CMLS HW1 - Music Genre Classification

{ 10482867 10521088 10539533 10702368 10751919 }@mail.polimi.it

May 18, 2020

Abstract

In this assignment we take a subset of the database from GTZAN [1] and perform automatic music genre recognition on the dataset. We first extract a set of features from the audio files and then implement two different classifiers (KNN and SVC) to get the prediction for unseen music samples. We then output the predicted genre results with two confusion matrices. In this report we are going to explain the detailed research approaches and how we come to the final conclusion.

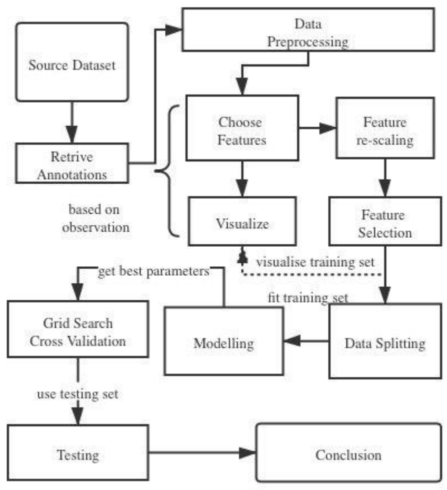

Figure 1: Workflow diagram.

1 Introduction

Music genres are categories used to describe different music with their special characteristics. These categories are complex and loosely-defined. They can not be defined easily using basic music features, but it is clear that instances of the same genre share a series of traits (instrumentation, rhythmic patterns, harmonic features etc.) that can be extracted quantitatively and used for the classification of digital audio files.
This can be done through supervised machine learning methods that allow us to extract genre information from large pools of audio data.
Thus, in this homework, we extract a group of features from a set of tracks and use these to train two different classifiers that should be able, in the end, to assign a genre to any new given track. We are using the Python package LibROSA[2] to extract the main features we need for analysis and further steps. We mainly use the Python package scikit-learn[3] for supervised learning and Numpy, Pandas for data analysis[4][5].
The code used for this project is also available on GitHub (`https://github.com/yilin10/MusicalGenreClassification`).

2 Dataset

The original dataset we are using can be found on GTZAN[6]. It is the most widely used public database in the field of music

genre recognition.
It consists of 1000 audio tracks each 30 seconds long, with 10 genres, each represented by 100 tracks. The tracks are all 22050 Hz monophonic 16-bit audio files in `.au` format.
In our Task (Assignment 1), we only need to classify 4 out of the 10 genres: Blues, Metal, Hip-Hop, and Reggae.

2.1 Annotation Retrieval

Since for each audio track, the labelled music genre is provided in the folder name, it meets the *a priori* conditions needed to use supervised machine learning methods to solve the problem. Therefore, we went through the whole set and saved the 400 rows of retrieved basic info: `genre` and `path` for each audio piece as pandas DataFrame object and saved a copy in `genres.csv` file.

2.2 Preprocessing

Next, we need to perform some preliminary operations on the data.
First of all we need to check that the dataset is homogeneous: we check every file for its duration and sample rate, and verify that, indeed, they all have a duration of 30 seconds and a sample rate of 22050 Hz. We also check the static tempo of the tracks to make sure that the tempos fall into a sensible range[1].
Lastly, we need to split the dataset into a training and a test set. This is done using the `train_test_split` from the `model_selection` module of scikit-learn, which computes a random split of the input dataset it is fed, ensuring that the training set is a representative sample of the whole. We chose to split the set in 80 % training vs. 20 % testing.

[1]This is done using the `beat.tempo` function from LibROSA.

2.3 Limits in the Dataset

Before going on, we must in some way check the quality of the dataset, i.e. whether it has been compiled sensibly or if it presents flaws or biases. According to [7], the GTZAN dataset presents several issues, such as the repetition of some of the tracks or the mislabelling of others. In the cited paper, the dataset was checked using both fingerprinting of the audio and also human ears.
The following is a list of the problems with the 4-genre subset we used in this assignment, which are reported also in Tab. 1 for clarity.

