IAM Project

Informatica Applicata alla Musica: Artificial Intelligence Module

Program behaviour

This MATLAB project focuses on the classification of a personal music collection under both normal and noisy conditions. Several techniques are employed for the analysis and classification of musical tracks belonging to three different genres [electronic, jazz, and metal], by extracting features such as **Chromagrams** and **MFCCs**. Clustering algorithms (k-means) and classification algorithms (k-NN and decision trees) have been implemented to evaluate the system's performance, even when noise is added to the test data. The ultimate goal is to analyze the results and determine the effectiveness of different approaches in identifying the musical genre.

ProcAndMrgAudio

```
function procAndMrgAudio(genreName)
```

This function creates a folder called '**noisyTestDir**' if it is not already found in the MATLAB path. This temporary directory is used to store test files to which noise is added, with a signal-to-noise ratio (SNR) of **5dB** relative to the clean files. After creating the directory, the function proceeds to generate these merged-noisy files and moves a copy of each into noisyTestDir. Checks are implemented to prevent array indexing errors when reading the various audio files and merging them with the noise file '**babble.wav**'. After the features have been extracted from the noisy files, the folder is deleted.

normalization

```
function [trainMatrix, testMatrix, mn, st] = normalization(trainMatrix, testMatrix)
```

This function normalizes two matrices, one for training and one for testing, by subtracting the **mean** and dividing by the **standard deviation**, both calculated from the training matrix. The values of '**mn**' and '**std**' will then be reused for normalizing the matrices of features extracted from the noisy test files.

Results analysis

Premise: the selected songs all come from different artists, in order to avoid evaluating only a single "style" from one artist per genre. This approach aims to extract more **general features** related to the genre itself rather than characteristics specific to individual performers.

Introduction

To start, a **window length** of 2 seconds and a **step length** of 0.5 seconds were chosen to ensure good frequency resolution, which is crucial for the features that will be extracted. These parameters may seem unusual for audio analysis, but given the very small training and testing set, it was necessary to compensate by using these values. **Chroma features** represent the tonal content of a musical piece in a condensed form. Essentially, they capture the distribution of sound energy across the twelve pitch classes (the notes C, C#, D, D#, E, F, F#, G, G#, A, A#, B) that make up the Western musical octave, regardless of the specific octave in which these notes occur. **MFCCs** (Mel-Frequency Cepstral Coefficients), on the other hand, represent the general shape of the spectral envelope of a sound over a short time interval, in a way that approximates human auditory perception. They aim to capture the timbre-related characteristics of a sound. From the outset, it can be anticipated that chroma features are less distinctive for signal analysis, whereas MFCCs are more effective in discriminating between different musical genres.

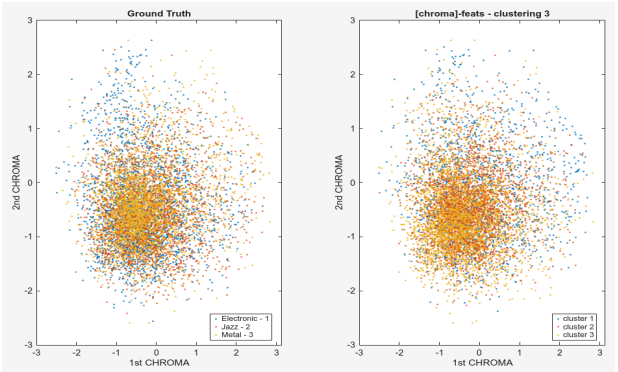
Algorithm

k-means to clusters

K-means clustering is an **unsupervised algorithm** used to divide a dataset into k distinct groups (or clusters) based on the similarity between the data points. The algorithm operates by initializing k (3 in our case) random centroids, then assigning each data point to the nearest centroid and updating the centroids by computing the mean of the points assigned to each cluster. This process is repeated iteratively until the assignments no longer change or a stopping condition is met. K-means is particularly useful for discovering hidden structures within data, reducing dimensionality, or serving as a preliminary step for more complex analyses.

