Music Genre Classification

Xi Liu
Santa Clara University

Shruthi Ramprasad
Santa Clara University

Abstract

This paper aims to develop a machine learning model that automatically classifies a given audio file into its respective music genre using the k-nearest neighbors algorithm and Bayes theoretic classifier with multivariate Gaussian approaches. Humans invented musical genres as categorical designations for various types of musical works. A musical genre is defined by the shared characteristics of its members. These qualities typically have something to do with the musical instrumentation, rhythmic structure, and harmonic content. Genre hierarchies are commonly used to organize the vast amounts of music available on the Internet. Musical genre annotation is typically done by hand. Machine learning algorithms are used to automatically predict genres. Automatic musical genre classification can supplement or replace the human user in this process, with an accuracy of 74% for ten musical genres using KNN and 72% for Bayes theoretic classification. The results are comparable, and we can observe that the KNN produces better results in the experiments we conducted.

Keywords: K-Nearest Neighbor (KNN), Bayes Theoretic Classifier(BTC), Multivariate Gaussian Distribution(MGD), GTZAN.

1. Introduction

1.1. Problem Description

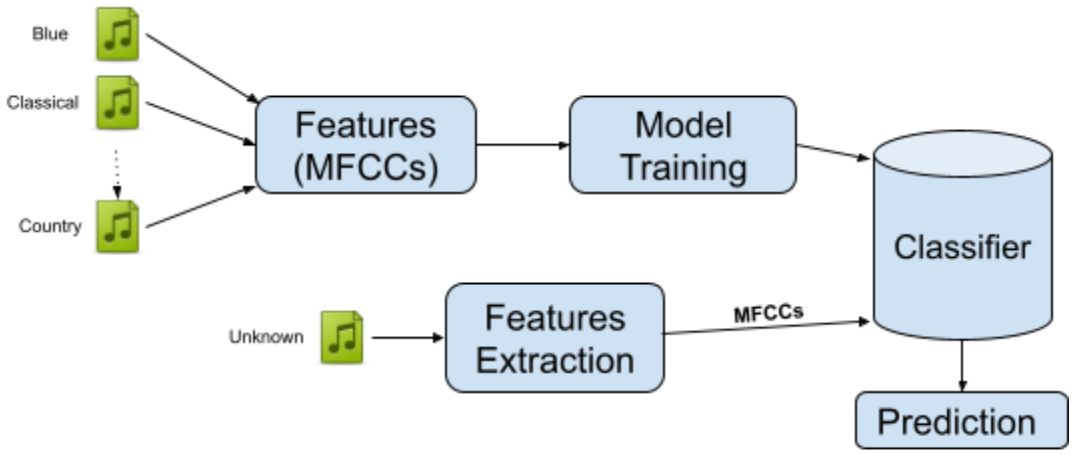


Figure 1 Model Structure

The classification of genres is a critical task with numerous real-world applications. As there are a growing number of music files available digitally on the Web, it is becoming more important to automatically extract music information in order to structure and organize the data. One of the services that music content distribution companies will offer to attract customers is automatic music analysis. The recent legal attention given to organizations like Napster is another sign of the growing significance of digital music distribution. A 2016 statistic suggests that tens of thousands of songs were released each month on Spotify. As a result, the amount of accurate meta-data needed for database management and search/storage purposes rises proportionally. This is especially true for internet platforms like Soundcloud and Spotify. The ability to instantly categorize songs in any playlist or library by genre is a crucial feature for any music streaming or purchasing service, and the statistical analysis that accurate and thorough labeling of music and audio provides is essentially limitless.

We tested a k-nearest neighbors and Bayes theoretic classifier with multivariate gaussian approaches. We experimented with both raw amplitude data and transformed mel-spectrograms of that raw amplitude data as input to our algorithms. We then generate a predicted genre from a list of ten common music genres. Converting our raw audio into mel-spectrograms and using the k-nearest neighbors algorithms produced better results, with around 74% accuracy on all of our models.

