

**MUSIC GENRE CLASSIFICATION**

***Submitted By***

TAARINI V-18BCS089 VISNU SANKER SS-18BCS112

PRIYADHARSHINI T-18BCS090 SITTHANADHAN K-18BCS106

KALAICHANDRAN C-18BCS100 SAKTHIVEL S-18BCS208

***In partial fulfillment for the award of the degree***

*of*

**BACHELOR OF ENGINEERING**

in

**COMPUTER SCIENCE AND ENGINEERING**

**KUMARAGURU COLLEGE OF TECHNOLOGY**

**COIMBATORE-641 049**

(An Autonomous Institution Affiliated to Anna University, Chennai)

**December 2020**

### TABLE OF CONTENTS

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | | **PAGE NO.** |
|  | **ABSTRACT** | | 2 |
| 1 | **INTRODUCTION** | | 2 |
|  | 1.1 | CONCEPTUAL STUDY OF THE PROJECT | 2 |
|  | 1.2 | OBJECTIVES OF THE PROJECT | 2 |
|  | 1.3 | Proposed Approach | 3 |
| 2 | **LITERATURE REVIEW** | | 3 |
| 3. | **PROBLEM DEFINITION** | | 4 |
| 4. | **RELATED WORK** | | 4 |
| 5. | **DATASET** | | 5 |
|  | 4.1 | DATASET DESCRIPTION | 5 |
|  | 4.2 | DATASET EXPLORATION | 5 |
| 6. | **PROPOSED SYSTEM** | | 5 |
|  | 4.1 | METHODOLOGIES TO SOLVE | 5 |
|  | 4.2 | IMPLEMENTATION | 6 |
| 7. | **SOURCE CODE** | | 8 |
| 8. | **CONCLUSION** | | 12 |
| 9. | **REFERENCES** | | 13 |

**ABSTRACT**

In the area of music information retrieval (MIR), categorizing music according to their genre is a challenging task. We studied and implemented k-nearest neighbors , a classification algorithm admitting two different types of inputs.The model is trained end-to-end, to predict the genre label of an audio signal,with the help of spectrogram. Being able to instantly classify songs in any given playlist or library by genre is an important functionality for any music streaming/purchasing service.

**INTODUCTION**

CONCEPTUAL STUDY OF THE PROJECT:

This project is primarily aimed to create an automated system for classification model for music genres. Genre usually assumes high weight in music recommender systems. Genre classification, till now, had been done manually by appending it to metadata of audio files or including it in album info. This project however aims at content-based classification, focusing on information within the audio rather than extraneously appended information.

OBJECTIVES OF THE PROJECT:

Companies nowadays use music classification, either to be able to place recommendations to their customers (such as Spotify, Soundcloud) or simply as a product (for example Shazam). Determining music genres is the first step in that direction. Machine Learning techniques have proved to be quite successful in extracting trends and patterns from the large pool of data. The same principles are applied in Music Analysis also.

Proposed Approach:

The automatic music genre classifier, here, is aimed to categorize a musical data into 9 broad categories. The categorization is done by using a classifier upon a vector of features computed from the musical data.  
Humans are remarkably good at genre classification as they can identify genre of a music by 250 milliseconds of an audio. This suggests that genre classification methodology should be as close as possible to human perception of music rather than any higher-level theoretical description. Therefore, here we have used chroma-based features as they are closely correlated to harmonic and melodic aspects of music, while being robust to changes in timbre and instrumentation. Chromagram is also very widely used to analyze and map human perception of music to signal processing techniques. Further, we compared performance of these chroma-based audio features or pitch class profiles with the performance of other features like MFCC, zero-crossing, rhythm based features, etc to establish classification efficiency associated with each.

**LITERATURE SURVEY**

Many companies nowadays use music Classification, either to be able to place Recommendations to their customers (such as Spotify, Sound cloud), or simply as a product (for example Shazam). Determining specific Music genres is a first step towards this goal. In this paper, we will present our successive Steps towards building different classifying Methods allowing us to identify a specific Genre from an initial audio file. We will first Describe how we collected data, and why this Choice of data was pertinent. Then, we offer a Possible conversion of this data into exploitable Information, and perform feature selection. We Will then progress onto presenting our various Algorithms and machine learning techniques Used for classification. The final output of our Algorithms is the prediction of the genre of Each input. We also quickly diverge into composer classification for classical music. Finally, we present Our results that we have obtained while studying this problem.