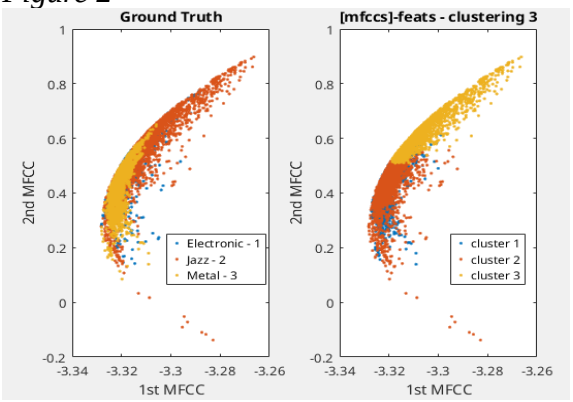
As can be observed, **chroma features** do not provide characteristics that are capable of clearly distinguishing one genre from another. Since they are inherently limited by the concept that defines them (the twelve notes of the tempered scale), they tend to be represented in a **similar way** across all three genres.

Figure 1



To the right, there's a representation of the **Mfccs features**. The first thing that stands out is the **shape** formed by the data when plotted, resembling a **semi-moon**. This arrangement is due to the correlation between the first and second MFCCs: the first MFCC represents the overall energy or basic spectral shape of the signal, while the second MFCC depends on the slope variation of the spectrum (how much energy is concentrated in the high or low frequencies).

Figure 2



Focusing on the analyzed genres:
In electronic music, if the tracks are rich in synthesizers, there might be a recurring spectral signature (a lot of energy in the low range or in specific bands). In jazz, with mostly acoustic instruments, the harmonic tails and timbral color tend to vary less in the first few MFCCs and become more evident in later MFCC components (or in other features). In metal, the distortion and the energy concentration in the mid-high range could introduce some differences, but the first and second MFCCs may still appear similar to those of electronic music due to the dense "energy tails" - a challenge that will also arise when trying to distinguish between these two genres using decision trees (DT) and k-NN classifiers.

For the clustering using **all the features**, I applied the algorithm on the first MFCC and the first column of the chroma features from the full feature matrix; otherwise, the graph would have looked identical to the one based solely on the MFCCs. That said, even after applying a PCA algorithm, I was not able to extract any particularly useful information from this plot, which highlights the difficulty of the task we are dealing with.

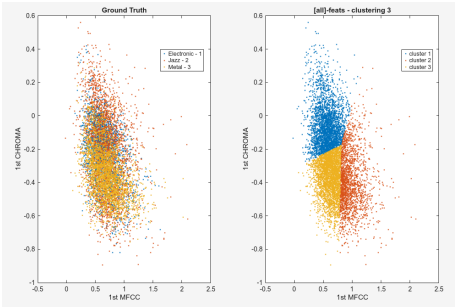


Figure 3

Decision Tree Classifier

The **decision tree algorithm** is a supervised learning technique employed for both classification and regression tasks. It operates by constructing a hierarchical tree structure where each internal node represents a test on a feature, each branch corresponds to an outcome of the test, and each leaf node assigns a class label or a prediction value. At each split, the algorithm selects the feature that provides the optimal partitioning of the data according to a specific criterion. This recursive process results in a decision-making structure that models complex relationships within the data and enables efficient predictions.
Looking at Figure 4, we can immediately notice (as previously anticipated) that the model trained using **chroma features** is not very efficient, especially for the 'metal' genre. A possible explanation is the lack of clearly defined notes, in contrast to 'jazz' and 'electronic', which are rich in harmonic components.

As for the confusion matrix of the model trained on **MFCCs**, some interesting results begin to emerge: the three genres are recognized much more accurately due to the nature of the extracted features.
The most frequently confused genres are 'metal' and 'electronic', as they share similarities in the concentration of spectral energy, particularly in the lower frequencies (both genres feature regular attacks from drums and bass). The confusion matrix representing the model trained on the **full feature set** lies somewhere between that of the chroma and MFCC models, balancing their weaknesses and strengths. By examining the confusion matrices in Figure 4, we could construct a small "map" of proximity among musical genres, with 'electronic' being closer to 'metal', while also highlighting the harmonic connection between 'jazz' and 'electronic'.

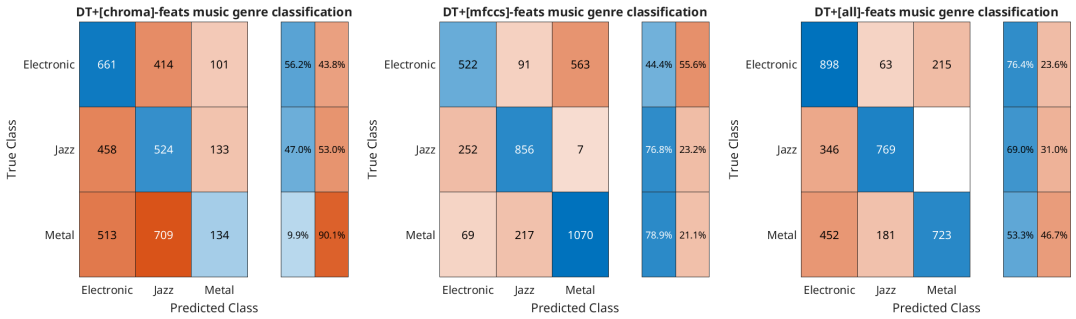


Figure 4

As for the Decision Tree trained under **noisy conditions** (Figure 5), we can observe which genres are more affected by the noise (Jazz) and which ones actually show improved model performance. This is again due to the small size of the training dataset. By making the signals noisier, or more "generic", the model becomes less tied to the specific song and is better able to recognize the genre itself.

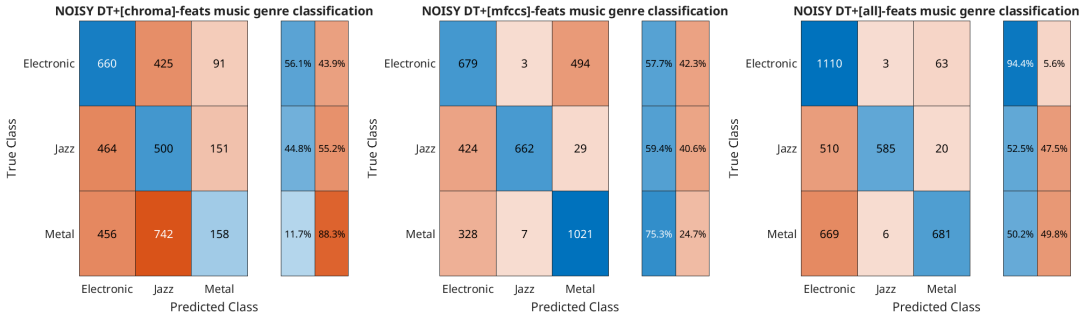


Figure 5

k-NN Classifier

Regarding the k-NN models, three different results are observed. The plot representing the **chroma features** shows a very low recognition rate, which could be due to the high values of the window length and step length. As noise increases, the recognition rate in this plot decreases even further-an expected outcome. On the other hand, the **MFCCs** plot demonstrates significant robustness to noise. In fact, the same phenomenon observed with the DT's recognition algorithm occurs here: the presence of noise actually increases the recognition rate of the different genres. As for the plot containing **all the features**, it initially follows a theoretically consistent trend when looking at the noise-free training set. However, with the introduction of noise, an overlap between the chroma and MFCC features occurs almost immediately, which leads to a sharp decline in the recognition rate.

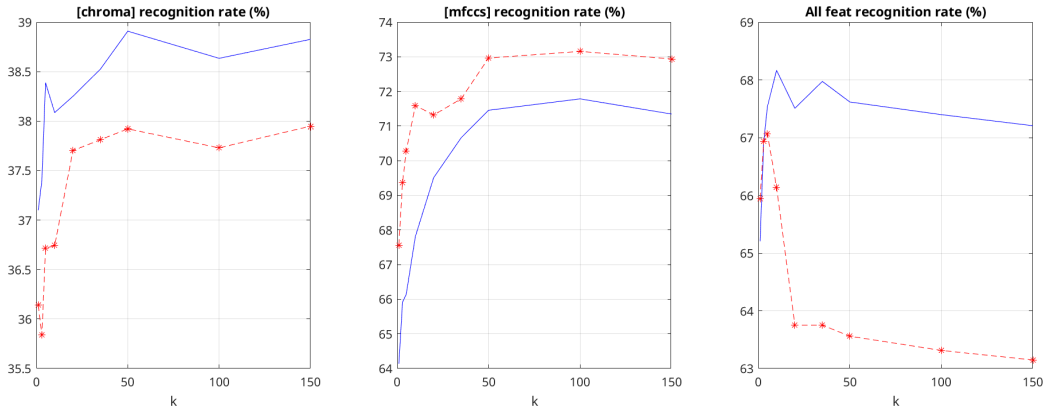


Figure 6

Conclusions

Automatic music genre classification is a challenging task in the field of music information retrieval, particularly when using small datasets. Music genres often exhibit overlapping acoustic characteristics, and individual tracks may incorporate multiple stylistic elements, making clear categorization difficult. The objective proof of this statement is provided by the graphs shown below: the low recognition rate is caused by the significant overlapping among the data.