- **Unbalanced pool of artists**: all blues tracks come from only 9 artists; 34 % of the Reggae pieces come from artist Bob Marley (no.00-27 and no.54-60).
- **Exact and Recording repetition**: there appear to be repetitions of the same recording in many cases in the database. This happens 0 times in blues, 8 times in hip-hop, 8 times in metal, 12 times in reggae.
- **Version repetition**: some of the pieces in the dataset occur more than once as different versions, e.g. Metal 33 is “Enter Sandman” by Metallica, and Metal 74 is a parody of the same song. The instances of this problem are 0 in blues, 2 in hip-hop, 3 in metal and 2 in reggae.
- **Mislabelling**: the labeling of some of the tracks is dubious: for instance, Queen’s “Tie Your Mother Down” , “Tear it up” and “We Will Rock You” are labelled as Metal (58, 59, 60 respectively), whereas they might be more aptly labelled as Rock instead. In all, there are 0 errors in blues, 4 in hip-hop, 13 in Metal and 1 in reggae.

These inconsistencies might affect the accuracy of the classifiers.

Genre	Rec./Exact Repetition	Unbalance	Version	Mislabelling
Blues	–	only 9 artists	–	–
Hip-hop	6	–	2	4
Metal	–	–	3	13
Reggae	3	30 % from one artist	2	1

Table 1: A Summary of the known errors in the dataset.

3 Feature Extraction

For this assignment, we have deemed these following features to be helpful in highlighting the differences between different music genres, especially for blues, metal, hip-hop and reggae. The features in question are: MFCC(1-13), Chroma(1-12), Spectral Centroid, Spectral flatness, Zero Crossing Rate, Tempo and Root-Mean-Square(RMS) value. First of all, we will visualize the values of these features on each genre subset to have a sense of how they vary from one genre to the other; then we'll apply the extraction to all audio tracks in the dataset to get all the feature values for the next step: providing the input data to train our classifiers. We will finally extract all the values of the features for each audio track and store them in a csv file (`feature_values.csv`).

3.1 Choose the right features

MFCC - MFCC feature values are calculated by applying a series of transformations to the input signal, in this order: Windowing $\rightarrow$ Discrete Fourier Transform $\rightarrow$ Mel-scaling $\rightarrow$ Log-scaling $\rightarrow$ Discrete Cosine Transform $\rightarrow$ MFCC features. After extracting the mfcc for each audio piece, we then compute the mean value of MFCC(1-13) over all the frames for each track. The plots in Fig. 2 are examples to visually compare the difference between 4 genres: blues, metal, hip-hop and reggae.

Chroma - Chroma is the feature that maps all the pitch information in a given frame to 12 coefficients that represent the 12 semitones of the equally-tempered chromatic scale. Here too for each piece we take the mean values of the coefficients over all the frames; this allows us to visualize the distribution of chromas (1-12) for the 4 genres as seen in Fig. 3.

Spectral Flatness - Spectral Flatness is used to indicate how "tone-like" an audio excerpt is from the peak structure of its spectrum. Low flatness stands for pure note and higher values means noisy sound. In Fig. 4 we can see the average value of Spectral Flatness in different genres. For the classification we use the average and standard deviation over all frames.

Tempo - Tempo is definitely a characteristic feature of different music genres. We choose the estimated tempo for the whole audio piece for each track, extracted with the `beta_tempo` function from LibROSA.

Zero crossing rate - The Zero Crossing Rate is a measure of how many times the audio waveform crosses the zero-axis. In the context of discrete-time signals, a zero crossing is said to occur if successive samples have different signs. It can be interpreted as a measure of the noisiness of a signal. For example, it usually exhibits higher values in the case of noisy signals. It is also known to reflect the spectral characteristics of a signal (the rate at which zero-crossings occur is a simple measure of the frequency content of a signal).

