Music Genre Classification: Traditional Methods vs. Deep Learning

This report shows the procedure/workflow on how music genre classification can be done using open-source signal processing and machine learning python libraries. The related code can be checked from following github repository:
https://github.com/serkanardaa/Musical-Genre-Classification

For a detailed implementation of the project, please check file main.ipynb under the repository

EQ2470, Group 4, November 2021

Outline

1. **Introduction and problem definition**
2. **Choosing dataset and partitioning it**
3. **Theory about signal processing, fourier analysis and features**
4. **Feature selection and extraction**
5. **Training models on training dataset**
6. **Selecting models using validation dataset**
7. **Evaluation of models on testing dataset**

References

1. Introduction and problem definition

Music genre classification is a topic within the area of music informatics. The pipeline of performing applications such as genre classification but also other applications involves in general similar steps as mentioned in the outline.

In this project we want to classify genres from audiofiles. The goal is to train some classifier which takes as input a chosen audiofiles and outputs a prediction of their genres.

Benefits of categorizing pieces of music in genres is many. For example, building search engines, organizing digital audio databases, and recommending songs. For instance, Spotify is a company that gets much value from using this type of application.

2. Choosing dataset and partitioning it

For this project, we have chosen a dataset called Free Music Archive(FMA) prepared by researchers from EPFL and NTU. The related paper[1] of the dataset was also among papers of ISMIR 2017. Although GTZAN dataset[2] is the benchmark dataset for musical genre classification in the literature, due to its flaws[3] we have chosen this dataset considering it also had a good amount of tracks (25.000 for medium size). This database also has some other advantages such as:

- Large scale (25.000 tracks, 16 genres, 44100Hz): avoids overtraining
- Available audio: Not dependent only on the features provided as in some databases. Can be used in end-to-end training systems like DL.
- High quality audio: no disturbance of noise
- Proper audio extraction: 30 seconds excerpts are extracted from middle of tracks which is the best interval as it is proven in [4].

When it comes to partitioning, we have chosen 500 tracks from 8 genres which are hip-hop, pop, rock, folk, experimental, electronic, Classical, Old Time/Historic. The percentage ratios for training, validation, and test datasets are chosen as 80%, 10%, and 10%.

While partitioning the data, we use two different approaches. The first approach is we choose tracks for these datasets for both genres such that there will be no common artists between training and validation/test dataset. The second approach is we choose tracks for the datasets such that there will be maximum amount of artists between training and validation/test dataset. The reason for these approaches is to see how much having common artists between different datasets affect the genre classification result and "mislead" the observer.

For the first approach, we ordered tracks according to artist name and selected datasets as separate as possible. For the second approach, we have chosen tracks for different datasets as close as possible such that training, validation, and test datasets will have maximum amount of common artists.

The result can be seen as in the following:

Common Artist Comparisons						
No common artists			With Common artists			
Labels	Genres	Training vs Validation	Training vs Test	Labels	Genres	Training vs Validation	Training vs Test
0	Hip-Hop	1 common artist	0 common artist	0	Hip-Hop	8 common artist	14 common artist
1	Pop	1 common artist	0 common artist	1	Pop	13 common artist	24 common artist
2	Rock	1 common artist	0 common artist	2	Rock	16 common artist	26 common artist
3	Folk	1 common artist	0 common artist	3	Folk	17 common artist	13 common artist
4	Experimental	1 common artist	0 common artist	4	Experimental	17 common artist	29 common artist
5	Electronic	1 common artist	0 common artist	5	Electronic	9 common artist	17 common artist
6	Classical	1 common artist	0 common artist	6	Classical	9 common artist	11 common artist
7	Old-Time / Historic	1 common artist	0 common artist	7	Old-Time / Historic	14 common artist	20 common artist

3. Theory about fourier analysis of signals

Fourier analysis of audio/music signals

A music or audio signal is in general seen as non-stationary, this means that statistical properties such as frequencies are changing over time. By taking the fourier transform of such a signal all together will then give the range of frequencies over the whole signal. But if we want to look more deeply into different parts of the signal, and get more detailed information on what frequencies are located in different parts of the signal we use the short time fourier transform.