1.2. Dataset

For this project, we have downloaded the dataset from the Kaggle website[11]. The GTZAN dataset is the most widely used public dataset for music genre recognition evaluation in machine learning research (MGR). In order to represent a variety of recording conditions, the files were collected in 2000 and 2001 from a variety of sources, including personal CDs, radio, and microphone recordings. The dataset consists of 1,000 audio tracks, each lasting 30 seconds with 661794 signals. It includes ten genres, each with 100 tracks. The tracks are all 22050 Hz, mono, 16-bit.wav audio files.

The genres are blues, classical, country, disco, hiphop, jazz, metal, pop, reggae, and rock.

1.3. Features Extraction

The process of creating a concise numerical representation that can be used to describe an audio segment is known as "feature extraction". The main challenge in developing pattern recognition systems is the design of descriptive features for a specific application. Once the features have been extracted, standard machine learning techniques that are not specific to one application domain can be applied. MFCC is the most widely used technique for extracting features from the audio signal.

Mel-frequency cepstral coefficients (MFCC) is a perceptually motivated STFT-based feature. The FFT bins are grouped and smoothed using the perceptually motivated Mel-frequency scaling after taking the log-amplitude of the magnitude spectrum. Finally, a discrete cosine transform is used to decorrelate the resulting feature vectors. Although 13 coefficients are typically used for speech representation, we discovered that the first five provided the best performance in genre classification. The MFCC feature extraction technique consists of windowing the signal, performing the DFT, calculating the log of the magnitude, and then warping the frequencies on a Mel scale before performing the inverse DCT.

1.4 Approaches

There are several approaches available to perform classification on this dataset. Some of the widely used approaches are:

- Multiclass support vector machines
- K-means clustering
- K-nearest neighbors
- Convolutional neural networks

We have decided to proceed with the K Nearest Neighbors and Bayes's Theoritic Classifier with Multivariate Gaussian Distribution. Here the predictions are made on a case-by-case basis by locating the closest training examples and predicting the label that appeared with the greatest frequency among the labels.

2. K-Nearest Neighbor(KNN)

2.1. Research for KNN

KNN is a supervised learning-based machine learning method. Because it is a non-parametric algorithm, it does not make basic data assumptions. The KNN algorithm is used for deceleration and partition, but it is most commonly used for

partition. The KNN algorithm detects similarities between new and existing data and adds new data to the group that most closely resembles the existing data. The KNN algorithm is also known as the “lazy learner” algorithm because it stores data rather than quickly learning from training data and adds action to the database during segmentation.

The nearest neighbor k (KNN) is a non-parameter separator that votes for adjacent training areas "k" in a given test area and divides the result by the majority vote of the nearest "K" neighbor [6]. KNN is based on the labeling of data and works well with details defined by multiple dense clusters, with each collection representing a given label [6] [9]. While this algorithm is simple to use, it is vulnerable to large amounts of input. It can be demonstrated that increasing data size reduces KNN performance because the K neighbors closest to a given test site become less and less available in that test area as data size increases.

The following algorithm can be used to explain how K-NN works [7]:

- Step 1: Count the number of neighbors K.
- Step 2: Determine the Euclidean distance between K neighbors.
- Step 3: Sort the distances from the closest to the farthest.
- Step 4: Determine the K closest neighbors based on the calculated Euclidean distance.
- Step 5: Count the number of data points in each category among these K neighbors.
- Step 6: Assign the new data points to the category with the greatest number of neighbors.
- Step 7: Our model is complete.

To calculate the distance between the new point and each training point, a number of methods are available, such as Euclidean, Manhattan (for continuous), and Hamming distances (for categorical).