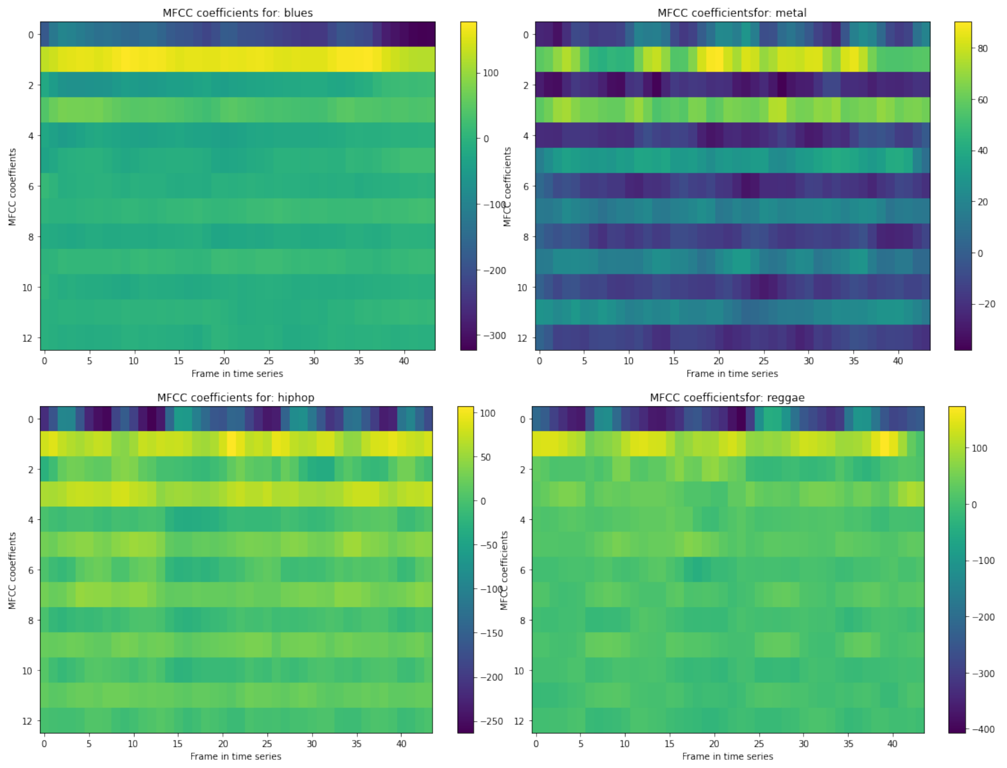

Figure 2: MFCCs for different genres.