Short time fourier transform

The goal of this transform is first slice the audio/music signal into smaller frames i.e. analysis windows, and then take the fourier transform of each and one of all these frames. The final transform will then be a frequency vs time representation of the signal. The size of these windows and the amount of so-called overlap between each analysis window can be chosen differently depending on application.

Long or short analysis windows, pros and cons

For example, having shorter windows will give more frequency information around the chosen time instance but will also mean that the fourier transform for that frame will be of lower quality i.e. a fourier transform of fewer samples will have limitations. And having longer analysis windows will give a higher quality fourier transform, but the frequency information for a specific time instance will be more overlapped by previous and past time instances of the signal.

4. Feature selection and extraction

As we checked some of the papers in the literature[4], the most useful features for musical genre recognition are spectral features and first five coefficients of MFCCs. Although features from chroma and beat can be also useful, we had to go with the most useful ones due to our computational limitations.

Feature selection

We want to classify genres. Our goal is here to find features which can characterize properties of different music genres as distinct as possible. If we manage to find such features, we can then train machine learning classifiers based on them. Feature extraction and selection of relevant features for a given music informatics application is a crucial step but difficult.

1. Spectral centroid

Spectral centroid is a measure of the spectral characteristics of a signal, it tells the center of mass of the spectrum which gives a good estimate of the brightness of a sound. It is calculated by taking a weighted average of the frequencies present in the signal, with weights being the magnitude of the frequency bins from the fourier transform of the signal.

2. Spectral roll-off

Spectral rolloff is similar to the spectral centroid in the sense that it also captures spectral properties, but this feature looks at the frequency in each frame under which a certain percentage of the total energy of the spectrum is contained. This percentage can be seen as a cutoff, or so called rolloff frequency.

3. Spectral flatness

Spectral flatness is a measure on how noisy each frame in the signal is as opposed to being tone-like. For example, white noise is maximum noisy and a pure sinusoid would be close to minimum noisy signal.

4. Spectral bandwidth

Spectral bandwidth is the difference between the upper and lower interrupting frequency with the spectral centroid as center frequency. The upper and lower frequency can be varied depending on application.

5. Spectral contrast

Spectral contrast is similar to the spectral flatness feature, estimating how noisy or tonal the frames are. The difference is that spectral contrast divides the frequency bins of each frame into a number of sub bands, and then basically calculates the spectral flatness of each band separately.

6. Zero-crossings

Zero-crossing rate is related to the number of zero crossings of the signal. The more zero crossings, the more noiselike the signal is. The rate is given by normalizing the number of zero crossings to get a number per frame in the range (0,1).

7. MFCC

MFCCs (Mel frequency cepstral coefficients) is calculated as follows

1. Take the Short Time Fourier Transform of the signal.
2. Map the power amplitude onto mel-scale using overlapping triangular windows. The mel as in melody is a perceptual scale of frequencies heard by humans as being the same distance from each other.
3. Taking the logarithm of the mel spectrum.
4. Taking the discrete cosine transform of the log-mel-spectrum which gives the Mel Frequency Cepstrum (MFC)
5. Choosing a range of coefficients from the MFC for each time frame gives the MFCC

In order to extract these features, we have calculated stft of each song with window size of 46 ms and hop size of 23 ms. Then we have used the popular library called Librosa to deploy easy feature extraction functions.

Due to the huge size of our datasets, instead of features of each frame we calculated mean and averages of features over the frames that cover 1 second of a song as it was implemented the famous GTZAN paper[2]. By doing that, although we doubled our features (17 x 2 = 34), we achieved a huge amount of reduction on number of samples which in the end provided us a reduction ratio of 20.