**PROBLEM DEFINITION**

Music plays a very important role in people’s lives. Music bring like-minded people together and is the glue that holds communities together. Communities can be recognized by the type of songs that they compose, or even listen to. Different communities and groups listen to different kinds of music. One main feature that separates one kind of music from another is the genre of the music. The aim of this project is:

1. To build a machine learning model which classifies music into its respective genre.

2. To compare the accuracies of this machine learning model and the pre-existing models, and draw the necessary conclusions.

**RELATED WORK**

Machine learning techniques have been used for music genre classification for decides now. In 2002, G. Tzanetakis and P. Cook used both the mixture of Gaussians model and k-nearest neighbors along with three sets of carefully hand-extracted features representing timbral texture, rhythmic content and pitch content. They achieved 61% accuracy. As a benchmark, human accuracy averages around 70% for this kind of genre classification work. Tzanetakis and Cook used MFCCs, a close cousin of mel-spectrograms, and essentially all work has followed in their footsteps in transforming their data in this manner.These results have far exceeded human capacity for genre classification, with our research finding that current state-of-the-art models perform with an accuracy of around 91% when using the full 30s track length. Many of the papers which implemented CNNs compared their models to other ML techniques, including k-NN, mixture of Gaussians, and SVMs, and CNNs performed favorably in all cases. Therefore we decided to focus our efforts on implementing a high-accuracy CNN, with other models used as a baseline.

**DATASET**

DATASET DESCRIPTION:

* Dataset used - GTZAN dataset
* Repository – MARYSAS

It contains 9 music genres, each genre has 100 audio clips in .wav format. The genres are – blues, classical, country, disco, pop, jazz,reggae,rock, metal. Each audio clips has a length 30 seconds, are 22050Hz Mono 16-bit Files.

DATASET EXPLORATION:

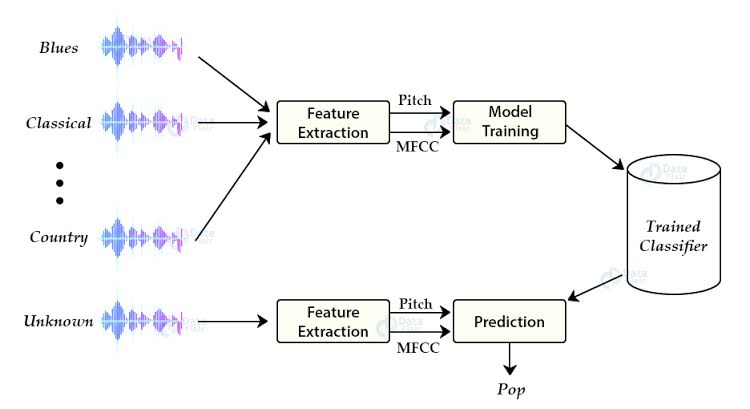
The dataset incorporates samples from variety of sources like CDs, radios,Microphone recordings etc.We split the datset in 0.9 : 0.1 ratio and used 5-fold cross validation for reporting the results.

**PROPOSED SYSTEM**

METHODOLOGIES TO SOLVE:

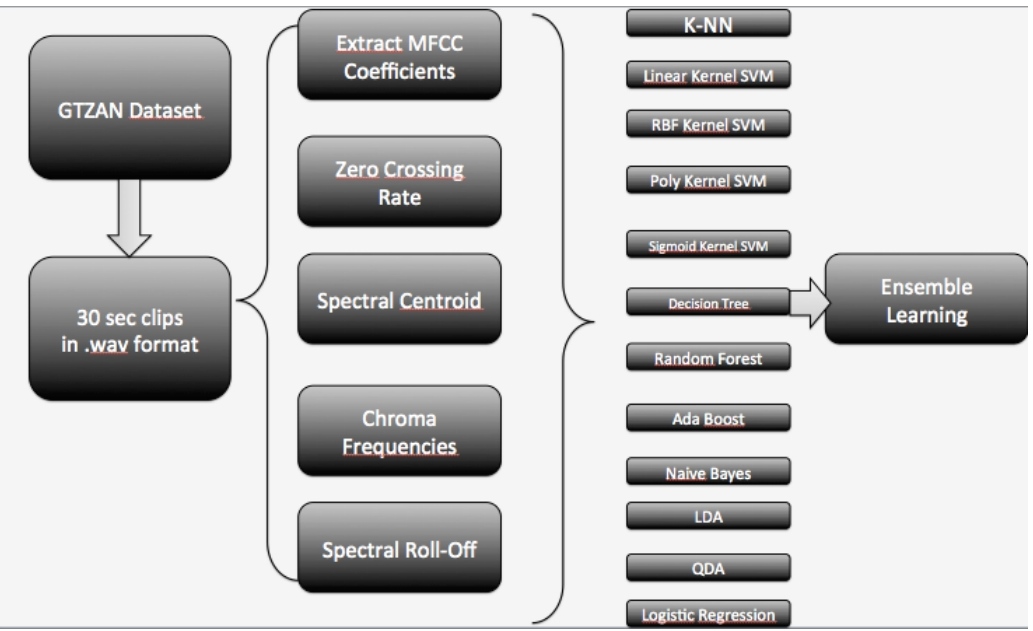
To classify our audio clips, we chose 5 features: Mel-Frequency Cepstral Coefficients, Spectral Centroid, Zero Crossing Rate, Chroma Frequencies, Spectral Roll-off. These 5 features are appended to give a 28 length feature vector. Then,we used dierent multi-class classiers and an ensemble of these to obtain our results.