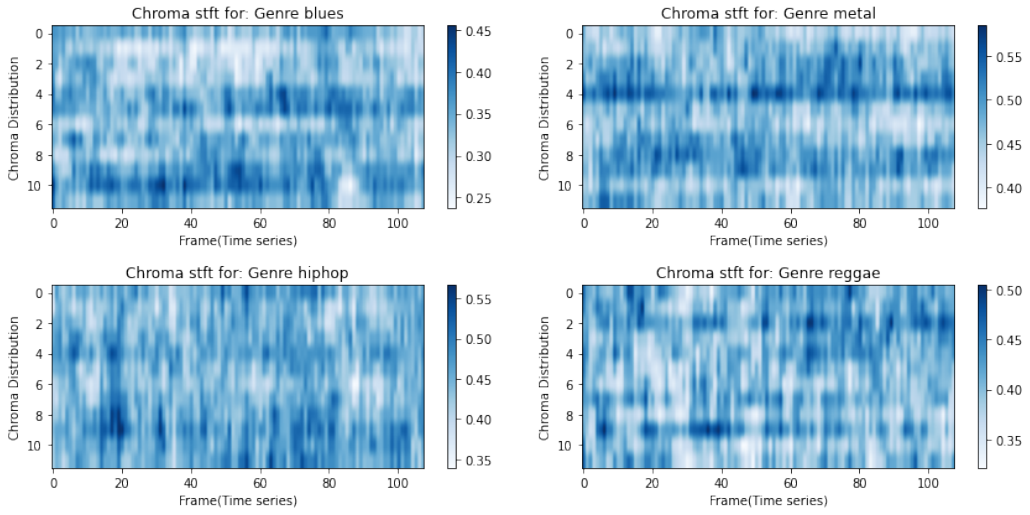

Figure 3: Short-time chromagrams for different genres.

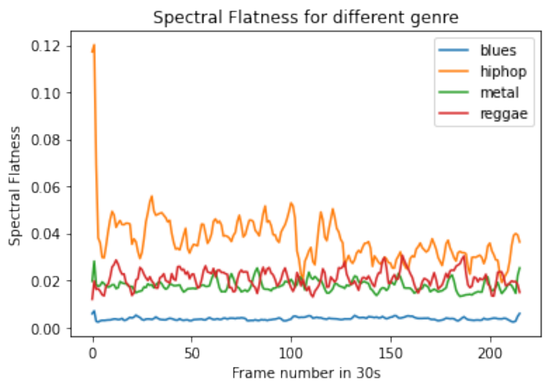

Figure 4: Average spectral flatness for each frame of each genre.

Here too we calculate the feature over each frame and extract the average and standard deviation for each track.

RMS - The Root-Mean-Square value is the effective RMS value of the total waveform. This method helps to detect the "vitality" in a music excerpt by calculating the power of the signal averaged over each frame. As usual, frome the RMS if evry single frame we extract the mean and the standard deviation over the whole track. The scatter plot of the RMS standard deviations vs. mean values for each track, labeled by genre, can be seen in Fig. 5.

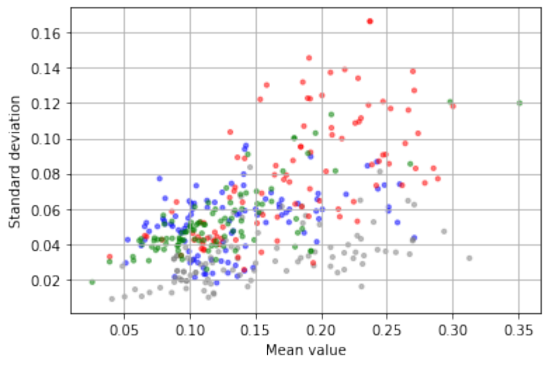

Figure 5: Root Mean Square for the four musical genres (red is hip-hop, green is reggae, blue is blues, grey is metal)

3.2 Feature re-scaling

As the range for each feature value differs, we need to standardize all the values in the `feature_value` table. We can use the `StandardScaler` method provided by sklearn to do this. The formula for standardization process is:

$$x' = \frac{x - \mu}{\sigma}$$

where x' is the rescaled feature value, x is the value before rescaling, μ is the mean value for the feature and σ is the standard deviation. The new feature values are saved in a new file called `re_features.csv`.

3.3 Feature selection

Since we have a multi dimensional feature space, we decide to apply the `VarianceThreshold` method from sklearn to reduce the dimension for further usage. The basic idea of this method is that if one feature does not vary much over different audio pieces, then it would not provide much help in predictions. However, in our case, the final number of dimensions is not reduced by this method, all 34 feature values are valid for the next process.

4 Modelling the Classifier

While there are a lot of supervised learning methods, our approach is to use K-Nearest Neighbour Classifier and Support Vector Classifier. The process to use the classifiers it to pick the right parameters to build the classifier and fit the training dataset. We also use the Grid Search method to optimize the parameters for those two classifiers.

Since we have already split the dataset into 80 % and 20 % to have totally different dataset for training and testing. Thus, we are only working on 320 audio pieces in total as training set for this model fitting section.

4.1 K-Neighbors Classifier

The first classifier we used is a K-Neighbor Classifier as it is swift and commonly used. This classification is computed from a simple majority vote of the k nearest neighbors of each point. The choice of k depnds on the nature of the training set: in general a larger k lets us deal with a noisier set but it's going to output regions with less clear boundaries.

4.2 Support Vector Classifier

The SVC is a multi-layer classifier based on support vector machines. Support vector machines rely on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships (in our case the differences between Blues, Metal, Hip-Hop and Reggae).The basic idea of SVM is finding the best separator function (classifier/hyperplane) to separate the two kinds of objects.
The derived classification parameters are used to discriminate different musical genres.

5 Validation

We used the training dataset for fitting the model for the two classifiers, while we saved the other 20 % from the original dataset as testing set, which would be totally new data to our classifiers. This strategy is applied to prevent from using the same data both in training and testing and achieve unreal high accuracy results as is written in **2.2 Preprocessing Section**.
Secondly, in order to tune our classifiers, we still need to do validation to see the prediction result of music genres and decide how to set the parameters of the classification method we used.
In order to make the best use of the data we have without adding extra data as validation set, we decide to use K-folding Cross Validation method to achieve this goal in the validating process. The general procedure for Cross-validating[3, User Guide 3.1] is:

1. Shuffle the dataset randomly;
2. Split the dataset to K groups;
3. For each group, hold one for testing, other remaining groups for training. Fit the model and evaluate the result;
4. After K runs then we can get a final score in average for all.

6 Optimization

In the KNN Classifier, the parameters we need to tune are k and p, where k is the number of neighbors we check and p defines the p-norm we use to compute the distance[2]. Thus we can write a grid search to check each combination of different parameters.The results are reported in Tab. 2 together with those for the SVC.

Classifier	parameters	best score
KNN	p:1, k:5	0.8375
SVC	'C':10, 'kernel':'rbf', 'gamma':0.001	0.797(+/-0.126)

Table 2: Grid Search to find best parameters

Also, for the SVC, we used the same grid search method but using the method `GridSearchCV` from sklearn. We set the parameter set for searching as following, to look

[2]Possible values of p are chosen in the range 1-11.

for the best combination of kernel and C value. Again, the results are reported in Tab. 2

7 Results and Conclusion

We use the confusion matrix to represent the result of the testing. The diagonal elements of this matrix represent the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, since this indicates many correct predictions. Each row in the matrix is the real genre, while each column is the predicted genre category classified by the model we build. As for classifiers, one can try different approaches. Here a K Neighbour Classifier and a Support Vector Classifier are used.

Testing with KNN

We get the percentage of the accuracy for the prediction result and generate the Confusion Matrix, which can be seen in Fig. 6.

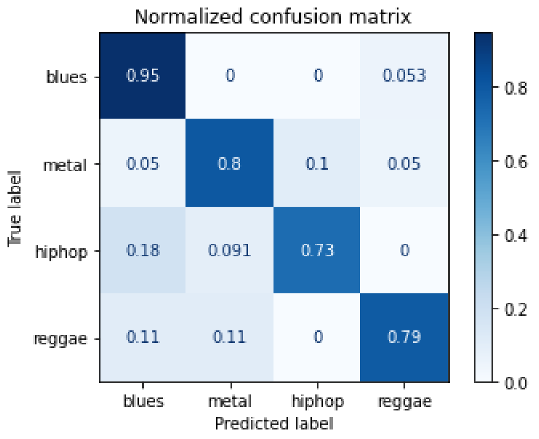

Figure 6: Confusion Matrix for the KNN classifier.

Testing with SVC

We tested the accuracy also employing the Support Vector Machine classifier, since it has good performances for the classification between musical genres (and to make a comparison). Though the results of the confusion matrix are a little lower than our previous (the one testing with knn). The confusion matrix for the test with SVC is shown in Fig. 7.

