Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

Dissertação de Mestrado

Unsupervised Automatic Music Genre Classification

Luís Filipe Marques Barreira (aluno nº 28070)

Orientadora: Prof(a). Doutora Sofia Cavaco
Co-orientador: Prof. Doutor Joaquim Ferreira da Silva

Trabalho apresentado no âmbito do Mestrado em Engenharia Informática, como requisito parcial para obtenção do grau de Mestre em Engenharia Informática.

2º Semestre de 2009/10
28 de Julho de 2010

Agradecimentos

À Professora Doutora Sofia Cavaco, minha orientadora, pelo conhecimento que me tem transmitido e pela disponibilidade total demonstrada para com este projecto. Desejo também as maiores felicidades para a família que vai agora acolher um novo membro.

Ao Professor Doutor Joaquim Ferreira da Silva, meu co-orientador, pela sua ajuda fundamental na elaboração deste projecto. O meu sincero obrigado por todas as horas dedicadas a este tema.

Aos meus familiares e namorada pelo apoio e incentivo que me foram fundamentais para conseguir elaborar esta dissertação.

A todos os meus amigos, colegas e professores que, de alguma forma, colaboraram na discussão de ideias e ajudaram na escolha deste tema. Um agradecimento especial aos que comigo fizeram parte do denominado *grupo de áudio*, a todos eles muito boa sorte.

Este trabalho fez parte do projecto Videoflow e foi parcialmente financiado pelo *Quadro de Referência Estratégica Nacional* (QREN), *Fundo Europeu para o Desenvolvimento Regional* e *Programa POR Lisboa*.

Resumo

A música sempre foi um reflexo das diferenças culturais e uma influência na nossa sociedade. Actualmente, a música é catalogada sem obedecer a nenhuma norma universalmente seguida, sendo assim este processo propício a erros. Contudo, o mesmo é fundamental para a correcta organização de bancos de dados que podem conter milhares de títulos. Neste trabalho, interessamo-nos pela classificação por estilo musical.

Propomos uma metodologia capaz de criar *n* grupos de músicas que apresentem propriedades musicais semelhantes (utilizando um processo de aprendizagem) assim como de classificar uma nova amostra de acordo com os grupos criados (utilizando um processo de classificação). O processo de aprendizagem tem como objectivo agrupar diferentes músicas *apenas* com base nas suas propriedades musicais e sem qualquer conhecimento prévio sobre o género das amostras, seguindo assim um método denominado como não-supervisionado. Assim, não é dada ao sistema nenhuma informação acerca do número de estilos musicais representados nas músicas a analisar, seguindo uma abordagem chamada "*Model-Based*" na criação dos conjuntos. As principais propriedades extraídas das músicas estão relacionadas com o ritmo, timbre, melodia, entre outros. A distância de *Mahalanobis* é utilizada no processo de classificação com o intuito de entrar em linha de conta com a forma das nuvens criadas aquando do cálculo das distâncias entre a música e os conjuntos formados.

O processo de aprendizagem proposto obtém uma percentagem de sucesso de 55% quando são submetidas músicas representantes de 11 estilos musicais distintos: *Blues*, *Classical*, *Country*, *Disco*, Fado, *Hiphop*, *Jazz*, *Metal*, *Pop*, *Reggae* and *Rock*. A percentagem de acerto neste processo aumenta significativamente quando o número de estilos é reduzido; com 4 estilos musicais (*Classical*, Fado, *Metal* e *Reggae*), é obtida uma percentagem de acerto de 100%. Quanto ao processo de classificação, 82% das músicas submetidas são classificadas correctamente.

Palavras-chave: Reconhecimento Automático de Géneros Musicais, Propriedades do Som, Classificação Não Supervisionada

Abstract

In this study we explore automatic music genre recognition and classification of digital music. Music has always been a reflection of culture differences and an influence in our society. Today's digital content development triggered the massive use of digital music. Nowadays, digital music is manually labeled without following a universal taxonomy, thus, the labeling process to audio indexing is prone to errors. A human labeling will always be influenced by culture differences, education, tastes, etc. Nonetheless, this indexing process is primordial to guarantee a correct organization of huge databases that contain thousands of music titles. In this study, our interest is about music genre organization.

We propose a learning and classification methodology for automatic genre classification able to group several music samples based on their characteristics (this is achieved by the proposed learning process) as well as classify a new test music into the previously learned created groups (this is achieved by the proposed classification process). The learning method intends to group the music samples into different clusters only based on audio features and without any previous knowledge on the genre of the samples, and therefore it follows an unsupervised methodology. In addition a Model-Based approach is followed to generate clusters as we do not provide any information about the number of genres in the dataset. Features are related with rhythm analysis, timbre, melody, among others. In addition, Mahalanobis distance was used so that the classification method can deal with non-spherical clusters.

The proposed learning method achieves a clustering accuracy of 55% when the dataset contains 11 different music genres: Blues, Classical, Country, Disco, Fado, Hiphop, Jazz, Metal, Pop, Reggae and Rock. The clustering accuracy improves significantly when the number of genres is reduced; with 4 genres (Classical, Fado, Metal and Reggae), we obtain an accuracy of 100%. As for the classification process, 82% of the submitted music samples were correctly classified.

Keywords: Automatic Music Genre Classification, Audio Indexing, Unsupervised Classification

Contents

List of Figures

List of Tables

Acronyms

AFTE Auditory Filterbank Temporal Envelopes. 30

ANN Artificial Neural Networks. 26, 30, 31

ASC Audio Signal Classification. 10, 14

BIC Bayesian Information Criterion. 34

DDL Description Definition Language. 2

DWCHs Daubechies Wavelet Coefficient Histograms. 18, 29

EM Expectation Maximization. 26, 33

ETM-NN Explicit Time Modeling with Neural Networks. 30

FFT Fast Fourier Transform. 7, 8

FT Fourier Transform. 7

GHSOM Growing Hierarchical Self-Organizing Map. 31

GMM Gaussian Mixture Models. 25, 29, 30

GS Simple Gaussian. 30

HMM Hidden Markov Model. 26, 31

KNN K-Nearest Neighbor. 25, 30

LDA Linear Discriminant Analysis. 26, 29, 35

LPC Linear Prediction Coefficients. 18, 30, 31

MBCA Model-Based Clustering Analysis. 33–35, 41, 54, 55, 65

MFCCs Mel-Frequency Cepstral coefficients. 14–18, 29–31, 35, 51

MIR Music Information Retrieval. 17, 51

MP3 MPEG-1 Audio Layer 3. 2, 11

MPEG-7 Multimedia Content Description Interface. 2, 11

NASE Normalized Audio Spectral Envelope. 17, 29, 35

OSC Octave-Based Spectral Contrast. 17, 29, 35

PCA Principal Component Analysis. 31, 32, 36, 40, 42, 45, 65

PDF Probability Density Function. 26

RH Rhythm Histogram. 21, 23

RMS Root Mean Square. 17, 52

RP Rhythm Patterns. 21, 23

SSD Statistical Spectrum Descriptor. 13, 15, 21, 23, 51, 52

STFT Short-Time Fourier Transform. 8, 14, 15, 20, 51, 52

SVM Support Vector Machines. 26, 29, 30

WT Wavelet Transform. 20

XML Extensible Markup Language. 2

ZCR Zero-Crossing Rate. 14–16, 30, 51

1. Introduction

1.1 Introduction and Motivation

In this study we explore automatic music genre recognition and classification of digital music. Music has always been a reflection of culture differences and an influence in our society. Today's digital content development triggered the massive use of digital music. Nowadays, digital music is manually labeled without following a universal taxonomy, thus, the labeling process to audio indexing is prone to errors. A human labeling will always be influenced by culture differences, education, tastes, etc. Nonetheless, this indexing process is primordial to guarantee a correct organization of huge databases that contain thousands of music titles.

These databases are growing everyday and it is now very easy to choose, beyond such offer, to which music, artist or genre we want to listen. Such amount of data needs to be well organized such that the constant updating does not interfere with the ability to generate correct query answers. A correct classification of each music can be the key to maintain a well structured and organized database. Many properties can be used to classify music, although, music genre is, perhaps, the most commonly applied. Often, music can be associated to one or more musical genres. Such genres can be seen as single leafs in an enormous hierarchical tree of genres that is always growing up. Currently, musical genre classification is used in music stores, Internet sites, etc. to organize music in different sections so clients retrieve, without difficulty, their favorite music. In this study, our interest is about music genre organization.

A genre is, by definition, *"a style or category of art, music, or literature"* [37] or *"a class or category of artistic endeavor having a particular form, content, technique, or the like"* [1]. Boundaries between multiple musical genres are not easy to describe. A music can be defined using different criteria such as geographic places, history time, instruments used, etc. For instance, while a certain music can be labeled as an "Indian Music" in Europe, it will, for sure, not be recognized as "Indian Music" in India.

Although everyone understands music like an universal language, the labeling process is not a simple problem. To solve it, most of the time an artist is associated to one musical genre. This can be a possible solution although, is not a strong one. An artist does not have to follow the same music references in his entire career, even a single album can mix multiple genres. So, which line can we follow to associate a correct music genre with a song? Is there any global music genre taxonomy followed by everyone?

Unfortunately, there is no such thing. Starting from the subjective definition of genre, crossing culture differences and ending in human interpretations, a common music genre taxonomy is very hard to achieve. One study related with this problematic is discussed in Section 2.2.1. Despite this universal taxonomy non existence, almost everyone with more or less difficulty has some music knowledge to classify a music by its genre, which is why we can find genre classification in most of places that deal with music (stores, web-sites, etc.).

As mentioned above, digital music is now the most common way to listen to our favorite songs. Beyond a wide range of music formats, we will highlight two in particular: Multimedia Content Description Interface (MPEG-7) and MPEG-1 Audio Layer 3 (MP3). Those two formats have, between other functionalities, a specific way to deal with extra information relative to multimedia content. Next we will briefly explain why these formats are important in dealing with music, and more precisely, why they gain a significant importance in music genre classification.

MPEG-7 is a standard that has the purpose of describing multimedia content. It supports information interpretations which can be read by many applications by simplified mechanisms. This information enriches multimedia content to automatic systems but also to human users. Metadata information follows a structure defined by a Description Definition Language (DDL). DDL is a schema language to represent the results of modeling audiovisual data (descriptors and description schemes). This structure definition is important to allow different applications to manage multimedia information always following the same protocol. To keep metadata information attached with multimedia content, Extensible Markup Language (XML) is used. This format brought a possibility to manage music information in a new approach. More information about this multimedia standard can be found in [33].

MP3 is a widely used digital audio format. Such popularity is due to its power to compress while maintaining a reasonable audio quality. MP3 reduces drastically the size of music files, which become easy to manage, store and share. We will not deepen MP3 specifications but an important source of information about this encoding algorithm can be found in [12]. There is a particularity in those files that is important to highlight, each one is capable to store extra information about music in different tags called *ID3 tags*. These tags contain information like title, artist name, album, year, etc. In newer versions, advanced data can be found, such as music lyric, album art or user comments. Once filled, most commonly used MP3 players are able to present such properties during a reproduction. Generally, *ID3 tags* are filled manually among digitalization process. Misinformation introduced at this stage can compromise search

processes results. A wrong decision made by a "tagger" can quickly spread over the network and become hard to correct being often detected human errors in music genre classification. So why don't we try an automatic labeling process? Can this automation improve reliability in genre classification?

An automatic classification based on good features and using a correct classification algorithm may prevent the occurrence of errors related to manual labeling. As a consequence it will raise information reliability, which will benefit music consumers and also industrial companies. A perfect automatic classification is hard to achieve but we believe, based on related work results, that it is possible to obtain good results in such task.

Our main goal is to create an audio indexing system which is able to respond to this classification issue. An unsupervised automatic music genre classification is intend to be implemented to organize a set of music by their genre *only* based in their audio properties. With an unsupervised approach, new genres can be also classified since we do not restrict the number of genres as a supervised model would do. For that, we propose a learning methodology for automatic genre classification which is able to group several music samples by their music characteristics (learning process). This intends to group the music samples into different clusters only based on audio features and without any previous knowledge on the genre of the samples, and therefore it follows an unsupervised methodology. In addition a Model-Based approach is followed to generate clusters as we do not provide any information about the number of genres in the data set. It would not be plausible to initially give the number of existing genres to our system, this would be the complete opposite of our main goal, which is to identify the number of genres represented in a music data set. Features are related with rhythm analysis, timbre, and melody, among others. As these features represent a large number of dimensions, a redundancy reduction technique is necessary to reduce dimensionality of our extracted data. As a final result, we will have several clusters composed by music samples which present identical audio properties. These groups will be the result of our genre classification, which we can query to know which musics are in a specific cluster or which musics have the same characteristics as our favorite music, for instance.

With this first goal accomplished, we had our indexing problem solved. Then, after the clustering process is completed, a music classification based on the previously created groups makes sense. That way, we proposed ourselves to create a classification methodology to associate a new test music to one of the previously created clusters. For that, Mahalanobis distance is used to attempt to cluster shapes, volume and orientation (since music clusters tend to present

dispersions along reference axes) when calculating distance between the submitted music and a learned cluster. As result, the tested music will be associated with one of the existing clusters.

To test our work, we need a music database. The proposed learning method achieves a clustering accuracy of 55% when the data set contains 11 different music genres: Blues, Classical, Country, Disco, Fado, Hiphop, Jazz, Metal, Pop, Reggae and Rock. The clustering accuracy improves significantly when the number of genres is reduced; with 4 genres (Classical, Fado, Metal and Reggae), we obtain an accuracy of 100%. As for the classification process, 82% of the submitted music samples were correctly classified.

We organize this document in 6 chapters: **Introduction** (Chapter 1), where we discuss the aim of our project and present our motivations; **Fundamental Concepts** (Chapter 2), where we introduce some sound fundamental concepts in Section 2.1, a discussion about music genre recognition in Section 2.2, more precisely concerning genre taxonomy in Section 2.2.1 and automatic music genre recognition in Section 2.2.2; **Methodology and State-of-the-art** (Chapter 3), where related work on automatic music genre recognition is discussed as well as different methodologies that will be adopted in our solution. Section 3.1 introduces the main features tested and explains some properties of those features; Section 3.2 presents the main classification algorithms explored in sound classification; accuracy results are presented and discussed in Section 3.3; an introduction to the redundancy reduction approach is discussed in Section 3.4; and finally a discussion concerning the clustering problematic is shown in Section 3.5; **My Contribution** (Chapter 4) will explain how the learning system (Section 4.1) and the classification system (Section 4.2) are implemented; **Results** (Chapter 5) presents all the important results and details about the submitted tests; and finally, **Conclusion and Future Work** (Chapter 6) where we analyze our system accuracy as well as approach some possible improvements to the implemented solution.

2. Fundamental Concepts

This chapter introduces some fundamental aspects that, we believe, are important for music genre recognition. Section 2.1 discusses basic knowledge about digital sound and Section 2.2 introduces the music genre recognition thematic with a reflection about genre taxonomy and the main steps that have to be taken in order to achieve an automatic music genre categorization.

2.1 Sound

We live in an environment where the air has an important role in the propagation of signals. Air molecules tend to move in a random direction. When an object vibrates, it triggers a movement to the closest air molecules and forces them to follow the movement direction creating a wave. This wave is then propagated in a spherical form. Once it arrives near the human ear, the eardrum (tympanic membrane) receives these vibrations which are then transmitted to the ear structures and may result in audible sound. More information about this topic can be found in [46].

Joseph Fourier [1] derived an important theorem which states that any vibration, including sounds, can be resolved into a sum of sinusoidal vibrations, where each sinusoid corresponds to a frequency component of the sound. The sum of sinusoidal vibrations is called Fourier Series. Any sound can thus be represented by different sinusoids.

This section introduces and clarifies important concepts concerning sound analysis that can be helpful to understand the developed work. Complementary explanations of those concepts can be found in [2, 3, 13, 32, 46]. In the next Section 2.1.1 some concepts around digital sound representation are discussed.

2.1.1 Digital Sound Representation

In Figure 2.1 we show a pendulum movement that is tracing a sinusoid, also called sine wave (particular sinusoid with starting phase = 0°). This wave repeats infinitely the same movement with identical periods (cycles of oscillation), and can easily be described as a continuous oscillatory movement characterized by three important aspects: amplitude, frequency and starting

[1] Jean Baptiste Joseph (1768-1830), Baron de Fourier French engineer, mathematician and physicist best known for initiating the investigation of Fourier series and their application to problems of heat transfer.

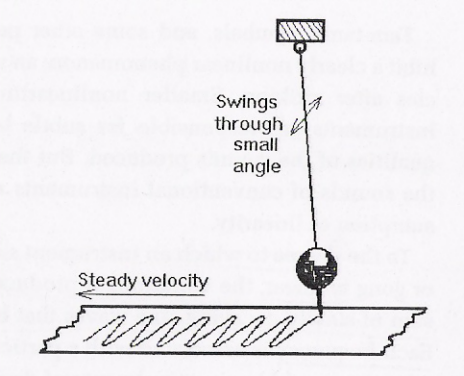


Figure 2.1: A swinging pendulum drawing a sinusoid (sine wave), from [3].

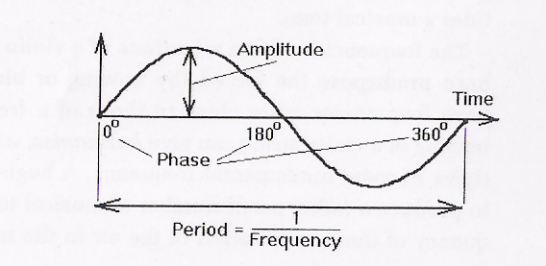


Figure 2.2: A cycle of a sinusoid, with amplitude, phase and frequency, from [3].

phase (Figure 2.2).

Amplitude is often expressed in decibels (dB) and it refers to the amount of vibration displacement. Frequency is the number of cycles that a sinusoid performs per second and is measured in Hertz (Hz). If a sinusoid completes 100 complete cycles in 1 second, then it has a frequency of 100 Hz. Starting phase represents the point in the displacement cycle of a sinusoid in which the sine wave starts. If the wave begins with a positive amplitude, starting phase must be between 0° and 180° (0-π radians). Otherwise starting phase lies between 180° and 360° (π-2π radians). The relationship between amplitude displacement and time can be described by the following equation:

$$D(t) = A\sin(2\pi ft + \theta) \tag{2.1}$$

where A is the maximum amplitude, f is a frequency measurement, t is a time measure, θ represents the starting phase and finally $D(t)$ is the correspondent instantaneous amplitude of the sinusoid at time t.

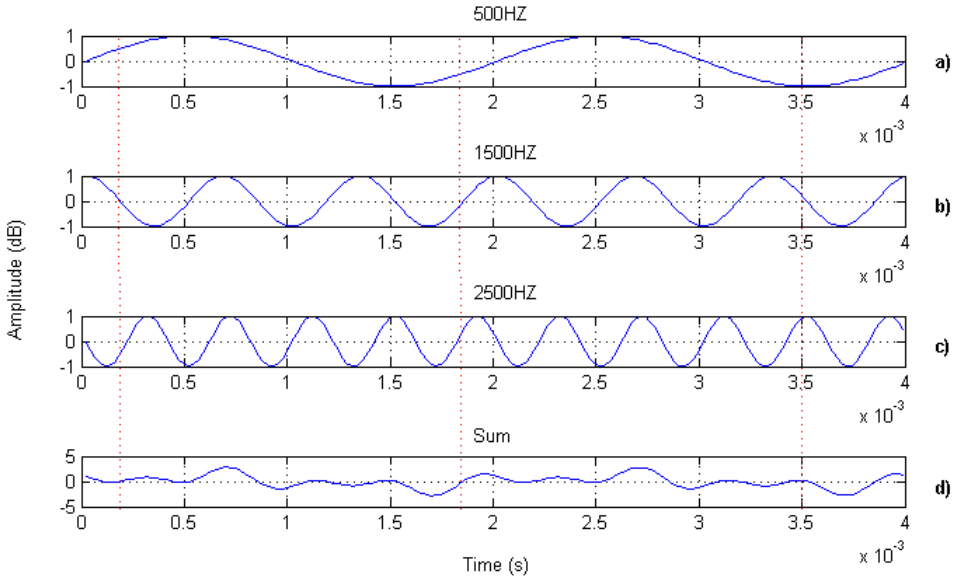


Figure 2.3: (a), (b) and (c) represent sinusoids with frequencies 500, 1500 and 2500 Hz respectively and with initial phases 0, $\pi/2$ and π respectively. Adding those three sine waves, the result is a complex vibration (waveform) that is represented in (d).