In our project we are using the Euclidean distance formula to calculate the distance between the new point and the training point. The formula to calculate Euclidean distance between two points, p and q, with p,q∈ $\Re^n$, is:

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \qquad (1)$$

2.2. Implementation

Initially we use the MFCC feature to extract features from audio wav files in python_speech_features. Each of the 30-sec audio files contains 661794 signals constituting 2994 frames and the number of dimensions of each mfcc is defaulted to 13.

```
for files in os.listdir(file_path+"/"+folder_list):
    samplerate,signal = wav.read(file_path+"/"+folder_list+"/"+files)
    mfcc_feat = mfcc(signal, samplerate, winlen=0.020, appendEnergy = False)
```

Figure 2. Code snippet depicting feature extraction for KNN

Further, we calculate the Euclidean distance between the test file and the training files and sort the k nearest neighbors.

```
def getNeighbors(trainingSet, new_audio, k):
    distances = []
    for x in range (len(trainingSet)):
        dist = euc_distance(trainingSet[x], new_audio)
        distances.append((trainingSet[x][1],dist))
    distances.sort(key=operator.itemgetter(1))
    neighbors = []
    for x in range(k):
        neighbors.append(distances[x][0])
    return neighbors
```

Figure 3- Code snippet to calculate K nearest neighbors

The training set is the actual training data, and k is the hyperparameter indicating how many nearest neighbors we want. The first part calculates the distances

between all of the points in the training set, then we sort the data and find the nearest neighbor to find the k nearest neighbors.

```
# predict the genre
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]
```

Figure 4- Code snippet depicting the genre prediction

In this section, we define a set that will aid in determining the response of each class present in nearest neighbors.Actually, the idea behind this function is that we already have a list of nearest neighbors, and because data is labeled data, each neighbor will hold some specific class. Now, from that nearest neighbor, whatever the maximum frequency is for each class, the new data will be assigned to that class.

For example, if k=10, the list of 10 nearest neighbors will be generated, and if (Class A=4, B=3, C=3), a new point will be assigned to A as having the highest frequency.

3. Bayes Theoretic Classifier(BTC) with Multivariate Gaussian Distribution(MGD)

3.1. Research for BTC with MGD

BTC is a pattern classifier. The rule of the Bayes Decision is to minimize the overall risk. It always selects the action that minimizes the conditional risk. The discriminant function is the tool used to calculate the minimum error rate and

make decisions by Bayes rule. Let's assume the data in this project obeys the MGD and the discriminant function is based on the MGD here.

The Multivariate Gaussian is a probability distribution that models the joint distribution of multiple random variables, where each variable may be dependent on the others. It is also known as a multivariate normal distribution.

A multivariate Gaussian is specified by a mean vector and a covariance matrix. The mean vector is a vector of the expected values of each random variable, while the covariance matrix specifies the pairwise covariance between each pair of variables.

The multivariate Gaussian is a common distribution used in machine learning for modeling continuous variables. It has several useful properties, including:

It is a flexible distribution that can approximate a wide range of continuous distributions.

It is easy to work with mathematically, which makes it suitable for many analytical and computational tasks. It is often used as a prior distribution for Bayesian machine learning models. In practice, the multivariate Gaussian is used in many areas of machine learning, such as generative models, clustering, and anomaly detection.

The multivariate Gaussian distribution is a vector-form generalization of the Gaussian distribution. The probability density function for an n-dimensional vector x=[x1, x2,..., xn] with a multivariate Gaussian distribution can be expressed[10]:

$$p(x) \sim \aleph(x|\mu, \Sigma)$$

$$p(x) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}} exp(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)) \quad \textbf{(2)}$$

- D: $dimension$
- X: $D - dimensional\ column\ vector$
- μ: $E[x] \in \Re^D : D - dimensional\ mean\ vector$

- Σ: $covariance\ matrix\ (D \times D)$
- $|\Sigma|$: $determinant\ of\ covariance\ matrix\ \Sigma$
- Σ^{-1}: $inverse\ of\ covariance\ matrix\ \Sigma$

The mean vector μ and covariance matrix Σ are computed separately by

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \textbf{(3)}$$

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu)(x_i - \mu)^T \qquad \textbf{(4)}$$

where x_i is the samples and m is the number of the samples.