5. Training models on training dataset

For model training, we have benefited from scikit learn library which has a lot of different classifiers that can be used for our project. The chosen classifiers are as in the following:

- K Nearest Neighbor Classifier
- Support Vector Classifier
- Gaussian Naive Bayes Classifier
- Multi-layer Perceptron Classifier

In addition to that, as in the main purpose of the project we tried to implement a deep neural network for comparison with the traditional methods. Due to the limited time and computational resources, we couldn't implement an example for these models. But as we did our research on related papers[5][6], DNN and CNN based modelling achieve higher accuracies than the traditional classifiers using traditional signal structured features.

6. Selecting models using validation dataset

In order to achieve the highest accuracies in our experiments, we need to optimize the classifiers by finding the best values for their hyperparameters. For that reason, we have searched for the best values by using validation dataset. The method of model selection for each classifier has been chosen as in the following:

- K Nearest Neighbor Classifier: Best k value is seached such that it will give highest f1 score for macro average. After our trials, we achieved the highest accuracy with k = 1.
- Support Vector Classifier: For scikit learn library, we didn't have a hyperparameter for that classification. We have just used gaussian kernel setting which had the highest capacity for complex features
- Gaussian Naive Bayes Classifier: For that classifier, we didn't need any hyperparameter setting.
- Multi-layer Perceptron Classifier: Considering the similarity of its structure to neural networks, we had hyperparameters to optimize for that classifier such as number of hidden units and initial value for learning rate. After our search, the most time and f1 score efficient values for number of hidden units and initial value for learning rate are chosen as 50 and 0.001.

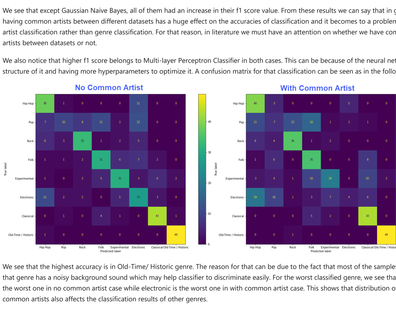
7. Evaluation of models on testing dataset

In the end of our experiments, we had the chance to compare macro average f1 scores of different classifiers in two different cases where we have no common artists and common artists. The results can be seen as in the following:

Macro Average Comparisons f1 score		
Classification Type	No common artists	Common artists
KNN	0.44	0.54
SVC	0.45	0.50
Gaussian Naïve Bayes	0.46	0.43
MLP	0.55	0.64

We see that except Gaussian Naïve Bayes, all of them had an increase in their f1 score value. From these results we can say that in general having common artists between different datasets has a huge effect on the accuracies of classification and it becomes to a problem of artist classification rather than genre classification. For that reason, in literature we must have an attention on whether we have common artists between datasets or not.

We also notice that higher f1 score belongs to Multi-layer Perceptron Classifier in both cases. This can be because of the neural network structure of it and having more hyperparameters to optimize it. A confusion matrix for that classification can be seen as in the following:



We see that the highest accuracy is in Old-Time/ Historic genre. The reason for that can be due to the fact that most of the samples from that genre has a noisy background sound which may help classifier to discriminate easily. For the worst classified genre, we see that pop is the worst one in no common artist case while electronic is the worst one in with common artist case. This shows that distribution of common artists also affects the classification results of other genres.

References

[1] Michaël Defferrard, Kirell Benzi,Pierre Vandergheynsty, Xavier Bresson et al. "FMA: A DATASET FOR MUSIC ANALYSIS" ISMIR (2017).

[2] George Tzanetakis et al. "Musical Genre Classification of Audio Signals" 2002.

[3] Bob L Sturm et al. "The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use." 2013.

[4] Carlos Silla,Celso A. A. Kaestner, Alessandro L Koerich et al. "A Machine Learning Approach to Automatic Music Genre Classification" 2008.

[5] Siddharth Sigtia, Simon Dixon et al. "IMPROVED MUSIC FEATURE LEARNING WITH DEEP NEURAL NETWORKS"

[6] Philippe Hamel and Douglas Eck et al. "LEARNING FEATURES FROM MUSIC AUDIO WITH DEEP BELIEF NETWORKS" ISMIR (2010).