As mentioned above, a vibration consists of a sum of one or more sinusoids. A vibration that is composed by several sinusoids is called a *complex vibration*, although, if it consists of only one sinusoid, it is a *simple vibration*. Figure 2.3.a-c shows three sinusoids with frequencies 500, 1500 and 2500 Hz respectively. In the bottom graph (Figure 2.3.d), a complex vibration is plotted representing the sum of the sinusoids of 500, 1500 and 2500 Hz. The representation of such sum is called *waveform*.

A mathematical procedure called Fourier Transform (FT), converts a waveform (time domain) into a spectrum (frequency domain). An efficient implementation of FT that uses discrete-time signals is called Fast Fourier Transform (FFT). With this technique (FFT), the *magnitude spectrum* and *phase spectrum* of a specific waveform can be obtained. Those two elements are enough to totally define a waveform. The magnitude spectrum describes the maximum magnitude of each sinusoid whereas the phase spectrum represents the starting phase of each sinusoid, both in a frequency-domain. Figure 2.4 shows how the waveform from Figure 2.3.d is portrayed in a frequency-domain. The left graphic represents the amplitude spectrum whereas the right one represents the phase spectrum.

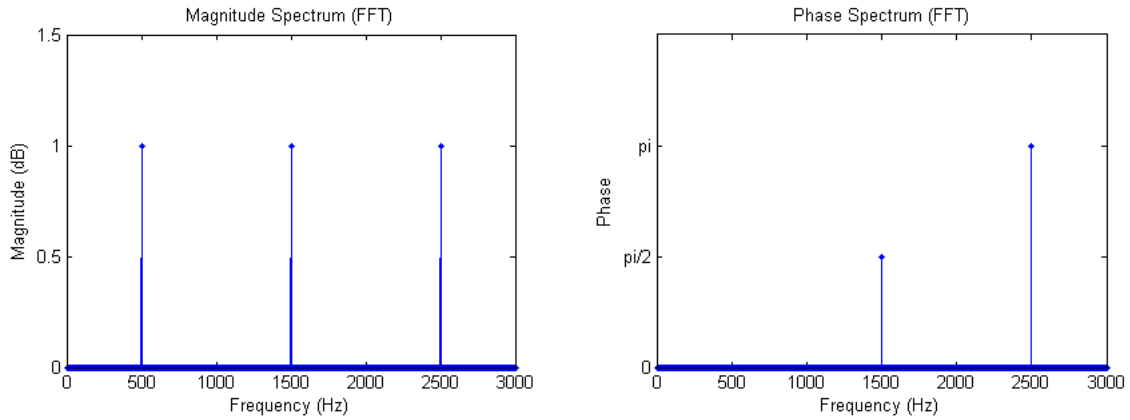


Figure 2.4: Magnitude spectrum and phase spectrum of the waveform in Figure 2.3.d, frequency domain representations.

Sound may also be portrayed using another approach entitled *spectrogram*. While a spectrum is a 2D representation (frequency × magnitude) a spectrogram corresponds to a 3D graphic (time × frequency × magnitude). A spectrogram illustrates the magnitude variation along time and frequency. Figure 2.5 shows a *spectrogram* where higher magnitude values are expressed in dark colors while light colors are intended for lower amplitude values.

The Short-Time Fourier Transform (STFT) uses FFT to compute the time-varying spectra of the signal. The term *short-time* derives from a process called *windowing process* where an input signal is divided into small segments with a specific time duration, usually between 1 ms and 1 second. Figure 2.6 illustrates this process. Since a Hamming function, which is not a rectangular function, is usually chosen to multiply with the signal segment (like illustrated in Figure 2.6), there is a need to overlap those segments to avoid some loss of information. The next step is to extract a magnitude and phase spectrum from each window using the FFT. Thus, STFT gives us a representation of the time-spectral variation of the signal (Figure 2.5).

Three sound representation techniques were presented in this section: waveform, spectrum and spectrogram. Those methods are used in sound analysis each one with a different purpose. For instance, a spectrogram is better if we want to find a specific sound (instrument, voice, etc.) in the middle of a music, while waveforms are useful when looking for a weak reflection following a short sound, and a spectrum representation can be useful to observe the sound's *fundamental frequency*.

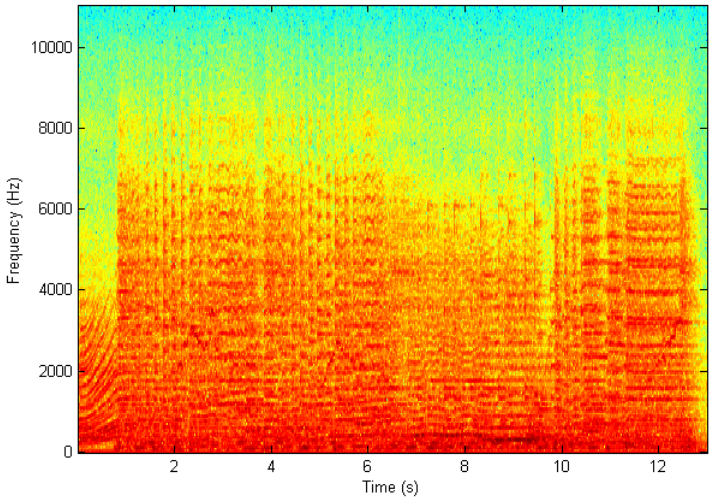


Figure 2.5: A spectrogram of a music.

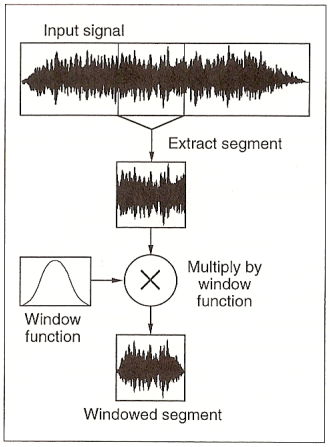


Figure 2.6: Windowing an input signal, from [32].

2.2 Music Genre Recognition

2.2.1 Music Genre Taxonomy

In this section we discuss about music genre taxonomy trying to find some answers about the boundaries that are, or are not, possible to specify. It is a subject that brings many questions due to its subjectivity. In our best knowledge, only one article has been published concerning music genre taxonomy.

F.Pachet and D.Cazaly brought some interesting information about the number of different music taxonomies on the Internet [29]. They analyzed the classification methods of three different sources, which are all well known Internet music retailers (Amazon[2], AllMusicGuide[3] and MP3 web site[4]) and looked for similar structures between them. Those three taxonomies presented huge differences concerning the treatment of relevant information like genealogical hierarchies, geographical information, historical period, etc. and also how they are prepared to receive a new genre. It was easily concluded that the skeletons were extremely different and their combination would be an hard task. Yet, those taxonomies work perfectly when used by a client that is searching for a specific music, the aim is accomplished with more or less effort.

An inconsistency appears when the data is exploited by software (for instance, search mechanism). To solve this problem, F.Pachet and D.Cazaly suggested a new method to create a genre taxonomy following some basic criteria. They reduced considerably the number of different genres applying a stronger connection between each one and introduced new descriptors (like danceability, audience location, etc.). It is clear that a world taxonomy is hard to implement but this solution can be a good starting point to develop an universal music genre classification.

2.2.2 Automatic Music Genre Recognition

Audio Signal Classification (ASC) is a research field that explores different areas such as speech recognition, music transcription and recently speech/music discrimination. The aim of ASC is to label an audio signal based on its audio features, i.e., with a computational analysis of audio properties, to be able to identify to which class the analyzed signal belongs. Two main steps are needed during this process: feature extraction and classification. As part of ASC,

[2]http://www.amazon.com/
[3]http://www.allMusic.com/
[4]http://www.mp3.com/

music genre recognition also needs to follow these two steps.

2.2.2.1 Feature Extraction

Feature extraction is the first step in music genre recognition. This process is very important: automatic genre recognition can only be successful if music samples from different genres are separated in the space formed by the extracted features. When a set of music begins to be analyzed, a lot of values are extracted from each single music, and from that point, these values can be seen as the music files "lawyers". These values will represent each music during all the genre recognition process and, from now one, all complementary information that we could access, from a MP3 file or MPEG-7 file, will not be used. The importance of this process is to guarantee that the values extracted will be enough to distinguish different music genres and emphasize boundaries between different music styles.

A feature is, by definition, "a distinctive attribute or aspect" [37]. To better understand what a feature is, we propose a simple exercise. Imagine that we want to know the average height of a class. To obtain it, we will imperatively need to know the height of each student. Once we have this information, we are able to use it for different purposes such as, average height, maximum, minimum, etc. In this simple example, the feature used is human body height.

Features from audio signals can be related to dimensions of music as melody, harmony, rhythm, timbre, etc. Two main feature groups can be set: computational features and perceptual features. Those groups are exploited in Section 3.1 where most commonly used musical features are described as well as articles in which those feature sets have been tested.

2.2.2.2 Classification

Once the features are extracted, the classification process is the next step. Basically, this step will use the extracted values to define boundaries between different genres, and afterwards it associates each music to one (or more) genre. Three distinct paradigms can be followed at this stage, each one with their own properties: expert systems, unsupervised learning or supervised learning. Before a decision about which paradigm is the right one do adopt, we need to study their differences. The goal can be hit with any of these paradigms, but some are more useful to solve our problem. Let us find out why.

Expert systems demand for a explicit set of rules, which in automatic genre classification is very hard to achieve, if not impossible. As discussed before, a global genre taxonomy does

not exist until now, and we believe that it is an utopia. In our best knowledge, not a single experiment was done using this paradigm in genre classification.

Unsupervised learning is a completely different way to board the question. In this paradigm, clusters are formed from a set of data, i.e., without any taxonomy or rule set, clusters are formed from data based on some similarity measures. In genre classification, it can be very useful since not requiring a taxonomy definition lowers the chances of having music samples that do not fit in any genre. However, with such approach, the created groups may not follow an expected human music genre taxonomy. That is a possible reason for the reduced number of studies in genre classification using this approach. Some examples that use this approach are [31, 35].

Supervised learning is the most used paradigm to classify music samples according to their genre. Manually labeled data is required to train the system in a first stage. During this training stage, relationships between features values and genres will be created. After this period, the system is able to classify new unlabeled data based on the previous training. This paradigm differs from the expert system paradigm since it does not need a complete genre taxonomy, it only requires labeled data that will be used to automatically create the relationships between the features and the categories. This technique is explored in [16–20, 22, 25–27, 30, 38, 42, 45].

These paradigms can be implemented using several algorithms. The most used algorithms in music genre recognition are presented in 3.2. In the next chapter (Chapter 3) the state-of-the-art is discussed. We also analyze results from published articles and methods for music genre classification.

3 . Methodology and State-of-the-art

In this chapter we present all the details concerning features extraction and the classification process used in our implementation. Here we also present the state-of-the-art concerning automatic music genre recognition and discuss about other features and algorithms used in this area. As mentioned in Section 2.2.2, two main steps are needed in automatic genre recognition: feature extraction, which is discussed in Section 3.1, and classification, described in Section 3.2. In Section 3.3, results from several studies are compared based on features and classification used.

3.1 Feature Extraction for Automatic Music Genre Recognition

Feature extraction is the first process to be executed in automatic music genre recognition. Next we describe all the features we have explored as well as some commonly known features that are used in music genre recognition. We grouped the features into three distinct groups: computational features, perceptual features and high-level features. We considered as computational features (Section 3.1.1) the ones that does not present any musical meaning, they only describe a mathematical analysis over a signal. In the opposite, perceptual features (Section 3.1.2) represent, in some way, the perception that a human has listening to a music. Finally, we also focus some features named as high-level features (Section 3.1.3) which are able to represent a music event using a different perspective.

Before a more detailed description of each feature, it is important to note that some of these feature sets have a very high dimensionality and it is more efficient to describe them with less dimensions, always ensuring their "meaning". For this purpose, we use Statistical Spectrum Descriptor (SSD) which consist in seven statistical descriptors: mean, median, variance, skewness, kurtosis, min- and max-value [21]. Using these mathematical values, T.Lidy and A.Rauber presented interesting results. With this representation, we drastically reduce the number of dimensions to seven ensuring that the main statistical properties of the analyzed feature set are maintained. Whenever this property is calculated, we clarify over which set these statistics are processed. These properties can limit the description of the analyzed features since they assume a perfect Gaussian data representation although we choose to use these descriptors since they already present interesting results.

3.1.1 Computational Features

Computational features are extracted from digital audio signals but they do not assume a musical meaning or are related to any human perceptual measure. These features are often used in ASC and they can be important to discern between different musical genres. Many studies on automatic music genre problematic use these type of features [11, 16–18, 18–20, 22, 23, 27, 30, 42, 45].

A set of features called *timbral texture features* [42] handles with several feature sets commonly used. *Timbral features* is composed by: Spectral Centroid, Spectral Rolloff, Spectral Flux, Zero-Crossing Rate (ZCR), Low Energy and Mel-Frequency Cepstral coefficients (MFCCs). The spectral properties can follow two different approaches, calculate values over each window of a STFT (obtain a set of spectral values for each window) or they can be calculated directly over an audio spectrum (and we only have one value for that music). Usually, these values are calculated after a STFT. As long as we obtain a set of values with a significant dimension, means and variances are calculated and used as another feature value. This two additional properties represent the "behavior" of these spectral values. How do we calculate these properties? Let us look in detail to each feature:

Spectral Centroid represents the "center of gravity" of the magnitude spectrum. Next we present how the centroid is calculated:

$$C_t = \frac{\sum_{n=1}^{N} M_t[n] * n}{\sum_{n=1}^{N} M_t[n]}$$

where $M_t[n]$ is the magnitude at frame t and frequency bin n.

Spectral Rolloff corresponds to the frequency R_t which concentrates 85% of the magnitude distribution below it.

$$\sum_{n=1}^{R_t} M_t[n] = \sum_{n=1}^{N} M_t[n] * 0.85$$

Spectral Flux corresponds to the square difference between the normalized amplitudes of successive spectral distributions.

$$F_t = \sum_{n=1}^{N} (N_t[n] - N_{t-1}[n])^2$$

where $N_t[n]$ and $N_{t-1}[n]$ are the normalized magnitude of the Fourier transform at the current frame t, and the previous frame $t-1$, respectively.

ZCR represents the number of times the audio waveform crosses the zero axis per time unit:

$$Z_t = \frac{1}{2}\sum_{n=1}^{N} |sign(x[n]) - sign(x[n-1])|$$

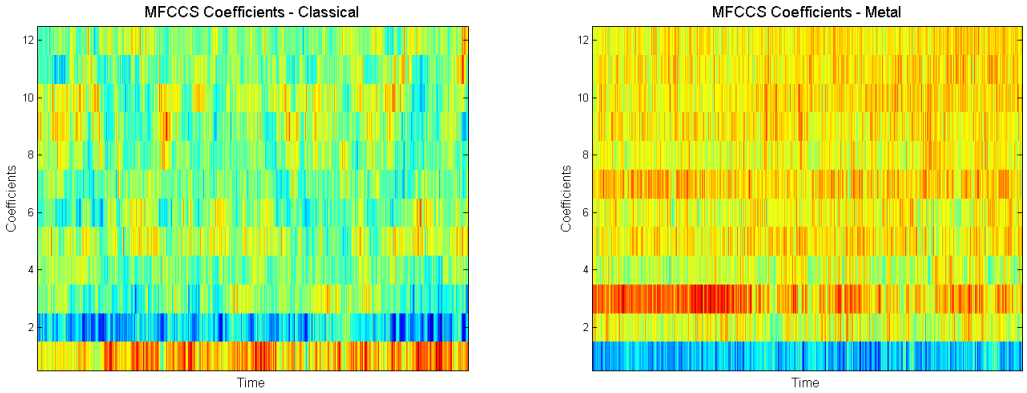
where the *sign* function is 1 for positive arguments and 0 for negative arguments and $x[n]$ denotes the time domain signal for frame t.

Low Energy is the percentage of frames that have lower energy than the average energy over the whole signal. It measures the amplitude distribution of the signal and can be a good feature to distinguish between music genres. As an example, a music which presents long silence periods will have a larger low-energy once compared to a music with few silence periods. This feature is based on the analysis over the spectrogram of a sound.

MFCCs are one of the most popular set of features used in pattern recognition, particularly in speech recognition. Based on the auditive human system model, it uses a Mel-frequency scale to group the FFT bins. After a STFT transformation the log-based magnitude is filtered by a triangular filter bank that is constructed based on 13 linearly-spaced filters (133.33Hz between center frequencies). A 10 log base is calculated and a cosine transformation is applied to reduce dimensionality.

Although this feature set is based on human perception analysis, we classify it as a computational feature since it may not be understood as human perception of rhythm, pitch, etc. It would be acceptable to classify the MFCCs as perceptual as they try to simulate the human auditory perception based on the Mel scale.

As this audio feature is well known we do not discuss minutely each step of the extraction process (please refer to [32] for further details). Instead of using the MFCCs directly, we use the SSD of the MFCCs: once the extraction process is concluded, the SSD is calculated to describe each coefficient. These are the values we use in the classification process. Therefore, this results in a 7 dimension vector for each of the 13 coefficients. Figure 3.1 shows the differences between the MFCCs from a sample from Classical music and a sample from Metal, respectively.



(a) Classical (b) Metal

Figure 3.1: MFCC's feature from 2 music samples, Classical and Metal.

As mentioned above, these presented features were combined in a set called *timbral texture features* in some previous works [16, 20, 22, 42]. This set is used in many studies related with automatic music genre classification.

G.Tzanetakis and P.Cook were the firsts to extract these features from several music files [42]. Later, T.Li and G.Tzanetakis presented a refinement of this article achieving, with the same features but different classification algorithms, better accuracy results [20]. S.Lippens et al. compared accuracies between human and automatic music genre classification also with timbral texture features [22] and they conclude that the use of features derived from an auditory model have similar performance with features based on MFCCs.

A. Koerich et al. also include these features in their study [16]. A particular difference exists in their approach concerning the extraction of features. Features are not extracted from the whole music file or from a single part of the music, they are extracted from three different sections of the music file: begin, middle and end. A. Koerich's team believes that this small detail could make a difference when a music does not behave well in time, i.e., the amplitude variation can be very high.

As described above, MFCCs are a particular feature that belongs to the timbral texture set. M.McKinney combines low-level signal parameters (such as Spectral Centroid, Spectral Roloff, ZCR, etc.) with MFCCs into a 36 and 52 features vectors respectively [27]. D.Pye tried two distinct feature sets in his experience: MFCCs and MP3CEP [30] (the MP3CEP set will not be explored in this document since it presents worst results than those obtained with MFCCs).

T.Li and al. also exploit timbral features in their researches [18, 20].

Apart from the timbral texture features, there are also other useful computational features:

Root Mean Square (RMS) is an approximation of the volume/loudness value:

$$p_n = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} s_n^2(i)}$$

where N is the frame length, $s_n(i)$ denotes the amplitude of the ith sample in the nth frame.

Bandwidth is a energy-weighted standard deviation and it measures the frequency range of the signal [11, 23]:

$$B_t = \sqrt{\frac{\sum_{n=1}^{N}(C_t - log_2(n))^2.M_t[n]}{\sum_{n=1}^{N} M_t[n]}}$$

B_t is the bandwidth of frame t, with C_t as the centroid and n as a frequency bin.

Uniformity measures the similarity of the energy levels in the frequency bands [11, 23]:

$$U_t = -\sum_{n=1}^{N} \frac{M_t[n]}{\sum_{n=1}^{N} M_t[n]}.log_N \frac{M_t[n]}{\sum_{n=1}^{N} M_t[n]}$$

U_t is the uniformity of frame t.

While the features described so far were all explored in this work, there are many more features that are not used by our classifier. Below we present some of those features that can lead to good results in Music Information Retrieval (MIR).

C.Lee et al. extracted MFCCs features combined with two other feature sets [17]:

Octave-Based Spectral Contrast (OSC) represents the spectral properties of a music signal. A spectral contrast can be defined over a spectral analysis, in which peaks and valleys represent harmonic and non-harmonic components of the music respectively.

Normalized Audio Spectral Envelope (NASE) represents the power spectrum of each audio frame which corresponds to the normalized square magnitude of each frequency sub-band.

C.Lee et al achieved good results with a classification algorithm based on modulation of the spectral analysis of these feature sets.

Besides MFCCs features, C.Xu et al. considered another feature set called Linear Prediction Coefficients (LPC) [45]:

LPC were settled based on the idea that a music can be approximated as a combination of past music samples. A set of predictors can be determined minimizing the squared differences between the actual music sample and the linear predictive ones. This feature is commonly extracted for vocal music analysis.

Daubechies Wavelet Coefficient Histograms (DWCHs) was proposed by T.Li et al. and is discussed here since it presents good results in automatic genre classification [18, 19]. In summary, this feature set extraction presents the following steps:

1. Wavelet decomposition of the music signal;
2. Construction of a histogram of each subband;
3. Computation of the first three moments of all histograms;
4. Computation of the subband energy for each subband;

To illustrate the advantages of this feature set, Figure 3.2 shows DWCHs from ten different music genres. In this figure we can see DWCHs features of ten music genres based in G.Tzanetakis and P.Cook database with Blue, Classical, Country, Disco, Hip hop, Jazz, Metal, Pop, Reggae and Rock music [3]. Analyzing the different graphics, we can see that for each music type we have a different representation. For instance, Blues songs show a very different DWCHs graphic from Pop or Rock. If DWCHs are capable to present distinct values for each music genre, they can be very useful in automatic categorization of music.

Besides distinct representations for each music genre, DWCHs also present similar features inside a single music genre. Figure 3.3 shows DWCHs from ten blues songs and all present a similar graphic representation. Thus, it may be possible to achieve a good classification accuracy using this feature.

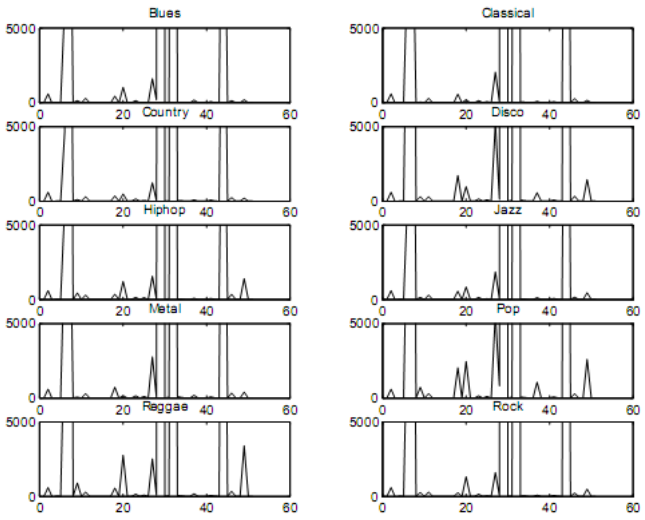


Figure 3.2: DWCHs of 10 music signals in different genres. The feature representation of different genres are mostly different from each other (from [19]).

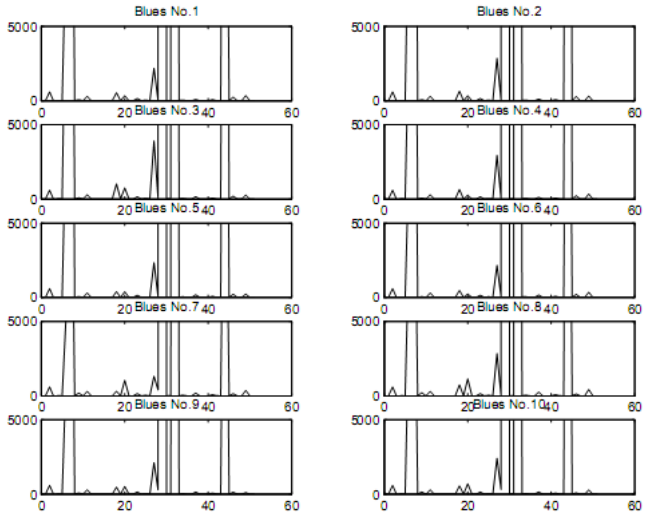


Figure 3.3: DWCHs of 10 blues songs. The feature representation are similar (from [19]).

3.1.2 Perceptual Features

Perceptual features, unlike computational ones, have a musical meaning. They mathematically represent music properties based on the human hearing system. In genre classification, those features gain more importance and are often used as they may simulate some properties that a human would use to classify a music by its genre. Next, we will present the usually tested features in automatic musical genre recognition and describe how they can be extracted from digital audio signals.

Rhythmic Content (Beat)

The rhythmic content of a music can be important in genre recognition. Rhythm has not a precise definition but it can be seen as a temporal description of music, i.e., it contains information such as: the beat, the tempo, the regularity of the rhythm and time signature. This feature set is obtained from the beat histogram analysis, which allows to explore the periodicities of the signal.

The beat histogram is obtained by decomposition of a music signal using Wavelet Transform (WT), an alternative technique to the STFT. From this decomposition, the envelopes of each band are summed and the autocorrelation of the resulting function is calculated. Analyzing the autocorrelation function, the peaks are accumulated into a beat histogram.

From this beat histogram, the usually meaningful information extracted is:

1. Relative amplitude (divided by the sum of amplitudes) of the first and second histogram peak;
2. Ratio of the amplitude of the second peak divided by the amplitude of the first peak;
3. Periods of the first and second peak;
4. Overall sum of histogram;

A repetitive music will present strong beat spectrum peaks at the repetition times revealing both tempo and relative strength of particular beats. Thus it can be used to distinguish between two different kinds of rhythms at the same tempo.

G.Tzanetakis and P.Cook proposed a procedure to extract these features [42]. Figure 3.4 illustrates some of the results they obtained: four distinct beat histogram for four different

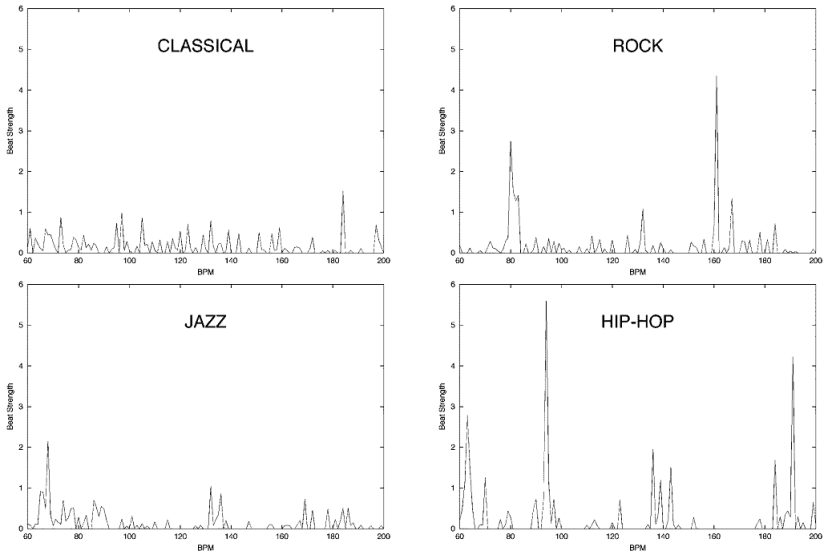


Figure 3.4: Beat histogram examples, from [42].

music genres are presented where is perceptible the differences between each histogram. Other studies also used this feature set [16, 18, 19, 22, 45].

Rhythm Patterns

Rhythm Patterns (RP) are not a complete description of rhythm neither a total pitch describer [21]. This feature represents the loudness sensation for several frequency bands in a time-invariant frequency representation. To obtain RP of one music some psycho-acoustic processing steps are required, and once all transformations are completed, two distinct features can be calculated: SSD and Rhythm Histogram (RH). First, let us start with the RP extraction process. Below we explain the main steps required to obtain this feature.

1. A Short-Time Fourier Transformation is applied to the signal representation. A spectrogram is obtained (Figure 3.6a); (as default values, 512 samples windows are used with a Hanning window function of 23ms and 50% overlap)

2. A Bark scale[1] is applied. The spectrogram is grouped using 24 critical bands. More details in [47] (Figure 3.6b);

[1]Bark scale is an absolute frequency scale which is a measure of critical-band number. In table 3.5 the Bark Scale from Zwicker is plotted.

Bark	lower (Hz)	center (Hz)	upper (Hz)	Bark	lower (Hz)	center (Hz)	upper (Hz)
0–1	0	50	100	12–13	1720	1850	2000
1–2	100	150	200	13–14	2000	2150	2320
2–3	200	250	300	14–15	2320	2500	2700
3–4	300	350	400	15–16	2700	2900	3150
4–5	400	450	510	16–17	3150	3400	3700
5–6	510	570	630	17–18	3700	4000	4400
6–7	630	700	770	18–19	4400	4800	5300
7–8	770	840	920	19–20	5300	5800	6400
8–9	920	1000	1080	20–21	6400	7000	7700
9–10	1080	1170	1270	21–22	7700	8500	9500
10–11	1270	1370	1480	22–23	9500	10500	12000
11–12	1480	1600	1720	23–24	12000	13500	15500

Figure 3.5: The Bark Scale (from [13]).

3. Transform the last spectrogram into a decibel scale (dB).

4. Compute loudness levels through equal-loudness coutours (Phon).

5. For each critical band, calculate the specific loudness sensation (Sone).

6. At this step, a new Fast Fourier Transform is applied to the Sone representation. This new transformation presents a time-invariant representation of the 24 critical bands which shows an amplitude modulation with respect to modulation frequencies that can be seen as a rhythmic descriptor. As humans are not able to perceive rhythm beyond a range from 0 to 43Hz, the considered amplitude modulation is between 0 and 10Hz.

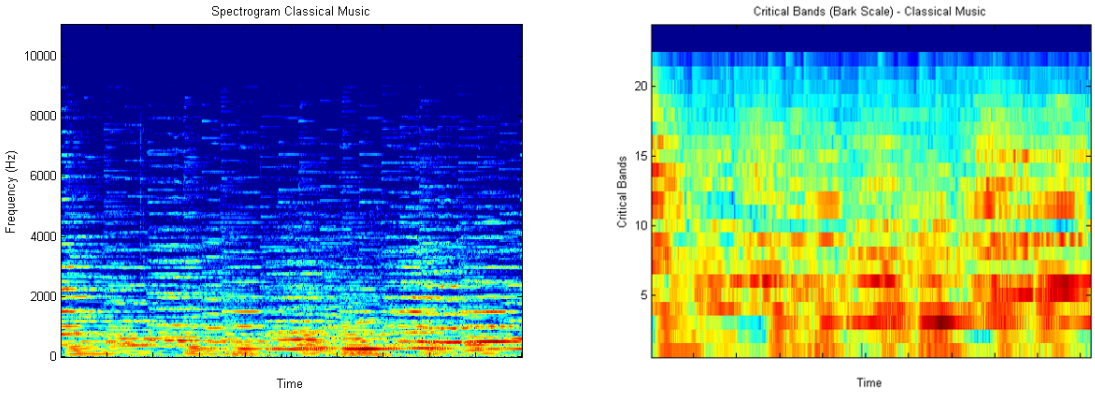
7. Weight modulation according to human hearing sensation and emphasizing distinctive beats from the previous results.

As earlier mentioned, psycho-acoustic phenomena are incorporated in this analysis. In steps 2 to 5 and 7 some studied techniques involving human hearing system are incorporated to increase accuracy results.

Once RP obtained, some properties can be computed as describers of those values.

After step 6 a SSD (Figure 3.7a and Figure 3.7c) feature set is calculated which is able to describe the audio content according to the occurrence of beats or other rhythmic variation of energy. For each one of the 24 bands, 7 statistical moments are calculated: mean, median, variance, skewness, kurtosis, min- and max-value. The resulting feature set is a vector with 168 dimensions.

Unlike RP and SSD extraction, to obtain RH (Figure 3.7b and Figure 3.7d) features we do not store information per critical band, rather, the magnitudes of all frequencies modulation



(a) (b)

Figure 3.6: Two firsts steps of the RP extraction for a Classical music. a) A spectrogram representation. b) A spectrogram representation after a Bark Scale applied.

are summed up to reach a histogram of "rhythmic energy" per modulation frequency. This histogram will present 60 dimensions which reflect frequency range between 0 and 10Hz.

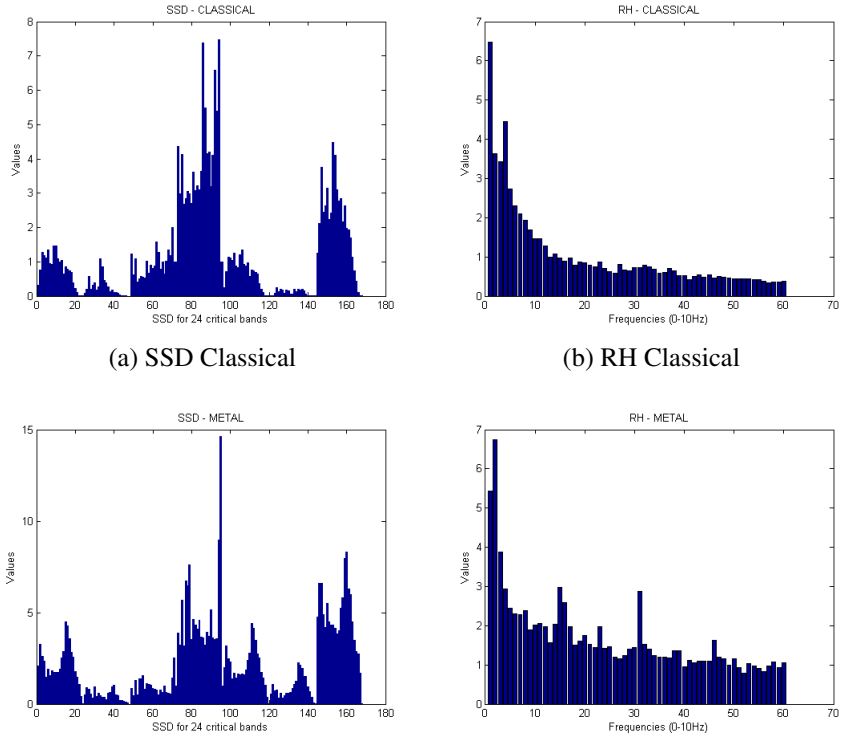
Pitch Content (Melody, Harmony)

Pitch content features are used to describe melody and harmony of a music signal. The extraction method can be simply explained and is based on various pitch detection techniques. The main goal is to obtain a pitch histogram from where pitch content can be extracted.

Pitch histograms are obtained following four main steps:

1. Decompose the signal (FFT);
2. Obtain envelopes for each frequency band and sum them;
3. From that sum, obtain dominant peaks of the autocorrelation function;
4. Accumulate dominant peaks into pitch histograms;

A more precise explanation about pitch histogram can be found in [43]. Once the pitch histograms are obtained, pitch content can be extracted and it typically includes the amplitudes and periods of maximum peaks in the histogram, pitch intervals between the two most prominent peaks and the overall sums of the histograms.



(a) SSD Classical (b) RH Classical

(c) SSD Metal (d) RH Metal

Figure 3.7: Feature set used.

G.Tzanetakis et al. demonstrated how relevant the pitch content feature may be in genre classification [43]. G.Tzanetakis and P.Cook also used this feature set [42] following a multipitch detection algorithm described by Tolonen and Karjalainen [40]. Later, T.Li and G.Tzanetakis refined this work using the same features but with another classification method [20]. As mentioned before, some other papers related with automatic genre classification followed features proposed by G.Tzanetakis and P.Cook being pitch content analyzed too in [16, 18, 19, 22].

This feature set is widely tested in automatic music genre classification and good accuracy results are achieved when it is used. Music melody is an important property for labeling a music by its musical genre therefore pitch content is often extracted.

3.1.3 High-Level Features

Digital audio data can be represented in two distinct form: with symbolic data and audio data. The difference between them concerns the representation of the audio signal and how

it is stored. Symbolic data, or "high-level" representation, stores musical events and parameters while audio data, or "low-level" representation, encodes analog waves as digital samples. Description features can be extracted from both types of representations.

C.McKay brought a different approach to genre classification [25,26]. He defined two types of features, *low-level features* and *high-level* features that are extracted from these representations. Low-level features are based in signal-processing with no explicit musical meaning, e.g., ZCR, RMS, etc. while high-level features are characterized by having a musical meaning, based on musical abstraction, e.g., tempo, meter, etc. Both features can be extracted from low-level data. Still, a better accuracy is hit with high-level features when high-level data is used.

C.McKay tests high-level features using MIDI format, which is a high-level data representation. C.McKay explains the main advantages and disadvantages of using the MIDI format [26]. Some features commonly used in genre classification are unavailable in such format which could reduce the average accuracy classification, although, since symbolic data was an unexploited area concerning musical genre classification, C.McKay developed an application based on high-level features that could bring a new focus into symbolic data representation. Some important musical knowledge such as, precise note timing, voice and pitch are available which opens a door to exploit those new features. At the end, this experience revealed that some losses of usual information, as timbral content, could not be as significant as it may seem.

3.2 Classification Algorithms for Automatic Music Genre Recognition

After the feature extraction phase, we have now data that can be analyzed in the classification process. Two different approaches can be adopted in music genre recognition: supervised recognition and unsupervised recognition (see Section 2.2.2 for more details). Below, we describe the most commonly used algorithms in automatic music genre recognition. As we will be able to see in this section, most of these algorithms follow a supervised approach.

K-Nearest Neighbor (KNN) is one of the most popular algorithms in instance-based learning. It is based on the idea that there are high chances that a point belongs to the same class as its closest neighbors. For a given test feature vector, the K nearest vectors are selected (according to some distance measure, traditionally Euclidean distance) and the test vector is labeled with the class that occurs more often within its neighbors. KNN is a non-parametric classifier. This algorithm is very popular in automatic music genre recognition

and it was used in [17–19, 22, 25, 26, 42, 45]. A detailed explanation about KNN can be found in [10].

Gaussian Mixture Models (GMM) is a widely used algorithm in music information retrieval with the aim to model the distribution of different genres. For each genre we assume a Probability Density Function (PDF) expressible as a mixture of a number of multidimensional Gaussian distributions. The iterative Expectation Maximization (EM) algorithm is typically used to estimate the parameters to each Gaussian component and the mixture weight. This algorithm, compared with others, presented some interesting results in several studies [17–19, 22, 30, 42, 45]

Hidden Markov Model (HMM) is widely used in speech recognition due to its capacity to handle time series data. It is a double embedded stochastic process that handles two distinct processes: one process is hidden and can only be observed over another stochastic process (observable) which produces the time set of observations. This method can be defined by its number of states, the transition probabilities between its states, the initial state distribution and the observation symbol probability distribution. It can be used in supervised systems [34, 38] but also in unsupervised systems [35].

Linear Discriminant Analysis (LDA) aims to find the linear transformation that best discriminates among classes. The classification is performed in the transformed space using some metric (e.g. Euclidean distances). This classification algorithm presents one of the best accuracy results known and it was used in several different articles [17–20]. An extensive explanation of this algorithm can be found in [10].

Support Vector Machines (SVM) is a very popular binary classification method applied in pattern recognition [44]. This algorithm intends to find a hyper plane that separates negative points from positive ones with a maximum margin. Initially designed for binary classification, it can also be used to a multi-class classification applying different approaches such as pair-wise comparison, multi-class objective functions or one-versus the rest. This algorithm is explored in [18–20, 45].

Artificial Neural Networks (ANN) are composed by a large number of highly interconnected processing elements (neurons) that work together to solve a problem. When first proposed, this method was based on human biological learning system [28]. Music genre classification can be done using this method [16, 25, 26, 38].

Below, in Section 3.3, results from different studies are presented. Different combinations between features and classification algorithms are discussed.

3.3 Databases & Results

3.3.1 Supervised System Recognition

Now that the main features and classification algorithms were presented, it is time to discuss about the accuracy results that each analyzed article achieved. For that, we summarize this information in Figure 3.8. It is important to notice that not all tests used the same music databases, which reduces the possibility to compare results equally between all studies. To highlight studies that shared the same music database, a blue background color is used in rows number 2, 6, 10-12.

	AUTHORS	YEAR	KNN	GMM	HMM	LDA	NN	SVM	GS	TreeQ	Best result achieved with:	Algorithm:	Features:	Nr. Musics	Nr. Genres
SUPERVISED SYSTEM RECOGNITION															
1	A.Koerich,C.Poitevin	2005					x				95,97%	NN (Multilayer Perceptron)	Beat+Musical surface Features	414	2
2	C. Lee, J. Shih, K.Yu, H.Lin	2009	x	x		x					90,60%	LDA	MMFCC + MOSC + MNASE	1000	10 (100/genre)
3	C.McKay, I.Fujinaga	2000	x				x				98% (root) - 90% (leafs)	KNN + NN (GA for feat.selection)	High-Level (109)	950	3 (root) + 9 (leaf)
4	C.Xu et al.	2003	x	x	x			x			93,14%	SVM	Beat+LPC+ZCR+MFCC+SpecPower	100	4
5	D.Pye	2000		x						x	92,00%	GMM	MFCC (energy terms + delta)	175	6
6	G.Tzanetakis,P.Cook	2002	x	x					x		61,00%	GMM	Timbre,Beat,Pitch	1000	10 (100/genre)
7	H.Soltau	1998			x		x				86,10%	ETM-NN	Analysis of Temporal Structures	360	4
8	M.McKinney,J.Breebaart	2003		x							74% (static&temporal features)	GMM	AFTE	188	7
9	S.Lippens et al.	2004	x	x					x		58,00%	GS	Beat,FFT,MFCC,Pitch	160	6
10	T.Li,G.Tzanetakis	2003				x		x			71,10%	LDA	Beat,FFT,MFCC,Pitch	1000	10 (100/genre)
11	T.Li,M.Ogihara,Q.Li	2003	x	x		x		x			78,50%	SVM (one-versus-the-rest)	DWCH	1000	10 (100/genre)
12	T.Li,Ogihara	2006	x	x		x		x			78,50%	SVM (pari-wise)	MFCC+DWCH+FFT	1000	10 (100/genre)
UNSUPERVISED SYSTEM RECOGNITION															
13	X.Shao et al.	2004			x						89,00%	HMM	Rhytmic+MFCC+LPC+Delta&Acc	50	4

Figure 3.8: Summary of previous studies in automatic genre recognition. Each row represents one study and for each one we present: authors, published year, algorithms tested, best accuracy result, algorithm used to achieve the best result, features extracted to achieve the best result, number of music samples classified and number of different genres tested. There are twelve studies that used a supervised approach while only one study used an unsupervised approach. Blue background is used to specify the studies that used the same database (rows 2, 6, 10-12).

Five distinct studies used the same database which contains ten musical genres (Blues, Classical, Country, Disco, Hip hop, Jazz, Metal, Pop, Reggae, and Rock) with one hundred excerpts of music for each genre (rows 2, 6, 10-12). These excerpts of the data set were taken from radio, compact disk, and MP3 compressed audio files. They were stored as 22 050 Hz, 16-bit, mono audio files. An important effort was made to ensure that the musical sets truly represent the corresponding musical genre [17–20, 42].

In 2002, G.Tzanetakis and P.Cook created this database to exploit musical genre classification of audio signals [42]. They achieved an accuracy of 61% combining timbre, beat and pitch feature sets with a GMM classification algorithm. In 2003, T.Li and G.Tzanetakis, realized new tests to this database with other classification methods [20]. They achieve a correct classification result in 71,1% of the tested music using the LDA classification algorithm applied to a feature set composed by beat, FFT, MFCCs and pitch content.

Also in 2003, T.Li et al. achieved an accuracy of 78,5% using DWCHs to classify music with an SVM algorithm (one-versus-the-rest) [19]. In a later work, T.Li and Ogihara presented more details and shown that DWCHs can achieve good classification results when combined with FFT and MFCCs feature sets [18]. They also obtained a classification accuracy of 78,5% although, it is reached with a pair-wise SVM algorithm. The information in these two articles is contradictory. The presented results are exactly the same, not only the best accuracy, but the confusion matrix is the same using different features and a different classification algorithm. In the first article [19], we believe that some information is missing since the later article [18] presents a different combination (SVM + DWCHs,MFCCs,FFT) that is more plausible (with the same results).

In 2009, C.Lee et al. achieved really good results with the same database on long-term modulation spectral analysis of spectral (OSC and NASE) and cepstral (MFCCs) features [17]. Their best accuracy is 90,6% achieved using LDA classification algorithm.

These five studies apply several combinations of features and classification algorithms to obtain the better accuracy possible. It is clear that the results have improved along the years and now an accuracy rate of 90,6% has been achieved. Although, other studies with better performances are reported in the past years, a direct comparison should not be made since those studies tested different databases.

A.Koerich and C.Poitevin had a 1000 music titles database for a total of about 50 hours of music. From those titles, 414 samples were randomly selected as data set. These samples represent 2 distinct classes (genres): classical and rock. They achieved an accuracy of 95%

classifying beat-related features and musical surface features (spectral centroid, flux, etc.) with an ANN (Multilayer Perceptron Neural Network) [16].

C.McKay and I.Fujinaga use high-level musical features [25]. For this, they use a different digital format, MIDI. Their music database contains 900 MIDI records and the basic taxonomy includes 3 main genres that are part of the "root genre". 9 other musical genres derive from those 3 root classes. An accuracy of 98% in "root" correct classification and 90% in "leaf" was obtained combining two classification methods (KNN and ANN) to classify 109 high-level features. Those features are detailed in McKay thesis [26].

C.Xu et al. employ a data set with 100 music samples collected from music CDs and the Internet [45]. This data set contains music from different genres as Classic, Jazz, Pop and Rock. Only 15 titles from each genre were used as training samples. An accuracy of 93% was achieved combining several feature sets (beat spectrum, LPC, ZCR, spectrum power and MFCCs) classified with a SVM.

D.Pye's database contains 175 samples representing 6 different musical genres, Blues, easy listening, Classical, Opera, Dance and Indie Rock [30]. Musics were split evenly between the training and test sets. By extracting an MFCCs feature set and classifying with a GMM algorithm, D.Pye achieved an accuracy of 92%.

H.Soltau et al. had a database composed by 3 hours of audio data for four categories: Rock, Pop, Techno and Classic [38]. They achieved a recognition rate of 86,1% using their Explicit Time Modeling with Neural Networks (ETM-NN) approach that combines discriminative power of neural networks with a direct modeling of temporal structures.

M.McKinney and J.Breebaart classified 7 different musical genres (Jazz, Folk, Electronica, R&B, Rock, Reggae and Vocal) with a total of 188 samples [27]. An accuracy rate of 74% was achieved using Auditory Filterbank Temporal Envelopes (AFTE), classified by a GMM algorithm.

S.Lippens et al. studied music genre classification on a 160 music database considering different music genres: classical, dance, pop, rap, rock and "other tracks" [22]. With this database, the best accuracy rate was 58%. This can be compared with a human classification study that achieved 76% classification accuracy. They extracted the same feature set as that used in [42] by G.Tzanetakis in addition to a feature that explored the use of an auditory model. The classification method tested in this article that achieved better results is Simple Gaussian (GS).

3.3.2 Unsupervised System Recognition

X.Shao et al. presented a study using unsupervised classification for music genre recognition [35]. They achieved an accuracy rate of 89% with a 50 music database that represents 4 distinct musical genres (Pop, Country, Jazz and Classic). Rhythmic content, MFCCs, LPC and delta and acceleration (improvements in feature extraction) were applied to a classification performed by a HMM. However, a quite detailed confusion matrix is presented with a perfect accuracy results in Classical music recognition (100%). The worst results are achieved with the Jazz recognition process, only reaching 76% success. Jazz samples were assumed as Pop music 20% of times. The presented accuracy results allow us to conclude that this unsupervised approach can benefit automatic genre classification since there is no need to previously label a data set to obtain good results in music genre recognition.

A.Rauber et al. applied a Growing Hierarchical Self-Organizing Map (GHSOM) (which is a popular unsupervised ANN) to classify psycho-acoustic features (loudness and rhythm) obtaining a hierarchical structuring music tree [31]. They tested their implementation with two distinct music databases, one with 77 samples and another with 359. They presented no confusion matrix or accuracy results although, an analysis of each created cluster is made looking at the grouped samples. Their results are encouraging despite the few features extracted. It is noticeable that music samples belonging to the same cluster present similar rhythms and "sound styles" for human perception. It is also interesting to notice that music from a single band can fall into several clusters, which reveals a plurality of music genres within a music band. Unfortunately, no confusion matrix has been presented turning a comparison with other discussed studies impossible, that is why this paper is not included in Figure 3.8.

3.4 Redundancy Reduction

Principal Component Analysis (PCA) is one of the most popular techniques used to reduce the dimensionality of a data set. In this section, our goal is to provide a simple explanation of this technique and discuss some details that we consider important to understand why we use this algorithm. A detailed study about this statistical tool can be found in [24, 39].

A huge data set, represented in a space with several dimensions, can be hard to work with for several reasons, including those related to machine resources, that is, computational limitations. PCA can be used to reduce the dimensionality of the data set and therefore preparing it for

a computationally easier analysis. PCA is able to reduce the dimensionality of the data by decreasing the data redundancy and not considering the less informative dimensions (that is, dimensions with very low variability). Reduction of dimensionality by PCA is obtained when a high correlation between variables can indicate a huge redundancy in the data. Usually, a high variance in some variables reveals the most important dimensions. While doing this, PCA compresses the data and filters some noise from the data.

Algebraically, PCA returns the basis functions, or axis, of a new space where the data is now represented. These axis are linear combinations of the initial variables (that is, the initial attributes or axis of the initial space). These axis are the uncorrelated final attributes which are presented in a matrix, ordered by descending variances of their values, from the matrix leftmost to the rightmost columns [15]. Since global information is mostly concentrated on the leftmost columns of that matrix, usually, only the first few columns are used to describe the initial data.

3.5 Clustering Problematic

A clustering or grouping process can follow several ways pursuing one common goal: create clusters based on a data set. We will approach two different solutions for such problematic in this section: Partitioning grouping (Section 3.5.1) and Model-Based grouping (Section 3.5.2).

3.5.1 Partitioning Group

In this family we can highlight two commonly known algorithms: *k-means* and *k-medoids* (details in [14]). Once applied to a data set, they both return as result k clusters. This number (k) has to be given by the user.

In *k-means*, a cluster is represented by the centroid of its elements. The algorithm attempts to find a cluster combination such that it maximizes the similarity between each element (music, document, etc.) and the centroid of its belonging group. In the other hand, in *k-medoids* a cluster is represented by one element chosen by the algorithm. Essentially, these two algorithms differ in such a way that while *k-means* gives an equal weight to each element of a cluster to get the centroid, *k-medoids* tends to ignore the outliers from a cluster.

Another peculiarity concerning these algorithms is the use of the *Euclidean* or *Manhattan* distance which confine clusters to have an equal hyper-sphere volume. Neither variances nor

covariances have an important role in the cluster creation which implies the uniform volume to all clusters. Nonetheless, clusters do not always present a hyper-spherical volume, as it can be seen in [7].

Since, the number of clusters must be set a priori by the user and clusters can present non-spherical volumes, we do not use this approach in our solution. As it will be seen later (Section 3.5.2) in order to allow non-spherical volumes, we opted for a different kind of distance measure in our solution, and we also do not set the number of clusters a priori to let the algorithm find the most appropriate number of clusters. As already mentioned in Chapter 1 our goal is to find the number of clusters of a music data set without any initial information besides the music samples. With a partitioning group model, we would be forced to initially define the number of groups that we wanted to create. However, we would like that this information (number of groups) would be automatically given by our implementation.

3.5.2 Model-Based Group

The Model-Based approach aims to answer two distinct questions:

- Which is the more accurate configuration for each cluster?
- How many clusters must be created?

Just like in the partitioning approach, the model-based approach aims to create clusters based on an initial data set, although, it does not know the number of clusters, nor their shape or orientation in a restrict dimension space.

Based on this model, C.Fraley and A.E.Raftery [7] developed a Model-Based Clustering Analysis (MBCA) in which they represent the data by several models. Each model is composed by some clusters and presents different geometric properties from the other models. As we will see in more detail later, a configuration *model-number_of_clusters* more plausible is suggested for each initial data set.

Groups are merged combining the EM algorithm to use a maximum-likelihood criterion and a hierarchical clustering algorithm (see [7] for details). This methodology is based on multivariate normal distributions (*Gausians*). Thus, the density function has the form:

$$f_c(x_i|\mu_c, \Sigma_c) = \frac{e^{(-\frac{1}{2}(x_i-\mu_c)^T\Sigma_c^{-1}(x_i-\mu_c))}}{(2\pi)^{\frac{p}{2}}|\Sigma_c|^{\frac{1}{2}}} \quad , \tag{3.1}$$

Σ_c (model)	Distribution	Volume	Shape	Orientation	Ref.
λI	Spherical	equal	equal	NA	EI
$\lambda_c I$	Spherical	variable	equal	NA	VI
λDAD^T	Ellipsoidal	equal	equal	equal	EEE
$\lambda_c D_c A_c D_c^T$	Ellipsoidal	variable	variable	variable	VVV
$\lambda D_c A D_c^T$	Ellipsoidal	equal	equal	variable	EVE
$\lambda_c D_c A D_c^T$	Ellipsoidal	variable	equal	variable	VEV

Table 3.1: Parameterization of matrix Σ_c in the Gaussian model and their geometric interpretation.

where x_i is an object that belongs to group c which is centered in μ_c. Each group has an ellipsoid volume. Covariance matrix Σ_c is responsible for the geometric property of each group, which is why different models result from different parameterizations of its eigenvalues decomposition:

$$\Sigma_c = \lambda_c D_c A_c D_c^T, \tag{3.2}$$

where D_c is the orthogonal matrix of eigenvectors which determine the orientation of the axis; A_c is a diagonal matrix whose elements are proportional to the eigenvalues of Σ_c, and which determine the shape of the ellipsoid; and the volume is defined by the scalar λ_c. That way, shape, orientation and volume of clusters can be allowed to vary between them, or be constrained to be the same for all groups.

In Table 3.1 we can find several models explored by the MBCA algorithm. By analyzing this table we can see that one of the considered strategy matches with the *k-means* specification, using *Euclidean* or *Manhattan* distances which generate hyper-spherical clusters with the same volume for all clusters ($\Sigma_c = \lambda I$); the $\Sigma_c = \lambda_c I$ model, groups are hyper-spherical although their volume can be variable; with the $\Sigma_c = \lambda DAD^T$, all groups have an ellipsoidal volume and present the same shape and orientation; the $\Sigma_c = \lambda_c D_c A_c D_c^T$ model is the least restrictive model in which shape, volume and orientation can differ in all groups; with $\Sigma_c = \lambda D_c A D_c^T$, only the orientation can change between groups; lastly, when $\Sigma_c = \lambda_c D_c A D_c^T$ all groups present the same shape;

Once all models are created, there is a need to compare them and choose which one is the most accurate. MBCA provides a Bayesian Information Criterion (BIC), that is a measure to compare each pair (*model-nr_of_clusters*) using Bayes factor. A simply comparison is made and the larger the value of BIC, the stronger the evidence for the model. With this technique,

the most reliable model will be chosen and consequently, the number of clusters is defined based on such choice.

As a multi-oriented technique, this approach can be used in several areas. Some interesting results can be seen in [4, 5] concerning document clustering. This algorithm fits perfectly our solution. With MBCA we will be able to, after a feature extraction process, attribute a correct cluster to each music from the data set without any a priori information on the number of clusters. Therefore, MBCA will provide us a finite number of clusters and will tell us which music belongs to each cluster (that is exactly what we were looking for!).

3.6 Conclusion

In this chapter we presented the most relevant and commonly used features in music genre recognition as well as the most studied classification algorithms used for the same purpose. As mentioned above, supervised recognition has much more documentation related to genre classification than unsupervised recognition.

In supervised systems, the high number of studies published until now demonstrates a clearly improvement in accuracy results along the past years. Once again, it is important to understand that results achieved from one study can only be compared with another one if both use the same music database. As mentioned before, five studies based their implementation in the same database, and from those studies, the best accuracy achieved was 90,6% extracting MFCCs, OSC and NASE features [17]. A spectral modulation analysis of each feature set was applied and classification was performed by a LDA algorithm. This accuracy was achieved in a 1000 music database representing 10 different genres.

Concerning unsupervised systems, a fair comparison between studies may be not done since we do not have enough detailed information from the two analyzed papers, however, an encouraging accuracy of 89% was achieved using this approach with few features, which is a very good indicator from the potential that this method can achieve in automatic music genre recognition.

In a global analysis, we can highlight that in several studies some feature sets are usually present, such as: beat, pitch, timbre. This is not surprising. A human can easily distinguish a music genre based on some properties as: the music rhythm (beat content), different types of sounds, instruments heard (timbre) and music melody (pitch). Another very popular feature set used is MFCCs since it is based on the human auditory model.

In this chapter we also introduce the redundancy reduction problematic. A PCA based technique is adopted to reduce the dimensionality of our data (Section 3.4 explains the main ideas behind such technique).

Finally, we approach the clustering problematic analyzing two different models: Partitioning Group (Section 3.5.1) and Model-Based Group (Section 3.5.2). We discuss the main differences between each methodology and explain why we follow a Model-Based approach. Also in that section, we introduce the Model-Based algorithm adopted.

4. My Contribution

Now that all the important concepts concerning our problem were introduced, this chapter will present a detailed explanation of the proposed solution. There are two main goals on this implementation: on the one hand, we want to cluster music samples (from the training data) according to their genre, and, on the other hand, we want to be able to classify new (test) music samples. Clustering music is a learning process which is able to organize them according to their audio features. Once the clustering is done, the classification of new test samples is done according to the clusters learned during the learning phase. This chapter is organized in two sections: one that explores how clustering of the training samples is done (Section 4.1) and another that discussed the classification of the test samples (Section 4.2).

4.1 Learning Process

The learning process aims to organize several music samples into clusters without any initial information besides the feature set values of these samples. Let us try to clarify the reader about which steps are needed in the learning process. Figure 4.1 plots the system organization where different transformation steps and data representation can be clearly identified. To easily understand how it works, we will follow the sequence shown in the referred figure and explain it step-by-step. We use rectangles for input/output values while ovals represent computation processes.

1) The system extracts several features (previously selected) from the music samples in the data set;

2) As a result of step 1 we have a data set matrix (**M**) in which each line characterizes one music sample while columns represent a specific feature;

$$\mathbf{M} = \begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,F} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,F} \\ \vdots & \vdots & \ddots & \vdots \\ m_{N,1} & m_{N,2} & \cdots & m_{N,F} \end{pmatrix},$$

where $m_{m,f}$ is the value of the fth feature for music sample m.

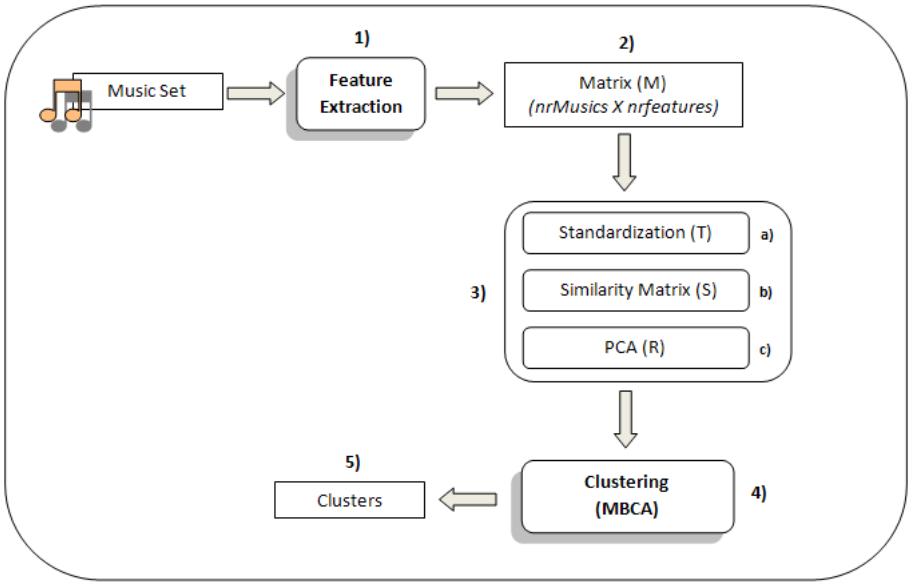


Figure 4.1: Learning process illustration where rectangles represent data (vectors or matrices) while oval boxes are used for transformations/processes.

3) Once our data set matrix is obtained, some transformations need to be performed.

(a) Standardization - This process aims to scale each feature (that is, each column of the data set matrix **M**) such that the features have equal variances (i.e., importance). For this purpose, it creates a new matrix **T** that has the same dimension as matrix **M**.

$$\mathbf{T} = \begin{pmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,F} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,F} \\ \vdots & \vdots & \ddots & \vdots \\ t_{N,1} & t_{N,2} & \cdots & t_{N,F} \end{pmatrix}$$

where $t_{m,f}$ stands for the value of the standardized value of music m for feature f. To calculate each new cell of **T** we apply the equation below:

$$t_{m,f} = \frac{m_{m,f} - mean(m_{.,f})}{\sqrt{var(m_{.,f})}}, \tag{4.1}$$

where $m_{.,f}$ is the fth column of matrix **M**. Variance is obtained from:

$$var(m_{.,f}) = \frac{1}{N}\sum_{i=1}^{N}(m_{i,f} - mean(m_{.,f}))^2, \tag{4.2}$$

and the mean from

$$mean(m_{.,f}) = \frac{1}{N}\sum_{i=1}^{N} m_{i,f}, \tag{4.3}$$

where N is the number of music samples.

(b) Correlation Matrix - At this step, matrix **S** (the similarity matrix) is calculated from matrix **T**. **S** presents different dimensions from **T**. Since it is a correlation matrix, **S** is a symmetric matrix, where the number of lines and columns is the number of music samples in the data set. Each cell of this matrix represents the similarity between two music samples. With such similarity matrix, it is expected that music titles which belongs to the same genre present, higher correlation values between them. Based on the **T** matrix we next present the calculation of the similarity for a generic cell of this matrix:

$$s_{i,j} = \frac{cov(i,j)}{\sqrt{cov(i,i)} * \sqrt{cov(j,j)}}, \tag{4.4}$$

where i and j refer to music samples. **S** can be seen as:

$$\mathbf{S} = \begin{pmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,N} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ s_{N,1} & s_{N,2} & \cdots & s_{N,N} \end{pmatrix}$$

The covariance between samples, $cov(i,j)$, is obtained based on the equation:

$$cov(i,j) = \frac{1}{F-1}\sum_{f=1}^{F}[t_{i,f} - t_{i,.}] * [t_{j,f} - t_{j,.}], \tag{4.5}$$

where F is the number of features used (that is, the number of columns in **M** and **T**) and $t_{i,.}$ is the ith line of matrix **T**.

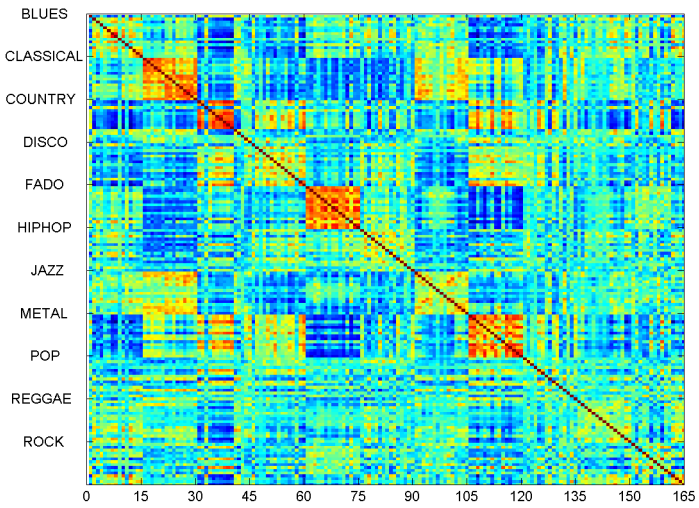


Figure 4.2: Similarity matrix from 165 music titles (11 different genres).

Figure 4.2 plots an image of correlation matrix **S** for 165 music samples (from 11 different genres). The symmetry of the matrix can easily been confirmed in the figure. The color spectrum in the figure goes from dark blue (for lower values) to red (for higher values) and brown (for the highest value, which is 1): the diagonal (brown) has the maximum correlation value (1).

This similarity matrix can be seen as representing a set of music samples by a special set of attributes: where each attribute characterizes the similarity of the music sample to another music sample in the data set.

(c) PCA - Now that our similarity matrix is calculated, it is important to built a matrix where objects are characterized by a reduced number of final attributes, so it can be submitted to a classification algorithm. Since the number of samples in the data set is usually high, the similarity matrix **S** usually has a high number of features (i.e., attributes). Therefore, there is a need to reduce the number of features (that is the dimensionality of the data). In the example given in Figure 4.2, there are 165 attributes, which are too many attributes to characterize the same number of objects. A technique based on PCA will be used to reduce dimensionality.

Matrix **S** may be seen as a representation of several music samples characterized by a set of attributes and, since it presents strong correlations between dimensions, a

reduction in dimensionality can be applied such that we ensure that the main data properties are maintained. As a symmetric matrix, $\mathbf{S}$ can be described as $\mathbf{S} = \mathbf{P\Lambda P^T}$ with $\mathbf{P}$ orthogonal ($\mathbf{P} = [\mathbf{e_1}, ..., \mathbf{e_n}]$ is the matrix of normalized eigenvectors of $\mathbf{S}$) and $\mathbf{\Lambda}$ diagonal (more details in [15]). In $\mathbf{\Lambda}$ we have the eigenvalues of $\mathbf{S}$ $\lambda_1, ..., \lambda_2$ such that $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n \geq 0$. Since $\mathbf{\Lambda}$ is symmetric, $\mathbf{\Lambda} = \mathbf{\Lambda}^{1/2}\mathbf{\Lambda}^{1/2}$ and $\mathbf{\Lambda}^{1/2} = (\mathbf{\Lambda}^{1/2})^T$ so:

$$\mathbf{S} = \mathbf{P\Lambda^{1/2}\Lambda^{1/2}P^T} = \mathbf{P\Lambda^{1/2}(\Lambda^{1/2})^TP^T} = \mathbf{P\Lambda^{1/2}(P\Lambda^{1/2})^T} = \mathbf{QQ^T}, \tag{4.6}$$

where

$$\mathbf{Q} = \mathbf{P\Lambda^{1/2}}. \tag{4.7}$$

The lines in matrix $\mathbf{Q}$ represent the music samples while the columns represent new uncorrelated attributes. See [6] for details about this PCA-based technique.

In order to obtain matrices $\mathbf{P}$ and $\mathbf{\Lambda}$, we used the Matlab function named PCACOV. Once the axes of the new space are calculated, a decision concerning the number of dimensions (columns of $\mathbf{Q}$ matrix) to use, needs to be made. There are two known criteria: one is selecting the first k columns of this matrix such that those k columns contain at least 95% of the total variance given by all columns; the other criterion is based on rejecting the columns for which the corresponding variance (given by cells in $\mathbf{\Lambda}$) is lower than 1. We used this second criterion as it provided a more reduced number of columns. With this technique, we were able to drastically reduce our matrix dimensionality from 165 to 7 dimensions with the data set mentioned above. In other words, we have obtained a new matrix $\mathbf{R}$ which is a copy of the 7 leftmost columns of matrix $\mathbf{Q}$. This way we built a k-dimension space in which music samples will be represented. Now, we are able to submit the resulting matrix $\mathbf{R}$ to the clustering stage as well as to the classification algorithm.

4) Clustering using MBCA - We adopted MBCA to perform clustering. As it was mentioned in section 3.5.2, MBCA does not need to initially know how many clusters exist as well as their characteristics (shape, volume and orientation).

We use the *MCLUST package* [8, 9], developed in R[1] for that purpose (as our system is

[1] R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. More details in http://www.r-project.org/.

developed in *Matlab*, an integration was required to use this package). Once configured, the *mclust* function is called with matrix **R** as input. The result obtained is a vector which tells us which music sample belongs to which group (cluster).

5) Once all the previous steps are concluded, the final result is a vector with the group id for each one of the submitted samples. That way, we are able to organize our clusters and identify which music belongs to which cluster.

As it turned out, this approach suits our goal. In this learning/training process, by combining the techniques discussed above, we do achieve an audio indexing system which is able to group music files only based on some audio features. So, after this learning phase, it is possible to query the system in order to know which group is associated to a given music (provided the music is in the data set), as well as what are the other samples of the same cluster. This clustering process answers our initially proposed problem, to create an audio indexing solution based on music genre classification (Chapter 1). Later, we will analyze some results in order to know how accurate this system can be (Chapter 5).

Even though initially we only proposed to develop a musical genre indexing system that was able to index the music samples in a data set, as we believed that we could upgrade this system, we actually went further and developed an extension to this unsupervised clustering approach: once clusters have been built, a classification process might be developed in order to classify new test music samples according to what was learned in the learning phase. We present the details of this classification process in next section (Section 4.2).

4.2 Classification Process

In this section we explain how we are able to classify a music sample (not included in the training set) after the learning process has been completed. An illustration of the main steps of such a classification system is plotted in Figure 4.3 and discussed below. It is important to mention that this process is not autonomous from the clustering process (Section 4.1) since it needs to use some results obtained from the PCA-based technique previously processed, as well as the T matrix generated in the clustering phase.

Let us explore, in more detail, each step of this system. To easily understand how it works, we will follow the sequence shown in Figure 4.3 and explain it step-by-step. In the figure we

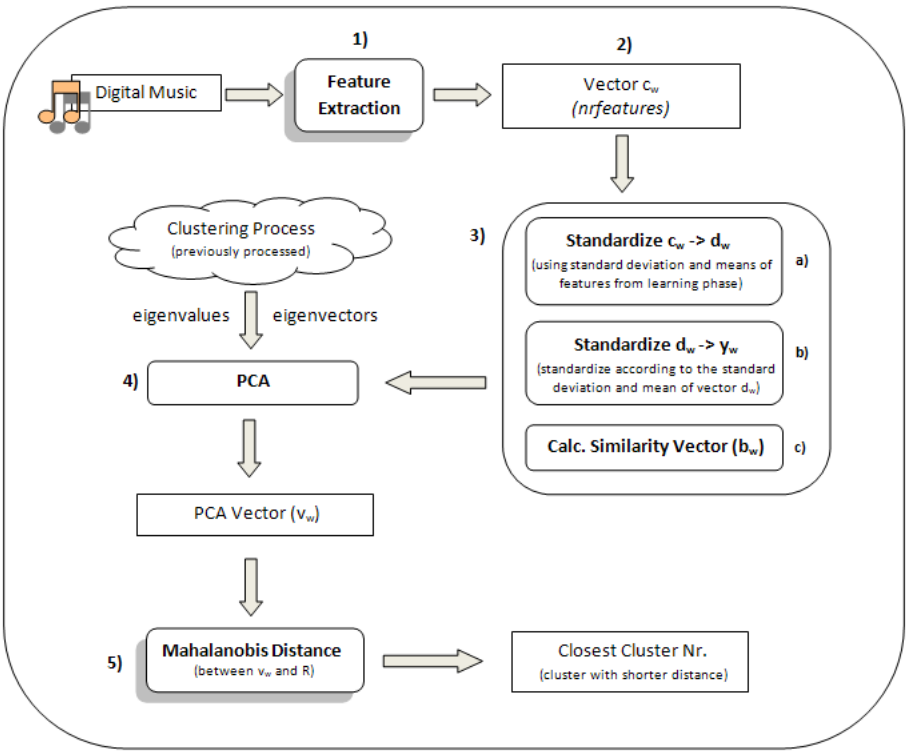


Figure 4.3: Classification process illustration where rectangles represent data (vectors or matrices) while oval boxes are used for transformations/processes.

use rectangles for input/output values while ovals represent computation processes.

1) Based on a new music sample w, the same initial feature set as that extracted in the learning process will be here used to characterize the music (just one at a time) we want to classify. This extraction follows exactly the same steps adopted during the clusters creation.

2) The previous step outputs a vector (with as many dimensions as the number of extracted features) where each cell contains the value for one of the features for music w. We call this vector $\vec{c_w}$, where $\vec{c_w}^T = [c_{w,f_1}, ..., c_{w,f_F}]$. Each cell of vector $\vec{c_w}$ represents the value of one feature in the range $f_1, ..., f_F$ of music w. If you remember matrix $\mathbf{M}$ (from Figure 4.1), it is nothing else than a matrix in which each line is a music representation as vector $\vec{c_w}$.

3) Vector $\vec{c_w}$ needs to be transformed into a new vector that represents music w in the k-dimensional space built in the clustering process.

Some values obtained during the learning phase need to be used to process the next few steps, which transform vector $\vec{c_w}$ into the new vector:

(a) Standardize $\vec{c_w}$ - As it was done in the learning phase, this transformation aims to set equal variances (importance) and scale to each column (feature) of the vector $\vec{c_w}$. Despite $\vec{c_w}$ had only one value for each feature, standardization will be processed according with all the other music features extracted during the learning phase. Thus, it creates a new vector $\vec{d_w}$ that has the same dimension as $\vec{c_w}$. To standardize $\vec{c_w}$, means and standard deviation calculated during the learning process are loaded and integrated such that each cell of the new $\vec{d_w}$ vector is obtained with an equation similar to equation 4.1 used above:

$$d_{w,f_i} = \frac{c_{w,f_i} - mean(m_{.,f_i})}{\sqrt{var(m_{.,f_i})}}, \tag{4.8}$$

where mean and variance results from equation 4.3 and 4.2 respectively. Vector $\vec{d_w}$ can be represented as $\vec{d_w}^T = [d_{w,f_1}, ..., d_{w,f_F}]$.

(b) Standardize $\vec{d_w}$ - The next step aims to calculate a similarity vector between music w and all the training set. For that, we could use equation 4.4 which is a calculation of a correlation. However, since a correlation using non-standardized variables is equivalent to a covariance using the standardization of those variables, for reasons of computational weight, we followed this last option to get the same results. So, once we have vector $\vec{d_w}$, the referred standardization has to be computed to obtain a new vector $\vec{y_w}^T = [y_{w,f_1}, ..., y_{w,f_F}]$. It uses mean and standard deviation of $\vec{d_w}$, such that each cell is given by:

$$y_{w,f_i} = \frac{d_{w,f_i} - mean(d_{w,.})}{\sqrt{var(d_{w,.})}} \tag{4.9}$$

(c) Calculate Similarity Vector $\vec{b_w}$ - In order to obtain a similarity vector, we need the information given by the similarity matrix **S**, which may also be given by another matrix **Z** – later, in this section, we will prove the need of this matrix – such that each column of **Z** is a vector $\vec{z_m}$ where $\vec{z_m}^T = [z_{m,f_1}, ..., z_{m,f_F}]$ represents the training set sample m using standardized values. In other words, each of these standardized

values z_{m,f_i} is calculated by:

$$z_{m,f_i} = \frac{t_{m,f_i} - mean(t_{m,.})}{\sqrt{var(t_{m,.})}} \tag{4.10}$$

where t_{m,f_i} is given by 4.1.

At this step we do need to relate our music w, now represented by $\vec{y_w}$, with the samples analyzed during the learning process, that is the training set samples. To do that, vector $\vec{b_w}$ represents the similarity vector between $\vec{y_w}$ and each sample in the training set:

$$\vec{b_w}^T = \frac{1}{F-1}\vec{y_w}^T\mathbf{Z} \tag{4.11}$$

where F is the number of features extracted.

4) PCA - Now, by using the information obtained by the PCA-based technique from Section 4.1, that is with $\mathbf{\Lambda}$ and $\mathbf{P}$, which reflect the uncorrelated dimensions of the training space, we can translate the music w, by using similarity vector $\vec{b_w}$, into a vector $\vec{u_w}$:

$$\vec{u_w} = [u_{w,1}, ..., u_{w,N}] = \vec{b_w}^T\mathbf{P}\mathbf{\Lambda}^{-\frac{1}{2}} \tag{4.12}$$

where N is the number of samples used in the learning phase. As focused in Section 4.1, with the data set we used to test our approach we only use $k = 7$ dimensions, so only the k leftmost cells of $\vec{u_w}$ will be used to obtain a final vector $\vec{v_w}$ that represents music w in the k-dimensional space built in the clustering process. In other words

$$\vec{v_w} = [v_{w,1}, ..., v_{w,k}] \text{ where } v_{w,i} = u_{w,i} \text{ for } i = 1, \ldots, k \tag{4.13}$$

Next we will prove why $\vec{v}$ is the translation of music w in the k-dimensional learned space.

Proof. Let us suppose we want to classify a sound, say the first music of the training set, which is available in $\vec{z_1}$, the first column of matrix $\mathbf{Z}$. So b_1:

$$\vec{b_1}^T = \frac{1}{F-1}\vec{z_1}^T\mathbf{Z} \tag{4.14}$$

being $\vec{z_1}^T = [z_{1,1}, ..., z_{1,F}]$ then

$$\vec{b_1}^T = [s_{1,1}, ..., s_{1,N}] \tag{4.15}$$

where

$$s_{1,j} = \frac{1}{F-1} \sum_{i=1}^{F} z_{1,i}.z_{j,i} \tag{4.16}$$

notice that, by statistics theory, equation 4.16 and equation 4.4 are similar since $s_{i,j}$ in equation 4.4 is a correlation using non-standardized values, and $s_{i,j}$ in equation 4.16 is a covariance using the standardization of those values.

Then, in order to simplify this proof, let us suppose that we wanted to classify not just one music from the training set, but the whole training set. Then it is easy to conclude that:

$$\mathbf{B} = \frac{\mathbf{1}}{\mathbf{F-1}} \mathbf{Z^T Z} \tag{4.17}$$

B would be obtained instead of $\vec{b_1}$ in equation 4.15. Note that **B** = **S** because it contains the similarity vectors between each training set music and the other music samples of the same data set.

Now let us work with the entire **S** as if we wanted to translate all training sounds in vectors in the k-dimension space. Then, from 4.12 we would obtain

$$\mathbf{A = SP\Lambda^{-1/2} = SP\Lambda^{-1}\Lambda^{1/2}} \tag{4.18}$$

but since $\mathbf{P^T P = I}$ (where **I** is the identity matrix) and $\mathbf{S = P\Lambda P^T}$, then

$$\mathbf{A = P\Lambda P^T P\Lambda^{-1}\Lambda^{1/2} = P\Lambda\Lambda^{-1}\Lambda^{1/2} = P\Lambda^{1/2}} \tag{4.19}$$

But $\mathbf{P\Lambda^{1/2}} = \mathbf{Q}$, the matrix characterizing all sounds by the PCA-based method presented before (see equation 4.7). Since we used a copy of the whole training set for classification instead of just one music, we obtained a matrix (**Q**) instead of a vector $\vec{u_1}$.

Then, choosing the $k = 7$ leftmost columns of this matrix we would obtain the **R** referred in Section 4.1, which is the matrix reflecting the whole training set in the k-dimensions

space. By this, we proved that $\vec{v_w}$ in equation 4.13 reflects the music we wanted to classify in the k-dimensions clustering space built in the training phase. □

5) Mahalanobis Distance - We are almost there! Now that we have our music w represented in the same k dimensions from the learned space, somehow we need to relate the created clusters with our vector $\vec{v_w}$. Mahalanobis distance was adopted for this purpose since it takes into account the geometric properties of each cluster, which is very important because distances take different impact depending on the dispersions along each axis [15]. This characteristic is not achieved when using other metrics such as Euclidean or Manhattan distances. With this distance calculated between each cluster centroid and our music sample (vector $\vec{v_w}$), the system proposes the class represented by the cluster having a smaller Mahalanobis distance as the most likely class for music w. In other words, class p will be associated to $\vec{v_w}$ if $D(\vec{v_w}, \vec{\mu_p}, \boldsymbol{\Sigma_p}^{-1}) = \min_i D(\vec{v_w}, \vec{\mu_i}, \boldsymbol{\Sigma_i}^{-1})$ where $D(\vec{v_w}, \vec{\mu_i}, \boldsymbol{\Sigma_i}^{-1})$ is given by:

$$D(\vec{v_w}, \vec{\mu_i}, \boldsymbol{\Sigma_i}^{-1}) = (\vec{v_w} - \vec{\mu_i})^T \boldsymbol{\Sigma_i}^{-1} (\vec{v_w} - \vec{\mu_i}) \tag{4.20}$$

where $\vec{v_w}$ corresponds to the new vector to be classified; $\vec{\mu_i}$ is the centroid concerning all samples that belongs to a specific cluster index i with $\vec{\mu_i} = [\mu_{i_.,1}, ..., \mu_{i_.,k}]$ with k dimensions and $\mu_{i_.,l} = \frac{1}{\|G_i\|} \cdot \sum_{g \in G_i} r_{g,l}$ where g refers to a music sample of the cluster (group) G_i and $\| G_i \|$ is the size of that cluster, $r_{g,l}$ is the value of the music g for axis l which is available from matrix $\mathbf{R}$ calculated in the training phase (see section 4.1, step 3.c). So, $\vec{\mu_i}$ represents, say, an *average* music sample of cluster i; $\boldsymbol{\Sigma_i}^{-1}$ is the inverse matrix of:

$$\boldsymbol{\Sigma_i} = \begin{pmatrix} D_{i_{1,1}} & D_{i_{1,2}} & \cdots & D_{i_{1,k}} \\ D_{i_{2,1}} & D_{i_{2,2}} & \cdots & D_{i_{2,k}} \\ \vdots & \vdots & \ddots & \vdots \\ D_{i_{k,1}} & D_{i_{k,2}} & \cdots & D_{i_{k,k}} \end{pmatrix}$$

and,

$$D_{i_{l,m}} = \frac{1}{\| G_i \| - 1} \cdot \sum_{g \in G_i} (r_{g,l} - \mu_{i_.,l}).(r_{g,m} - \mu_{i_.,m}) \ . \tag{4.21}$$

This matrix reflects the geometric properties of the cloud made by cluster i in the k-dimension space. So, each cell measures: the variance of the values along an axis; or the covariance reflecting the *correlation/dependence* of the values along a pair of axes.

It is important to notice that most calculations needed in the classification phase, can be done just once in the training phase which improves the classification performance. For instance, the clusters' centroid can be calculated in the learning process.

4.3 Conclusion

In this chapter we presented all details of our unsupervised automatic music genre recognition system: Section 4.1 presents the learning system and explains its 3 main steps: 1) Feature Extraction; 2) Matrix Transformations; 3) Classification. Once these steps are concluded, a vector containing cluster indexes is obtained. Each music from the initial data set is associated with one, and only one, cluster. Section 4.2 explains a second stage, to classify a new test sample according with the previous clusters. This system is probably more complex than the previous one but it does not deal with a higher complexity, on the opposite, it was developed to quickly answer to a new query. In the next chapter (Chapter 5), we present the results obtained with our clustering and classification system.

5. Results

This chapter presents all the results obtained from our system. Many test were performed and we discuss them next. We start mentioning which data set was used in our test (Section 5.1) and then, the most relevant figures are shown and analyzed (Section 5.2).

5.1 Data Collection and Validation

It is important to remember that our system is an unsupervised music genre recognizer. Thus, beyond analyzing the classification results (Section 5.2.2.3), there is also a need to analyze the clustering results, that is, the clusters created by our learning system. After an analysis of many studies related with automatic music genre recognition (Section 3.3), it would be interesting to use a data set which already has been object of study to easily compare our results to other studies that used the same data collection (yet, this may not be enough to do a direct comparison, as we will discuss later in this chapter). That is why the results presented are based on a data collection proposed by G.Tzanetakis called GTZAN [1] which has 10 different genres each with 100 different music samples [42]. In addition, we added a new genre (with 100 music samples) to the GTZAN data collection since we also would like to analyze a Portuguese typical music genre named Fado.

Therefore, our data collection has 1100 music files (all stored as 22050 Hz, 16-bit, mono audio files) representing 11 different music genres: blues, classical, country, disco, fado, hiphop, jazz, metal, pop, reggae and rock.

Once we have the data previously labeled, we can validate our clustering and classification results based on such labeling. To reinforce our main idea, we do not want to use such information to influence any results and, therefore, this information is not used in the learning process. Yet, this labeling is used to validate the results obtained. Assuming that this labeling is correct, an error calculation can be performed. The error percentage for a set of N music samples is obtained in the following way:

1. Get the most represented genre g_i for each cluster i; g_i will be seen as the correct genre labeling for this cluster;

[1] http://marsyas.info/download/

2. Calculate the number of *outliers* o_i, that is, the number of music titles from cluster i which actually are not labeled as g_i - we use the concept of outliers in this case since those music samples are different of the others;

3. Process this information to get an error percentage rate, e = $(100 * \sum_i o_i)/N$

5.2 Discussion

5.2.1 Learning Process

As earlier explained (Section 4.1) our system aims to create several clusters based on music features such that it organizes and distinguishes music from different genres. In this section we present all details concerning the training data set used, features details and obtained results. The referred results presented in this section are shown using accuracy rate or error rate depending on which interpretation is more convenient.

5.2.1.1 The Training Data Set

As mentioned above, we use a data collection of 1100 music titles. In order to train the system and test its clustering capabilities, we use a subset of samples from this data collection: a set of 15 music files were chosen from each genre (the first 15 from each genre in the data collection) to form the training set. This training set was named as "Data set A". We believe that 15 music samples from each genre (in a total of 165 music samples) are enough to get some results to analyze the accuracy of our implementation. Here we assume that all musics are *correctly* manually labeled.

A second training data set, "Data set B", was also created with less genres than "Data set A" nevertheless, it maintains the same number of music samples per genre. We selected the following 4 distinct genres to be represented in this data set: Classical, Fado, Metal and Reggae. Each genre is represented with 15 music titles, giving a total of 60 samples. Just like "Data set A", this data set is based on the main data collection (1100 music files).

5.2.1.2 Features

As mentioned in Section 3.1, many music features used for genre recognition purposes were studied . By analyzing that section it is easy to conclude that some feature sets are clearly more studied than others, and after a comparison between their achieved results, we chose a combination of features to test our system. In addition, we also used some extra properties that we believed could improve the performance of the recognizer. Their implementation details are explained below.

To extract most of the features, two principal libraries were used: *Marsyas* v0.2 [2] and *MIRtoolbox 1.2.4* [3] [41]. While MIRtoolbox is an integrated set of functions developed in Matlab to extract musical features from audio files, using several different approaches, Marsyas is a well known software used in MIR that is also a great tool to feature extraction.

'ttf' + 'mfccs' - A feature set named *timbral texture features* was explored during our test phase. As explained in Section 3.1, it is composed by: Spectral Centroid, Spectral Rolloff, Spectral Flux, ZCR, Low Energy and MFCCs. For this set, a STFT is applied using windows of 512 samples, with a hop size of 512 samples and a Hamming window.

These properties, excluding MFCCs, are extracted using *MARSYAS* software for each short-time frame. For MFCCs extraction, we used some functions developed in *Matlab* which are integrated in an *Auditory Toolbox* developed by Malcolm Slaney [36]. More information about this toolbox can be found here [4].

'scentroid','srolloff','sflux','zcr','lener' - As mentioned earlier, some of these spectral values can be calculated either after STFT ('ttf') or over the entire audio spectrum; we explore both approaches. To extract these same features only based on the audio spectrum, the MIRToolbox is used. Each one of these properties results in a single value per sample, except the 'sflux' values which corresponds to the SSD calculation over the temporal

[2] Marsyas (Music Analysis, Retrieval and Synthesis for Audio Signals) is an open source software framework for audio processing with specific emphasis on Music Information Retrieval applications. It has been designed and written by George Tzanetakis (gtzan@cs.uvic.ca) with help from students and researchers from around the world. Marsyas has been used for a variety of projects in both academia and industry. More information concerning this software can be found in http://marsyas.info/.

[3] MIRtoolbox offers an integrated set of functions written in Matlab, dedicated to the extraction from audio files of musical features such as tonality, rhythm, structures, etc. The objective is to offer an overview of computational approaches in the area of Music Information Retrieval. More details in https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox.

[4] http://cobweb.ecn.purdue.edu/ malcolm/interval/1998-010/

function of the spectral flux values (obtained with a frame length of 50ms and a hop factor of 0.5).

'rmsFrame' + 'rms' - To extract RMS feature, two solutions are followed: to 'rmsFrame', a STFT transformation is calculated with a frame length of 50ms and half overlapping. A MIRToolbox function gives us a temporal evolution of the energy which is used to calculate the SSD values that will represent this feature; and 'rms' is simply calculated over the entire magnitude spectrum.

'kurt', 'skew', 'entro' - Some statistical features are directly calculated over the spectrum (spectral kurtosis, spectral skewness and spectral entropy). Each one of these features are extracted using MIRToolbox functions.

'specStat' - Once the STFT transformation is applied, the SSD are calculated for each frame, i.e., for each magnitude spectrum, we calculate statistical values to represent each spectrum (mean, median, variance, skewness, kurtosis, min- and max-value).

'centBandUnif' - An analysis over each frame of the STFT can bring us two other features: Uniformity and Bandwidth. Already explained in section 3.1, we calculate each one of these properties over each frame of the signal in analysis. Another property can be calculated as shown in [11, 23]: differences between uniformity and bandwidth values from consecutive frames present another feature set that can characterize our samples.

Some rhythm features are also explored. Mentioned in section 3.1.2, we extract: beat properties and rhythm patterns properties.

'beat' - Beat properties are extracted using the *MARSYAS* software and they follow exactly the same criteria as mentioned in [42].

'tempo' - As beat the properties from the *MARSYAS* software were not as complete as we would like, we decide to also use the MIRToolbox to obtain the autocorrelation function for each music. Once the function is calculated, we process the SSD over the autocorrelation function. This analysis brings us an important descriptor of music repetitions (autocorrelation). Tempo estimation (bpm) is also added to these values to compose the feature set.

'rssd_rh' - Rhythm patterns were extracted with a Matlab function[5] presented in [21]. Details

[5]http://www.ifs.tuwien.ac.at/mir/audiofeatureextraction.html

about its implementation were already presented in section 3.1.2.

5.2.1.3 Results (Test A)

With so many different features it would be extremely difficult to test all the possible combinations of features and see which combination(s) is/are the best(s). Although, we established some criteria to minimize the chance to have a feature combination that would have clearly better results than all the others and it would not be tested. We tested several combinations of features for each training set used. For each combination, an error percentage is calculated. Even though we tested many combinations of features, here we report the results of only a few of those combinations: those with lower error rate.

Our first experience, "Test A", uses combinations of features extracted from "data set A". These features are: 'ttf' 'rssd_rh' 'beat' 'rmsFrame' 'mfccs' 'lener' 'scentroid' 'zcr' 'srolloff' 'sflux' 'rms' 'kurt' 'skew' 'entro' 'centBandUnif' 'tempo' 'specStat'. As it would be expected, several combinations present the same error percentage as well as the same cluster composition. This reveals that some of the listed features do not characterize significantly our musics, at least, in what concerns music genre analysis.

As mentioned above, here we only report the results of a few of the tested combinations. In table 5.1 we present the error percentage as well as the feature combination extracted to obtain those results. Since this error percentage may not be enough to understand the accuracy of our system, we plot the result of each clustering process as an image that relates the learned clusters to genres, that is, to the initial labeling. In each illustration of Figure 5.1 we have all the 165 musics from "data set A" represented with dots (.) or crosses (x). In the vertical axis, each music is grouped according to its genre (based on the initial labeling): we have 11 musical genres represented each with 15 music samples. To easily understand the created cluster, different samples from different genres were plotted in different colors. The horizontal axis represents the learned clusters where each number represent a cluster *id*. With these illustrations we can now analyze in detail which clusters where created and which musics are grouped together.

If you remember how our system is implemented (Chapter 4, more precisely Section 4.1), there is a similarity matrix **S** which is created during the leaning process. Each cell of this matrix represents the similarity between two music samples. We plot these matrices in Figure 5.2 for each feature combination here analyzed. Each matrix plots the similarity between 165 musics ("Data set A") where red colors are used to higher values while blue ones concern lower

similarity values. Since we are dealing with similarity matrices, the objects in the horizontal axis are the same as in the vertical axis, although, we labeled them differently: in the y axis we define the music genres in analysis while in the x axis we deal with the "music number".

As presented in table 5.1, our *best*[6] accuracy result was obtained with a combination of 3 feature sets: 'ttf', 'rssd_rh', 'lener'. An accuracy of 55% was obtained, in other words, in a data set with 165 musics, 90 musics where well classified.

Based on the error calculation explained above, we have 45% of wrong classified musics. Although, as we can see in Figure 5.1a, particularly looking at cluster number 3, we have musics labeled as Fado, Jazz and Blues, which cannot necessarily be considered as a *wrong clustering* since boundaries between these genres are thin! Nevertheless, some music genres are easily identified.

Looking to the classical samples, grouped in cluster 4, all the 15 musics were grouped in the same cluster. As well as Classical musics, Fado musics were also combined together (13 out of 15) in cluster number 8. From 15 Reggae samples, only 2 are not in cluster number 2, being all the other grouped together. Hiphop samples were also grouped together (12 out of 15) despite other several samples labeled as disco "sharing" cluster number 7. From the 15 musics labeled as Metal, only 4 were grouped in an "outsider" cluster. For all the other genres (Rock, Pop, Jazz, Country and Blues), they have been clustered into, at least, 4 clusters. This means that the features used are not capable to completely discriminate all musical genres in a clearly way such that MBCA is able to create the *correct* clusters.

In this image (Figure 5.1a) we can also see that with 11 genres our learning system creates only 9 different clusters, which is not as perfect as we would like, although it is important to analyze and see how we can improve such accuracy. Later on, in Chapter 6, we discuss about future work that can be done to improve the system.

If we now look at the similarity matrix created during the learning process for this feature combination (Figure 5.2a), we can see that there are too many red colors beyond the diagonal. In a perfect solution, we would like to present a similarity matrix such that the similarity between musics with the same musical genre would be very high (values near 1) whereas similarity between musics with different genres would be lower (near 0). In this case, along the diagonal we would have *red squares* (with a 15*15 dimension) while all the other values would be near

[6]How can we say if this is really our best result? We based our classification accuracy on the error percentage obtained! Since, we did not test every possible combination of features, this does not mean that the clusters obtained with these features are better than all the others.

Feature Combination	Error Rate (%)	Covariance Matrix	Result Fig.
‘ttf’, ‘rssd_rh’, ‘lener’	45	Figure 5.2a)	Figure 5.1a)
‘ttf’, ‘rssd_rh’, ‘beat’, ‘scentroid’	48	Figure 5.2b)	Figure 5.1b)
‘ttf’, ‘rssd_rh’, ‘beat’, ‘rmsFrame’, ‘sflux’	50	Figure 5.2c)	Figure 5.1c)
‘ttf’, ‘rssd_rh’, ‘sflux’	50	Figure 5.2d)	Figure 5.1d)
‘ttf’, ‘rssd_rh’, ‘specStat’	51	Figure 5.2e)	Figure 5.1e)
‘rssd_rh’	52	Figure 5.2f)	Figure 5.1f)

Table 5.1: Best results achieved in "Test A".

zero and present a blue color.

We will not do such a thorough analysis of the remaining images in Figures 5.1 and 5.2. We will only highlight some interesting or important details about these images.

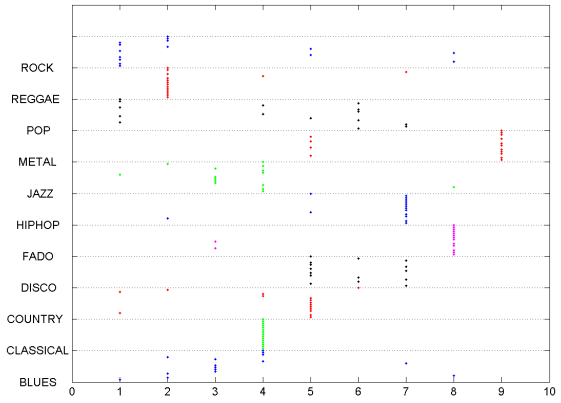
In a more global observation of Figure 5.1, it can be observed that some feature combinations correctly group samples from a specific genre while other combinations are more suitable to correctly group other musical genres. Let us look at Figure 5.1c or Figure 5.1e in which all the 15 Fado musics are correctly grouped, while using other combinations this does not happen. As well as Fado musics, we can highlight other similar situations (for instance, Reggae music in Figure 5.1f). The developed system does not perform any feature selection such that only the best features are submitted to the MBCA algorithm and we believe that with such selection an improvement could be achieved. We will discuss this subject and other important conclusions concerning "Test A" in the next Chapter (Chapter 6).

In the next section, we will use our system with a different data set, "Data set B", to see how the system performs with less musical genres.

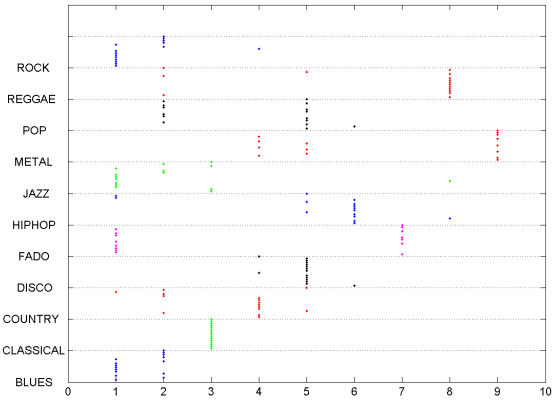
5.2.1.4 Results (Test B)

As in "Test A", we use the same feature range to this second experience. Following the same processing methods (Section 4.1), a learning process will be performed over "Data set B" (see Section 5.1 for more details). Like we did before, we plot an image for each one of the best six results obtained in which we can see how many clusters were created as well as the musics belonging to each cluster (Figure 5.3). In Figure 5.4 the respective similarity matrices are plotted.

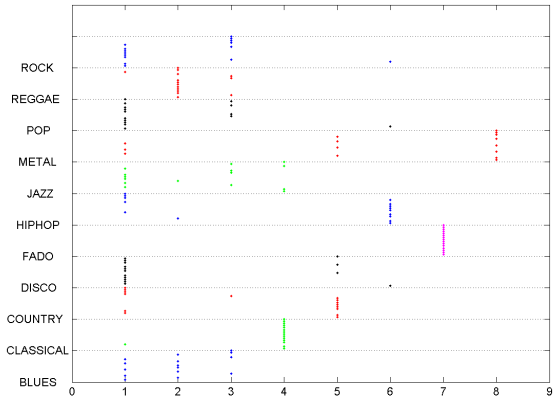
With "Data set B" the results are really good! In table 5.2 the best 6 feature combination are



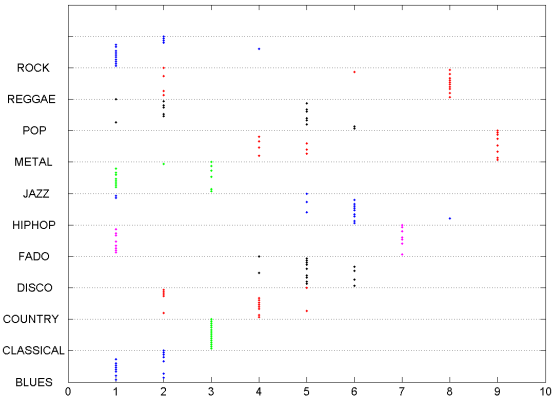
(a) ‘ttf’, ‘rssd_rh’, ‘lener’



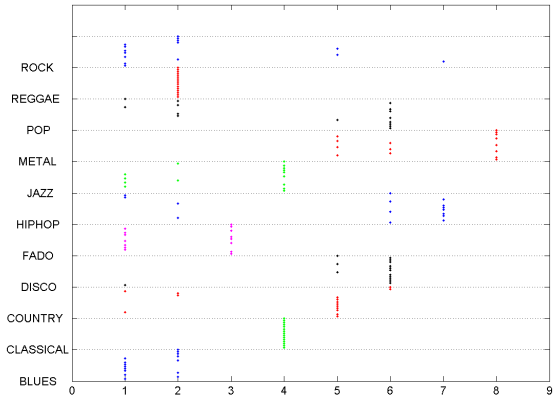
(b) ‘ttf’, ‘rssd_rh’, ‘beat’, ‘scentroid’



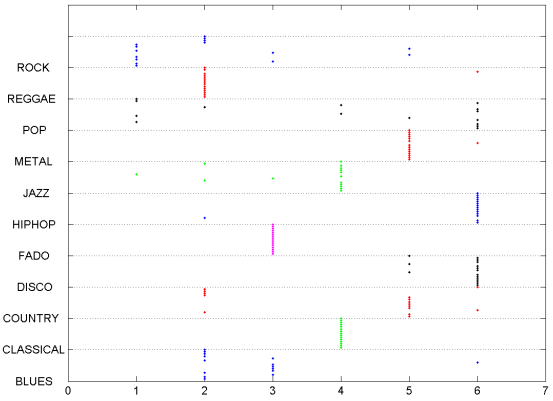
(c) ‘ttf’, ‘rssd_rh’, ‘beat’, ‘rmsFrame’, ‘sflux’



(d) ‘ttf’, ‘rssd_rh’, ‘sflux’

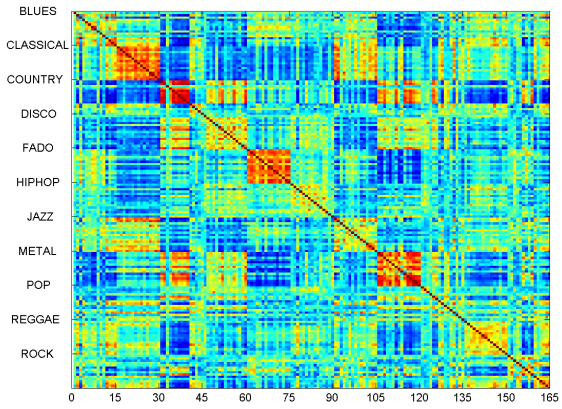


(e) ‘ttf’, ‘rssd_rh’, ‘specStat’

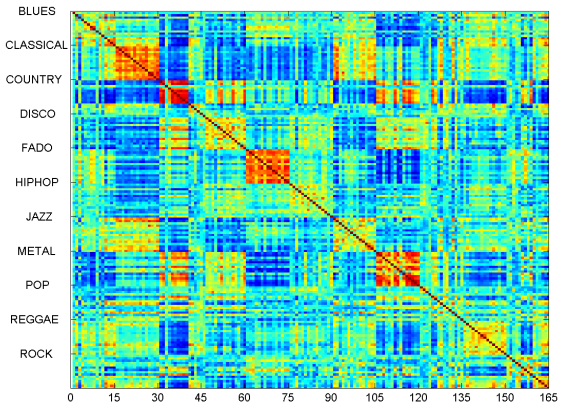


(f) ‘rssd_rh’

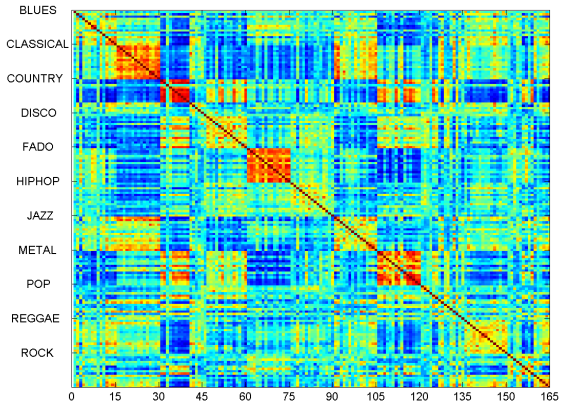
Figure 5.1: Results illustration of learning system for "Test A" - MBCA clusters.



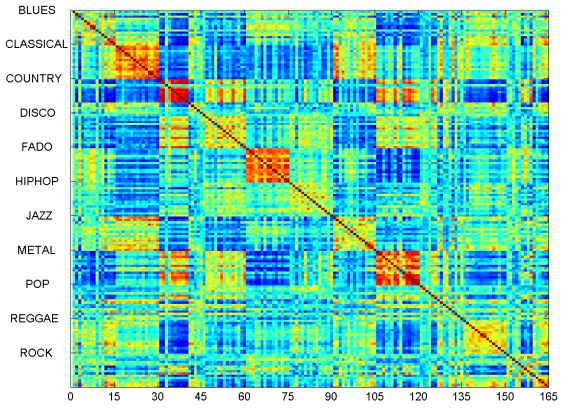
(a) ‘ttf’, ‘rssd_rh’, ‘lener’



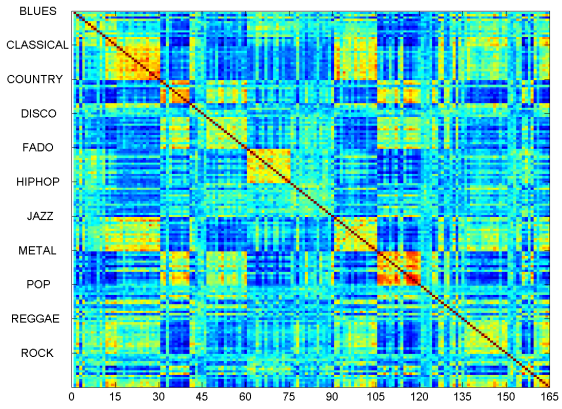
(b) ‘ttf’, ‘rssd_rh’, ‘beat’, ‘scentroid’



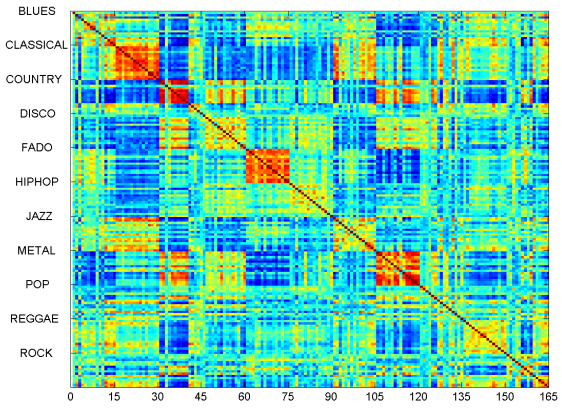
(c) ‘ttf’, ‘rssd_rh’, ‘beat’, ‘rmsFrame’, ‘sflux’



(d) ‘ttf’, ‘rssd_rh’, ‘sflux’



(e) ‘ttf’, ‘rssd_rh’, ‘specStat’



(f) ‘rssd_rh’

Figure 5.2: Results illustration of learning system for "Test A" - Similarity Matrices.

Nr.	Feature Combination	Error (%)	Covariance Matrix	Result Fig.
1.	‘ttf’, ‘rssd_rh’, ‘beat’, ‘rmsFrame’, ‘mfccs’, ‘centBandUnif’	0	Figure 5.4a	Figure 5.3a
2.	‘ttf’, ‘rssd_rh’, ‘beat’, ‘rmsFrame’, ‘mfccs’, ‘specStat’	0	Figure 5.4b	Figure 5.3b
3.	‘ttf’, ‘rssd_rh’, ‘centBandUnif’	0	Figure 5.4c	Figure 5.3c
4.	‘ttf’, ‘rssd_rh’, ‘specStat’	2	Figure 5.4d	Figure 5.3d
5.	‘ttf’,‘rssd_rh’,‘beat’,‘rmsFrame’,‘mfccs’, ‘lener’,‘scentroid’,‘specStat’	3	Figure 5.4e	Figure 5.3e
6.	‘ttf’,‘rssd_rh’,‘beat’,‘rmsFrame’,‘mfccs’, ‘lener’,‘scentroid’,‘zcr’,‘specStat’	3	Figure 5.4f	Figure 5.3f

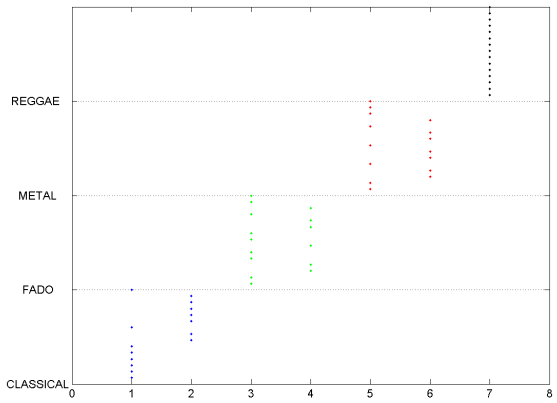
Table 5.2: Best results achieved in "Test B".

shown and we achieved an error rate of 0% (features combinations nr.1, 2 and 3). This shows that with a smaller number of genres, our system is able to achieve perfect clustering (at least compared with the initial labeling!).

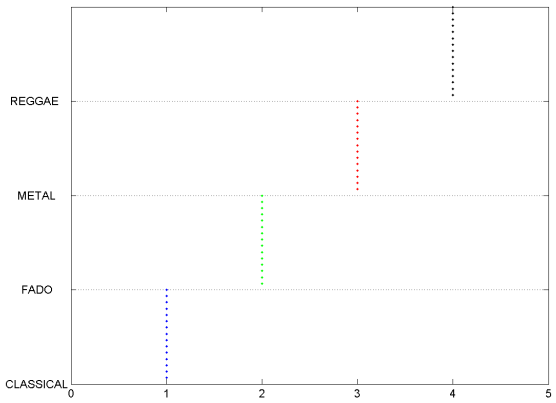
As we have 3 combinations which presents an error of 0% we can do a comparison and see if there is one combination that achieves better results. Based on table 5.2, it is clear that with combination nr.3 we do need to analyze less features than with the other ones. On the other hand, if we look at the clusters created, combination nr.2 presents a perfect cluster combination since it exactly creates 4 cluster with the 15 correct musics. Although this does not mean that the other analysis are incorrect or worse. It may actually be the case that combinations nr.1 and 3 are learning sub-genres within Classical, Fado and Metal.

Analyzing Figure 5.4, the presented similarity matrices are much more better looking than the ones obtained in "Test A". From these images it is hard to define which is the most well defined similarity matrices. It is clear that the squares along the diagonal are well defined (with red colors panel) which is a great sign concerning the similarity between musics with the same musical genre.

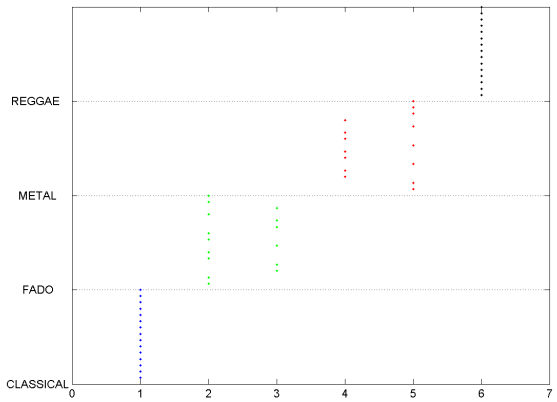
With such results, we felt that a second step would be almost mandatory: we are talking about a classification process! With an accuracy rate of 0% with 4 music genres (60 samples), we believed that we could perform a classification process with successful results. As described in Section 4.2 a classification system was implemented despite it was not our first goal. In the next section (Section 5.2.2) we present some results from our classification system.



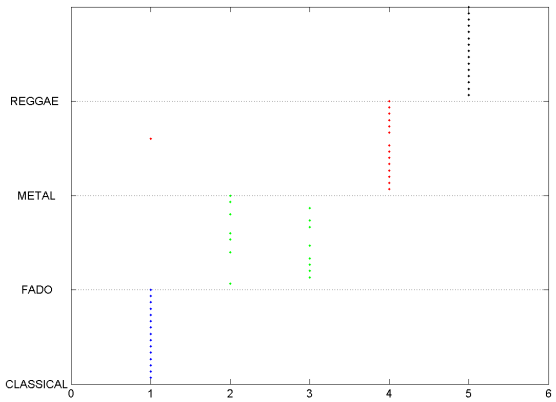
(a) Combination nr.1



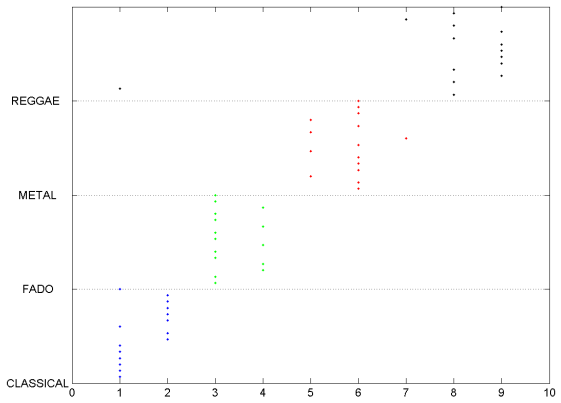
(b) Combination nr.2



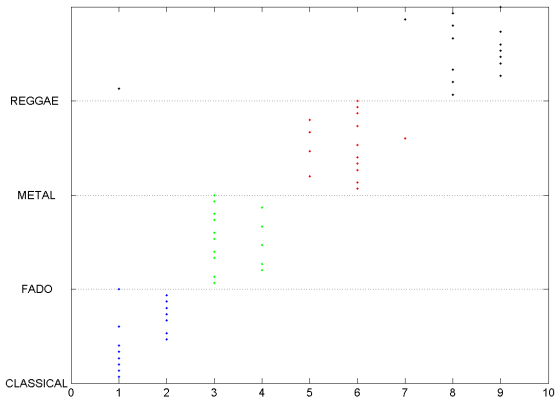
(c) Combination nr.3



(d) Combination nr.4

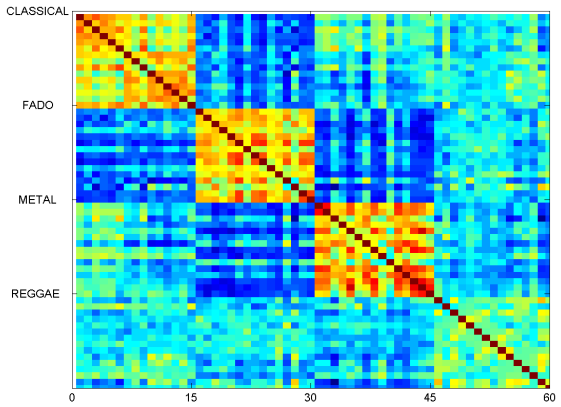


(e) Combination nr.5

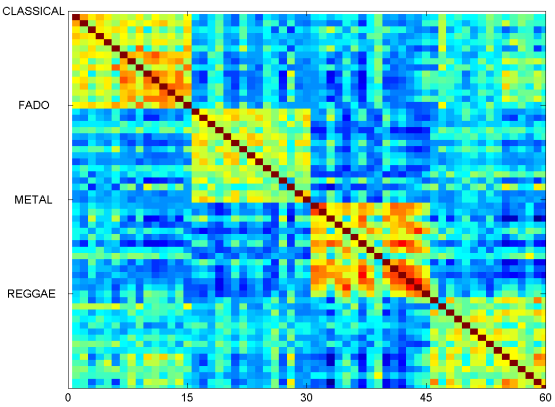


(f) Combination nr.6

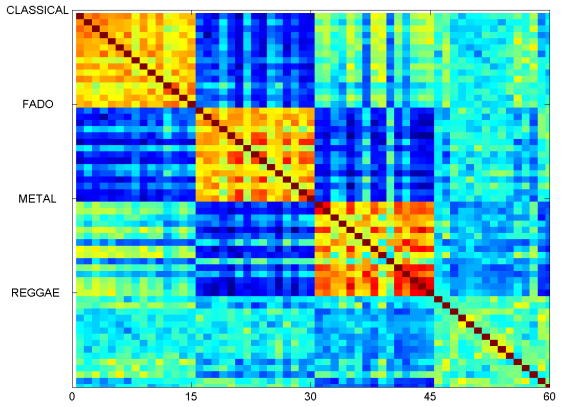
Figure 5.3: Results illustration of learning system for "Test B" - MBCA clusters.



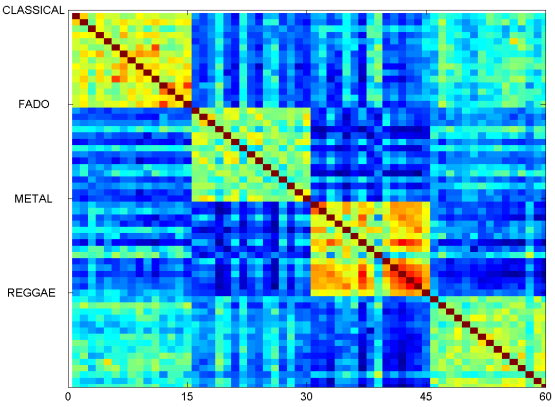
(a) Combination nr.1



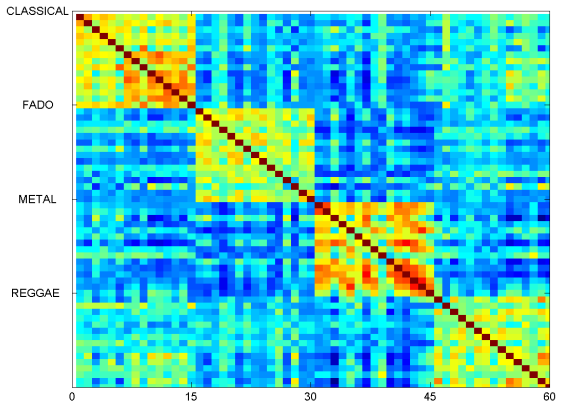
(b) Combination nr.2



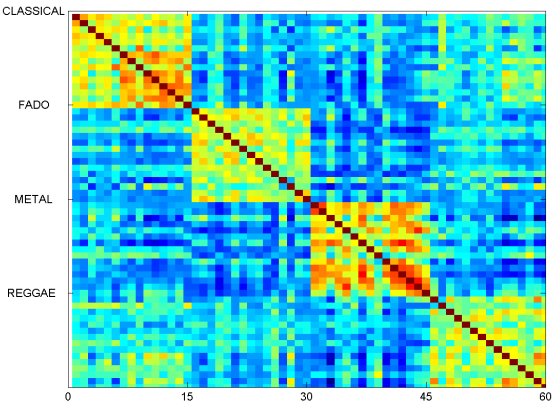
(c) Combination nr.3



(d) Combination nr.4



(e) Combination nr.5



(f) Combination nr.6

Figure 5.4: Results illustration of learning system for "Test B" - Similarity Matrices.

5.2.2 Classification Process

Once the learning process is concluded, we are able to classify any other music according with the created clusters. A detailed explanation of the implemented system can be read in Section 4.2. In this section we discuss some interesting results that we obtained testing the classification system. In this classification process, a submitted music is *always* associated with one of the existing clusters, that is, the musics in the test set have the labels as the musics in the training set.

5.2.2.1 The Test Data Set

With an accuracy rate of 100% in the learning process, we are "forced" to use the same data set which obtained these results. This result was obtained with "Data set B", and therefore we will only use musics belonging to the 4 genres represented in "Data set B": Classical, Fado, Metal and Reggae. As the reader may recall from Section 5.1, our data collection has 1100 musics from 11 different genres. As the learning phase used the first 15 musics from each of the 4 genres, we now will classify the remaining 85 musics from each genre, with a total of 340 (85musics*4genres) musics to classify.

5.2.2.2 Features

As previously discussed in Section 5.2.1.4, we achieved an error of 0% with 3 different feature set combinations. Here we analyze the results of the classification process for each one of these 3 combinations, see table 5.2. The combination number is given by the first column presented in the referred table.

5.2.2.3 Classification Results

As result from the clustering process, we obtain, with the data set used, a cluster id (cluster to which the music sample is closer) and all the Mahalanobis distances (that is, the distances of the new music to each of the learned clusters). To easily understand the accuracy of our system, we present: a confusion matrix for each combination, that shows the clusters assigned to the samples in the test set (tables 5.3, 5.4 and 5.5); and a table with the accuracy rates for each combination (table 5.6).

It is important to remember that we only use combinations in which the learning process gets

0% error rate, that way, we can label each cluster with a music genre. Thus, in the confusion matrices, the first line corresponds to the music genre associated to each cluster number. For instance, using combination number 1 we have obtained 7 clusters from learning process (see Figure 5.3a) so, as each cluster only has one music genre represented, we can associate this label to the cluster number. That way the two firsts lines from each confusion matrix will associate one cluster to a music genre (based on the learning process). For each set of musics submitted to our classification process (85 musics from 4 different genres), we will add a new line in which each cell represents the number of musics classified in the cluster identified by an id. Table 5.6 works as a summary of all the confusion matrices. To all combinations an accuracy rate is calculated and plotted as well as the number of correctly classified musics.

Let us analyze the different confusion matrices obtained. Based on combination number 1 (table 5.3), we notice that several musics labeled as Classical were classified as Reggae (27 musics) and Metal (17 musics). This is strange and it is not easy to explain. Perhaps the classical samples used during the learning process were not enough to establish a correct "range" for Classical musics. Other than Classical musics, this feature combination achieved a reasonable performance. In 340 musics, using this combination, our classification system correctly classified 260 musics (76.5%).

Before a discussion concerning feature combination number 2, let us first analyze the results from combination number 3. This combination present the worst result with an accuracy rate of 73.8%. Since this combination is composed by only 3 features, it is comprehensible that the results obtained were not that good once extracted from a larger number of musics.

Combination number 2 achieves an accuracy of 81.8%. This was the best result during the classification test. Analyzing Figure 5.3b, only 4 clusters were created during the learning process. Indeed, with only 4 clusters, it helps our classification system to easily choose which cluster is the more appropriate to the submitted music.

As discussed in Section 4.2, Mahalanobis distance is calculated between each learned cluster centroid (a vector) and the music to be classified, that is a test music (another vector). In order to know to which group the test music belongs, this distance is calculated for all the 340 musics. Obviously, it makes no sense to present all the calculated distances, yet, we would like to highlight some of the obtained results that we consider important.

As our best accuracy result was achieved with feature combination number 2, we present in Table 5.7 several distances calculated for 4 test music samples. Each line represents a different music and for each music we start by plotting the correct cluster, next the group number to

	Classical		Fado		Metal		Reggae	
	1	2	3	4	5	6	7	Accuracy Rate (%)
Classical	**26**	**15**	0	0	0	17	27	48,2
Fado	0	0	**28**	**41**	0	6	10	81,2
Metal	0	0	0	0	**0**	**77**	8	90,6
Reggae	0	0	0	1	0	11	**73**	85,9

Table 5.3: Confusion matrix of classification process using feature set combination nr.1. Number of musics labeled with a specific genre associated to a cluster id (7 clusters).

	Classical	Fado	Metal	Reggae	
	1	2	3	4	Accuracy Rate (%)
Classical	**69**	0	10	6	81,2
Fado	8	**63**	4	10	74,1
Metal	0	0	**82**	3	96,5
Reggae	0	0	21	**64**	75,3

Table 5.4: Confusion matrix of classification process using feature set combination nr.2. Number of musics labeled with a specific genre associated to a cluster id (4 clusters).

	Classical	Fado		Metal		Reggae	
	1	2	3	4	5	6	Accuracy Rate (%)
Classical	**51**	0	0	33	0	1	60,0
Fado	0	**28**	**29**	28	0	0	67,0
Metal	0	0	0	**83**	**1**	1	98,8
Reggae	0	0	0	26	0	**59**	69,4

Table 5.5: Confusion matrix of classification process using feature set combination nr.3. Number of musics labeled with a specific genre associated to a cluster id (6 clusters).

	Accuracy Rate (%)	Correctly Classified
Combination nr.1	76.5	260/340
Combination nr.2	**81.8**	**278/340**
Combination nr.3	73.8	251/340

Table 5.6: Accuracy results of a classification process for 3 different feature combinations.

Correct Group	Selected Group	Dist. to G1	Dist. to G2	Dist. to G3	Dist. to G4
1	1	**0.805338**	42.22051	19.78257	13.32368
1	1	**1.862967**	51.04241	21.69921	15.07404
1	4	9.005724	50.9507	19.63604	**8.784079**
1	3	17.53428	29.53328	**17.22235**	20.50123

Table 5.7: Mahalanobis distance examples calculated from 4 distinct musics based on feature combination number 2. The first column represent the correct group index, the second corresponds to the attributed group by our system, and next, the Mahalanobis distances calculated are shown (each distance is from the music to a specific cluster). Distances in bold are those chosen by the system.

which the music was associated by our system and, finally, the distances for each different cluster centroid. The top two lines (in green background) show correct classifications in which the musics are grouped to cluster number 1. In the bottom 2 lines (in a red background), we have wrong classifications where musics are classified as belonging to cluster 3 or 4, while they should be classified as belonging to cluster number 1.

In a closer analysis, if we look at the firsts two lines (green ones), we can see that a music that "belongs" to cluster 1 is correctly associated to that cluster. A significant difference between distances is obvious and the music in analysis is clearly closer from group 1 than from all the others. In the other hand, the two red lines represent wrong classifications. Despite music titles belong to group number 1, our system classifies them in cluster number 4 and 3, respectively. In fact, looking at the Mahalanobis distances calculated for both, we can see that there is no significant differences between the distances to the closest cluster and the second closest cluster. Our system associates the analyzed music to the closest cluster despite the slightly differences presented between distances, i.e., in the last table line the system concludes that the music belongs to cluster number 3 as it presents a distance of: 17.22235; although, it presents a very similar distance to cluster 1: 17.53428. A slight difference between distances may be handled with some attention as it does not clearly identify a single cluster as the correct one for the tested music: a possible future criterion to be considered.

The implemented system is able to classify new test musics, thus, our second goal was achieved. Despite the achieved results are already very good, we believe that these results can still be improved, and our classification system can be upgraded. In the next chapter (Chapter 6) we discuss some future work that may be important to improve the systems accuracy.

6 . Conclusion and Future Work

The proposed system is now implemented and tested. With 2 distinct systems, learning (Section 4.1) and classification (Section 4.2), our main goal was achieved. Only based on audio features, our learning system is able to create music clusters using a *Model-Based* approach (MBCA). Once the features have been extracted, a redundancy reduction was required to reduce the dimension of our data, for that purpose, a PCA-based technique was adopted. Our second implementation, the classification system, performs a feature extraction over the test music and compares it with the previously clustered samples. This comparison is based on *Mahalanobis* distances between the tested music and all the created clusters.

This chapter discusses the proposed music genre classifier as well as some improvements that we believe would lead to even better results. As mentioned in Chapter 1, our initial and main goal was to develop a system which is able to learn several music clusters according to the music samples' genres only based on feature analysis and with no previous knowledge on the musics' genres. This first goal was successfully achieved with the creation of the learning system (see Section 4.1 for details), which was successfully implemented. In addition, we also aimed to go further and create a classification system grounded on the results achieved in the learning phase. This second goal was also successfully achieved (see Section 4.2 for details).

As a global appreciation, it is interesting to see that, despite the criticism concerning the use of an unsupervised model in a music genre recognition system, the achieved results show that even with such an approach, we are able to obtain clusters as expected by a common human taxonomy. With this model approach, we obtain clusters following the initial labeling proposed, creating one cluster for each music genre. This kind of approach has the advantage to be totally independent from any influence coming from a human taxonomy. Thus, such system is able to create several music clusters only based on audio features. It is also useful to distinguish between songs that may present no significant differences for human taxonomy but in fact belongs to distinct *clusters*. For instance, someone who is not able to distinguish differences between Classical songs may have a temptation to group all similar samples in one group. Nevertheless, with an unsupervised music genre recognition, this kind of mistake can be avoided.

6.1 Learning Process

As our system follows an unsupervised approach (see Section 2.2.2.2 for details) a validation technique had to be adopted. We used manual labeling of the data to validate the clustering results, that is, the accuracy rate was calculated with basis on the data collection existing labeling. It may seem a contradiction to create an unsupervised classification system and get an accuracy rate based on a previously labeled data set. While it can be senseless, it is the best approach, in our opinion, to easily validate our system and see how accurate it can be. Of course that with an unsupervised approach there is not a perfect validation method as we begin with nothing else but a music data set completely "naked" of information besides its audio characteristics, thus, created clusters can be correct even if some samples, previously labeled with one genre, are grouped in a cluster which is composed by samples labeled with a different genre, in other words, our music samples can be correctly grouped even if a previous labeling does not say so. As previously discussed, a Human labeling will always be fuzzy and could not be seen has a universal truth. Again, we do need a reference basis for validation purposes, and that is why we decide to use the initial labeling to have accuracy percentages.

With a data set composed by 1100 music samples from 11 different genres (Blues, Classical, Country, Disco, Fado, Hip-Hop, Jazz, Metal, Pop, Reggae and Rock, we achieved an accuracy rate of **55%**. This result was the best from several tests in which different feature sets where used, in other words, we did not use always the same features, we combined them into different sets and verified the accuracy of the system using the distinct feature combinations. Our best accuracy was obtained using a combination of 3 feature sets: 'ttf', 'rssd_rh' and 'lener' (see Section 5.2.1.3 for details). It is not a perfect result accuracy, although, as we are following an unsupervised approach for music genre classification, it would be unexpected to obtain higher accuracies in a first experience. In a total of 11 music genres, it is comprehensible that some genres present less specific audio properties than others, turning a clustering process much harder. Thus, we believe that with some updates in our methodology a higher result can be obtained.

Once the accuracy rate is obtained, we have the temptation to compare these results with those from other music genre classification studies. Although, there have been several such studies (see Chapter 3), a direct comparison could not be plausible since most studies follow a supervised approach and we believe that differences between each methodology (supervised and unsupervised) are too many to compare such methods. Still, from the 2 known unsupervised

approaches [31, 35] we only can compare our results with X.Shao et al. [35] study since it presents an accuracy percentage while A.Rauber et al. do not present the accuracy results [31]. Then, X.Shao and his team achieved an accuracy result of 89% based on a data set with 50 musics from 4 music genres (Pop, Country, Jazz and Classic). Although our 55% accuracy is worst, we must keep in mind that it is obtained with 65 musics from 11 genres data set. As we will see below, our accuracy will increase as soon as we reduce the number of music genres. There is another important detail that may not be ignored: in such comparison, the different data sets used may drastically change the accuracy rate between studies.

In a second test to our learning system, we used a training set with less genres. This data set had 65 musics where 4 different genres are represented: Classical, Fado, Metal and Reggae. Proceeding exactly as described in Section 4.1, an accuracy rate of **100%** was obtained. This result is very satisfactory since we were able to perfectly create 4 clusters without any wrong classified music. As mentioned above, the only study with an unsupervised model which present an accuracy result achieved an accuracy of 89% with 4 genres [35]. As we already mentioned, a direct comparison is not fair since they are not based in the same data set (in this study they do not provide the used data set) then, we can only mention this article to be aware about another unsupervised model based approach to classify musics by their genres.

From this 2 tests some conclusions can be drawn. We are aware that an accuracy of 55% does not present a perfect result for a music genre clustering system. Although, once we reduce the number of genres, the accuracy results clearly improves (100% with 4 genres). As it would be difficult to present all the performed tests and we were looking for a high accuracy rate, we would like to highlight that even with 5/6 genres the accuracy result was good (around 90%). More results may be shown in a future work thus, we are pleased with the obtained accuracy despite we always pursue better performances.

6.2 Classification Process

In order to test the classification phase, we used the feature combination that lead to 100% clustering accuracy in the clustering phase. As shown in Section 5.2.2.3, we achieved a classification accuracy of 82% using a test set composed of 340 musics from 4 genres (Classical, Fado, Metal and Reggae). Looking at the classification results obtained with combination number 2 (Table 5.4), Metal musics are easily classified compared with the other genres. In 85 Metal

musics, only 3 musics were not correctly classified, being classified as Reggae. Between the other 3 styles, there is not a huge difference, around 65 musics are correctly classified from the 85 submitted.

During this classification phase, Mahalanobis distance between a music (being classified) and each one of the clusters centroids (created during the learning phase) is determined. Once all the distances are calculated, the system picks the closest group and associates it to the music. Analyzing 4 different distances plotted in Table 5.7 some conclusions can be drawn. Once all Mahalanobis distances are calculated, it would be interesting to analyze the differences between these distances to prevent wrong classifications. That way, if the smallest distance is too close to the second smallest distance, we could, for instance, give the classification a smaller degree of confidence or associate the test sample with more than one cluster.

Finally, it is important to highlight that there is no rejection in our classification system. This means that each submitted music is always associated with one existing cluster and cannot be considered as from another genre not represented by the training set.

6.3 Future Work

Since we already analyzed our systems results, let us now discuss about some possible improvements that could lead to a better performance. We are pleased about the obtained results, nevertheless, during the implementation stage, some details were not treated with too much attention since an initial schedule had to be respected and we tried to achieve all our initial goals. Let us try to give you an idea of how these systems could reach higher accuracy results.

Starting by the foundations of our system, and by foundations we mean the extracted features, an important set of features was analyzed and tested. As we referred in Section 3.1 there are many other audio features that we do not explore in this work but could be very important to discriminate the represented genres and smooth the classification process difficulty. We have no doubt about the importance of a continuous necessity to explore more features since the better the features characterize and discriminate audio properties, the better our system would create clusters. Also concerning features, during the learning phase, there are some details that could make a difference. Notice that we do not explore a possible "filtering" over the extracted features to only process those that present higher variances between musics. This approach would be important to get "better looking" similarity matrices.

Considering the classification system, we do believe that an analysis of the variances between the calculated Mahalanobis distances would be important. As mentioned in the above section, small differences between those distances can be problematic for our classification system. We do not allow our system to associate one music with more than one cluster, thus, a strict choice of the group that is based only on the smallest distance could be not as good as we wanted to. With small variances between distances, it would be interesting, perhaps, to associate the analyzed music to more than one cluster or alternatively, to associate a lower degree of confidence to the results. As for those test samples that are too far away from all clusters, we could also consider to totally reject them.

We are conscious that many updates can be performed in the proposed methodology. With our approach, we wanted to raise a discussion about this model (unsupervised clustering) since we believe that music genres do not present clear boundaries between each others. That way, a supervised system would be tendentious as it would be trained based on a previously labeled data set. The obtained results show that it is possible to achieve good accuracy results using an unsupervised model to automatic music genre recognition.

Bibliography

[1] Dictionary.com unabridged, January 2010.

[2] S. Cavaco. "Lecture i - introduction and audio analysis" in interaction and visualization in multimedia environments - audio, March 2009.

[3] P.R. Cook. *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics*. The MIT Press, March 2001.

[4] J.F. da Silva, J. Mexia, C. Coelho, and G. Lopes. Multilingual document clustering, topic extraction and data transformations. In *Progress in Artificial Intelligence*, pages 55–98, 2001.

[5] J.F. da Silva, J. Mexia, C. Coelho, and J. Lopes. Document clustering and cluster topic extraction in multilingual corpora. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 513–520. IEEE Computer Society, 2001.

[6] Y. Escoufier and H. L'Hermier. A propos de la comparaison graphique des matrices de variance. *Biometrical Journal*, 20(5):477–483, 1978.

[7] C. Fraley and A. E. Raftery. How many clusters? which clustering method? answers via Model-Based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.

[8] C. Fraley and A.E Raftery. MCLUST: software for Model-Based cluster analysis. *Journal of Classification*, 16:297–306, 1999.

[9] C. Fraley and A.E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97:611–631, June 2002.

[10] K. Fukunaga. *Introduction to Statistical Pattern Recognition, Second Edition*. Academic Press, 2 edition, October 1990.

[11] S. Golub. Classifying recorded music. Master's thesis, University of Edinburgh - Division of Informatics, 2000.

[12] S. Hacker. *MP3: The Definitive Guide*. O'Reilly Media, illustrated edition edition, May 2000.

[13] W.M. Hartmann. *Signals, Sound, and Sensation*. American Institute of Physics, corrected edition, September 2004.

[14] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice Hall College Div, March 1988.

[15] R.A. Johnson and D.W. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, 6 edition, April 2007.

[16] A.L. Koerich and C. Poitevin. Combination of homogeneous classifiers for musical genre classification. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 1, pages 554–559, 2005.

[17] C. Lee, J. Shih, K. Yu, and H. Lin. Automatic music genre classification based on modulation spectral analysis of spectral and cepstral features. *IEEE Transactions on Multimedia*, 11(4):670–682, 2009.

[18] T. Li and M. Ogihara. Toward intelligent music information retrieval. *Multimedia, IEEE Transactions on*, 8(3):564–574, 2006.

[19] T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 282–289, Toronto, Canada, 2003.

[20] T. Li and G. Tzanetakis. Factors in automatic musical genre classification of audio signals. In *Workshop on Applications of Signal Processing to Audio and Acoustics IEEE.*, pages 143–146, 2003.

[21] T. Lidy and A. Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *ISMIR*, pages 34–41, 2005.

[22] S. Lippens, J.P. Martens, and T. De Mulder. A comparison of human and automatic musical genre classification. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 4, 2004.

[23] R. Malheiro, R.P. Paiva, A.J. Mendes, T. Mendes, and A. Cardoso. Sistemas de classificação musical com redes neuronais. In *Gestão e Desenvolvimento - Universidade Católica Portuguesa*, volume 12, pages 167–195, 2004.

[24] E. Mankin. Principal component analysis: A How-To manual for r.

[25] C. McKay. *Automatic Genre Classification of MIDI Recordings*. PhD thesis, 2004.

[26] C. McKay and I. Fujinaga. Automatic genre classification using large high-level musical feature sets. In *ISMIR*, pages 525–530, 2004.

[27] M.F. Mckinney, J. Breebaart, and Prof Holstlaan. Features for audio and music classification. In *ISMIR*, 2003.

[28] T. Mitchell. *Machine Learning*. McGraw Hill Higher Education, 1st edition, October 1997.

[29] F. Pachet and D. Cazaly. A taxonomy of musical genres. In Joseph-Jean Mariani and Donna Harman, editors, *RIAO*, pages 1238–1245. CID, 2000.

[30] D. Pye. Content-based methods for the management of digital music. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 6, pages 2437–2440 vol.4, 2000.

[31] A. Rauber, E. Pampalk, and D. Merkl. Using Psycho-Acoustic models and Self-Organizing maps to create hierarchical structuring of music by sound similarity. In *ISMIR*, 2002.

[32] C. Roads. *The Computer Music Tutorial*. The MIT Press, February 1996.

[33] P. Salembier. *Introduction to MPEG 7: Multimedia Content Description Language*. Wiley, Har/DVD edition, June 2002.

[34] N. Scaringella and G. Zoia. On the modeling of time information for automatic genre recognition systems in audio signals. *IN PROC. INTERNATIONAL SYMPOSIUM ON MUSIC INFORMATION RETRIEVAL*, pages 666–671, 2005.

[35] X. Shao, C. Xu, and M.S. Kankanhalli. Unsupervised classification of music genre using hidden markov model. In *IEEE International Conference on Multimedia and Expo (ICME)*, volume 3, pages 2023–2026, 2004.

[36] M. Slaney. Auditory toolbox techique report (version 2), 1998.

[37] C. Soanes and A. Stevenson. *Oxford Dictionary of English*. Oxford University Press, USA, 2 edition, May 2004.

[38] H. Soltau, T. Schultz, M. Westphal, and A. Waibel. Recognition of music types. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 1137–1140, 1998.

[39] J.V. Stone. *Independent Component Analysis: A Tutorial Introduction*. The MIT Press, illustrated edition edition, September 2004.

[40] T. Tolonen and M. Karjalainen. A computationally efficient multipitch analysis model. *IEEE transactions on speech and audio processing*, 8(6):708–716, 2000.

[41] G. Tzanetakis and P. Cook. MARSYAS: a framework for audio analysis. *Org. Sound*, 4(3):169–175, 1999.

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

[43] G. Tzanetakis, A. Ermolinskyi, and P. Cook. Pitch histograms in audio and symbolic music information retrieval. *Proceedings of the third international conference on music information retrieval (ISMIR)*, pages 31–38, 2002.

[44] V.N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, September 1998.

[45] C. Xu, N.C. Maddage, X. Shao, F. Cao, and Q. Tian. Musical genre classification using support vector machines. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 429–32, 2003.

[46] W.A. Yost. *Fundamentals of Hearing, Fourth Edition: An Introduction*. Academic Press, 4 edition, July 2000.

[47] E. Zwicker and H. Fastl. *Psychoacoustics: Facts and Models (Springer Series in Information Sciences)*. Springer, 2nd updated ed. edition, April 1999.