The three-sigma rule of thumb is a very useful tool for detecting anomalies in univariate Gaussian distributions. Similarly, in the multivariate Gaussian distribution, a fault sample is one whose probability density of new vector x′ is less than a certain threshold:

$$y(x') = \begin{cases} 1\ (faulty) & if\ p(x') < \varepsilon \\ 0\ (normal) & if\ p(x') \geq \varepsilon \end{cases} \qquad \textbf{(5)}$$

where y represents the prediction label of the new vector x*.

Recall the minimum error rate discriminant:

$$g_i(x) = ln\, p(x|\omega_i) + ln\, P(\omega_i) \qquad \textbf{(6)}$$

Because $p(x|\omega_i) \sim \aleph(x|\mu_i, \Sigma_i)$, the general discriminant for densities with multivariate Gaussian distribution is:

$$g_i(x) = -\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i) - \frac{D}{2}ln^{2\pi} - \frac{1}{2}ln\left|\Sigma_i\right| + ln\, P(\omega_i) \qquad \textbf{(17)}$$

Denote a decision region $\Re_i$ for ω_i, if $g_i(x) > g_j(x)\ \forall j \neq i$, then x is in $\Re_i$.

3.2. Implementation

The statistics for the Gaussian distribution are mean and covariance. Therefore, the first step is to find out the mean vectors of features from each audio file in the training set.

```
for files in os.listdir(file_path+"/"+folder_list):
    samplerate,signal = wav.read(file_path+"/"+folder_list+"/"+files)
    mfcc_feat = mfcc(signal, samplerate, winlen=0.020, appendEnergy = False)
    mean_cur = mfcc_feat.mean(0)
    train[j] = mean_cur
    j+=1
```

Figure 5- Code snippet depicting feature extraction for BTC using MGD

Then, the 13-dimensional mean vector and the covariance matrix are calculated as follows:

```
mean_matrix = train.mean(0)
covariance = (train-mean_matrix).T.dot(train-mean_matrix)/len(train)
```

Figure 6- Code snippet to calculate the mean and covariance matrix

The general discriminant of the test file for each genre is calculated from the equation (17). Then, compare these general discriminants and make prediction:

```
    general_discriminant = g[0]
    predict[count] = 1
for k in range(10):
    if general_discriminant < g[k]:
        predict[count] = k+1
        general_discriminant = g[k]
```

Figure 7- Code snippet to make the genre prediction for MGD

4. Result

The dataset used in this paper is the GTZAN dataset. It includes 10 genres of music and each genre contains 100 audio files. The training set and the test set for our models are randomly extracted from the GTZAN dataset with split rate 0.9. This rate is changed in BTC implementation in order to test how the rate affects the accuracy of the classifier.

4.1. K-Nearest Neighbor

The accuracy of the K-Nearest Neighbor algorithm, to some degree, depends on the value of K. Generally speaking, using the K-Nearest Neighbor algorithm, the accuracy is around 70% after simulation. By adjusting the value of K, the accuracy can be improved on a small scale.

KNN Accuracy with Values of K					
K	4	6	8	10	12
Accuracy	69.1%	65%	74%	68%	69%

Figure 8 - Experimental Results for different K values

The accuracy for each genre prediction is as follows:

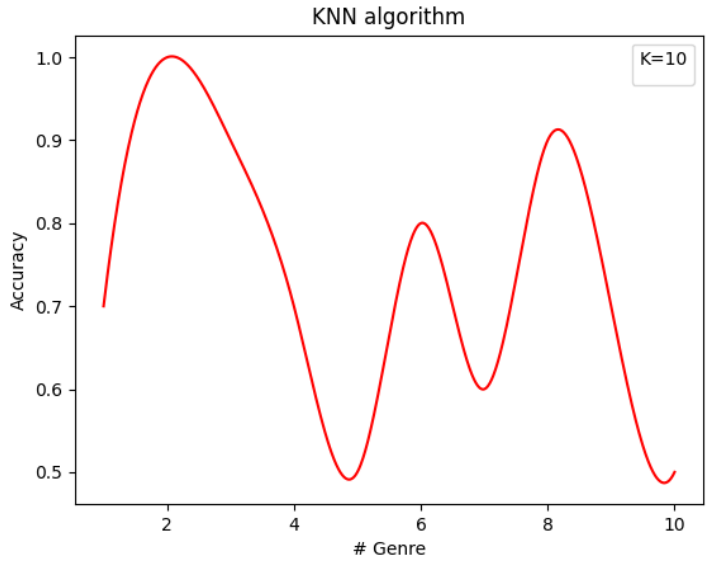


Figure 9 - Accuracy Prediction of music genre classification using KNN

The best prediction is made for the type of Classical with an accuracy of 100%.The worst predictions are made for Hip Hop and Rock, with an accuracy of less than 50%.