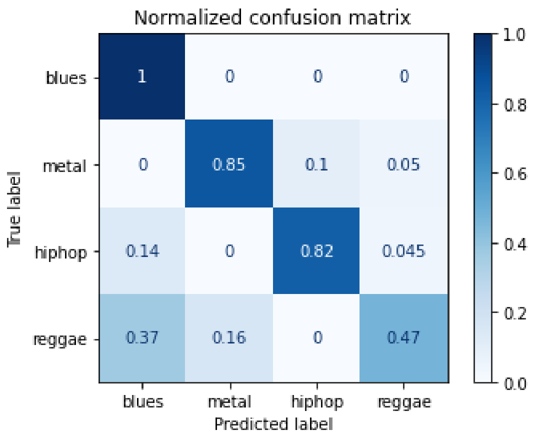

Figure 7: Confusion Matrix for the SVM classifier.

The results show good performances for both the confusion matrices, with a slightly better accuracy for the KNN classifier. We point out, however, that the SVC performed better on Blues, Metal and Hip-hop, but it exhibits a significantly worse result for Reggae: this could be attributed to the flaws in the Reggae dataset we mentioned in Section 2.3.

We can both use the *Accuracy* of the results and also the *F1 Score* to evaluate different performance of the two classifiers we implemented. According to the definition of F1 score in multi-class classifier cases, the current F1 score is calculated as the average score of each music genre. The results are reported in Table 3.

Our results can be compared for example to those in [1]: their accuracy over the entirety of the 10-genre dataset is 61 %, which is lower than ours. More recently with the use of SVCs accuracies of up to 90 % have been obtained over a four genre subsample of the dataset[8]. It must be remarked that, as the authors in [1] write, the accuracy of human recognition of genres is also limited, given the fuzzy nature of the concept of genre itself: a study they cite [9] found a 70 % accuracy of human participants in genre recognition. The accuracy of automatic genre recognition methods, therefore, should be compared to this figure.

Classifier	Accuracy	F1 Score
KNN	0.8125	0.812
SVC	0.785	0.79

Table 3: The final result on Accuracy and F1 score

8 Future Works

As shown in this report, we have succeeded in applying two classifiers to the data set to extract music genre information. Regarding the results we have shown in the last section, we could still have ways to optimize our classifiers with future research.

First, we should enlarge the dataset from the current dataset of 400 pieces of music tracks. We can add in other music pieces of blues, metal, hip-hop and reggae but from different artists and different times. With the enlarged new dataset, we would be able to validate the current methods in a more balanced way. Secondly, we could try other advanced supervised classifiers tuned to the new dataset.

However, with the current and limited dataset, we have already applied the theory concepts seen in class into practice and obtained some interesting results.

References

[1] G. Tzanetakis and P. Cook. "Musical genre classification of audio signals". In: *IEEE Transactions on Speech and Audio Processing* 10.5 (2002), pp. 293–302.

[2] *LibROSA Documentation.* URL: `https://librosa.github.io/librosa/`.

[3] *Scikit-learn.* URL: `https://scikit-learn.org/stable/`.

[4] *NumPy.* URL: `https://numpy.org/`.

[5] *Pandas documentation.* URL: `https://pandas.pydata.org/pandas-docs/stable/index.html`.

[6] *Data Sets. GTZAN Genre Collection.* URL: `http://marsyas.info/downloads/datasets.html`.

[7] Bob L. Sturm. "The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use". In: *CoRR* abs/1306.1461 (2013). arXiv: `1306.1461`. URL: `http://arxiv.org/abs/1306.1461`.

[8] Changsheng Xu et al. "Musical genre classification using support vector machines". In: *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).* Vol. 5. 2003, pp. V–429.

[9] D. Perrot and R. Gjerdigen. "Scanning the dial: An exploration of factors in identification of musical style". In: *Proc. Soc. Music Perception Cognition* (1999), p. 88.