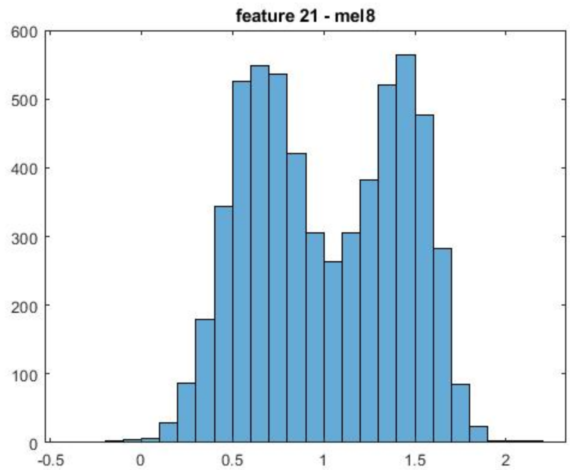

feature 16 - mel3
700
600
500
400
300
200
100
0
-22
-20
-18
-16
-14
-12
-10
-8
-6
-4
-2

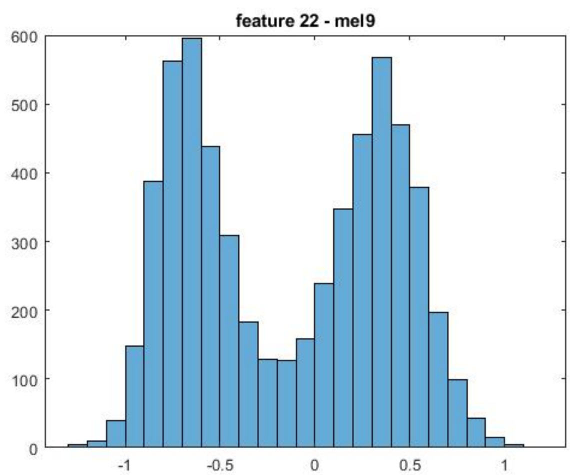

feature 17 - mel4
0
50
100
150
200
250
300
350
400
450
500
3.5
4
4.5
5
5.5
6
6.5
7
7.5
8

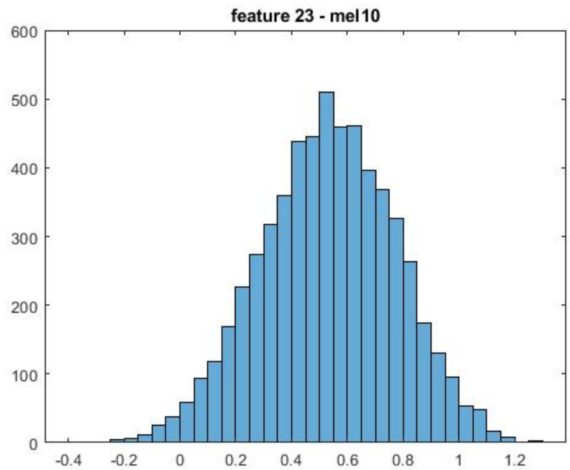

feature 18 - mel5
0
100
200
300
400
500
600
700
800
900
-4.5
-4
-3.5
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5

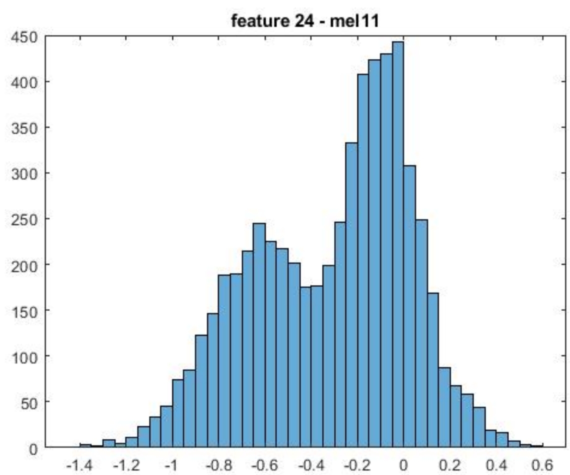

feature 19 - mel6

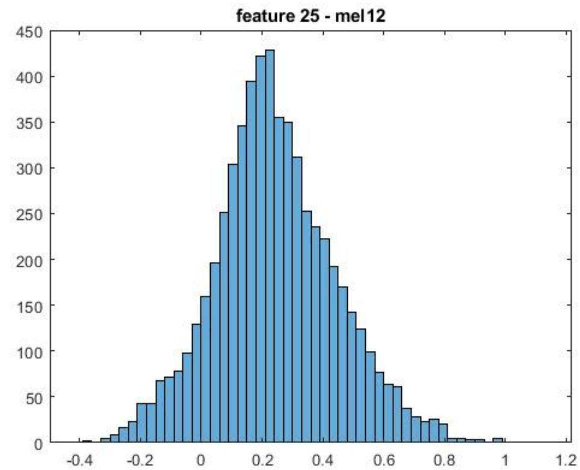

feature 20 - mel7

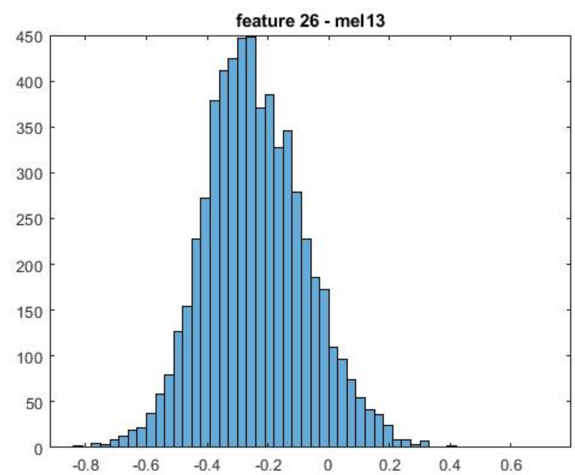

feature 21 - mel8
600
500
400
300
200
100
0
-0.5
0
0.5
1
1.5
2

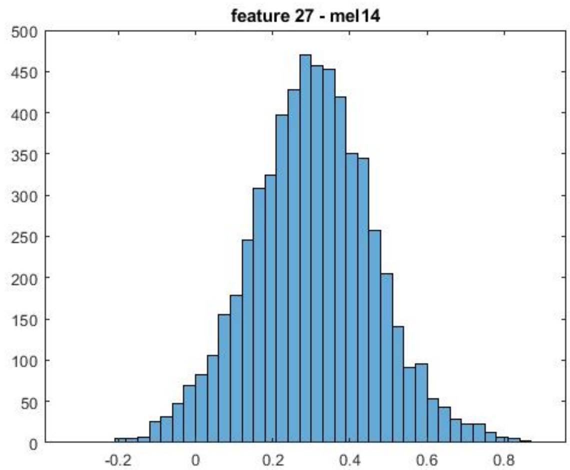

feature 22 - mel9
600
500
400
300
200
100
0
-1
-0.5
0
0.5
1

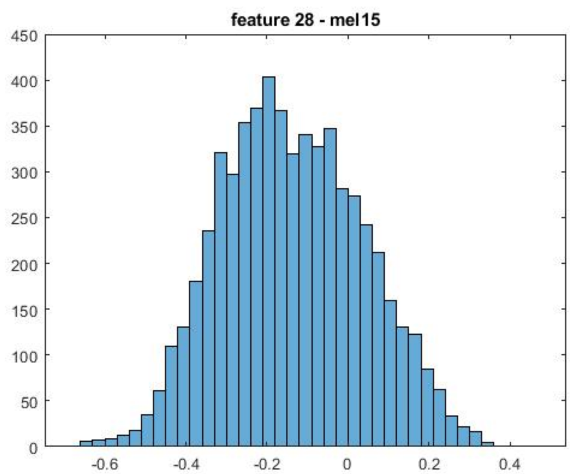

feature 23 - mel10
600
500
400
300
200
100
0
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
1.2


feature 24 - mel11
450
400
350
300
250
200
150
100
50
0
-1.4
-1.2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6


feature 25 - mel12
450
400
350
300
250
200
150
100
50
0
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
1.2


feature 26 - mel13
450
400
350
300
250
200
150
100
50
0
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6





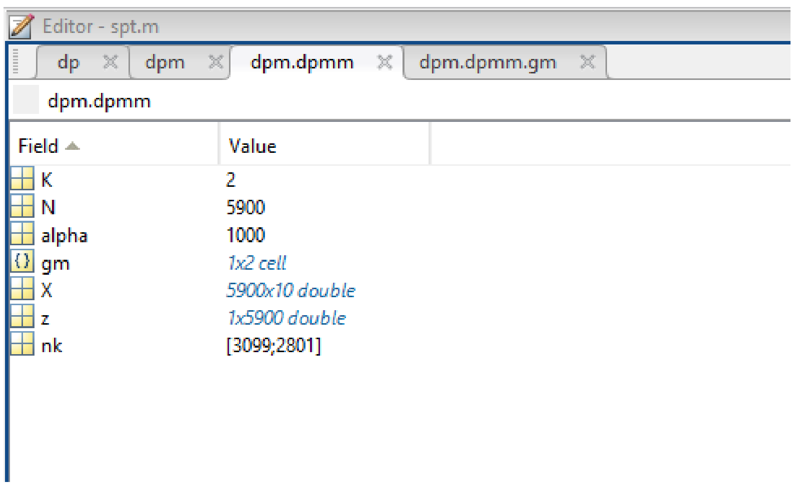
Important Observations

- The feature 11 (formant frequency 1) is one single frequency i.e., it is the same for both words.
- Almost all features follow a gaussian distribution with some and variance.
- Some features are a mixture of two gaussians. This can be observed in mel frequency coefficient features, namely, features, 14, 20, 21, 22 and 24.

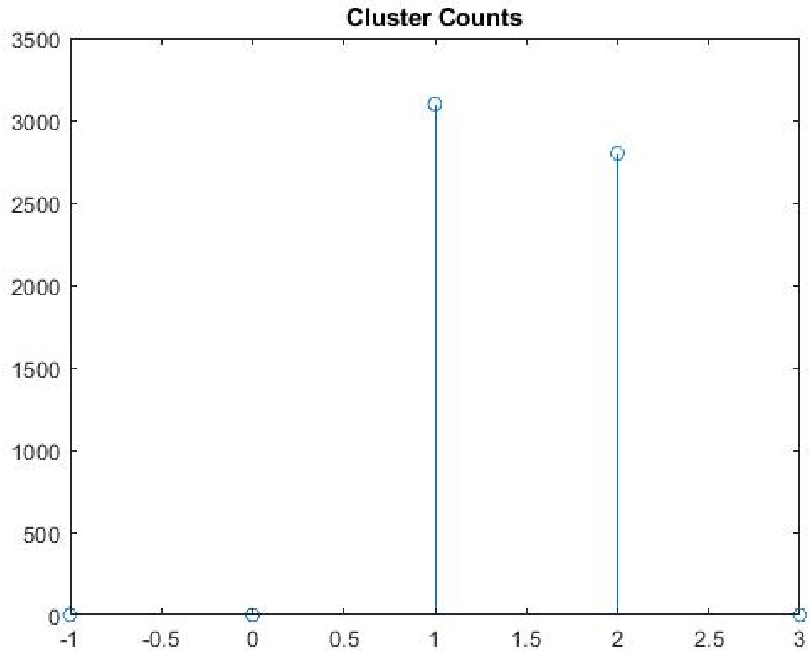
10.4 DP clustering

Before we go the results of DP Clustering, we note and use the observations made in this section. Some features were observed to have a mixture of two gaussians, prompting us to expect the number of clusters in DP clustering to be 2. Another important consideration is the prior distribution, G_0. In this case, it is easy and intuitive to take the gaussian as G_0.

We then cluster these 28-dimensional points using the dirichlet process clustering. For processing simplicity, we transformed the features to 10 dimensional vectors using the concepts of sparse filtering discussed above. The first figure shows the metrics of the built model. As you can see, it has two clusters. The variable z is list of labels assigned to each point.



The second figure shows the cluster count. Cluster 1 has been assigned 3099 points from the dataset and the remaining points were put into cluster 2.



The next four figures show the mean and covariance matrix for the clusters obtained. Mean is denoted mu and is highlighted whereas the covariance matrix is denoted as sigma. These values will be used for further for predicting the cluster of new points.

dp | dpm | dpm.gm | dpm.gm{1, 1} | dpm.gm{1, 1}.Sigma

dpm.gm{1, 1}.Sigma

	1	2	3	4	5	6	7	8	9	10
1	14.3681	6.1590	2.8025	2.6698	2.6687	12.8329	2.6529	2.6648	2.6867	1.7935
2	0	31.3585	4.9945	4.7900	4.7996	0.3027	4.8200	4.8098	4.8135	3.7067
3	0	0	14.0276	12.4923	12.5241	0.2115	12.4827	12.5324	12.4859	4.1787
4	0	0	0	6.5692	3.3512	0.0235	3.3444	3.3236	3.2923	1.0980
5	0	0	0	0	5.9335	0.0096	2.1786	2.1318	2.1661	0.3459
6	0	0	0	0	0	6.4460	0.0019	0.0048	0.0112	0.0896
7	0	0	0	0	0	0	5.4818	1.4316	1.4366	0.2259
8	0	0	0	0	0	0	0	5.2621	1.0741	0.1392
9	0	0	0	0	0	0	0	0	5.1736	0.1469
10	0	0	0	0	0	0	0	0	0	21.4526
11										

dp | dpm | dpm.gm | dpm.gm{1, 2}

dpm.gm{1, 2}

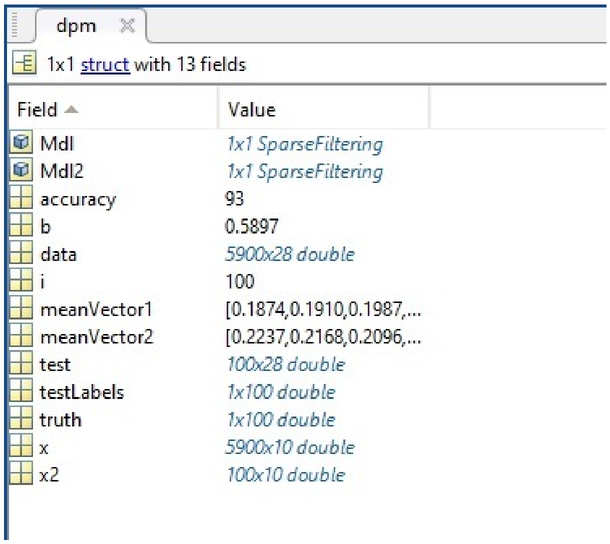
Field	Value
d	10
n	2801
rr	2.8011e+03
nu0	2812
Sigma	*10x10 double*
mu	[165.5274;1.4071e+03;655.0822;645.9489;648.9726;166.3715;647.5308;648.3494;648.2369;317.9733]
Z0	-145.5818

dp | dpm | dpm.gm | dpm.gm{1, 2} | dpm.gm{1, 2}.Sigma

dpm.gm{1, 2}.Sigma

	1	2	3	4	5	6	7	8	9	10
1	8.1410	10.1634	3.8579	3.6359	3.6009	5.4462	3.5922	3.6267	3.6115	1.4382
2	0	33.7398	4.5054	4.2191	4.1988	0.8227	4.2058	4.2014	4.2251	2.6956
3	0	0	14.3274	12.7921	12.8303	0.4864	12.7873	12.8338	12.8212	2.0582
4	0	0	0	6.5758	3.3624	0.0539	3.3558	3.3320	3.3072	0.6710
5	0	0	0	0	5.8970	0.0099	2.1408	2.1037	2.0921	0.4746
6	0	0	0	0	0	5.9884	0.0049	0.0186	0.0095	0.1153
7	0	0	0	0	0	0	5.4776	1.4271	1.4140	0.2832
8	0	0	0	0	0	0	0	5.2612	1.0672	0.1660
9	0	0	0	0	0	0	0	0	5.1522	0.1853
10	0	0	0	0	0	0	0	0	0	13.8397

Finally, we need to validate our model by testing it against some known chunks. For this we took 100 labelled chunks and tested our model. As the figure below shows, we achieved a 93 percent accuracy.



10.5 Application - Speech to Text

We built a model and tested it. Now, it is time to apply this model to convert detected speech to text. There is another challenge to this problem apart from detecting the word utterance. We also need to find where exactly this word was spoken. For this we propose two ways.

- We can exploit the assumption made that we have a word every second. We can therefore process the speech signal every second i.e., slotwise.
- Or we can use a sliding window to detect the exact starting and ending point of each utterance.

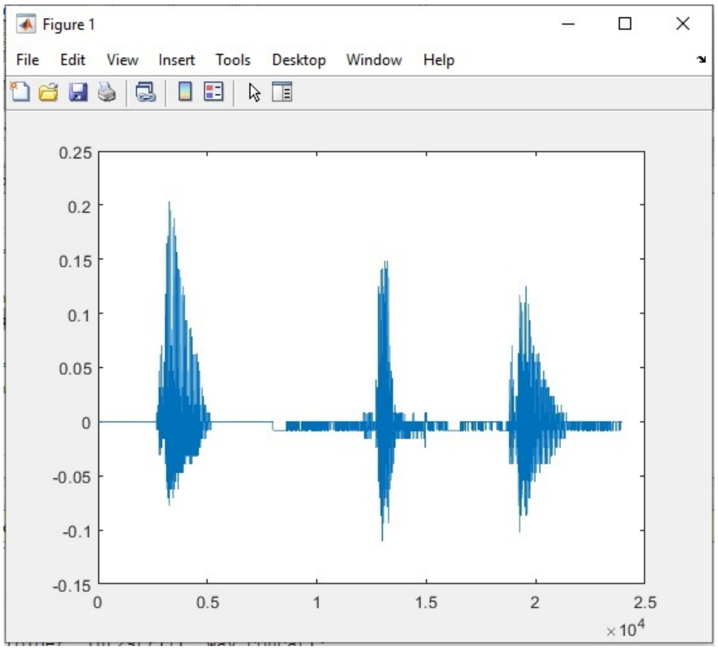
Another important output of this application is to count the number of utterances of each word in the vocabulary. Such an analysis can be used for various interesting purposes, the most interesting being context extraction.

10.5.1 Slotted Speech

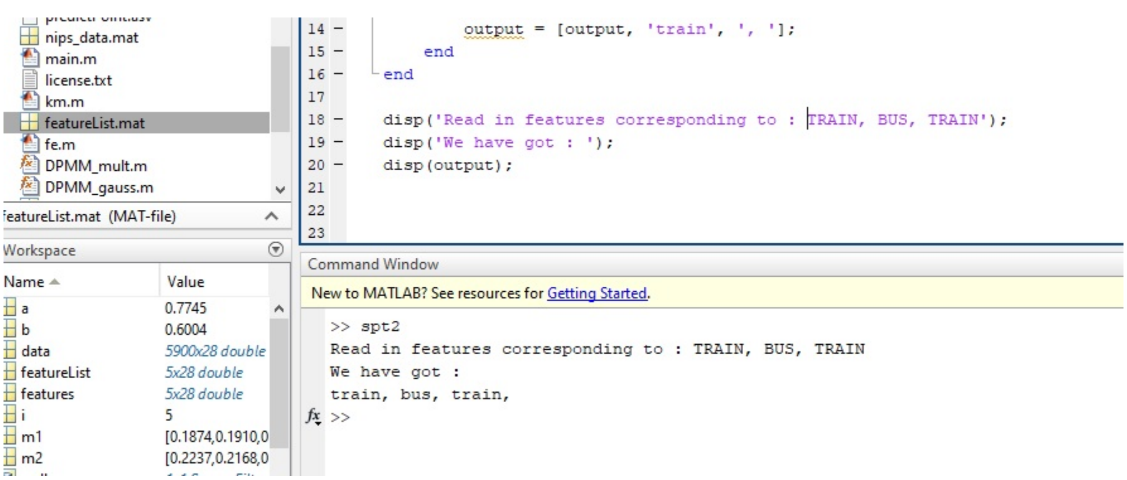
The ideas discussed above are formally put down as follows:

- Use a window of 1 second.
- Extract features.
- Predict the cluster.

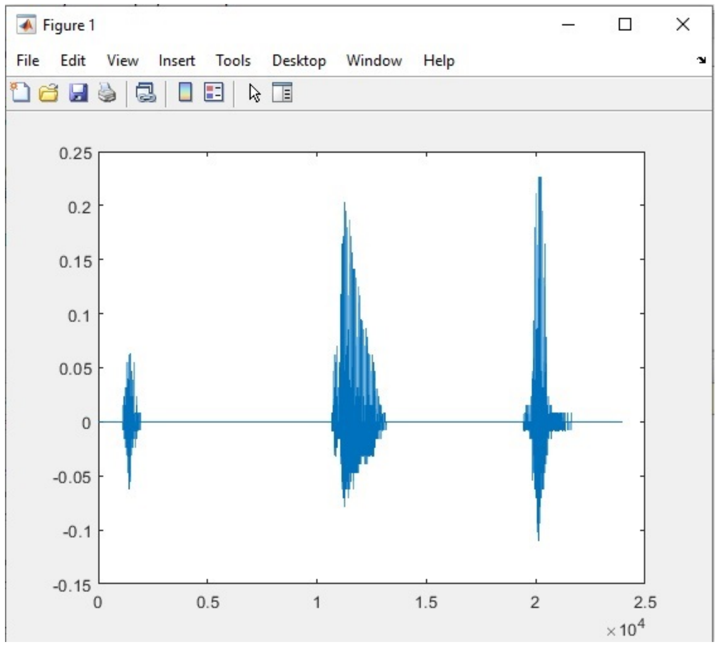
This is a simple approach given our crucial assumption that only one word is spoken in a second. We present the results for this algorithm. Consider a speech signal as shown below, for which we know what the output needs to be.



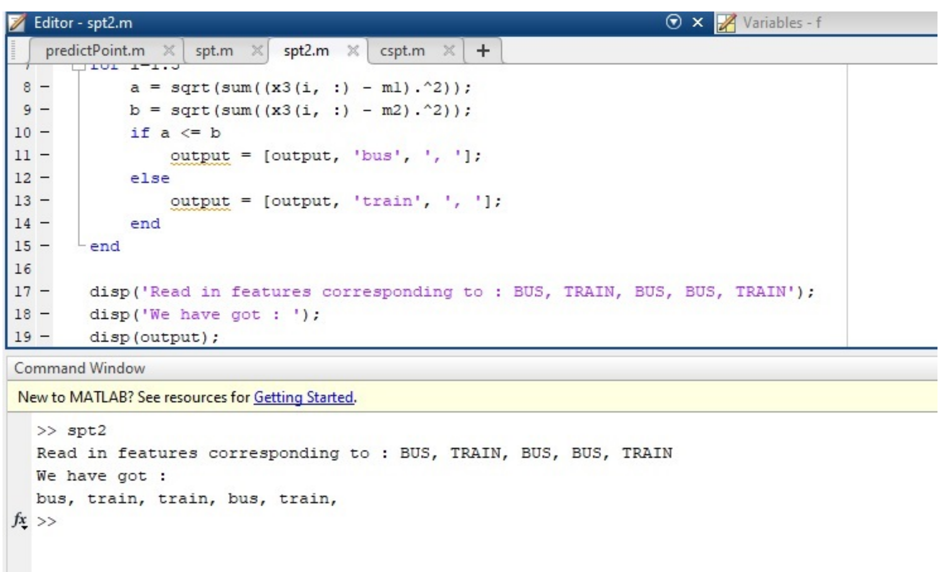
The speech was : Train Bus Train
Output obtained was : Train Bus Train
The screenshot is attached below.



We now consider another speech and repeat the process.



The speech was : Bus Train Bus Bus Train
Output obtained was : Bus Train Train Bus Train
The screenshot is attached below. As you can see the third word was incorrectly clustered.

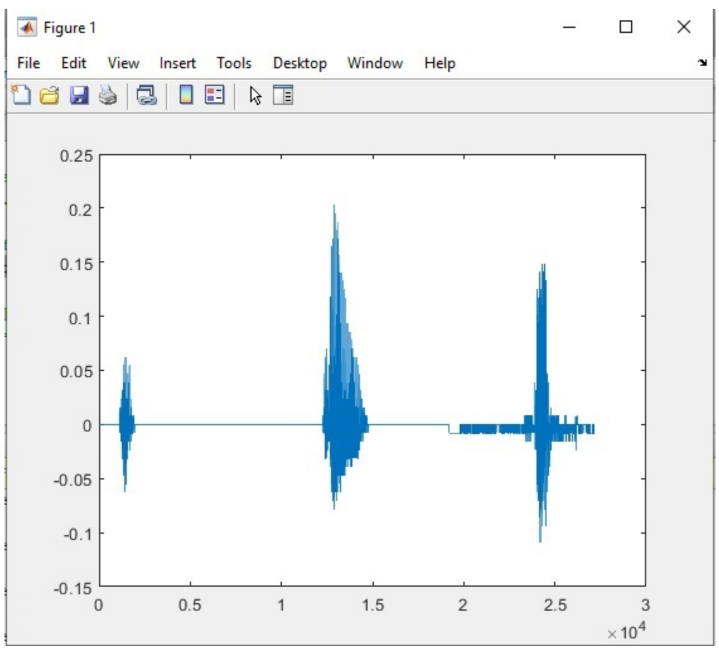


10.5.2 Continuous Speech

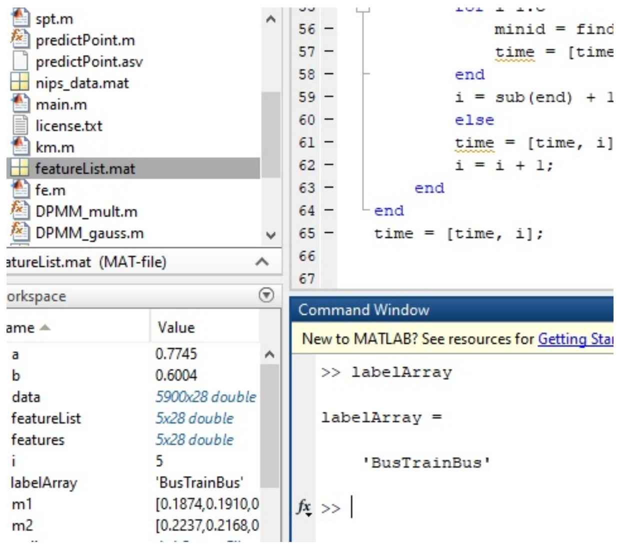
In this section, we discuss an approach to detect the locations of speech utterances using a sliding window.

- Take a window of size 1 second.
- Extract features and cluster it using the built model.
- Slide the window to a new position with overlap, say by 6000 samples.
- Repeat the process.

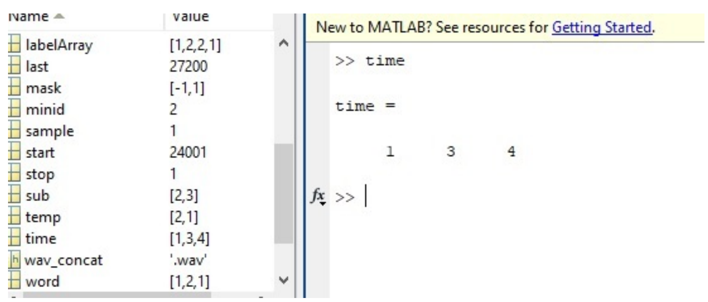
The above process will give multiple detections out of which we choose the one that corresponds to least distance from the corresponding cluster. Consider a speech signal shown below. The words spoken were Bus Train Bus.



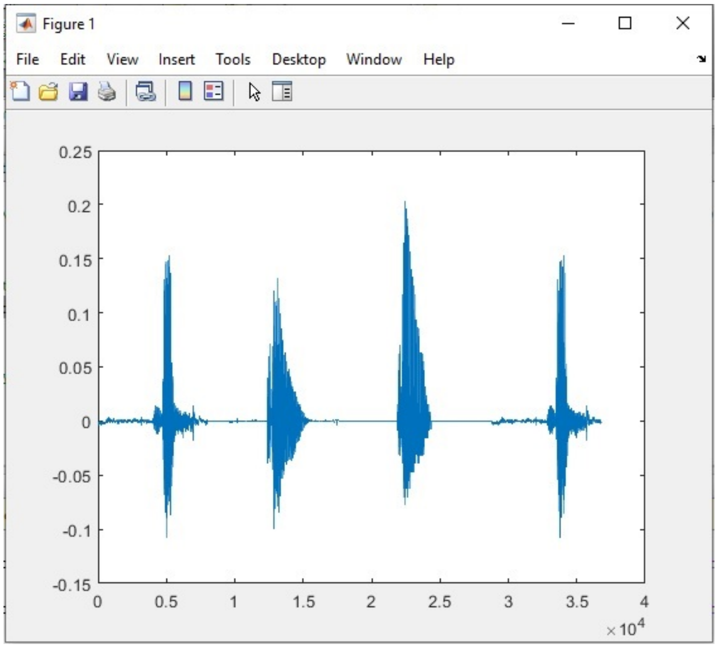
The words were detected successfully as shown in the figure below.



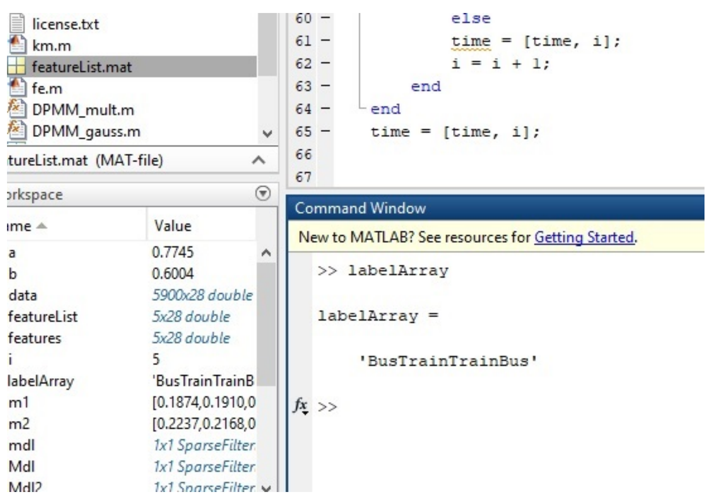
We were successfully able to locate the word utterances apart from counting the number of utterances. The screenshot below shows the time it was detected in terms of window number.



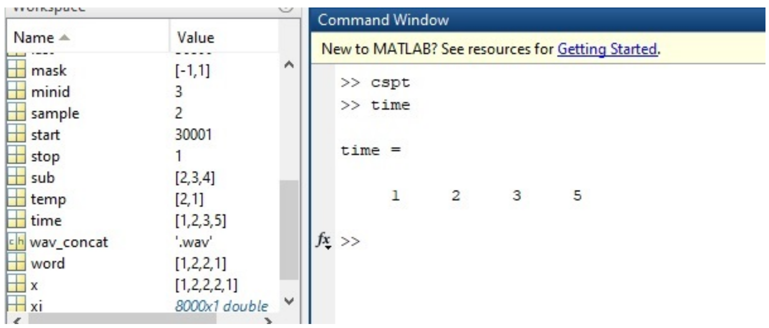
Since, we know the sampling rate and stride length(=6000 samples i.e., 6/8 seconds), we can find the start and end locations easily. Similarly, consider another speech wave with four word utterances i.e., Bus Train Train Bus.



The wave was successfully decoded as expected and shown in the figure below.



Again, the locations of the word utterances are shown below as window indices.



With this we end our discussion and conclude our findings.

11 Conclusions

- For this project, we first captured raw data sampled at 8000 hz, 8-bit, mono.
- From the raw data, we extracted features from three domains.
- From the extracted features we have built a dataset.
- By splitting the dataset into training and testing, we clustered the points using Dirichlet Process Clustering.
- We evaluated the model by comparing assignments to the known truth.
- We then applied the model to different speech waveforms to convert it to text by slotting the speech and considering it as a continuous signal, which it is.

12 Future Work

- We plan to take this project further by expanding the vocabulary and removing all constraints. We also plan to implement different speech codecs for the chunks and train a neural network to classify word utterances.

References

[1] Theory and Applications of Digital Speech Processing - Lawrence Rabiner and Ronald Schafer

[2] Cepstral Analysis
http://research.cs.tamu.edu/prism/lectures/sp/l9.pdf

[3] Chinese Restaurant Process
https://www.cs.princeton.edu/courses/archive/fall07/cos597C/scribe/20070921.pdf

[4] One more CRP Source
https://www.statisticshowto.com/chinese-restaurant-process/

[5] Dirichlet Distribution
https://towardsdatascience.com/dirichlet-distribution-a82ab942a879

[6] DPC-1
https://www.mathworks.com/matlabcentral/fileexchange/62202-dirichlet-process-mixture-model-dpmm

[7] DPC-2
https://datamicroscopes.github.io/ncluster.html

[8] DPC-3
https://www.cs.cmu.edu/~kbe/dp_tutorial.pdf

[9] Speech and Audio Lecture
http://kom.aau.dk/~zt/courses/Feature_extraction/Extraction%20of%20features,%20MM5.pdf

[10] Our Motivation
https://www.sigdial.org/files/workshops/workshop10/proc/pdf/SIGDIAL51.pdf