The incorrect classifications for Hip Hop music are predicted to Disco, Jazz, Metal and Rock.

```
Predicted genre is:  5  ,the true genre is:  5
Predicted genre is:  5  ,the true genre is:  5
Predicted genre is:  5  ,the true genre is:  5
Predicted genre is:  5  ,the true genre is:  5
Predicted genre is:  10  ,the true genre is:  5
Predicted genre is:  5  ,the true genre is:  5
Predicted genre is:  6  ,the true genre is:  5
Predicted genre is:  7  ,the true genre is:  5
Predicted genre is:  4  ,the true genre is:  5
Predicted genre is:  7  ,the true genre is:  5
```

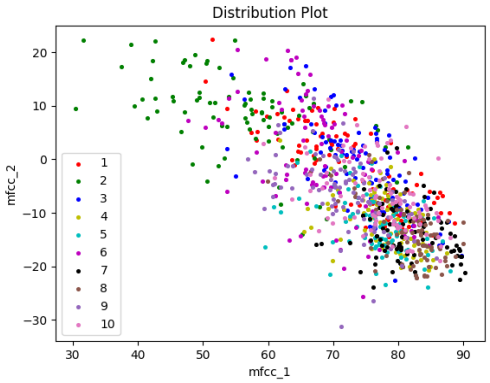
Figure 10 - Predictions for Genre 5

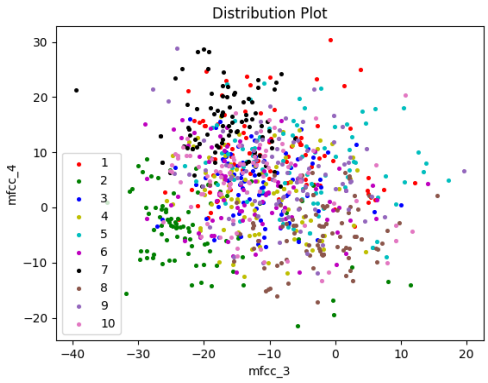
The incorrect classifications for Rock music are predicted to Blues, Disco, Jazz and Pop.

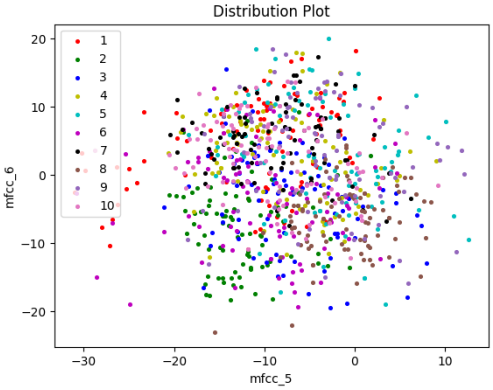
```
Predicted genre is:  10  ,the true genre is:  10
Predicted genre is:  10  ,the true genre is:  10
Predicted genre is:  6  ,the true genre is:  10
Predicted genre is:  8  ,the true genre is:  10
Predicted genre is:  1  ,the true genre is:  10
Predicted genre is:  8  ,the true genre is:  10
Predicted genre is:  4  ,the true genre is:  10
Predicted genre is:  10  ,the true genre is:  10
Predicted genre is:  10  ,the true genre is:  10
Predicted genre is:  10  ,the true genre is:  10
```

