Chords Classification using GMMs

Project report submitted in partial fulfillment
of the requirements for the degree of

Bachelor of Technology
in
Computer Science and Engineering

by

Namish Narayan: 15UCS080
Yash Nag: 15UCS173

Under Guidance of
Mr. Kshitiz Verma

Department of Computer Science and Engineering
The LNM Institute of Information Technology, Jaipur

December 2018

Copyright © The LNMIIT 2018
All Rights Reserved

The LNM Institute of Information Technology
Jaipur, India

CERTIFICATE

This is to certify that the project entitled Chords Classification using GMMs , submitted by Yash Nag (15UCS173) and Namish Narayan (15UCS080) in partial fulfillment of the requirement of degree in Bachelor of Technology (B. Tech), is a bonafide record of work carried out by them at the Department of Computer Science and Engineering, The LNM Institute of Information Technology, Jaipur, (Rajasthan) India, during the academic session 2018-2019 under my supervision and guidance and the same has not been submitted elsewhere for award of any other degree. In my/our opinion, this thesis is of standard required for the award of the degree of Bachelor of Technology (B. Tech).

Date

Kshitiz Verma

Dedicated to our amazing faculty and helpful peers

Acknowledgments

We would like to express our special thanks of gratitude to our mentor faculty, Kshitiz Verma who gave us a chance to perpetrate this project under his tutelage on this growing field of Machine Learning, we thank him for his constant support, clear answers and helping us improve this project along the way. It would also be unjust to not thank Andrew Ng numerous other creators and tutors on their amazing video or text based lectures and in-depth illustrations of Machine Learning, without which, it would've been impossible to accomplish this project. We would also like to thank our seniors and peers at LNMIIT, who helped us correct every silly bug and made us easily understand those notorious computations with the silliest analogies. This project has inspired us to commit to more such projects in the future in this field of Science and we're more than determined to continue along this path, thanks to all of our supporters and well wishers.

Abstract

Identifying chords from the audio of a music track has always been a challenge for traditional programs as the variation in the tones, background noise, pitch shift etc. makes it virtually impossible to rely comfortably on them to recognize the chords accurately from the audio. Due to the lack of reliable options, musicians still rely on thier '*trained ear*' to identify chords instead.

To remedy that, In this paper, we introduce Gaussian Mixture Models (GMMs) that are specific to a chord, trained with respect to it, and which performs admirably in our training and testing with a self-synthesized dataset of chords audio. This architecture is based to one similar to Speaker Recognition in Audio Classification, and focuses learning from the MFCC and delta MFCC values from the sound. The separate GMMs with Expectation Maximization (EM) algorithm calculates probability logs for each test case, and generates predicted class (chord) with an accuracy of 100% in our testing.

Contents

Chapter *1*

Introduction

1.1 Problem Overview

As fellow musicians, we've more than frequently encountered the problem to figure out chords of a song just by the audio of the song. Sometimes the distortion, tremolo effect or numerous other effects limit the human ability too figure out appropriate chords for a section of the song. As one of the most important mid-level features of music, chord contains rich information of harmonic structure that is useful for music reproduction and modification.

In this paper, we propose Chords Classification using separate Gaussian Mixture Models for each chord. We generated the audio data of Chords by ourselves with a standard acoustic guitar using default strumming and extracted each chord's audio MFCC values as features to train our model.

1.2 The Area of Work and Limitations

Our area of work will be mainly focused on extracting features from the separate audio files per chord and achieving a reasonable precision upon classification of the same. Our system will be limited to classifying 5 chords, played on a single instrument i.e. Acoustic Guitar, in their major variation. Note that our proposed model can be extrapolated to conform to all other chords and their variations with more audio data.

1.3 Existing Solutions

Following sub-sections give brief overview of current systems to tackle Chords Classification.

1.3.1 Machine Learning Systems

In Osmalskyj, Jacques et al paper on chords classification [3] they approached the challenging problem of music recognition with an effective machine learning based method using a feed-forward neural network for chord recognition. Their method incorporated the us of known feature vector for automatic

chord recognition called the Pitch Class Profile (PCP). In another paper by Henaff [2], they used Support Vector Machine, simple linear classifier combined with a fast feature extraction system allows our approach to scale well to large datasets which gave a GITZAN Dataset accuracy of over 80%. All these systems achieve noteworthy precision in respect to classification.

1.3.2 Other Systems

In Matti P. Ryynnen and Anssi P. Klapuri 2008 article [6], they proposed a method for automatic transcription of melody, bassline and chords in polyphonic pop music using frame-wise pitch pitch-salience estimator as feature extraction, followed by acoustic modelling of note events. The chords transcription maps the pitch salience estimates to a pitch-class representation and uses trained chord models and chord-translation probabilities to produce a transcription of major and minor triads.

1.4 Our Approach

Traditionally, chord classification lacked a stable architecture and required quite a bit of human element to have a reasonable success rate. However, Machine learning based approaches like the use of Support Vector Machines and Neural Networks are quite successful in classification of chords as seen in multiple research papers, but they require large amount of data to adequately train the model to differentiate between chords. Also, such models lack the route to expandability into new 'custom' chords and new patterns which may form their own class. Deep learning methods lack suck dynamic model and hence we focused on approaching the problem as a clustering problem rather than a classification problem.

Due to lack of training, development and testing data (i.e. audio samples of chords), we created our own samples using an acoustic guitar and generic microphone to record the samples. We used MFCC and computed their differential values (deltas) combined as features to be trained. Our Gaussian Mixture Model uses the above mentioned features to cluster a model based on expectationmaximization (EM) algorithm to form a model for each of the classes (in this case, chords) and then tested each of the training samples to the chords models to compute a log score, by which we we can classify the test sample to one of the model (or one of the chord) to which the score favors the most. Our implementation of GMM is broadly based upon the *Speaker verification* by DA Reynolds [5], and how it can be implemented in the context of identfying chords.

Considering all the available approaches, we chose to go with this approach as it provides a more sustainable and expandable approach in tackling the problem and gave encouraging results in our testing.

Chapter *2*

Pre-processing

2.1 Introduction

In this section, we'll discuss the chord's audio sample data that we created, along with the methods and techniques we used to extract features from the audio files and preprocessing it. We used the Linux environment for the development of the system, along with python as a preferred language due to availability of multiple open source libraries and easy-to-read architecture. We also extensively used PyAudio and SciPy for recording and preprocessing audio samples.

2.2 About the Data

Based on our extensive search online for audio chords sample, we were unable to retrieve adequate data to train our GMM, hence we decided to create the audio samples of the chords by ourselves. Our implementation is limited to classifying 5 chords, ie. A Major, B Major, C Major, D Major and E Major as they provide a good balance in being distinguishable to the ears and few samples would be enough to train our model to classify these chords

To start off, we used an acoustic 6-String guitar tuned to standard tuning (i.e. tuned to the notes E, A, D, G, B and E) for recording of the samples. For each of these 5 chords, we used 3 different strumming patterns to provide appreciable variance in how the chord can be played in the guitar, along with 2 different positional variation of each pattern on each chord. This gave us a total of **5 x 3 x 2 = 30 chord samples** to learn from.

All these samples have a time duration of about 10 seconds, are recorded on mono channel with a rate of 16000 and with 512 frames per buffer. Also, all the samples of each chord are stored in separate directories with the name of the chord, basically to treat these chords as labelled data. Sample Link for the data we used can be found here.

2.3 Data Preprocessing

The unstructured audio samples data that we acquired is converted to a structured feature vector per chord to train it's respective GMM for classification. The following sub-sections dives a little deeper into methods we used to preprocess the data and finally create a feature vector for the same.

1. Audio Sample Processing

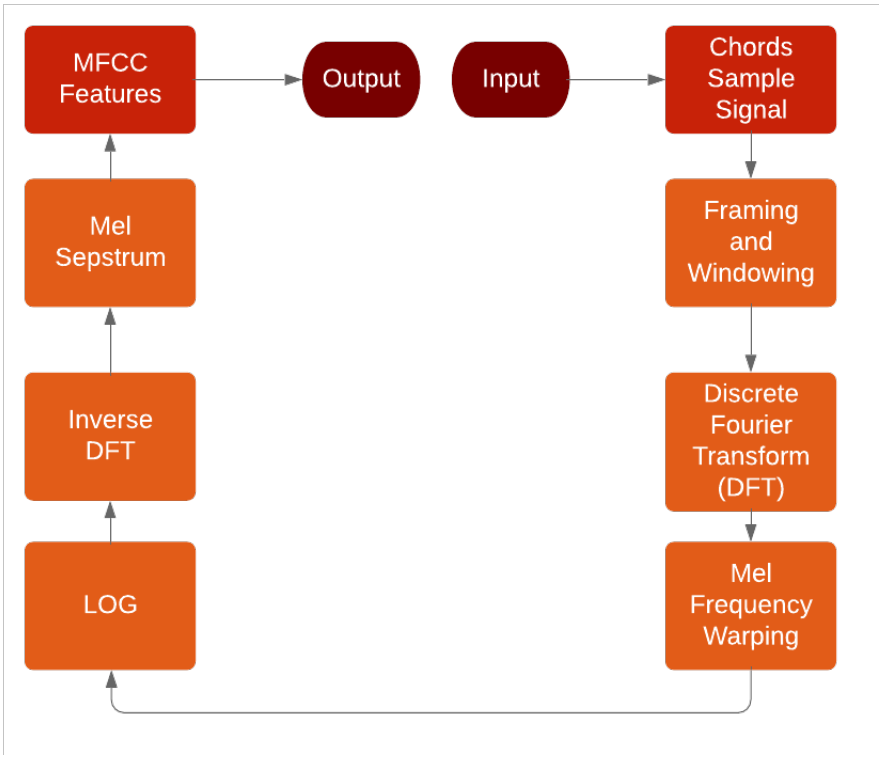
2. Extracting Features

2.3.1 Audio Sample Processing

The recorded audio sample had multiple blank phases. This may be due to the lag that was introduced during recording and playing of the instrument, along with time that elapsed before ending the recording. To form adequate feature vectors, we removed such lag by trimming any of the phases if it didn't meet the threshold (volume too low), as well as normalizing the entire data (averaging out the volume). This processed data is then forwarded to the next phase for feature extraction.

2.3.2 Extracting Features

We extract 40-dimensional features from each of the frames of each chord's sample. These features are derived from MFCC (Mel-frequency cepstral coefficients) and it's derivatives. Each frame consists of 20 MFCC features and 20 derivatives of MFCC features. Extraction of MFCC features is handled by an open source library, python-speech-features library.



After extraction of MFCC Features, we calculate the derivatives of MFCCs. Derivatives provides the information of dynamics of MFCCs over the time. To calculate delta features from MFCCs, we use the following equation upon the 20-Dimentional MFCC Features:

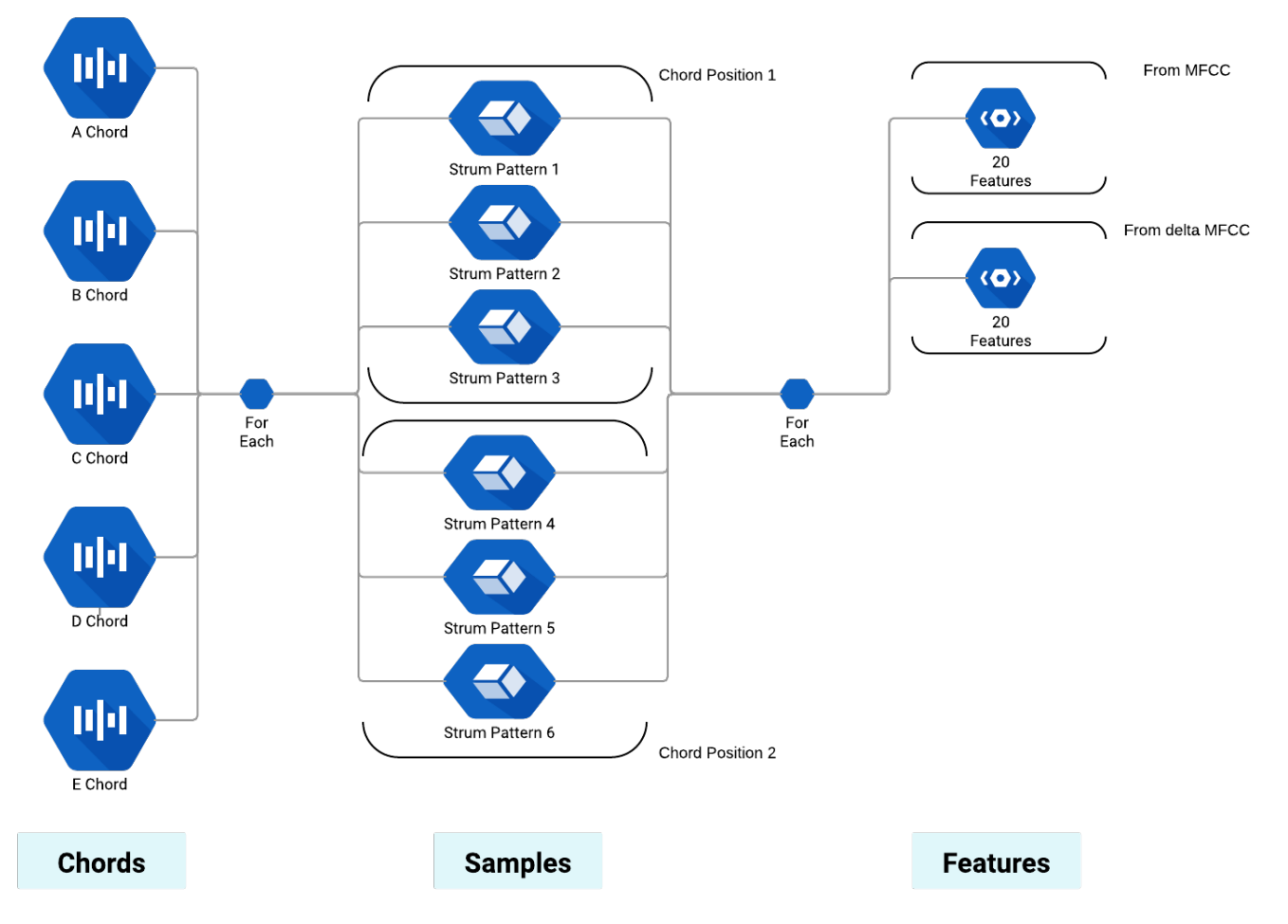
$$d_t = \frac{\sum_{n=1}^{N} n(c_{t+1} - c_{t-1})}{2\sum n^2}$$

Here, N=2

This operation gives us 20 more dimensions of the feature vector. It turns out that calculating the delta-MFCC and appending them to the original MFCC features increases the performance in analytic applications. From these successive operations, we get a feature vector of the 40 dimensions, 20 MFCC and 20 delta-MFCC features.

2.4 Exporting Features

Recalling the above procedures, for each of the 30 audio chord samples of different chords, we extracted 40 features from each frame of every sample. This gave us a reasonable data to perform machine learning techniques and apply our Gaussian Mixture Model to train on these features.



The following chapters will focus on these features feeding into the Gaussian Mixture Model for training and testing.

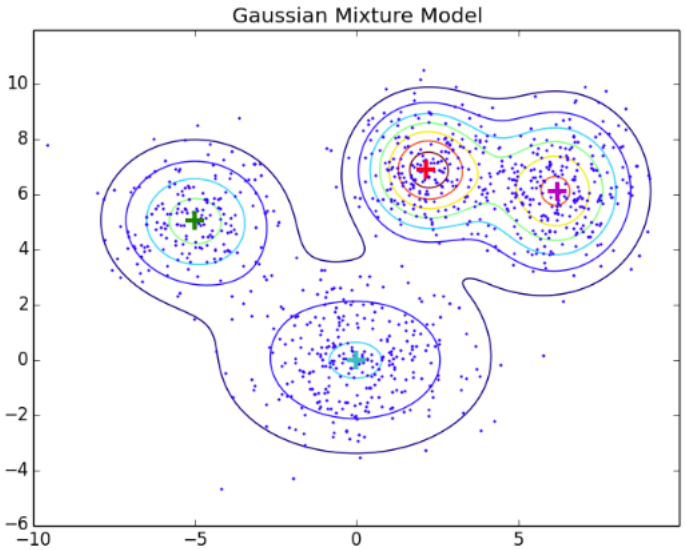
Chapter *3*

Gaussian Mixture Model

We used Gaussian Mixture Model (GMM) due to it's expandability and flexibility regarding fitting data points and it's accuracy over small datasets. GMM are faster than other machine learning alternatives like Neural Networks for learning mixture models and also are agnostic as this algorithm maximizes only the likelihood. To implement the GMMs, we used SciKit Learn's Gaussian Mixture Model [4].

3.1 About

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. A Gaussian Mixture Model is the weighted sum of different Gaussian distributions so we can represent the data as a single curve. Gaussian Mixture Model can be used for framing a given data into clusters. The data is assigned to a specific cluster/clusters on the basis of their probability of belonging to a Gaussian distribution.



Rather than assigning clusters of each class from the data, we instead created a new model to each class to train on data specific to that class (in this case, chord). Prompted by *Front-end factor analysis for speaker verification*[1], we used the GMM for classification by creating multiple GMM to be assigned to each class and later on testing, comparing their scores to assign the class of the test subject. This architecture gives more expandability in the future as new models can be made to incorporate different classes (i.e. chords) later.

3.2 Training

The idea of training a GMM is to approximate the probability distribution of a class by a linear combination of k Gaussian distributions/clusters, also called the components of the GMM. The likelihood of data points (feature vectors) for a model is given by following equation:

$$P(X|\lambda) = \sum_{k=1}^{K} w_k P_k(X|\mu_k, \Sigma_k)$$

, where $P_k(X|\mu_k, \Sigma_k)$ is the Gaussian distribution

$$P_k(X|\mu_k, \Sigma_k) = \frac{1}{\sqrt{2\pi|\Sigma_k|}} e^{\frac{1}{2}(X-\mu_k)^T \Sigma^{-1}(X-\mu_k)}$$

The training data X_i of the class λ are used to estimate the parameters mean μ, co-variance matrices Σ and weights w of these k components.

Initially, it identifies k clusters in the data by the K-means algorithm and assigns equal weight $w = \frac{1}{k}$ to each cluster. k Gaussian distributions are then fitted to these k clusters. The parameters μ , σ and w of all the clusters are updated in iterations until the converge. The most popularly used method for this estimation is the Expectation Maximization (EM) algorithm.

For each of the chords, we created a dedicated class and corresponding GMM. This is done by extracting all the audio files from the samples associated with each chord and feeding them to a single GMM. This GMM now fits and gives scores with respect to that particular chord. This method is repeated to generate 5 GMMs that map to 5 chords in our classification problem. A gist view can be seen below:

```
for this_class in class_labels:
    this_class_training_path = os.path.join(source_path, this_class)
    this_class_training_samples = [os.path.join(this_class_training_path, x) for
    features = np.asarray(())
    for index, sample_path in enumerate(this_class_training_samples):
# Read Audio and Rate
        sr, audio = read(os.path.join(source_path, sample_path))
```

```
# Extract Features (MFCC & delta MFCC)
        vector = extract_features(audio, sr)

        if features.size == 0:
            features = vector
        else:
         # Concatenate features from different samples
            features = np.vstack((features, vector))

        # Training Model on stacked features
        if (index+1 == len(this_class_training_samples)):
            gmm = GMM (n_components = 16, covariance_type='diag',n_init = 3)
            gmm.n_iter = 20
            gmm.fit(features)

            # Dumping Trained Model into Pickle
            picklefile = this_class + ".gmm"
            pickle.dump(gmm, open(os.path.join(models_path, picklefile),'wb'))
            features = np.asarray(())
```

As the code suggests, these models are trained with conventionally used parameters which are known to perform good in this scenerio. All these models are then dumped using Pickle library into a directory. It helps serialize all the data, and makes storage and accessing these models in python, easier.

3.3 Testing

For testing we used similar methodology like we used to train the model. Testing data consists of similar audio samples like the ones we synthesized for training. For each class (chord), we have 2 audio samples for training, which comprises of 2 different strum pattern on a unique chord position, separate from the ones used for training the chord model.

So for each of the 5 chords, we had 6 samples for training and 2 samples for testing the Gaussian Mixture Model. In total, we have **2 x 5 = 10 testing samples** to classify into one of the 5 classes. The testing samples follow the same preprocessing procedures like the training examples, discussed briefly in Chapter 2.

After extraction of the 40 MFCC and delta MFCC features, we require the log likelihood scores for each frame of the sample, $x_1, x_2, ..., x_i$, belonging to each class, i.e. $P(x_i|S_j)$ (for all j that belongs to S) is to be calculated. The likelihood of the frame being from a particular class is calculated by substituting the μ and Σ of that speaker GMM model in likelihood equation shown in previous sub-

section. This is done for each of the k Gaussian components in the model, and the weighted sum of the k likelihoods from the components is taken as per the weight w parameter of the model.

The logarithm operation when applied on the obtained sum gives us the log likelihood value for the frame. This is repeated for all the frames of the sample and the likelihoods of all the frames are added. The chord model with highest likelihood score is considered as the identified chord class for the sample.

For implementation, Pythons sklearn.mixture package provides with a score attribute for each model to retrieve log probability of the sample in the range $(-\infty, 0]$. The range of this value lies in with the value closer to 0, signifying a better constitution of that sample to the respective class. The following snippet in python gives a deeper isight about the process.

```
for this_class in class_labels:
    this_class_testing_path = os.path.join(source_path, this_class)
    this_class_testing_samples = [os.path.join(this_class_testing_path, x) for x

    for sample_path in this_class_testing_samples:
        # Read Audio and Rate
        sr, audio = read(os.path.join(source_path, sample_path))

        # Extract 40 Dimensional Features (MFCC & delta MFCC)
        vector = extract_features(audio, sr)
        log_likelihood = np.zeros(len(models))

        # Checking Score with each Model
        for i in range(len(models)):
            gmm    = models[i]['model']
            scores = np.array(gmm.score(vector))
            log_likelihood[i] = scores.sum()
      chosen_class = np.argmax(log_likelihood)
      log_likelihood_name = ('/').join([sample_path.split('/')[-2], sample_path.s
```

As evident from the code, all the log probabilities for each of the test sample on the 5 Gaussian Mixture Models are stored in a data frame object for better results analysis. Last chapter boils down into the Results and Conclusions associated with our system.

Chapter *4*

Results and Conclusion

In this section, we'll focus on the analytics of our classification of testing data with the Gaussian Mixture Models of each class, and hypothesize the cause and deviation of the log probabilities of each of the models with respect to actual class. Following that, we'll conclude this paper, along with listing our achievements, limitations and further scope of improvement.

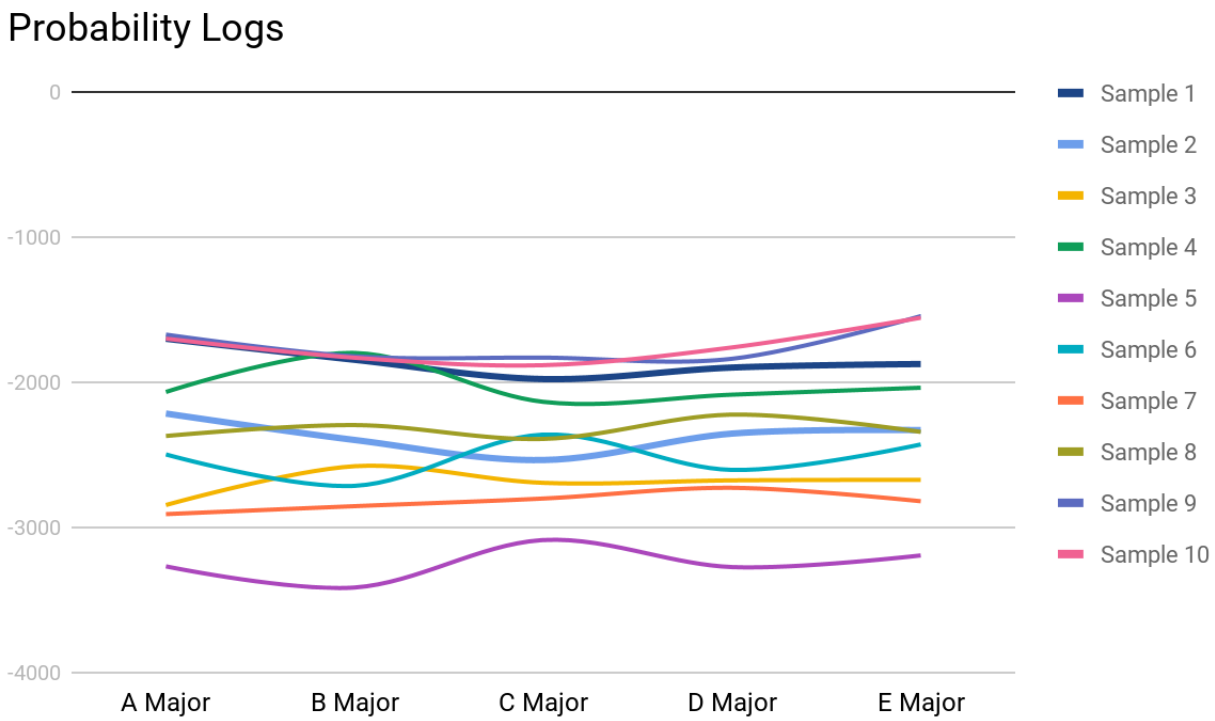
4.1 Results

For the 10 testing samples, the table visualizes the resulting python dataframe object which comprises of all the predictions and probability logs of GMMs per sample. Our system predicted the accurate class for 100% of the testing samples.

Sample Number	Probability Logs of GMMs					Predicted Class	Actual Class
	A-Chord	B-Chord	C-Chord	D-Chord	E-Chord		
1	-17.0017	-18.4507	-19.7808	-18.9865	-18.7445	A-Chord	A-Chord
2	-22.1549	-23.9753	-25.3556	-23.5348	-23.2974	A-Chord	A-Chord
3	-28.4556	-25.7817	-26.9279	-26.7579	-26.7206	B-Chord	B-Chord
4	-20.6685	-17.9802	-21.3493	-20.8546	-20.3773	B-Chord	B-Chord
5	-32.6795	-34.1376	-30.8674	-32.7297	-31.9277	C-Chord	C-Chord
6	-24.9708	-27.1348	-23.6059	-26.0378	-24.2825	C-Chord	C-Chord
7	-29.0821	-28.5340	-28.0077	-27.2699	-28.1952	D-Chord	D-Chord
8	-23.7003	-22.9481	-23.9012	-22.2280	-23.4184	D-Chord	D-Chord
9	-16.7173	-18.2124	-18.3035	-18.3811	-15.4429	E-Chord	E-Chord
10	-16.9948	-18.3114	-18.8240	-17.6006	-15.5553	E-Chord	E-Chord

The predicted class of the sample is derived from the probability logs across all the GMMs, and choosing the class that corresponds to the highest value of the probability log in it's respective Gaussian

Mixture Model. We can see that all of the predicted classes match with the actual class of the sample, signifying the accuracy and precision of GMMs in fitting and operating under small datasets.



The line graphs in the figure represents a smooth curve of a particular sample, derived from the probability logs of all the 5 GMMs. The maxima of each curve represents it's **Predicted Class**. We can see that due to the data being so small and lacking variation and modularity, the values of probability logs don't vary significantly as one would expect. However, in our testing, we consistently saw the performance being in the range of 95-100% under the margin of error.

There is also justification of some values of logs being closer to the predicted class's log. For example, in the Sample 5, value of E-Chord's GMM's log value is very close to C-Chord's GMM's log. This is due to the fact that C Major Chord comprises of 3 distinct semitones, major third (E) and perfect fifth (G), which are reflected into the DFT and eventually into the MFCC and delta MFCC features. These characteristics make the learning and prediction of these distinct chords a little similar. However, this can be resolved by expanding the dataset for each class.

4.1.1 Limitations

Apart from our limited addressing of different variations of chords that are widely used, we also would like to acknowledge the following limitations that our system suffers from.

- Minuscule number of chords are taken into consideration where in the music industry, a pool of thousands different variations are used everyday.

- Limited and less variant data, with scope limited to acoustic guitar. This system fails to acknowledge and incorporate different variants of the chords played on other instruments like Piano or Synthesizer.
- Lack better audio preprocessing, like including VAD (Voice Activity Detection), state of the art noise removal and silence truncation/.
- In this evaluation, we have not taken out-of-set chords into account i.e. if the audio does not belong to any speaker, still our system will identify and classify it to one of the classes in trained set depending upon highest likelihood.

4.1.2 Scope of Improvement

Following Open/Closed principles, we can extrapolate the same system to accommodate multiple chords rather than just 5, by training Gaussian Mixture Models for each of the chord. This can be achieved by training and testing using a bigger dataset than used in this paper, that constitutes more chords, more samples per chord and more environment and realistic noise for a broader and better clustering of the data points inside the GMM, which will improve a scattered input classification to it's respective chord.

4.2 Conclusion

This system is also a testament to how efficient and accurate GMM and other clustering mechanisms can be in terms of labelled data classification in general, by utilizing small datasets. We tried implementing a feed-forward neural network for the same, but it refused to converge efficiently to train each class respectfully, due to lack of data. Deep Learning requires alot of labelled data to be cost-efficient and worth implementing. GMMs in contrast proved how fast it can implemented for learning mixture models, and can be eventually used for classification.

From the above results we can conclude that with a small dataset of 60 seconds of audio of a guitar playing a chord, we achieved the accuracy of 100% on our chord classification system based on the 20 second testing samples, which really speaks volumes about the possible applications and scaling of our system. Our system boasts Open/Close Principle and is expandable to accommodate multiple classes as the dataset grows. Apart from the only 5 Major chords that we used to train the models, we can expand this same system into multiple chords including all of the 12 Major Chords, along with all minor, diminished, augmented, semi-augmented etc.

Bibliography

[1] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011.

[2] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCun. Unsupervised learning of sparse features for scalable audio classification. In *ISMIR*, volume 11, page 2011. Citeseer, 2011.

[3] J. Osmalskyj, J. J. Embrechts, M. Droogenbroeck, and S. Pirard. Neural networks for musical chords recognition. 01 2012.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[5] D. A. Reynolds. Speaker identification and verification using gaussian mixture speaker models. *Speech communication*, 17(1-2):91–108, 1995.

[6] M. P. Ryynnen and A. P. Klapuri. Automatic transcription of melody, bass line, and chords in polyphonic music. *Computer Music Journal*, 32(3):72–86, 2008.