A flowchart is been provided below for better understanding:



IMPLEMENTATION:

**Extraction of features**



**Mel-Frequency Cepstral Coefficients**

MFCC represents a set of short term power spectrum characteristics of the sound and have been used in the state-of-the-art recognition and sound categorisation techniques. It models the charac-terics of human voice. This features is a large part of the final feature vector (13 coefficients). The method to implement this feature is below :

– Dividing the signal into several short frames. The aim of this step is to keep an audio signal constant.

– For each frame, we calculated the periodogram estimate of the power spec-trum. This is to know frequencies present in the short frames.

– Pushing the power spectra into the mel filterbank and collecting the energy in each filter to sum it. We will then know the number of energy existing in the various frequency regions.

**Chroma Frequencies**

Chroma frequency vector discretizes the spectrum into chromatic keys, and represents the presence of each key. We take the histogram of present notes on a 12-note scale as a 12 length feature vector. The chroma frequency have a music theory interpretation. The histogram over the 12-note scale actually is sufficient to describe the chord played in that window. It provides a robust way to describe a similarity measure between music pieces.

**Spectral Centroid**

It describes where the ”centre of mass” for sound is. It essentially is the weighted mean of the frequencies present in the sound. Consider two songs, one from blues and one from metal. A blues song is generally consistent throughout it length while a metal song usually has more frequencies accumulated towards the end part. So spectral centroid for blues song will lie somewhere near the middle of its spectrum while that for a metal song would usually be towards its end.

**Spectral Roll-off**

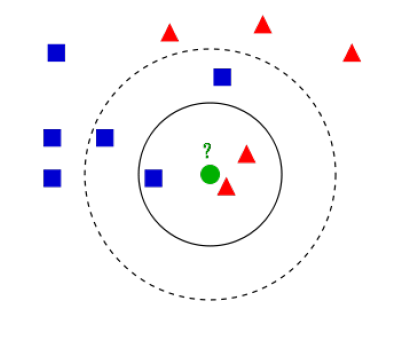
It is a measure of the shape of the signal. It represents the frequency at which high frequencies decline to 0. To obtain it, we have to calculate the fraction of bins in the power spectrum where 85% of its power is at lower frequencies.

**K-Nearest Neighbors Algorithm**

Our last approach was the widely used technique of K-Nearest Neighbors. This is often one of the first techniques applied to classification problems because of its accessibility of implementation. We first applied Principal Component Analysis to our features vector to reduce the dimension of our features space to three, and then ran the K-NN algorithm. The idea here is to "place" our training set in space, coloring each example using our labels, pick k, and then find the k "nearest" neighbors of the testing example that we are trying to classify. We then make our prediction as the most represented class amongst the neighbors.The parameters in this model are the distance function, and k. The latter is chosen by training on different values of k, and picking the be stone. The distance is often chosen to be the euclidean distance

d(x1, x2) = ||x1 − x2||2

An example of this method can be found below.



**SOURCE CODE**

------------------------------------------------------------------------------------

from python\_speech\_features import mfcc

import scipy.io.wavfile as wav

import numpy as np

from tempfile import TemporaryFile

import os

import pickle

import random

import operator

import math

# function to get the distance between feature vecotrs and find neighbors

def getNeighbors(trainingSet, instance, k):

distances = []

for x in range(len(trainingSet)):

dist = distance(trainingSet[x], instance, k) + distance(instance, trainingSet[x], k)

distances.append((trainingSet[x][2], dist))

distances.sort(key=operator.itemgetter(1))

neighbors = []

for x in range(k):

neighbors.append(distances[x][0])

return neighbors

# identify the class of the instance

def nearestClass(neighbors):

classVote = {}

for x in range(len(neighbors)):

response = neighbors[x]

if response in classVote:

classVote[response] += 1

else:

classVote[response] = 1

sorter = sorted(classVote.items(), key = operator.itemgetter(1), reverse=True)

return sorter[0][0]

# function to evaluate the model

def getAccuracy(testSet, prediction):

correct = 0

for x in range(len(testSet)):

if testSet[x][-1] == predictions[x]:

correct += 1

return (1.0 \* correct) / len(testSet)

# directory that holds the wav files

directory = " path\_to\_dataset "

# binary file where we will collect all the features extracted using mfcc (Mel Frequency Cepstral Coefficients)

f = open("my.dat", 'wb')

i = 0

for folder in os.listdir(directory):

i += 1

if i == 11:

break

for file in os.listdir(directory+folder):

try:

(rate, sig) = wav.read(directory+folder+"/"+file)

mfcc\_feat = mfcc(sig, rate, winlen=0.020, appendEnergy=False)

covariance = np.cov(np.matrix.transpose(mfcc\_feat))

mean\_matrix = mfcc\_feat.mean(0)

feature = (mean\_matrix, covariance, i)

pickle.dump(feature, f)

except Exception as e:

print('Got an exception: ', e, ' in folder: ', folder, ' filename: ', file)

f.close()

# Split the dataset into training and testing sets respectively

dataset = []

def loadDataset(filename, split, trSet, teSet):

with open('my.dat', 'rb') as f:

while True:

try:

dataset.append(pickle.load(f))

except EOFError:

f.close()

break

for x in range(len(dataset)):

if random.random() < split:

trSet.append(dataset[x])

else:

teSet.append(dataset[x])

trainingSet = []

testSet = []

loadDataset('my.dat', 0.66, trainingSet, testSet)

def distance(instance1 , instance2 , k ):

distance =0

mm1 = instance1[0]

cm1 = instance1[1]

mm2 = instance2[0]

cm2 = instance2[1]

distance = np.trace(np.dot(np.linalg.inv(cm2), cm1))

distance+=(np.dot(np.dot((mm2-mm1).transpose() , np.linalg.inv(cm2)) , mm2-mm1 ))

distance+= np.log(np.linalg.det(cm2)) - np.log(np.linalg.det(cm1))

distance-= k

return distance

# making predictions using KNN

leng = len(testSet)

predictions = []

for x in range(leng):

predictions.append(nearestClass(getNeighbors(trainingSet, testSet[x], 5)))

accuracy1 = getAccuracy(testSet, predictions)

print(accuracy1)

# testing the code with external samples

test\_dir = " path\_to\_test\_file "

# test\_file = test\_dir + "test.wav"

test\_file = test\_dir + "test2.wav"

# test\_file = test\_dir + "test4.wav"

(rate, sig) = wav.read(test\_file)

mfcc\_feat = mfcc(sig, rate, winlen=0.020, appendEnergy=False)

covariance = np.cov(np.matrix.transpose(mfcc\_feat))

mean\_matrix = mfcc\_feat.mean(0)

feature = (mean\_matrix, covariance, i)

from collections import defaultdict

results = defaultdict(int)

directory = " path\_to\_dataset "

i = 1

for folder in os.listdir(directory):

results[i] = folder

i += 1

pred = nearestClass(getNeighbors(dataset, feature, 5))

print(results[pred])

------------------------------------------------------------------------------------

**CONCLUSION**

The main quantitative metric which we used to judge our models is accuracy (that is, percentage of predicted labels which matched their true labels), and our main way of visualizing the performance of our best model is through the confusion matrices. Because the labeling was uniformly distributed on our data, cross-validation, and test sets, these confusion matrices offer not only a way to visualize our data, but more specific information than precision and recall values offer.We selected our hyper parameters based of empirical results and industry standards. Working with audio time-series data over 2 dimensions cause sparse gradient problems, similar to those often encountered in natural language or computer vision problems.

**REFERENCES**

[1] Hareesh Bahuleyan. Music genre classification using machine learning techniques

[2] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. IEEE Transactions on Speech and Audio Processing, 10(5):293–302, July 2002.

[3] Y. Panagakis, C. Kotropoulos, and G. R. Arce. Music genre classification via sparse representations of auditory temporal modulations. In 2009 17th European Signal Processing Conference, pages 1–5, Aug 2009.

[4] Mingwen Dong. Convolutional neural network achieves human-level accuracy in music genre classification. CoRR, abs/1802.09697, 2018.