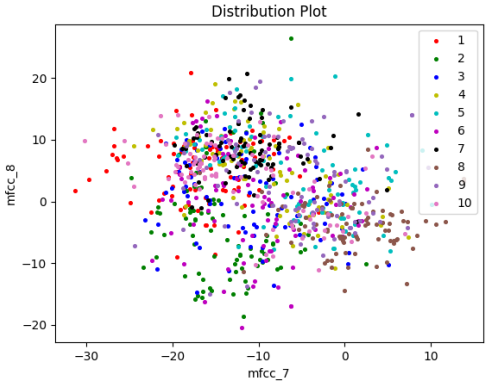
Figure 11 - Predictions for Genre 10

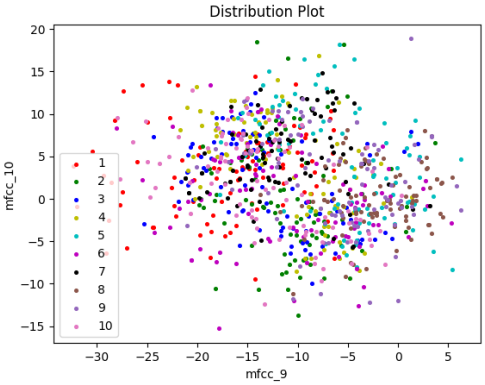
KNN is a lazy supervised algorithm. It classifies the instance by its distance to each class. Therefore, having a look at the distribution of each class is meaningful. There are in total 13 features of each instance in this project. The 2D figures between 2 of those features are plotted. In fact, high-dimensional datasets tend to be very sparse and they are going to lie close to a much lower-dimensional manifold. Therefore, these 2D figures are not quite representative for the case. The 2D distribution plots are as follows:

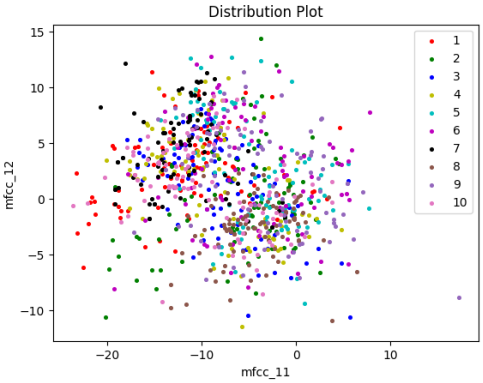

Distribution Plot
mfcc_2
mfcc_1
1
2
3
4
5
6
7
8
9
10

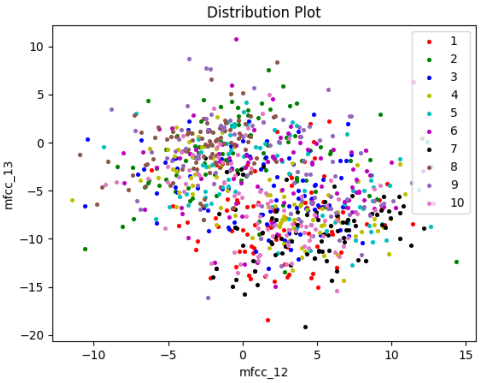

Distribution Plot
mfcc_4
mfcc_3
1
2
3
4
5
6
7
8
9
10


Distribution Plot
mfcc_6
mfcc_5
1
2
3
4
5
6
7
8
9
10


Distribution Plot
mfcc_8
mfcc_7
1
2
3
4
5
6
7
8
9
10


Distribution Plot
mfcc_10
mfcc_9
1
2
3
4
5
6
7
8
9
10


Distribution Plot
mfcc_12
mfcc_11
1
2
3
4
5
6
7
8
9
10


Distribution Plot
mfcc_13
mfcc_12
1
2
3
4
5
6
7
8
9
10

4.2. Bayes Theoretic Classifier

The accuracy of the BTC algorithm is also around 70%. The accuracy for each genre is shown below:

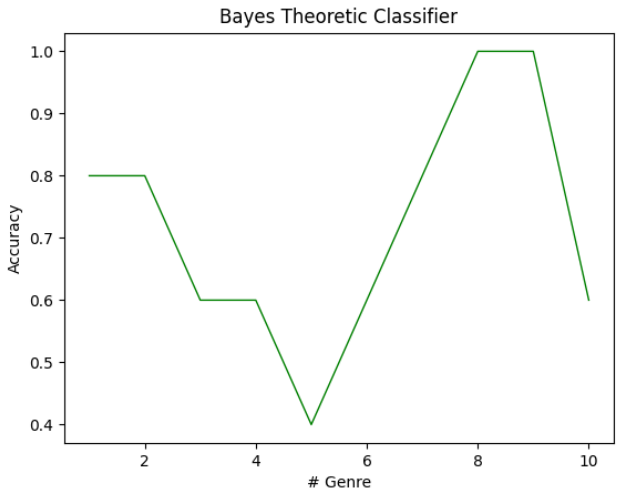


Figure 12 - Accuracy with BTC

Adjust the scale of the training set and the test set, and investigate the change of the accuracy:

	Bayes Classifier Algorithm		
Scale of Training Set	900	920	950
Scale of Test Set	100	80	50
Accuracy	68%	68.75%	72%

Figure 13 - Accuracies

The Bayesian Classifier assumes all the features are statistically independent, which means the feature data is independently, identically distributed from a multivariate Gaussian distribution. We estimate the parameters of the distribution via maximum likelihood and make the predictions by Bayes Rule.

The features used in this project are the 13 MFCCs, which are coefficients that collectively make up the mel-frequency cepstrum(MFC), that is a representation of the short-term power spectrum of a sound. MFCCs are derived from a type of cepstral representation of the audio clip, and they are quite decorrelated. This can be beneficial for the Multivariate Gaussian model.

5. Conclusion

This project has applied K Nearest Neighbour algorithm and Bayes Theoretic Classification algorithm to classify music genre of audio files. The dataset is GTZAN, which is the most-used public dataset for machine learning projects in music recognition. In GTZAN, each music is 30-second long, with 2994 frames. And the features extracted from each frame are 13-dimensional MFCCs. For the KNN algorithm, the distance between the instances is determined by Euclidean distance. Different K values are tested and the accuracy fluctuates in the range of 65% and 74%. The Bayes Classifier model in this paper is based on the multivariate Gaussian distribution and the accuracy is around 70%.

Reference

[1] Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow, 2nd edition, by Aurelien Geron, O'Reilly Media, 2019.

[2] Pattern Classification, 2nd edition, by R.O. Duda, P.E. Hart and D.G. Stork, Wiley, 2001.

[3] ProjectPro. (2022, June 17). Solved Music genre classification project using Deep Learning. ProjectPro. Retrieved February 16, 2023, from https://www.projectpro.io/article/music-genre-classification-project-python-code/566

[4] Viswanathan, A. P. (2016). Music genre classification. International Journal Of Engineering And Computer Science. https://doi.org/10.18535/ijecs/v4i10.38

[5] Lefaivre, A., & Zhang, J. Z. (2018). Music genre classification. Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion. https://doi.org/10.1145/3243274.3243310

[6] S. V. Chandan, M. R. Naik, Ashwini and A. V. Krishna, "Indian Instrument Identification from Polyphonic Audio using KNN Classifier," 2019 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET), Chennai, India, 2019, pp. 135-139, doi: 10.1109/WiSPNET45539.2019.9032860.

[7] https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4

[8]https://www.analyticsvidhya.com/blog/2021/04/simple-understanding-and-implementation-of-knn-algorithm/

[9] J. K. Bhatia, R. D. Singh and S. Kumar, "Music Genre Classification," 2021 5th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2021, pp. 1-4, doi: 10.1109/ISCON52037.2021.9702303.

[10] W. Xing, H. Yang, J. Sheng, X. Chang, W. Li and X. He, "Open-circuit Fault Detection and Location in MMCs with Multivariate Gaussian Distribution," 2020 4th International Conference on HVDC (HVDC), Xi'an, China, 2020, pp. 555-559, doi: 10.1109/HVDC50696.2020.9292753.

[11] https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification