CHARACTERIZING AND CLASSIFYING MUSIC GENRES AND SUBGENRES VIA ASSOCIATION ANALYSIS

ADAM LEFAIVRE
Bachelor of Science, University of Lethbridge, 2016
Bachelor of Music, University of Lethbridge, 2013

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

CHARACTERIZING AND CLASSIFYING MUSIC GENRES AND SUBGENRES VIA ASSOCIATION ANALYSIS

ADAM LEFAIVRE

Date of Defence: December 13, 2018

Dr. John Zhang Supervisor	Associate Professor	Ph.D.
Dr. Wendy Osborn Committee Member	Associate Professor	Ph.D.
Dr. Yllias Chali Committee Member	Professor	Ph.D.
Dr. Howard Cheng Chair, Thesis Examination Committee	Associate Professor	Ph.D.

To Mom, Dad, and Tom.

Abstract

In this thesis, we investigate the problem of automatic music genre classification in the field of Music Information Retrieval (MIR). MIR seeks to apply convenient automated solutions to many music-related tasks that are too tedious to perform by hand. These tasks often deal with vast quantities of music data. An effective automatic music genre classification approach may be useful for other tasks in MIR as well.

Association analysis is a technique used to explore the inherent relationships among data objects in a problem domain. We present two novel approaches which capture genre characteristics through the use of association analysis on large music datasets. The first approach extracts the characteristic features of genres and uses these features to perform classification. The second approach attempts to improve on the first one by utilizing a pairwise dichotomy-like strategy. We then consider applying the second approach to the problem of automatic subgenre classification.

Acknowledgments

First and foremost, I would like to thank Dr. John Zhang, for the endless hours spent working on papers together, supplying all of the necessary guidance for finishing this thesis, helping me contribute to music technology in some way, and for throwing in some much-needed humour throughout the whole process.

I would like to further extend my gratitude to my committee members, Dr. Wendy Osborn and Dr. Yllias Chali, for the indispensable comments, and advice. Many thanks must also be given to Dr. Howard Cheng, for the assistance regarding specific sections of this thesis, and for clarifying some of the connections between image processing and audio processing. Further appreciation must be given to Jordan Nickorick, for having a coffee with me while discussing music genre taxonomy.

To Mom, Dad and Tom: I would not be where I am today without your support.

To Auntie Debi: thank you for fostering my interest in music (and for introducing me to the blues).

To Nichole: thank you for your constant encouragement and understanding.

To my dear friends from the Mathematics and Computer Science department: I appreciate you guys a lot.

Many thanks to anyone else who supported me along the way, and to anyone else reading this thesis.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Music is a highly social construct that is tied to our evolution as a species and is as fundamental as language. Through it, musicians can express abstract ideas and orient these ideas to specific audiences [80]. The abstraction of these ideas is presented in a temporal form that can be recorded and further analyzed.

Recording technology began with Thomas Edison's invention of the phonograph in 1877, a year after Alexander Graham Bell patented the telephone. Edison stated that the phonograph would eventually be able to reproduce music, preserve languages, etc. [15]. Then, record players became available at the beginning of the 20th century. The new ability to record and distribute audio had an immense impact on composers and the archival of music. Eventually, magnetic cassette tapes and optical Compact Discs (CDs) became available later in the 20th century. However, some of the most disruptive technologies for music were the creation of the Internet and the digitization of music. At this point, storage shifted from physical mediums (e.g., LPs, CDs, and cassettes), back to the seemingly "intangible". This was also further accelerated by innovations in data compression (e.g., MP3, AAC, etc.) and networking. For the sharing of music, the only bottlenecks became the amount of digital storage available, and the transmission of music via the Internet. Peer-to-peer file sharing upended years of record label monopolies, and finally, Apple began selling digital music on their iTunes Music Store in 2003[1]. Most recently, music streaming services (e.g., Spotify, Deezer, and Google Play Music) have gained incredible popularity, for example, Spotify

[1] https://www.wired.com/2010/04/0428itunes-music-store-opens/.

had 83 million paying subscribers in 2018, with over 180 million active monthly users[2]. When using a streaming service, users no longer need to digitally download music and are even more likely to listen to a greater quantity and variety of music [21].

Music listeners require plenty of conveniences from their access to the massive amounts of music data available, such as: playlist and related artist recommendations, proper genre and mood categorization, and easy user querying. Automated approaches must be devised, since it is extremely time-consuming and expensive to have experts label genres or moods, and find similar artists. These are problems that can be solved by *Music Information Retrieval (MIR)*.

Before discussing MIR, it is important to mention that technologies outside of MIR that are still related to the computational analysis of music data have also flourished alongside MIR. Some of these developments include: algorithmic composition, automatic music transcription, and automatic analysis of music scores.

1.1 Music Information Retrieval

MIR was first introduced in the mid-1960's, by Michael Kassler [42], which was ahead of its time. The substantial development of MIR began in the late 1990s, which was followed by the first International Symposium on Music Information Retrieval (ISMIR) occurring in 2000 [13].

MIR offers solutions to problems that, above all, will enhance the listening experiences for music fans by improving their access to large music collections. It becomes an invaluable field when one takes into account the size of online digital music services, the number of listeners, and the current ease of access to vast digital music libraries.

MIR is a field that requires cross-disciplinary perspectives to devise various solutions. Some of these disciplines include: psychology, musicology, audio engineering, signal processing, acoustics, and computer science. Many computational approaches found in MIR

[2] https://www.theverge.com/2018/7/26/17616404/spotify-paid-subscribers-83-million-streaming-music-quarterly-results.

use data mining to extract patterns in large repositories, databases, the Web, or through data streams [37].

The problems that exist in MIR can be solved with several computational approaches in music data mining, as stated by Li et al. [51]. Such tasks can be grouped into various categories: data management, music similarity search, association mining, sequence mining, classification, clustering, and music summarization.

Data management must be done in order to facilitate the access to large music databases. This entails music data indexing, which ultimately allows for efficient querying. This indexing can be based on the extracted acoustic features of music pieces (see Section 2.1.2).

Music similarity search is the task of finding related music pieces to a specific input music piece, or a number of inputted music pieces. This is done by using specific acoustic features that are extracted before searching, or by using higher level descriptions, such as a piece's structure. Then, the distance between the features or descriptors extracted from the input to other music pieces can be calculated. Approaches to this can also be extended to include more than one input music piece. For instance, an artist's album or discography can be used as input to find similar albums or artists. Similarity can also be measured by using different segments from the same piece (e.g., identifying chorus or verse segments).

Association mining, which will be further discussed in Section 2.3, is used extensively in this thesis. Association mining is used to find correlations of various items in a dataset. There are several cases of its use in the MIR literature, including: 1) among various acoustic features, 2) between music and other document formats, and 3) among music features and other aspects of music. We will especially focus on the first and the last use cases.

Sequence mining detects patterns that are represented in a structured temporal sequence. Some examples are to examine the relationships and co-occurrences of chord progressions, rhythmic phrases, melodic motifs, song structure, and especially to find errors in music transcriptions.

Music summarization extracts concise representations of music pieces that can be used

for the organization of large databases. The extraction of a main melody can be done using *Musical Instrument Digital Interface (MIDI)* data, or by extracting acoustic features that may determine song structure through timbral analysis.

Clustering is a method of separating music data into different groups based on some similarity metric, without using any category (i.e., class) labelling. The lack of class labelling makes clustering unsupervised [37]. Acoustic features, lyrics, etc. are typically used for this purpose.

Classification is different from clustering, in that it uses class labels, and is therefore supervised. It is the process of correctly identifying various classes, for each new individual data instance. Classification in MIR takes on various tasks, including: genre classification, mood and emotion classification, instrument recognition, artist classification, and singer recognition. Each of these classification tasks poses different problems. However, if a classification algorithm is effective enough, it may work well in all of these cases. For this thesis, we focus on the problem of genre and subgenre classification.

Of course, any of the above tasks may combine several of the categories proposed by Li et al. [51], for instance, the approach stated in Chapter 3 of this thesis combines association mining with genre classification, and is also useful for the similarity search task.

1.2 Genre Classification

The automatic classification of music pieces into genres began with Tzanetakis and Cook [95], who state that a genre is a label that is placed on a music piece to somehow both summarize its similarity to other pieces sharing the same label, and differentiate this piece from other pieces that do not share this label. Furthermore, they state that a genre label is dependent on instrumentation, rhythmic, and harmonic properties.

Tzanetakis and Cook [95] also state that there is more to merely applying this label based on instrumentation, rhythm, and harmony, stating that historical, marketing, and cultural factors also determine a piece's genre. Therefore, the boundaries between genres

cannot be simplified, so we present some of the difficulties and the benefits of performing automatic music genre classification below.

1.2.1 Difficulties with Automatic Genre Classification

For the problem of automatic genre classification, there are some inherent difficulties that must be addressed. Firstly, many cultural factors come into play, and a genre's characteristics can, to a large extent, be socially motivated, i.e., dependent on such things as: age, race, sex, and further abstractions like political messages, and the sense of belonging to a community [62]. Other issues of genre classification relate to having reliably annotated benchmark data, i.e., whether human annotators completely agree on what they are annotating (disagreements can occur during labelling), the arduousness of manually labelling the genre of each piece in a benchmark dataset, the introduction of new genres and the changing of older ones, etc. [62].

Human classification of genres can also be imperfect. For example, first year university students' (non-music majors) average classification accuracy reached only ~70% (genre labels provided by record companies) for three (3) seconds of audio; a longer duration did not improve the accuracy [62]. This shows both the definite ambiguity of genres and that the ceiling for automatic classification accuracy can only be so high.

Further difficulties arise with the general evaluation of automatic genre classification approaches, and whether or not genres are even accounted for during classification, that is, a classifier may be finding similarities in music data without actually understanding a "genre" (i.e., identifying similar aspects of a recording like loudness, spectral characteristics, etc.) [87, 89]. Furthermore, Sturm [89] points out that various approaches' performance can be deflated (or inflated) quite easily. By using transformations such as very subtle time stretching (i.e., increasing the length of a piece), equalization (i.e., filtering certain frequencies), etc., and performing the classification tasks again, he shows that a proper classification may be dependent on musically irrelevant cues rather than what is consid-

ered "ground truth". Cases may also arise where single-label misclassifications should not necessarily be considered misclassifications, due to the subjective nature, and mixing of different genres [74].

1.2.2 Benefits of Automatic Genre Classification

The task of music genre classification can still be very beneficial for a variety of purposes. Genres are identified with at a cultural level and are crucial to an individual's way of explaining what music they are interested in [62]. Furthermore, listeners prefer to browse music by genres, compared to other methods, such as by similar artists or popularity [48]. Listeners are already accustomed to browsing music by genre, with detailed classifications (i.e., subgenre classifications), being particularly useful [62].

If an automatic genre classification approach is successful in actually determining music genres, it may offer other solutions to MIR problems, such as automatic tag annotation, mood classification, playlist recommendation, and instrument recognition. Finally, and as stated previously, an approach could assist in the curation of large amounts of digitized music data, since human labelling is ultimately expensive and time-consuming.

1.3 Contribution

This thesis first proposes two novel approaches for classifying music pieces into genres. In addition to categorizing music pieces into genres, the approaches are also able to statistically characterize the genres by storing the characteristic features associated with the pieces belonging to each genre. These approaches also offer new methods of applying association analysis to the genre classification problem in MIR. The first approach to be proposed is similar to one proposed by Arjannikov and Zhang [6], however, our approach uses a method of characterization that is genre-specific, without needing to derive specific rules for each genre. We also provide a number of novel ranking criterion to match an incoming music piece to the specific characteristics of a genre. The second approach presents

a new dichotomy-like pairwise comparison between genres that solves several issues related to the first approach and offers a mechanism of improving performance. For these approaches, we design several classification experiments on various datasets that demonstrate their accuracy. Various combinations of parameters are discussed, with regards to performance, so that further research can be done.

Next, we notice that there is very little literature regarding a music genre classifier's ability to perform finely detailed subgenre classification tasks, which can be a much more challenging and problematic task. Since there has not been much work done in this area, we design several replicable experiments that other researchers in MIR can compare against. We test our second approach with a series of subgenre experiments, using two (2) benchmark datasets, and notice that some fine-tuning of parameters may be needed when classifying subgenres due to the similarity between them.

Generally, if a genre classification approach performs successfully, it may assist in further music data management tasks (e.g., archive querying and playlist generation). The work presented demonstrates the usefulness of the above-mentioned approaches for characterizing and comparing genres, so that they can be further employed in industry on large scale digital music repositories and databases.

1.4 Outline

The remainder of this thesis proceeds as follows. In Chapter 2, we formalize the problem of genre classification further, we discuss various content-based features (derived from the audio signal) that are used in future chapters, we formalize association analysis, and we explore various classification methods in the literature, including some ensemble classification methods. Next, we discuss the related works found in the MIR literature concerned with genre classification, subgenre classification, ensemble techniques, and association analysis.

Chapter 3 presents our first approach. We examine how it characterizes genres, and

present various parameter combinations that ultimately affect the classification accuracy. These parameters are found throughout the classification process, starting at the feature discretization stage, where content-based features are encoded as feature-values, through to the stages after genre characterization, where scoring thresholds dictate the final classification accuracy according to the input music pieces used to evaluate our approach.

Chapter 4 discusses the improvement of our first approach, by way of further genre characterization and a dichotomy-like pairwise comparison of genres. Similar to Chapter 3, we examine parameter combinations. We analyze various methods of scoring that are attempted after genre characterization. We also observe how the genre classification accuracy changes when we uplift subgenres to their parent genre's hierarchical level. For example, promoting various subgenres to their parent genres makes certain genres less distinguishable.

In Chapter 5, we take the approach presented in Chapter 4 and attempt to classify various music subgenres using large benchmark datasets. We explore the effects of adding various descriptive features for the classification tasks and provide an analysis on parameter values, finding those parameter combinations that are successful for subgenre classification.

Finally, in Chapter 6 we conclude this thesis by providing an overview of the approaches and results presented in Chapters 3, 4, and 5, while restating our contributions. We also discuss the limitations of our approaches and describe potential future research directions.

Chapter 2

Background

The purpose of this chapter is to familiarize the reader with the overall classification process used in future chapters. This chapter is organized by presenting each section according to the various steps taken in our approaches, as broadly presented below in Figure 2.1, which also represents the standard process of training and evaluating a classifier for various content-based classification approaches in MIR.

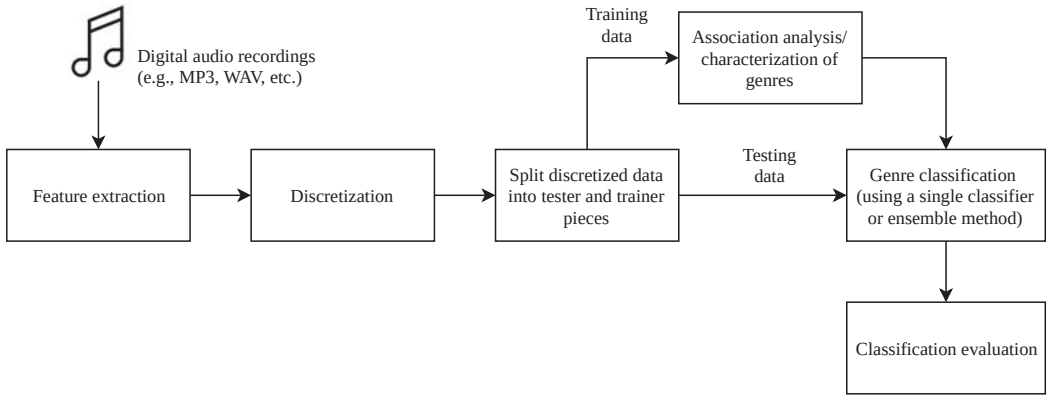


Figure 2.1: An overall depiction of our genre classification approaches.

We begin with a broad description of feature extraction and briefly summarize various features that are extracted from digital audio recordings to summarize the musical content. Next, we present various discretization steps taken in our approach. We then present association analysis in order to clarify how genres are characterized. Next, we discuss various single-classifier classification methods as well as ensemble techniques for classifiers. We then consider the multiple datasets used in our experiments and include the related works in the last section of this chapter.

2.1 Content-Based Analysis of Audio

In our experiments, we solely use content-based features. In this section, we will explore what these are, and how they are derived. Our focus will be on the content-based features that we use in our proposed approaches.

Content-based audio features play a crucial role in summarizing the information of an audio signal. When represented digitally, uncompressed audio requires large amounts of data, with a sampling rate of at least 44.1kHz[3], and often 16 bits per sample (for CD quality audio) for both stereo channels. Content-based features are ultimately used to represent the underlying audio content in a compact form [51]. These features are derived directly from the audio and stored as feature vectors for further processing. Obtaining these feature vectors is the first step taken by many classification systems in MIR and are often the only representation of the audio that a classifier might use [59]. Classifiers in MIR often benefit when there are high quality and numerous features [2, 59]. To represent various aspects of the music, features related to the underlying content of the music are used, such as *rhythmic*, *harmonic*, and *timbral textural* features, in order to recreate how humans perceive music [51]. First, we will discuss various preprocessing steps.

2.1.1 Preprocessing

In the preprocessing stages, certain actions can be taken to improve the performance of a classification system, decrease the complexity of the feature extraction step and the data mining task, or to ensure that all data are represented the same way (sometimes improving classification performance). Two methods of preprocessing will now be discussed: *downsampling* and *normalization*. Downsampling occurs when the sampling rate is decreased. It is generally done to ensure that all audio files have the same sampling rate before feature extraction since various sampling rates can be used to represent audio. However, it may also be used to decrease the computational complexity of the feature extraction process that

[3] 44.1kHz is chosen because of the Nyquist theorem. To prevent aliasing and ensure reproduction of the audio signal, the sampling rate must be twice as large as the highest frequency humans can hear ~20kHz.

follows. Downsampling does cause audio to lose its quality, however the most important low-frequency information is preserved, which is quite valuable for MIR tasks. Normalization occurs when the amplitude is adjusted to some specified value before feature extraction. This removes amplitude variability from recordings [61]. Some other preprocessing steps worth mentioning are: *channel merging*, converting multi-channel audio (i.e., stereo) to mono, and *rectification*, the removal of the negative component of the audio signal [61].

Next, a *short-time Fourier transform (STFT)* can be done to extract various features. Many important features modelled after human hearing are directly based on the STFT. The STFT is a time-localized version of the *discrete Fourier transform (DFT)*. An STFT may be done on evenly distributed time frames of a signal, such that a DFT is performed on each segment. To ensure that discontinuities or distortions, known as *spectral leakage*, at the edges of the windows do not occur, typically a smoothing window, such as a *Hamming*, *Hann*, or *Blackman* window is applied. The output is a spectrum for each window [51]. One purpose of the STFT for feature extraction is to convert from the time domain to the frequency domain, so that spectral features can then be extracted. For a formal definition of the DFT see Lyons [55].

2.1.2 Feature Extraction

After preprocessing is done, features will be extracted separately from analysis window to analysis window (with an STFT performed on each one, if needed). The analysis window size is dependent on the purpose of the content-based analysis that needs to be done and is used to separate the signal into small frames. Small analysis window sizes, ~20ms to 40ms, are typically used to provide a greater description of timbre, while medium analysis window sizes, ~1000ms to 2000ms, can provide a high-level description accounting for characteristics like instrumentation, or the changing of notes, and long windowed features, >2000ms, can provide beat content information [24, 66]. The decision of an analysis window size is a nontrivial consideration since it can directly affect the classification accuracies

of a MIR system [95]. Hop size accounts for how much the analysis window shifts, thus accounting for the amount of overlap between analysis windows. Smoothing is applied to the beginning and end of the analysis window. Normally, after features have been extracted across all of the individual analysis windows, some overall statistics (e.g., mean, standard deviation, etc.), can be calculated and stored in a feature vector.

Since our experiments described in the next chapters are strictly content-based, it is necessary that we define the extracted features used to summarize the music pieces from the chosen datasets. We do not provide an exhaustive depiction of all the features used in MIR. However, many common low-level and mid-level features used in MIR are discussed, since we do use time domain and spectral features. High-level musically meaningful information, such as the amount of chromatic motion [61], is outside the scope of this work.

2.1.3 Some Features Used in MIR

The features presented will be split into several categories: *temporal*, *spectral*, *cepstral*, *pitch-oriented*, *texture window*, and *beat-based* features. Each is used to summarize various aspects of a music piece, as will be described.

Temporal Features

Temporal features specifically use the time domain representation of the audio signal. The analysis windows used are typically the same as those spectral features. This way, the music piece can be summarized consistently. Let $x_t[m]$ be the time domain signal x's value at the sample m, with N total samples in the analysis window t.

Zero Crossings: A way to measure the noisiness of a signal via the amount the signal crosses the midpoint. A crossing of the midpoint is defined as $(x_t[m-1] < 0,\ x_t[m] > 0)$, $(x_t[m-1] < 0,\ x_t[m] > 0)$, or $(x_t[m-1] \neq 0,\ x_t[m] = 0)$ [61]. The zero crossing rate is

measured by the number of times a signal crosses the midpoint, and is defined as

$$Z_t = \frac{1}{2}\sum_{m=1}^{N} |sign(x_t[m]) - sign(x_t[m-1])|, \tag{2.1}$$

where the *sign* is 1 for positive values, and 0 is for negative values [95].

Strongest Frequency via Zero Crossings: A simple way to estimate the dominant frequency for an analysis window is to use the product of the zero crossing rate and the sampling rate *SR* [59, 61]. This is given by

$$SFZC_t = \frac{(Z_t)(SR)}{2N}. \tag{2.2}$$

Linear Predictive Coding (LPC): In music, LPC can be useful for estimating the instrumentation of a music piece [61]. Each sample of the signal is a linear combination of previous samples, plus an excitation, with the coefficients minimizing the mean square error between the prediction of the signal, and the actual signal. In essence, the coefficients are responsible for reproducing the resonances, so that the output is similar to the source. The coefficients are normally returned as a feature vector [61, 72]. The general LPC model is given by

$$S_m = \sum_{k=1}^{p} a_k s[m-k] + Gu[m]. \tag{2.3}$$

Where p is the order of the LPC filter, or the previous p speech samples, a_k is the k^{th} predictor coefficient, $s[m]$ is the m^{th} sample, $u[m]$ is the m^{th} excitation signal, and G is used for prediction error [72].

Spectral Features

Spectral features summarize the spectral domain of a given music piece and they are derived after taking the DFT of an analysis window. Since the spectral domain is concerned with frequency content, these features are useful for describing timbral characteristics. For

example, they can describe the instrumentation's timbre [51]. For the following descriptions below let $M_n[k]$ and $P_n[k]$ refer to the magnitude spectrum and power spectrum, at frequency bin number k, at the n^{th} analysis window. Similarly, let $X_n[k]$ represent the k^{th} frequency bin at analysis window n. Finally, let N be the highest index for frequency bins.

Magnitude and Power Spectrum: The relationship between the power spectrum and the magnitude spectrum is defined by Lyons [55] as

$$P_n[k] = M_n[k]^2 = X_{n_{re}}[k]^2 + X_{n_{im}}[k]^2, \tag{2.4}$$

where *re* and *im* respectively refer to the real and imaginary component of the DFT for the n^{th} analysis window's k^{th} frequency bin. Normally, it is only by convention to use the magnitude spectrum as a way to describe spectral features, as the spectrum itself, or the power spectrum, can also be used for extracting features. The magnitude spectrum is especially effective for describing lower energy spectral activity [61]. Furthermore, the energy of an analysis window is the averaged summation of power across the frequency domain bins in that analysis window, or the summation of power across the samples in the time domain for that analysis window (according to Parseval's theorem) [55, 84].

Spectral Centroid: Spectral centroid is used to estimate the perceptual "brightness" of the audio signal. In order to estimate this, the balancing point of the STFT's magnitude spectrum is extracted. It is formally defined as

$$C_n = \frac{\sum_{k=1}^{N} M_n[k] \cdot k}{\sum_{k=1}^{N} M_n[k]}. \tag{2.5}$$

Spectral Contrast: Spectral contrast is used to identify the difference between peaks and valleys for each individual frequency sub-band. This is done by dividing the spectrum of an analysis window into frequency sub-bands, and finding the difference between the

strength of spectral peaks and valleys [40]. Peaks in spectral contrast features correspond to harmonic components, and valleys correspond to noisier sounds. Spectral contrast is given formally as

$$SC_s = Peak_s - Valley_s, \tag{2.6}$$

where $Peak_s$ and $Valley_s$ are defined as

$$Peak_s = log\left(\frac{1}{\alpha N}\sum_{i=1}^{\alpha N} x_{s,i}\right), \tag{2.7}$$

$$Valley_s = log\left(\frac{1}{\alpha N}\sum_{i=1}^{\alpha N} x_{s,N-1+i}\right). \tag{2.8}$$

The DFT of the s^{th} sub-band is of a vector form, sorted in decreasing order of magnitude (i.e., $\{x_{s,1}, x_{s,2}, \ldots, x_{s,N}\}$). N is the number of DFT bins for the s^{th} sub-band and α is a neighbourhood constant for the sub-band, normally between 0.02, and 0.2. $s \in [1,6]$ [40].

Spectral Flux: Spectral flux is used to measure the amount of spectral change from analysis window to analysis window [61]. It is defined as

$$SF_n = \sum_{k=1}^{N} (M_n[k] - M_{n-1}[k])^2. \tag{2.9}$$

Spectral Rolloff: Spectral rolloff (SR) is the frequency below some fraction C, where the magnitude distribution is the most concentrated [95]. C is normally set to 0.85 or 0.95. Spectral rolloff shows the amount of energy at the lower frequencies, and will satisfy the inequality [61]

$$\sum_{k=1}^{SR_n} M_n[k] = C \cdot \sum_{k=1}^{N} M_n[k]. \tag{2.10}$$

Peak-Based Spectral Smoothness: This feature uses an average of the set of surrounding spectral peaks (i.e., the neighbours on either side), and calculates the difference between the log of the peak itself and the average of the logged peaks in the neighbourhood for a given

analysis window. The peaks are chosen algorithmically [58, 61]. Spectral smoothness is calculated as

$$SS_n = \sum_{m=2}^{M-1} \left(log(T_n[m]) - \frac{log(T_n[m-1]) + log(T_n[m]) + log(T_n[m+1])}{3} \right), \tag{2.11}$$

where $T_n[m]$ is a given peak, with $M \geq 3$ being the number of spectral peaks.

Strongest Frequency via FFT Maximum: This is a simple feature that finds the strongest frequency of the audio signal in Hz. Since features are extracted via analysis windows, this can be a simple method of tracking pitch changes [61].

Strongest Frequency via Spectral Centroid: Another way to find the strongest frequency of a signal is to derive it from the spectral centroid using the ratio to calculate the spectral centroid and mapping it to a frequency [59, 61].

Method of Moments: The area, mean, spectral density (or variance), skew, and kurtosis (or "peakedness") of the magnitude spectrum can be calculated and stored in a feature vector [61]. The higher ordered the moment calculation is, the more descriptive it is, however, lower ordered moments offer a concise description [32]. Note that skew and kurtosis can also be used to describe the overall characteristics of the spectral features after all of the features in each analysis window have been calculated (just like the mean and standard deviation), this is used for our experiments in Section 5. Let μ_n, σ_n^2, SK_n, and KU_n represent mean, variance (the twice power of the standard deviation), skew, and kurtosis of the n^{th} analysis window, respectively. The moments mentioned are given as

$$\mu_n = \frac{1}{N} \sum_{k=1}^{N} X_n[k], \tag{2.12}$$

$$\sigma_n^2 = \frac{1}{(N-1)} \sum_{k=1}^{N} (X_n[k] - \mu)^2, \tag{2.13}$$

$$SK_n = \frac{1}{\sigma^3 (N-1)} \sum_{k=1}^{N} (X_n[k] - \mu)^3, \tag{2.14}$$

$$KU_n = \frac{1}{\sigma^4 (N-1)} \sum_{k=1}^{N} (X_n[k] - \mu)^4 - 3. \tag{2.15}$$

Cepstral Features

Cepstral features are used to summarize the spectral envelope even further than spectral features. Also, there is some additional processing that typically needs to occur. In MIR, Cepstral features are dominated by the usage of *Mel-frequency cepstral coefficients*.

Mel-Frequency Cepstral Coefficients (MFCCs): MFCCs are a prevalent mid-level feature in MIR research. They were first introduced by speech recognition researchers, and are perceptually motivated, attempting to replicate how humans hear. This is due to the higher concentration of triangular Mel-scale filters for low-frequency bins when deriving MFCC features[4], and the use of the Mel-frequency scale, which accounts for humans' lack of ability to distinguish between lower frequencies, compared to higher frequencies. A typical conversion, given in [73], maps from Hz to Mels, with a steeper logarithmic mapping for lower frequencies. Let f represent some frequency, with m as the mapped Mel-scale value; the Mel-scale conversion is then defined as

$$m = 2595 log_{10}\left(1 + \frac{f}{700}\right). \tag{2.16}$$

After the DFT for a particular analysis window has been taken, the typical steps for extracting MFCC features, are: 1) The log amplitude of the magnitude spectrum is taken, 2) DFT bins are grouped and mapped to the Mel-frequency scale, and lastly, 3) a *discrete*

[4] According to Fletcher and Munson [28], human hearing is most sensitive around 2-4kHz, and less sensitive for lower frequencies, due to the resonance properties of the auditory canal.

cosine transform (DCT) is performed for each grouping and the DCT coefficients are stored in a feature vector. The log amplitude is taken to reduce potentially detrimental dynamic fluctuations. A DCT is similar to a DFT, except it uses only real numbers, and will describe the spectrum as a sum of cosine functions. A DCT is chosen because the Mel-scale filter-bank analysis windows overlap, and it is very good at decorrelating energies. Usually the lowest 13 coefficients are kept, and the coefficients corresponding to higher frequencies are discarded [51, 95].

Derivatives of MFCCs: The derivatives of MFCCs describe how much and how often the MFCCs change temporally. In general, finding the derivative is done discretely, by calculating the difference between the coefficients of analysis windows [59].

Pitch-Oriented Features

Pitch-oriented features can be derived from the spectral domain by mapping frequency content into pitches. This is especially useful for extracting features that are more musically intuitive in nature.

Chroma Features: Chroma features are widely used in the MIR community, and this thesis would be incomplete without mentioning them. Chroma features map the frequency domain bin to the nearest equally tempered frequency corresponding to a chromatic scale note (usually folded down to an octave), by computing the energy at each chromatic scale frequency. A pitch histogram can be constructed across the analysis window or the whole signal [51]. Certain musical characteristics can then be explored, such as: *key*, *harmony*, *interval* structures, and *melodies*. By mapping frequencies to pitch bins, as described above, the calculation of temporal attributes related to the bins can also be made. One such example is to find the *maximum period of the pitch peaks* [96].

Texture Window Features

Another strategy for feature extraction is the use of a *texture window*, which is a method of temporal feature integration. This is done by summarizing the information in the current analysis window and all of the previous analysis windows into a single feature vector, using a Gaussian model, for example. The number of past analysis windows chosen for the integration is dependent on the description needed, as adding more past analysis windows gets closer to a *song-level* analysis [51, 66].

In our experiments, we make use of one feature that is based on texture windows, i.e., *low-energy*. The following feature is common to include and has been used in the seminal works of Tzanetakis and Cook [95] and Silla et al. [41].

Low Energy: This feature describes the percentage of windows with a lower *root mean square (RMS)* energy than the texture window's average RMS energy. RMS is defined as the average energy of a window and is given by

$$RMS = \sqrt{\frac{1}{N}\sum_{m=1}^{N} x[m]^2}. \tag{2.17}$$

Low energy is also referred to as the *fraction of low energy frames* and may use some threshold different from the texture window's average RMS energy. For our purposes, it is the feature based on the texture window's RMS value [95].

Beat-Based Features

Beat-based features are an attempt to portray some high-level rhythmic information. They are still technically content-based, and can only give some estimation of note onsets and durations. Beat-based features make use of a *beat histogram*, which is constructed using *autocorrelation*. Brown [12] proposed the use of autocorrelation in music, and it was further popularized in MIR by Tzanetakis and Cook [95]. The definition of autocorrelation is

$$A[d] = \frac{1}{N} \sum_{m=1}^{N} Y[m]Y[m+d], \tag{2.18}$$

where m is the m^{th} sample point, N is the total samples, Y is a function applied to determine the autocorrelation value (this is explained shortly), and lastly, d is all integer values between $0 \leq d < N$. The value d exploits the time domain of the signal since it is used to calculate A for an evenly spaced number of samples. The overall value for A at the bin d will provide an indication of the strength in periodicity for that bin d.

The function Y is the representation of the signal after any number of preprocessing steps. For example, McEnnis et al. [59] implement an RMS function on the original signal. However, Tzanetakis and Cook [95] apply a pyramidal algorithm using *Daubechies wavelets* to perform the *discrete wavelet transform (DWT)*, which takes the signal and decomposes it in frequency using octave-spaced filters [95]. Next, they perform: *full wave rectification*, *low pass filtering*, downsampling, and normalization on the signal. Finally, they apply the autocorrelation function.

After the creation of the beat histograms using autocorrelation, several features can be extracted [59, 61, 95], including the following:

- *Beat Sum:* the sum of all the histogram bin values.
- *Strongest Beat:* the bin with the highest value in the beat histogram.
- *Strength of Strongest Beat:* the strongest beat divided by the beat sum.
- *Relative Amplitudes:* a vector of each bin value, divided by the beat sum.
- *Beats per Minute (BPM):* using each bin label to determine the periodicity of a music piece. This is done by detecting peaks returned by the autocorrelation function.

2.1.4 Existing Feature Extraction Toolboxes

Since feature extraction is such a fundamental step in MIR for various content-based tasks, researchers have created many toolboxes, such as: *stand alone applications*, *host application plug-ins*, or *software function libraries* [67]. There are possible benefits to using

a set of particular feature extraction tools, such as: the option of using a *graphical user interface (GUI)* or *command line interface (CLI)*, reduced computational complexity, the programming language it is implemented in (useful for the case of *application programming interfaces (APIs)*), the ability to perform batch processing, or the file output format (e.g., *attribute-relation file format (ARFF)*). In the experiment sections of this thesis, three feature extraction toolboxes are used.

jAudio [59]: jAudio is a standalone Java application that provides a GUI, CLI, and the ability to perform batch processing for a large number of audio files. It is able to extract only low to mid-level features (not counting beat-based features) [59, 67]. We use jAudio for its convenient batch processing, the selection of features it can extract, its CLI, and its ability to output into the ARFF format.

MARSYAS [94]: MARSYAS is a standalone framework written in C++. It is computationally efficient and offers a CLI. There are fewer features that can be extracted compared to jAudio, however, it can extract both high-level and low-level features [67]. It is chosen, by default, due to its use in several MIR papers, where the benchmark datasets are publicly available with features already extracted [41, 91].

LibROSA [60]: LibROSA is a feature extraction API for the *Python* programming language. It provides both high and low-level features. It is less efficient than MARSYAS and jAudio, and its interfacing is limited to API function calls [67]. LibROSA is used extensively in the extraction of features from the *Free Music Archive (FMA)* [9].

2.2 Discretization

The next step after the extraction of features is the *discretization* (or *binning*) of these features into *feature-values*. Frequent itemset mining via market basket analysis is used to

derive interesting patterns with discrete values only. Therefore, some method of converting real-valued features into discrete representations is necessary. In this section, we refer to the features in a vector of extracted features for a music piece as *attributes*, a music piece as a data *instance*, and the genre of a music piece as its *class*.

Discretization reduces the continuous values present in the data by dividing the range of the data attributes into a certain number of intervals. The labels for each interval are then used to replace all of the values that fell within that interval. Each interval is called a particular *bin*. *Supervised* and *unsupervised* discretization are the two types of binning most commonly used. Supervised discretization uses each data instance's class label (in our case the class label is the genre), whereas unsupervised discretization does not use a class label when splitting attribute values into bins. It is often the case that supervised binning increases classification accuracy, due to the binning of data based on class information [37].

Discretization can be a *top-down* or *bottom-up* approach. Top-down discretization finds several initial *cut points* for each bin, and then recursively discretizes these bins with further cut points. Bottom-up approaches will merge neighbourhood values with each other, and apply this process recursively.

Equal frequency and *equal width* binning are both examples of unsupervised, top-down binning methods, since they do not use a class label and find cut points based on a specified number of bins. Other various binning methods include: *entropy-based* discretization (supervised, top-down), *ChiMerge* (supervised, bottom-up), etc. For the interested reader, more binning methods and their details are found in [37].

Since we only use the equal frequency and equal width binning methods, we now direct our attention to these. At this time, only these binning methods are used, we leave further binning implementations to our future work. We use the API provided by the *Waikato Environment for Knowledge Analysis (Weka)* [30] for the implementation of the following binning methods.

Equal Width Binning

Binning based on equal width is to divide the value range of a continuous acoustic feature into a certain number of N bins. If we let w_i be the width of the bin for attribute A_i, with D total data instances, then the formal definition of equal width binning is given by

$$w_i = \frac{max(A_i) - min(A_i)}{N}. \tag{2.19}$$

Rice Rule Binning

There are other ways to determine the value of N instead of relying on a user to set it. One variation of equal width binning assumes the normal distribution of data points in a dataset and uses the so-called "Rice rule" [45] to determine N automatically, such that

$$N = \lceil 2 \cdot D^{1/3} \rceil. \tag{2.20}$$

Equal Frequency Binning

Binning on equal frequency generates N number of bins such that each bin contains the same number of attributes from different instances. If N is large, then the number of attributes that will fall into a particular bin will be smaller. If N is small, there will be a greater number of values per bin. Equal frequency binning addresses one of the issues of equal width binning, where specific bins may be empty due to the attribute values falling between only a subset of the total intervals specified.

2.3 Association Analysis

Association analysis is first proposed by Agrawal et al. [1]. It is used to discover interesting relationships and correlations within large datasets. In a problem domain, a set of data items that "frequently" occur together shows some statistical relationship among them. The example of *market basket analysis* is often given in this context, where businesses are interested in discovering what consumers are buying together frequently in supermarkets,

e.g., a number of customers could buy $\{bread, milk, honey\}$. If $\{break, milk, honey\}$ is purchased over a certain percentage of the time, we can consider it as frequent. Furthermore, association rules could provide evidence that if a customer buys bread and milk, then they are likely to buy honey too. This would be given as $\{bread, milk\} \Rightarrow \{honey\}$. The general steps for association analysis include *frequent itemset mining* and *association rule generation*. For our purposes, we are strictly concerned with frequent itemset mining.

2.3.1 Frequent Itemset Mining

Frequent items are put into *frequent itemsets* (i.e., a 3-item itemset means the three items in the set occur frequently together). The *support* of an itemset is the percentage of the co-occurrence of its items in the dataset. Only those itemsets whose support exceeds a threshold called *minimum support* (m_s), are considered frequent. We will now formalize the above concepts according to Han and Kamber's notation [37], as well as Agrawal et al.'s [1].

Let $I = \{I_1, I_2, \ldots, I_m\}$ be the set of all items, let T be a transaction from the set of transactions D, such that $T \subseteq I$. Let A be some set of items. A transaction T contains A if and only if $A \subseteq T$. Then, support is defined as

$$support(A) = \frac{|\{T : A \subseteq T, T \in D\}|}{|D|}. \tag{2.21}$$

The support of an itemset A is the percentage of transactions where A occurs over all of the transactions. The returned k-itemsets after applying the m_s threshold are said to be frequent itemsets. They are denoted as L_k, where

$$A \in L_k \iff support(A) \geq m_s : A \subseteq I,\ 1 \leq |A| \leq k. \tag{2.22}$$

2.3.2 Closed and Maximal Itemsets

When analyzing frequent itemsets in a large dataset, if m_s is set very low, there could be an explosion of frequent itemsets returned, as any subset of a frequent itemset is also frequent. To lessen the number of itemsets *closed frequent itemsets* and *maximal frequent itemsets* are used. An arbitrary closed frequent itemset X is an itemset with a support greater than the minimum support and has no proper super-itemset J existing. That is, all supersets must have a lower support if they are frequent, where $X \in L_k$. We formalize this relationship as

$$\forall J \supseteq X : \ support(X) > support(J). \tag{2.23}$$

A maximal frequent itemset is similar to a closed frequent itemset, except that the super-itemset J cannot be frequent. It is formalized as

$$\forall\, J \supseteq X : \ support(J) < m_s. \tag{2.24}$$

Thus, we see that maximal frequent itemsets have the most restrictive limitations placed on them, and can decrease the number of frequent itemsets in consideration [11].

2.3.3 The Apriori Algorithm

In our approaches, we adapt *Apriori*, a classical association analysis algorithm [1]. In this algorithm, the *Apriori property* is used to exploit the fact that all nonempty subsets of a frequent itemset also happen to be frequent. So to use our analogy again, if $\{bread, milk, honey\}$ is a frequent itemset, $\{bread, milk\}$, $\{bread, honey\}$, and $\{milk, honey\}$ will be frequent too. The classical implementation of the Apriori algorithm typically consists of two main steps, the join step and the prune step.

The join step and prune step are described as follows. For every iteration that the Apriori algorithm executes, L_{k-1} (a set of frequent itemsets) is joined with itself, i.e., $L_{k-1} \bowtie L_{k-1}$,

giving a new set of candidates C_k. If we let two itemsets be named l_1 and l_2, and assume they are sorted lexicographically, we say they are "joinable" if the first $k-2$ items are the same in each itemset. For example if $l_1 = \{a,b,c\}$ and $l_2 = \{a,b,d\}$, these are said to be joinable, and produce $\{a,b,c,d\}$. To achieve C_k all of the pairs of itemsets in L_{k-1} are tested to be joinable, and if they are, then they are included in C_k. Once C_k is achieved, a scan is made of all of the counts for each itemset in C_k, if they have a lower support than m_s they are removed from C_k. Here the Apriori property is used by removing itemsets whose subsets are not frequent. In summary, the general steps of the Apriori algorithm are to generate the candidates for L_1, and then prune those candidates with a lower support than the minimum support. Then, to perform a join step and prune step iteratively until there are no more frequent itemsets to be found. All of those frequent itemsets surpassing the minimum support are returned. For a more detailed description of this algorithm, please see Agrawal et al. [1].

Usage of the Apriori Algorithm

Throughout our experiments, we ensure that all itemsets are maximal and consisting of at least two items, in order to increase the efficiency of our approaches. We use Christian Borgelt's [11] implementation of the Apriori algorithm. The interested reader can refer to Goethals and Zaki's [35] comparison of Borgelt's implementation against others.

Other criteria can be used once association rules are mined, such as lift, or conviction. Several methods of improving the efficiency of the Apriori algorithm are also well-known [37].

2.4 Off-The-Shelf Classification Methods

The problem of determining a music piece's singular genre from a set of genres is known as the *single-labelled* genre classification problem. It is one of the simplest tasks in MIR. This does not make the task any less challenging since a music piece could be a multitude

of genres (e.g., *Latin jazz*, *folk rock*, etc.). As mentioned in Section 1.2.1, single-labelled misclassifications should be treated with some importance as well. In contrast, the task of *multi-label* genre classification is used to determine several matching genres from a set of possible genres for a music piece.

Since our approaches deal with only single-labelled classifications, we will briefly discuss several typical classification approaches that are used for this same problem. Most of these classification approaches have existed for some time and are prevalent in MIR.

2.4.1 Support Vector Machines

Support vector machines (SVMs) classify data with the use of hyperplanes. A hyperplane is a line (or plane, depending on the dimensionality) used to separate classes. They can be generalized to any dimension, with data that is linearly separable. Linear separability refers to data that can easily be split into two classes given some hyperplane. The best hyperplane to choose is the one that creates the most distance between both classes. The distance from the closest item from the hyperplane should be maximized for both of the classes, and the weight to the hyperplane should be minimized for both of the classes. The support vectors are the feature vectors that are the closest to the hyperplane, they often provide most of the information and are the crux of the classification mechanism [51].

The method of classifying non-linear data is given by a two-step approach. If the data can not be classified in the current dimension, then a further dimension is introduced to help separate the data. The mapping of a lower dimensional representation of the data to a higher one is done by a *kernel function*. Some notable kernel functions are *radial-basis functions*, *polynomial*, and *sigmoid* [69]. SVMs are generally more computationally complex but are more accurate compared to other classification approaches [51].

2.4.2 Gaussian Mixture Models

A *probability density function (PDF)* of instances is derived for each class, specifically, the PDF of each class is treated as a mixture of Gaussian distributions with parameters

that are unknown. An iterative algorithm, such as an *expectation-maximization* algorithm is used to estimate the parameters of each Gaussian component and mixture weights by finding local extrema [95]. Essentially, a *Gaussian mixture model (GMM)* classifier will model the PDF of a testing instance as the linear combination of multivariate Gaussian PDFs. An instance will belong to a class if it maximizes the probability of that class' GMM model producing that instance [79, 95].

2.4.3 Decision Trees

Each node of a *decision tree* classifier is a test on an attribute, with each leaf node representing a class. Essentially, a number of simple attribute-based decisions are used to model the difference of classes. Decision trees generally work as follows. A root node is created (if all training data has a different class the root becomes a leaf). Next, either an attribute is selected to be split on, or in the continuous case, a cut point is determined, with an attempt to make all of the instances of a partition belong to one class (i.e., as "pure" as possible). Then the partitioning occurs for the node's child nodes recursively. The recursive stopping conditions are when all instances belong to the same class (i.e., purity is achieved), no more attributes can be used for splitting, or there are no instances left for the next node.

The attribute to split on can be determined by various statistical methods. For example, it is done using *information gain* in the ID3 model, by the *Gini index* in the CART model, or with *gain ratio* in the C4.5 (or J48) model [37].

2.4.4 k-Nearest Neighbours

A *k-nearest neighbours (kNN)* classifier differs from the other classification methods discussed so far, in that it does not create a specific model of the data for classification. Therefore, it is considered to be a *lazy* classification method. Furthermore, no statistical distributions are assumed (i.e., *non-parametric*). Training instances are stored, and a testing instance's distance from the training instances will be calculated. The nearest number of instances to the testing data are referred to as its k-nearest neighbours. The distance

can be measured using *Euclidean*, *Manhattan*, *Tanimoto*, or *Mahalanobis* distances. kNN classifiers scale well, since they only need to store training information, and therefore can be updated easier. With the standard kNN approach the complexity of actually classifying instances can be quite intensive since the distance for each feature needs to be calculated, however, there are methods of improving the computation time [61].

2.4.5 Linear Regression

Linear regression is a simple prediction method that uses weight coefficients for each continuous feature in a feature vector. The idea is to solve for these weights by minimizing the *least square error*, giving an estimate for the line of best fit for each instance. *Linear logistic regression* is used as a classification method frequently. It uses a linear function of prediction weights, and the *method of maximum likelihood* to achieve those weights [36, 37]. There are plenty of other types of regression, including: *nonlinear*, *Poisson*, and *log-linear* [37].

2.4.6 Naive Bayes

A *naive Bayes* classifier is based on Bayes theorem. Bayes theorem is used to maximize the posterior probability that a testing instance belongs to a class. It is assumed that values of attributes are not dependent on one another (i.e., *class conditional independence*). Class labels are predicted by finding the highest probability of an instance given a specific class multiplied by the probability of that class occurring in the dataset. These classifiers are known to achieve similar accuracies to the classifiers discussed above.

When class conditional independence is not assumed and dependencies between attributes are found through probability distributions, this becomes a *Bayesian belief network*, where the arcs of a *directed acyclic graph* correspond to the probability that the two attributes are dependent on each other in some way, and are stored in a table [37].

2.4.7 Neural Networks

In MIR and other audio processing, it is common that *neural networks* be trained with the *spectrograms* of an audio signal. A spectrogram is created by performing many STFTs on a number of windows, then each frequency spectrum is used as a function of time to create an overall depiction of the signal. *Neural networks* are modelled after neural connections found in the brain. They are comprised of individual *perceptrons* (or with *sigmoidal* neurons which use an activation function) and weighted edge connections to the next neuron. One or more intermediate layers typically reside between the input and output neurons, and the output layer feeds back into the input layer a number of times. Neural networks can learn the "optimal" set of weights per edge by employing a *cost function*, and *backpropagation*, which will typically use a *gradient descent algorithm* which tunes the weights of individual neurons. Various neural network architectures are: *multilayer perceptrons* [71], *recurrent neural networks* [26, 27], *conditional neural networks*, *convolutional neural networks*, *masked conditional neural networks* [64, 65], *restricted Boltzmann machines* [46, 98], and *deep belief networks* [38, 98].

2.5 Ensemble Techniques for Classification

The technique of combining a multitude of classifiers for a single classification problem is called *ensemble-based classification*. The individual classifiers in the ensemble are called *base learners*, and the proper combination of them will typically allow for a better performance, assuming they are diverse enough to handle various data anomalies. In the case that some base learner performs poorly (i.e., has a high error rate), with a proper amount of diversity, the overall classification performance could remain stable. This is the benefit of using an ensemble method [61].

2.5.1 Bagging

Bagging (or *bootstrap aggregation*) is a simple ensemble technique based on the concept of a *majority vote*. Majority voting is such that each individually trained classifier only gets one vote toward which class a training instance is predicted to be. That is, a certain finite number of classification models are found by training them each with different subsets of the training data, this is called *sampling with replacement*. It is very often the case that an increase in accuracy is found, as the ensemble method is less affected by noisy data [37].

2.5.2 Boosting

Boosting can be thought of as a way to weight individual classifiers for more accuracy, instead of allowing each of them to have the same vote. Furthermore, each classifier is trained on a subset of the training data iteratively, that is, multiple passes of training the classifiers are performed. Each time training is done the classifiers provide increased weighting to the misclassifications that they performed in the last iteration, this provides a way of measuring the error rate per classifier (which can further influence the instance weights in future iterations). Finally, when the votes are counted from each classifier, it gives a chance to weight the less accurate classifiers less. A well-known boosting algorithm is the *Adaboost* (or, *Adaptive Boosting*) algorithm [37].

2.5.3 Stacking

Stacking differs from the above ensemble techniques as it takes the output of one or more classifiers, or *base learners*, and outputs the final classification in a way that prevents the base learners from actually voting directly. Intermediate steps weight the base learners to determine how they make errors relative to one another. This method is used with various neural net architectures, such as multilayer perceptrons [61].

2.5.4 Other Related Ensemble Methods

Since our approach in Chapter 3 is modified by introducing the following *pairwise dichotomy-like* strategy in Chapter 4, we must briefly explain how these concepts work in their regular context. Both pairwise (or *round robin*), and dichotomy classifier ensembles are normally used in the multilabel classification problem, where the objective is to find the correct number of labels for a given instance, not just one class label (as we are dealing with in this thesis).

The technique of training pairs of classifiers is useful when binary classifiers are used. A pairwise strategy will convert the task of classification into a number of binary classification steps, where a single classifier is trained to recognize the difference between one class and all others (OvA), or between two classes (OvO) [33]. In our case, we only use the OvO scheme, where the output is a vote towards one class or the other. A nested dichotomy can then be used to create a classification structure, where each internal node is associated with a particular class, and the other instances of that node that belong to other classes separate into different child nodes, each with a different associated class from their parent [31].

2.6 Datasets

In order to verify that a music genre classification approach is successful, data is needed for its training and testing, whether the approach is using association analysis or not. There are now many benchmark music genre datasets available for validating and comparing classification approaches. The genre datasets discussed below do not provide an exhaustive list of all of the possible genre datasets, as we see in Section 2.7. However, the following benchmark datasets are widely known in the MIR community and lay the foundation for genre classification tasks.

2.6.1 GTZAN Dataset

The *GTZAN* dataset is one of the most influential and most used datasets in music genre recognition tasks. So for completeness we must mention it in this section. The GTZAN dataset is comprised of 100 pieces across ten (10) genres. The 10 genres are: *classical*, *country*, *disco*, *hip hop*, *jazz*, *rock*, *blues*, *reggae*, *pop*, and *metal*. The GTZAN dataset is one of the earliest genre datasets and is cited a very large number of times in MIR papers [88]. However, it does have a number of well-known faults which must be taken into account when classifying music pieces into genres.

Sturm [88] states that classification accuracy can only be so high, and approaches should be tested on a wide variety of datasets, not just GTZAN alone. There are several reasons for this, including the disparity of the number of artists in a genre, for example, half of the reggae pieces are comprised of Bob Marley recordings, but disco has 55 different artists. There are also numerous instance repetitions, mislabellings, and distortions.

2.6.2 Free Music Archive

The *Free Music Archive* is derived from music pieces found in the actual Free Music Archive[5], which is an open library curated by the radio broadcasting company *WFMU*[6]. In total there are 106, 574 pieces in the dataset, with 16, 341 artists. It is the most recent dataset used in our experiments (2017), and also presents the best picture of how a classifier might perform given "real" data, since all pieces are dumped from the original dataset without concern for audio quality, song length, number of labels, sampling rate, etc. There are 16 top genres (e.g., rock, jazz, country, etc.), and 145 subgenres (145 labels total), each label is supplied by a piece's respective artist. The dataset also comes with potential subgenre labels per music piece, and an option is given to download the audio files in their entirety (full song lengths), which may provide experimentation for various tasks in MIR, such as song structure analysis [9].

[5] `http://freemusicarchive.org/`.
[6] `https://wfmu.org/`.

2.6.3 Cal10k Dataset

The *Cal10k* dataset, previously known as the *Swat10k* dataset, is comprised of 10, 267 music pieces from 4, 597 artists, with all genres hand-labelled by musicologists. There is a high level of agreement between musicologists, so this dataset is considered "objective". The genre tags are extracted via *Pandora's Music Genome Project*. There are 153 unbalanced, weakly-labelled genres tags per piece (meaning that when a tag is absent, a genre still might apply to that piece). D_{CAL} can allow for easy subgenre analysis since there are so many tags. To be exact, there are 18 main genres (e.g., jazz, blues, classical, rock, hip hop, etc.), and 135 subgenres [91].

2.6.4 Greek Audio Dataset

The *Greek Audio Dataset* is a rather small dataset, similar in size to the GTZAN dataset. It consists of 1000 Greek music pieces. The following genres included in the dataset are *rembetiko*, *laiko*, *entexno*, *modern laiko*, *rock*, *hip hop and rhythm and blues (rnb)*, *pop*, and *enallaktiko*. These genres present both a combination of traditional Greek music with some European influence (e.g., laiko) and more modern genres, such as enallaktiko, which is a combination of various Greek elements and alternative rock. There are 277 unique artists, so there is some artist overlap for each of the 1000 pieces.

One significant problem with the dataset is that the class distribution is skewed, and as already discussed, there is substantial artist repetition. The most music pieces are found in the genres rock and entexno (195), and the least are found in the genre enallaktiko (60) [56]. There may be further errors in the dataset that have not been addressed yet, as it is both recent, and not as heavily cited.

2.6.5 Latin Music Database

The *Latin Music Database* is quite a popular dataset in the MIR literature, and is used as a benchmark for other classification approaches [90]. There are 3, 227 music pieces in total, divided into ten (10) roughly balanced genres. Each genre is classified by experts as

one of: *axé*, *bachata*, *bolero*, *forró*, *gaúcha*, *merengue*, *pagode*, *salsa*, *sertaneja*, or *tango*. Most of these genres have deep societal and musical roots, with specific instrumentation and origins (e.g., Mexican, Brazilian, Puerto Rican, Afro-Cuban, etc.) Silla et al. [34, 83] plan on releasing subgenres of the Latin Music Database in the future.

Some important criticisms of the dataset are given by Sturm [90], which will now be discussed. One problem is that 7% of all its data is repeated. Another issue is that there is a skewed distribution of genre labels for the music pieces within particular folds, which acts too heavily on certain misclassifications (one missed classification can drop an approach's accuracy by several percentage points). Furthermore, the recording medium is quite specific for some genres, that is, genres like tango have a large number of older recordings, so a classifier may base the performance on the noise level in the recordings, and not the musical content itself.

2.7 Related Works in MIR

Now that we have briefly described the background information needed to further understand our approaches, as well as other single classifier, and ensemble classification methods, we will now describe various approaches in MIR which are related to our work. We begin with the problem of genre classification, followed by the more detailed problem of subgenre classification. Next, we acknowledge some ensemble techniques that have been used for genre classification since one of our approaches is ensemble-like. Finally, we observe the few approaches utilizing association analysis in MIR. There are relatively few subgenre classification attempts for typical genres (e.g., jazz, classical, rock, etc.), and also few implementations of association analysis in the MIR literature. Therefore, this section is used to further demonstrate the contribution of this thesis in the current landscape of MIR publications.

2.7.1 Genre Classification in MIR

Music genre classification is a core problem in MIR. Initial work is proposed by Tzanetakis and Cook [95], who provided the GTZAN dataset, which is now a well-used dataset, despite its deficiencies [88]. Tzanetakis and Cook also propose the usage of timbral, rhythmic, and pitch-based features. Since then, there has been plenty of work done for this task. Some notable works will now be mentioned.

An earlier paper from Ogihara et al. [50] proposes using *Daubechies wavelet coefficient histograms (DWCHs)* for features. Daubechies wavelet filters allow for the decomposition of audio signals into sub-band signals at different frequencies, where the wavelet coefficients are distributed across different frequency bands at different resolutions. Two self-constructed datasets with ten genres in one dataset and five genres in the other are used. Signals are sampled over 30 seconds. MARSYAS is used to retrieve standard content-based features. Using 10-fold cross-validation, they find that the SVM performs the best, and that DWCH features improve classification accuracy.

Ajoodha et al. [2] use magnitude, tempo, and pitch-based features, and several off-the-shelf classifiers. They show a relationship between using a larger number of features and an improvement in classification, especially for the first 100 features. They also find that a linear logistic regression model provides the best average accuracy (81% on all of the GTZAN genres).

Recent work is given by Rosner and Kostek [78], where genre classification is done based on instrument track separation. In their experiments they adapt a *non-negative matrix factorization* algorithm, as well as *iterative modified Kullback-Leibler divergence*, which then constructs a mask that is used to obtain a spectrogram of the drum part. The OpenBliSSART SVM classifier [81] is used to split up harmonic and drum instruments. They find accuracies of around 72% after applying feature vector reduction using Weka [30] on a subset of the Synat [39] database.

Other recent work is given by Medhat et al. [64] where the Ballroom music dataset [25]

is used. Masked conditional neural networks (MCLNNs), conditional neural networks (CNNs), SVMs with various feature combinations, and kNNs are examined. They show that having the ballroom data processed via a *Mel-scaled spectrogram* and then fed into a MCLNN gives the best accuracies of 92.12% on eight genres with 10-fold cross-validation.

Srinivas et al. [86] suggest using *on-line dictionary learning (ODL)* for genre classification. MARSYAS features for the three segments of the Latin Music Database (LMD) are obtained. The features obtained are timbral, pitch, and beat-based. During the training phase, dictionaries are developed for each class using ODL, during the testing phase, the sparsity of the test clip is computed, and the sparsest class determines which genre is assigned. This is done using the l1-lasso distance. They find that with a larger dictionary, the classification accuracy decreases.

Recently, there has been a trend towards the combination of acoustic features with other features, such as visual and textual features. For example, Pérez-Sancho, et al. [75] present a framework for genre classification based on *pitch class profiles*, which are constructed from algorithmic transcriptions of chords from audio signals into *chromagrams*. Using a language model (*n-grams*), they show that with a higher n-gram a higher accuracy is achieved. A further example is given by the proposed use of *Gabor filters* and *local phase quantization* on spectrograms of three 10-second segments by Costa et al. [20]. Genre classification is explored using acoustic and lyrical features by Mayer and Rauber [57], where content-based features and rhythmic features are combined with statistical style features, such as a bag of words approach using a tf-idf calculation, rhyme features using phonemes, part-of-speech features, words per minute, etc. They find an increase in performance using various feature modalities.

There are so many more approaches that contribute to the music genre classification problem in MIR, with hundreds of other works written. We will now focus on more closely related works, relevant to future chapters of this thesis.

2.7.2 Subgenre Classification in MIR

In contrast to the large number of works on general music genre classification, we have only found a small number of approaches dealing with subgenre classification. We will now mention some approaches focusing on the subgenre classification task.

Quinto et al. [76], examine the effectiveness of deep learning classifiers against standard classification methods using only MFCC features on a self-constructed dataset of three (3) jazz subgenres (*bebop*, *acid jazz*, and *swing/electroswing*). They find that a single-layered *long short-term memory (LSTM)* network with a multilayer perceptron added before it gives the best accuracy of 89.824%, their best neural network classifier obtains 79.39%, an SVM classifier obtains 81.67% and a kNN classifier obtains 77.43%.

Sousa et al. [23] create a dataset called the *Brazilian Music Dataset (BMD)*, with seven genres, and 30 pieces each. The BMD differs from the LMD by the introduction of other Brazilian genres, and the removal of several LMD genres. With 5-fold cross-validation and a 66% training set size, they achieve an accuracy of 79.7% using an SVM for 10 GTZAN genres, and an accuracy of 86.11% on the BMD genres.

Further recent work is also given that tests existing modern genre classification algorithms on newly constructed datasets with classes as subgenres from a particular region. One example is given by Kizrak and Bolat [44], where a self-constructed dataset of 93 pieces is used as data for the classification of the seven most frequent Turkish Makams. They test various classification schemes and find that the best accuracy is 96.57% via a deep belief network of five hidden layers using MFCC features. Another example is given by Soboh et al. [85], where they create a dataset based on the Arabic music styles of Moroccan, Egyptian, Shami, and Khaliji (100 pieces each). They derive dynamic, rhythmic, and timbral features. An overall best accuracy of 80.25% is achieved using a Weka decision tree and OneR attribute feature selection, using 10-fold cross-validation. The classification of Fado music is investigated by Antunes et al. [3], in the context of deciphering between Fado music and other genres. They extract rhythmic, timbral (MFCC), and dynamic features

from 10-second audio clips, and use Matlab and LibSVM [16] to perform the classification with SVMs using an RBF function. They find very high accuracies in distinguishing Fado from non-Fado music (i.e., greater than 90%).

Further work is found in various theses, especially concerning electronic and metal subgenre classification, otherwise there has still been, to our knowledge, a void in overall subgenre classification research. Two theses regarding the classification of electronic music are found, the first is given by Kirss [43]. 5 electronic music subgenres (50 pieces each) are used, including *house*, *techno*, *trance*, *drum and bass*, and *ambient*. The highest accuracy is 96.4% using an SVM classifier with spectral and rhythmic based features. Techno and house are often misclassified. The next thesis, by Chen [18], also discusses the classification of electronic music. Various spectral features are extracted, and Gaussian mixture models are used for classification. Deep house, dubstep, and progressive house are chosen as the subgenres. 20 training and 10 testing pieces are selected for each genre (30 seconds each). 3-fold cross-validation is done, and an 80.67% success rate is achieved.

The classification of metal takes place in two theses that will be discussed. The first is by Tsatsishvili [92]. Three classifiers are tested, two are off-the-shelf Weka classifiers (a C4.5 decision tree classifier, and a kNN classifier), and one algorithm is implemented. The implemented algorithm was originally proposed by Barbedo and Lopes in 2007 [8], where classification was carried out using a four layer genre hierarchy, and a voting process is done on several reference vectors. In Tsatsishvili's thesis, 30 tracks per subgenre are used. Timbral, rhythmic, dynamic, and pitch information is extracted. The highest accuracy achieved is 45.7% using spectral features, with AdaBoost and a J48 classifier. *Black metal* and *death metal* are the most correctly classified genres. The second thesis, by Mulder [68], also classifies metal music into subgenres. Chroma features as well as musical horizontal and vertical intervallic relationship features are derived. An average classification accuracy of 28% (using vertical intervals) over 17 subgenres is found. The collection used for classification is manually gathered.

2.7.3 Ensemble Techniques in Genre Classification

Since one of the approaches discussed in this thesis is a pairwise ensemble-like approach, it is necessary to review some works that deal with ensemble classification. Silla et al. [41] provide an early work using an ensemble of classifiers. Each classifier is trained on the beginning, middle, and ending 30 seconds of a piece. The best result combines SVM classifiers for the beginning, middle, and ending segments, and sums the probabilities for each class. Otherwise, they find that the middle 30 second section produces better results than the beginning and ending 30 seconds (we also find this to be the case for the LMD).

Almeida et al. [22] use two sets of features extracted from the LMD (e.g., spectral, timbral, chroma, etc.), and two pools of classifiers. They use a *k-nearest oracles* algorithm (i.e., at least one of the nearest neighbours in feature space must be recognized) and find that majority voting increases the accuracy. They also show an improvement in classification accuracy with a diverse number of classifiers.

Arjannikov and Zhang [7] report an empirical study on different nested dichotomies. They show that balanced nested dichotomies often perform better than unnested ones. However, many of the ensemble classifiers they test do not perform better than the individual base classifiers.

Chathuranga and Jayaratne [17] use *late fusion* for classification. Late fusion trains an individual classifier with a certain feature modality, in their case they use one classifier that models short-term features, and another that models long-term features (early fusion will combine modalities into a single feature vector first). They show that using a classifier ensemble technique with weighted majority voting and late fusion can improve classification accuracy (SVMs were the individual classifier). They apply *wrapping* to the short-term features and *filtering* to the long-term features.

Ariyaratne et al. [4] use decomposition, with a one-versus-one, majority voting technique, where they train on pairs of genres. They extract MPEG7 features, MFCCs, etc., using a self-made extension of the GTZAN dataset. For each genre, they perform feature

selection. Generally, they find that the one-versus-one scheme performs better, and that feature selection improves the classification accuracy.

2.7.4 Association Analysis in MIR

Association analysis is first presented by Agrawal et al. in 1993 [1]. Liu et al. [54], are some of the first to use association analysis in the context of classification (unrelated to MIR), in 1998. However, association analysis in MIR, like "at large" subgenre classification, has only been investigated by a handful of researchers, hence the need for this thesis and for further investigation. Some of the work proposed by these researchers will now be summarized.

Association analysis is proposed as an unsupervised way to retrieve genre-specific music files by Rompré et al. [77]. They obtain a frequency spectrum and then group frequencies in their method of discretization, and discard non-informative bins using the support and confidence thresholds. They use the GTZAN dataset and obtain a precision of 63% and a recall of 72%. With more association rules the recall drops and the precision increases.

Shan and Kuo [82] attempt to classify musical styles by using association analysis to construct a classifier through melody mining. They extract *Musical Instrument Digital Interface (MIDI)* melodic and harmonic information and find the chord-sets that are frequent. They compare the MIDI data (35 to 50 files each) of Enya, the Beatles, and Chinese folk pieces. They find that the minimum support, the length of feature extraction, and how the chords are being represented (e.g., as a set of bi-grams) affects their accuracies.

Data mining using *maximally general distinctive patterns* is carried out on various styles of local Cretan folk songs (106 total) by Conklin and Anagnostopoulou [19], where they propose a supervised descriptive rule discovery method. All songs are exported to MIDI format, and the type, supertype (East and West), area, and superarea that a song belonged to is determined. Specifically, interval relationships of melodies are mined and compared to a corpus and anticorpus (all other classes).

Similarly, non-content-based tasks are carried out by Neubarth et al. [70] to show the association of toponyms with folk genres using textual features. Association rule mining is utilized for the purpose of discovering genre and region relationships. Rule templates and evaluation measures are used on a corpus of Basque folk tunes. They use a set of constraints with annotations pertaining to geographical ontology and an ontology of hierarchically organized genres.

Arjannikov and Zhang [6] propose the use of association analysis with content-based features to explore genre classification. This work differs from our approach, in that antecedent and consequent rules are derived for itemsets and genres, as opposed to a genre specific derivation of frequent itemsets. Furthermore, they do not consider other potential parameters like the length of the itemsets during classification, as we explore using different ranking criteria in Chapter 3. Itemset intersection removal is also only done if the itemset occurs in two or more genres, however, we analyze different values of this parameter more. They implement four (4) evaluation methods, based on the matching of the itemset to the piece, these evaluation methods are: the normalized support sum, the normalized confidence sum, and the length sum. They also observe that the classifier will most often choose one genre for its classification, however, the minority of the time, it picks multiple genres (i.e., they measure the *multi-label rate*). They note that a lower support improves classification accuracy, classification accuracy is improved by selecting fewer genres, and that by implementing the removal threshold empty descriptions for a genre can be made. They use the LMD and subsets of the *Million Song Dataset (MSD)*, accuracies are approximately 50 to 60%, for 10 LMD genres.

Arjannikov et al. [5] use association analysis to verify tag annotations by matching music pieces against association rules, and utilizing a scoring method for evaluation. They perform several experiments, including using one dataset to evaluate another, splitting a dataset in half and checking that similar results are obtained, and amalgamating similar tags. The Cal10k [91], *Cal500* [93], *Magnatagatune* [47], and *LastFM* [10] datasets are

used. They select pieces with at least two tags in their cleaning stage. The general trend discovered is that with a higher confidence threshold, the normalized total of the number of pieces matching rules drops.

2.8 Summary

We have now presented a detailed description of the music genre classification task in MIR. The details for each of the typical steps of this problem are also provided, including: feature extraction, discretization, and the modelling of genres for the purpose of classification (whether using association analysis or not). Also included is a discussion on specific datasets and some related works.

Genre classification is a critical task not just for MIR, but for machine learning in general, since tackling a problem with such a high degree of subjectivity is useful for many purposes. Some applications of a good genre classification approach benefit the browsing, organization, and overall enjoyment of music.

Chapter 3

Capturing Genre Characteristics of Music Through Feature-Value Analysis

In Chapter 2 we saw how the problem of genre classification in MIR is approached. Recall that after acoustic features are extracted from a music dataset, they are then normally stored as feature vectors and sent to a classifier that will model the data. Then, testing data is classified in some way based on this model, or approach, and finally, the overall approach is evaluated.

In this chapter, we propose an adaptation of association analysis, in order to detect and extract the acoustic features of music genres and use the resultant relationships among them to classify new music pieces. In essence, for each genre, we obtain a set of characteristic features and their values that represent it.

The structure of this chapter is organized as follows. In Section 3.1 we contextualize the problem of content-based music genre classification for our approach and express the goals that we wish to accomplish. Next, in Section 3.2, we discuss our proposed approach in greater detail. In Section 3.3, we describe those music datasets, features, and songs per genre that are used in evaluating our approach. Then, in Section 3.4 we discuss the extensive experiment results on a set of music datasets, where we attempt to analyze our approach with respect to the various parameters that are used. Finally, in Section 3.5 we present a summary, that also provides some concerns for the next chapter and other potential directions for future work.

3.1 Introduction

From a computational viewpoint, each music genre has its own musical characteristics which are encoded in the acoustic contents of the genre. That is, there are specific patterns hidden and buried in the pieces of a genre, and it is our very task to detect and extract them. These patterns can then be used to classify music into genres. In essence, we utilize the acoustic features of music that are naturally found in music pieces, such as the content-based features presented in Section 2.1 (e.g., MFCCs, spectral features, time domain features, etc.). We extract these acoustic features from a music piece and use the most frequently occurring characteristics per genre. Next, we represent each genre uniquely by removing the similarities between genres. If we treat acoustic features and their values as data objects (i.e., we discretize the features of music pieces), then intuitively, frequent co-occurrences of particular features and their values from a genre will tell us something about its musical nature and therefore, can be used to conduct music genre classification. Lastly, we employ a genre's unique features to determine which genre an incoming music piece belongs to. As shall be seen, our proposed approach is flexible, in that it may be used with any number of acoustic features.

By adapting association analysis, we seek to detect and extract, for a music genre, a set of characteristic features and their values that statistically represent it. We further make use of the characteristic sets of individual genres to conduct genre classification, which results in practical classification accuracies. We also demonstrate that the final classification accuracy is affected when using different parameter combinations. We seek to determine what these parameters may be, and how they might improve classification accuracies. Our approach is applied on differing datasets with various sizes. We further attempt to determine if the approach performs in a stable manner. That is, we determine if the approach performs more successfully with better quality training data. This chapter's approach is provided with further detail below.

3.2 Characterizing Genres through Feature-Value Pairs

3.2.1 Approach

The acoustic content that is derived from a genre contains rich information that we can utilize. In our work, we consider adapting association analysis to detect and extract patterns from this content, capturing and storing the characteristics of individual genres and facilitating their classification.

Recall from Section 2.3 that with association analysis, we are interested in a set of data items, in a dataset, that frequently occur together. Intuitively, there is some statistical relationship among them. Those frequent data items are put together into *frequent itemsets*, i.e., a 3-item itemset means the three items in the set occur frequently together. Among the many parameters, the *support* of an itemset is the percentage of the co-occurrence of its items in the dataset. Only those itemsets whose support exceeds a threshold, called the *minimum support*, are considered frequent. We adapt the *Apriori* algorithm in our approach.

In the context of genre classification, each piece in a music dataset is represented as a *feature-value* vector $P = \{p_1, p_2, \ldots, p_n\}$, where p_i is the value of the feature $f_i \in F$ and $F = \{f_1, f_2, \ldots, f_n\}$ is the acoustic feature set used to conduct genre classification. It is often the case that a larger number of diverse and carefully chosen features yields higher classification accuracies [2], as a diverse and effective set of features will ultimately represent the musical, and acoustic properties of music pieces more thoroughly. We show that our approach provides a flexible method of determining the genre-dependent, frequently co-occurring features. Our intuition is that those feature-value pairs are genre-specific and can be detected and extracted from the acoustic features.

Our initial approach captures the intuition described above and provides a sound method of conducting music genre classification, as demonstrated shortly. In summary, a representative subset of music pieces from a genre is selected, and features for those music pieces are then extracted. The genre's acoustic features and their values summarize its musical nature. From a computational viewpoint, those feature-value pairs are treated as data items.

After proper discretization and encoding, they are used during association analysis. Given a minimum support, a set of frequent itemsets is obtained. For convenience, we name the set of feature-value pairs an *fv-set*, and also name the set of frequent feature-value pairs as the *frequent fv-sets*. We notice that there should be some musical elements that are common to all of the genres. The existence of these fv-sets will cause confusion in the genre prediction of new music pieces. So the frequent fv-sets will go through a cleansing stage, and we obtain, for each genre, a characteristic set of fv-sets representing it. This removal of intersecting (or, overlapping) frequent fv-sets yields those frequent fv-sets that are (more) unique to a genre. As we will see, we can control the amount of overlap detected and removed (see Section 3.2.2).

After the intersection removal step described above, the classification process then begins. A new music piece is represented as a set of feature-value pairs, and we use these feature-value pairs to compare against each genre's characteristic set (the genre's frequent fv-sets after the intersection removal) and find genre scores for this piece using various *Ranking Criteria*. A ranking criterion represents our views on how to use the frequent fv-sets we have obtained per genre. For instance, we may think that it is more important when a new music piece matches a larger frequent fv-set than a smaller one, or that a music piece should match the entirety of a frequent fv-set. This is an essential step in the classification process, as it ultimately determines how closely a music piece matches the frequent fv-sets of a genre. Our approach's high-level description is depicted below in Figure 3.1.

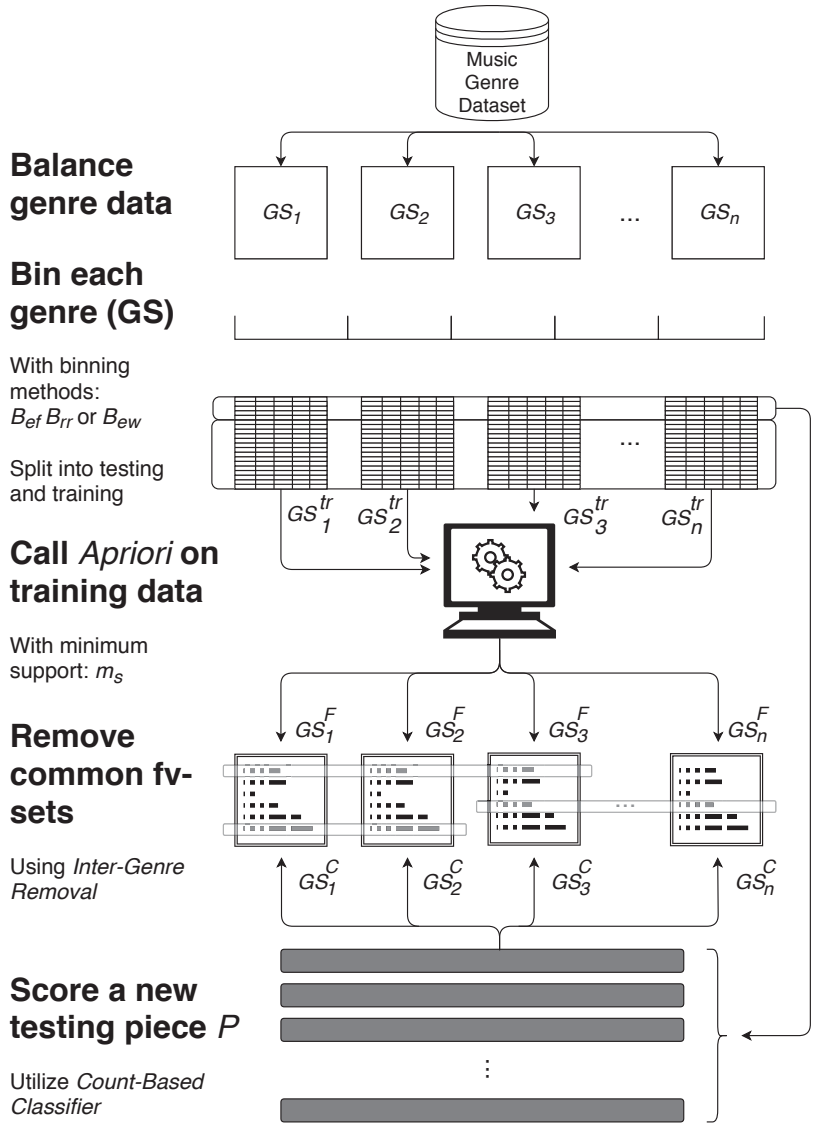


Figure 3.1: A high-level description of our first approach.

A detailed algorithmic description will now be presented. B goes through each binning method we are using (see Section 2.2 for a more detailed discussion), while the superscripts tr and te correspond to training and testing. Suppose that genre G has a dataset of music pieces labelled G. A set GS is randomly chosen from it to generate and evaluate the frequent fv-sets of genre G, this subset selection is done to ensure a similar treatment of genres during the derivation of frequent fv-sets. A proportionate amount of music pieces, per genre, provides a greater chance that each genre will be characterized fairly. Since Apriori han-

dles discrete categorical values instead of real values, we normalize their values and then discretize the real-valued acoustic features, according to different binning methods. During this process, we encode each value, for each feature systematically. For instance, the second feature's fourth value (or the fourth bin that the feature's value falls into) is encoded as $b4$. Note that the Apriori algorithm uses a minimum support, denoted as m_s, to find the frequent fv-sets. For each frequent fv-set, we set its size, i.e., the number of feature-value pairs in it, to be at least 2, and we ensure that Apriori returns only those maximal itemsets (see Section 2.3 for a more detailed discussion). Utilizing only maximal itemsets reduces redundancy and increases the efficiency of our approach. For the genre G, after this step, we obtain its corresponding set of frequent fv-sets, denoted as GS^F.

Characterizing and evaluating genres' characteristics through feature-value pairs

1. For each binning method B
2. For each genre G's GS
3. Apply B to GS
4. Split GS into its training set GS^{tr} and testing set GS^{te}
5. End of For
6. For each genre's GS^{tr} // Generate the characteristic set for each genre
7. GS^F = Call *Apriori* to GS^{tr} with m_s
8. End of For
9. Call procedure *Inter-Genre Removal* on all genres' GS^F
10. and for each GS^F we obtain GS^C
11. For each genre's GS^{te} // Test each genre's music pieces; collect accuracies
12. For each piece P in GS^{te}
13. For each variation *CBC* of procedure *Count-Based Classifier*
14. Call *CBC* on P with all genres' GS^C
15. End of For
16. Check whether P is correctly classified or not
17. End of For
18. Collect the classification accuracy of all of the pieces in GS^{te}
19. End of For
20. End of For

3.2.2 Inter-Genre Removal

Procedure *Inter-Genre Removal*, as called in the algorithm above, is the cleansing process to remove the common frequent fv-sets, which results in the characteristic set of fv-sets, denoted as GS^C, for each genre G. The criterion used in our current implementation is simple. If a frequent fv-set from a genre appears in more than a certain μ of other genres' characteristic sets, we remove it from the genres' characteristic sets, since it is common to all of them and will not contribute to their classification. In this procedure, we introduce the parameter ϕ, which determines the acceptable overlap between two fv-sets. For this parameter, there are two further considerations: strictness and relaxation (denoted as ϕ_s and ϕ_r, respectively). For instance, for three fv-sets {b4, c3}, {a2, b4, c3}, and {a2, b4, c2, d5}, if we set both ϕ_s to 60% and ϕ_r to be 60%, then {b4, c3} matches {a2, b4, c3} but not {a2, b4, c2, d5}. For ϕ_s, we delete both {b4, c3} and {a2, b4, c3}, whereas for ϕ_r, we only delete {b4, c3}.

3.2.3 Count-Based Classifier and Ranking Criterion

After we obtain each genre's GS^C, we use the procedure *Count-Based Classifier (CBC)* to count the number of times each frequent fv-set in GS^C appears in an incoming testing piece. It is an important step in our approach since it alters the final classification accuracy by checking how many matches occur between a testing piece and a genre's characteristic set (the best match is chosen as the genre). Currently, we propose the following variations of the CBC and call them Ranking Criteria, denoted as *RC*. Given a new music piece P, whose fv-set is denoted as P_i, we score it against genre G. Let C_i be a frequent fv-set in GS^C under consideration and let the length of C_i be $|C_i|$. In the following, α, β, and γ are user-supplied parameters to be tuned experimentally, where $\alpha+\beta=1$, for our experiments we set $\alpha=0.7$. This allows us to set the importance that the length of the frequent fv-set has when matching a music testing piece. We denote the score of a testing piece P under a ranking criterion as RC_*. Our proposed ranking criteria are given below:

- *Naive*, denoted as RC_n, increments the score of P by 1, if $C_i \subseteq P_i$.
- *Partial Power Threshold*, denoted as RC_{ppt}, increments the score of P by 2^l, where $l = |C_i \cap P_i|$, if $\frac{|C_i \cap P_i|}{|C_i|} \geq \gamma$.
- *Complete Power Threshold*, denoted as RC_{cpt}, increments the score of P by $2^{|c_i|}$, if $\frac{|C_i \cap P_i|}{|C_i|} \geq \gamma$
- $\alpha\beta$, denoted as $RC_{\alpha\beta}$, increments the score of P by $\alpha \cdot RC_n + \beta \cdot |C_i|$, where $C_i \subseteq P_i$.

Among those ranking criteria, the naive one simply increments the score of P by 1 whenever $C_i \subseteq P_i$, showing our "raw" view on the equal treatment of frequent fv-sets from a genre's characteristic set. The threshold criteria involves a parameter γ, which sets the relaxation of the comparison between P_i, and C_i from a genre's characteristic set. The power criteria represents the intuition that the larger the match between the two fv-sets from P and C_i, the better. We award an increment as a power factor of 2. The $\alpha\beta$ criteria combines the naive increment and the size of C_i. For instance, we may consider that the number of matching items between P_i and C_i is more important. We can then set β higher than α. For a testing music piece P, after going through a genre's characteristic set, it is awarded a score. After the total score is accumulated, it is normalized by the size of a genre's characteristic set. We select the genre which results in the highest score and assigns it to P. If there is a tie between multiple genres and the target genre is in the tied genres, we still consider it as a correct classification. In the rare situation where no scores are awarded to any genre for the new piece, we deem it to be a misclassification.

3.2.4 Discretization Methods Used

Several binning methods are used in our framework to preprocess music data. We briefly describe them here for convenience (more information can be found in Chapter 2). Binning based on equal width, denoted as B_{ew}, is to divide the value range of a continuous acoustic feature into a certain number of N bins. One variation of equal width binning assumes the

normal distribution of data points in a dataset and uses the so-called "Rice rule" [45], which sets N to be twice the cubed root of the number of data points in a dataset. We denote it as B_{rr}. Binning on equal frequency, denoted as B_{ef}, is to generate a number of bins such that each bin contains approximately the same number of data points from a dataset. The N number of bins manipulates how frequent fv-set mining reacts to the training data; a greater number of bins could create more fv-sets and affect how the m_s parameter behaves. For the binning methods, we use the implementations from the Weka environment [30]. We set $N = 10$ for equal frequency binning and $N = 15$ for equal width binning.

3.3 Experiment Preparations

It is desirable and beneficial to verify our approach through experiments on high-quality music datasets. For instance, we may gather a group of music experts and let them decide representative music pieces for each genre. This way, our approach will detect and extract the accurate characteristic sets for each genre. However, such an ideal situation involves considerable obligations. Therefore in our experiments, we select some *de facto* benchmark datasets that are publicly available and frequently used in the MIR community. It is necessary to test the effectiveness of our approach on several datasets so that a more transparent depiction of the approach's performance is provided.

3.3.1 Datasets

In order to verify our proposed approach, we apply it to a set of music datasets in the MIR literature, namely, the *Cal10k Dataset* (denoted as D_{CAL}) [91], the *Greek Audio Dataset* (denoted as D_{GAD}) [56], and the *Latin Music Database* (denoted as D_{LMD}) [41, 83].

For the datasets D_{CAL} and D_{LMD} we create three subsets by splitting the extracted features into the beginning, middle, and ending segments. We call those three subsets D^b_{CAL}, D^m_{CAL}, and D^e_{CAL}, for D_{CAL}. Similarly, we call those three segments D^b_{LMD}, D^m_{LMD}, and D^e_{LMD}, for D_{LMD}. For D_{GAD}, we use the beginning 30 seconds of each music piece. To

create each beginning, middle, and ending segment for D_{CAL}, we create three even subsets for every piece, by splitting the feature calculations over each segment. As for the splitting of D_{LMD} into segments, for each music piece, the beginning segment is from sample 0 to sample 1153 (30 seconds of audio in MP3 format), the middle segment is from sample $\frac{Q}{3} + 500$ to sample $\frac{Q}{3} + 1653$, and the final ending segment is from the sample $Q - 1453$ to $Q - 300$, where Q is the total number of samples in that piece, as provided by Silla et al. [41]. In order to achieve D_{LMD} we collapse the folds provided by Silla[7], and remove any duplication of feature vectors regardless of class label.

We will now describe the size of the dataset subsets (i.e., the number of music pieces per genre) used for the experiments. We design two series of experiments on D_{CAL} to determine whether different numbers of pieces from the same dataset influence the prediction accuracy. In the first series, each genre has 400 music pieces, and in the second one, each genre has 140 music pieces (for each of the 3 segments). For the experiments with D_{LMD}, we fix experiments to only have 300 songs for each of the 3 segments. For D_{GAD} we use 50 songs per genre in order to determine how our approach performs given a smaller dataset.

3.3.2 Genres

For the dataset D_{CAL}, we use a technique of hierarchically *uplifting subgenres* to their parent genres, with at least 2 subgenres per genre. The genres used include: *(H)ip (H)op*, *(M)etal*, *(C)lassical*, *(CO)untry*, *(J)azz*, *(E)lectronic*, and *(R)eggae*. For D_{GAD}, we include the genres: *(L)aiko*, *(H)ip (H)op/RnB*, *(R)ock*, and *(RE)mbetiko*. Lastly, for D_{LMD}, we use the genres: *(AX)é*, *(B)olero*, *(F)orró*, *(ME)rengue*, *(SE)rtaneja*, and *(PA)gode*. The organization of the genre classification experiments is provided below in Table 3.1. For the sake of convenience, we also include the m_s values in Table 3.2. These m_s values are used on each training subset for every genre. We also provide the testing and training percentages, since this determines how many music pieces from the training set are used to characterize

[7] https://sites.google.com/site/carlossillajr/resources/the-latin-music-database-lmd.

Table 3.1: A comprehensive description of the datasets used in the genre classification experiments of Chapter 3.

Dataset	Sections Used (Duration)	Genres Used in the 6 Genre Experiments (Songs per Genre)	Genres Used in the 4 Genre Experiments (Songs per Genre)	Minimum Support (m_s) Values Used%	Training%, Testing%
D_{LMD}	beginning, middle, end (30s each)	axé, bolero, forró, merengue, sertaneja, pagode (300)	tango, salsa, forró, axé (300)	5%, 8%, 10%	60%, 40%
D_{CAL} 140 song experiments	beginning, middle, end (1/3 the length of song)	classical, country, jazz, electronic, rock, hip hop (140)	hip hop, metal, classical, country (140)	3%, 4%, 5%, 6%	60%, 40%
D_{CAL} 400 song experiments	beginning, middle, end (1/3 the length of song)	classical, country, jazz, electronic, rock, hip hop (400)	classical, jazz, electronic, rock (400)	3%, 4%, 5%, 6%	60%, 40%
D_{GAD}	beginning (30s)	N/A	laiko, hip hop, rock, rembetiko (50)	8%, 10%, 13%, 15%	60%, 40%

the genres.

The genres and songs per genre for each dataset directly affect the classification accuracy. Generally, a greater number of genres reduces the performance of many classification approaches by causing confusion between genres, and that the amount of data used to model a genre could include (or remove) noisy data. These decisions will now be discussed in detail.

Specifically, the genre choices for D_{LMD} in the experiments with six (6) genres presents a challenge, as bolero and forró are both rhythmically slower, axé and merengue are both rhythmically faster, and merengue and forró both utilize similar instrumentation (i.e., the accordion is prominent in both) [83]. For the four (4) genre D_{LMD} experiments we choose slightly more distinguishable genres, especially with respect to tango. Recall from Section 2.6 that tango should be more distinct because of the genre's older recordings included in the dataset [90].

The genres chosen for D_{CAL} are quite diverse, with some obvious overlap of rhythmic and timbral characteristics between electronic and hip hop. This is why, for the four (4) genre experiments we replace electronic with metal when including 140 songs per genre. Since these genres are somewhat distinguishable, we experiment with the number of songs

included as well (i.e., 400 songs in one set of experiments, and 140 in the other).

Since D_{GAD} has many overlapping genres (e.g., alternative (enallaktiko) and rock, rock and pop) we limit the number of genres used in the experiments to four (4). However, the genres included may still contain overlapping characteristics. In addition, the subset of music pieces chosen is quite small. This provides a challenging classification task, so we use this task to evaluate our approach in a less than ideal scenario. This task is designed to demonstrate our approach's ability to scale down to smaller datasets, while still applying the Inter-Genre Removal method to remove commonalities between genres.

3.3.3 Features

There is a variety of features used in our experiment design. For D_{LMD} and D_{CAL} the features used are also included in previous studies [41, 91], so that potential comparisons can be made directly to other approaches. The dataset D_{CAL} has 13 MFCC features extracted over the entirety of each song only. The feature set used for D_{LMD} has a moderate amount of features, and D_{GAD} uses a greater number of features. The varying feature set sizes allow us to determine the affect that the number of features has on classification performance, even if the number of training pieces is not large. MARSYAS and jAudio are used to extract the features from D_{LMD}, and D_{CAL}'s music pieces, respectively.

3.4 Experiment Results and Discussions

Our approach involves several different parameter combinations. So, we report and examine the results of our experiments from certain angles. We use 10-fold cross-validation, for each experiment in this section, with the testing and training splits given in Table 3.1.

We first comment on some general observations. In Table 3.3 we see that the difference between ϕ_r and ϕ_s used in the procedure Inter-Genre Removal is not substantial, which is contrary to our original intuition that either the strict approach would yield more representative frequent fv-sets, or that it might be too strict during Inter-Genre Removal. Albeit,

Table 3.2: A comprehensive description of the features used in the genre classification experiments of Chapter 3.

Dataset	Features (Feature Extraction Software)	Feature Parameters	Feature Calculations
D_{LMD}	5 MFCCs, Spectral centroid, rolloff, and flux, zero crossings, low energy, relative amplitudes, beats per minute, max. periods of pitch peak (MARSYAS)	Unspecified in [41]	mean, variance
D_{CAL}	13 MFCCs (MARSYAS)	window size: 2048 ms hop size: 1024ms sampling rate: 22050Hz	mean, std. dev.
D_{GAD}	13 MFCCs, 5 method of moments, strongest beat, beat sum, strength of strongest beat, strongest frequency via zero crossings, spectral centroid, and FFT maximum, peak based spectral smoothness, 9 LPC (jAudio)	Unspecified in [56]	mean, std. dev.

Table 3.3: D^{m}_{LMD}, $\gamma = 50\%$, B_{ef}, $m_s = 5\%$, RC_{ppt}, 300 songs per genre.

	$\phi_s = 60\%$						$\phi_r = 60\%$					
	AX	B	F	ME	SE	PA	AX	B	F	ME	SE	PA
$\mu = 16\%$	0.51	0.74	0.46	0.87	0.58	0.55	0.48	0.66	0.47	0.87	0.56	0.58
$\mu = 50\%$	0.46	0.64	0.58	0.81	0.54	0.62	0.44	0.71	0.45	0.79	0.48	0.66
$\mu = 83\%$	0.46	0.64	0.58	0.81	0.54	0.62	0.43	0.66	0.53	0.82	0.48	0.49

ϕ_s does provide slightly higher accuracies. Across all experiments this is found to be the case for ϕ_r and ϕ_s, with there only being a few percentage difference per genre. We also observe that a more strict μ (i.e., a lower percentage value) often provides higher classification accuracies across all of the experiments. We believe that μ eliminates the frequent fv-sets that overlap, leaving the most unique ones. We further see that certain genres affect the overall prediction accuracy more than others, and sometimes can be considered distinguishable. For example, in Table 3.3 we notice that merengue and bolero typically achieve higher accuracies, which is also reflected in Table 3.4, which we will now use to examine various genre-specific confusions further.

For example, we note that: some sertaneja pieces are confused with fórro and axé, axé and fórro are not as distinguishable as other genres, some bolero pieces are confused with sertaneja, and lastly merengue is found to be the most distinguishable genre. We believe merengue is distinguishable due to its rich orchestration, with the common inclusion of

Table 3.4: Confusion matrix (10-fold cross validation) - D^{M}_{LMD}, $\mu = 16\%$, $\phi_r = 60\%$, $\gamma = 50\%$, B_{ef}, $m_s = 5\%$, RC_{ppt}, 300 songs per genre.

	AX	B	F	ME	SE	PA
AX	**0.48**	0.04	0.11	0.12	0.16	0.09
B	0.03	**0.66**	0.04	0.00	0.23	0.04
F	0.06	0.07	**0.47**	0.11	0.16	0.13
ME	0.06	0.00	0.03	**0.87**	0.01	0.027
SE	0.14	0.07	0.17	0.01	**0.56**	0.05
PA	0.09	0.05	0.11	0.04	0.13	**0.58**

brass instruments, and its direct-sounding rhythmic approach. Bolero and sertaneja both have slower rhythms where the guitar is prominent, which may cause some confusions. For the case of confusions with sertaneja pieces, it is reasonable to have some confusion with fórro, since both of these genres have a more relaxed rhythmic approach and sparser instrumentation. However, it is less reasonable that sertaneja be confused with axé, since axé is faster rhythmically, although, some misclassifications are due to pieces having similar timbral characteristics [34, 83].

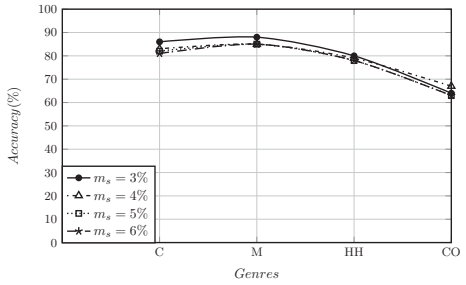


Figure 3.2: D^{m}_{CAL}, $\mu = 50\%$, $\phi_s = 0.6$, $\gamma = 50\%$, B_{ef}, RC_{cpt}, 140 songs per genre.

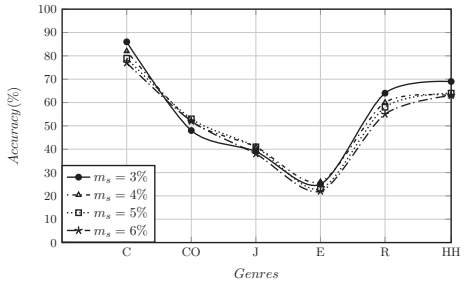


Figure 3.3: D^{m}_{CAL}, $\mu = 50\%$, $\phi_s = 0.6$, $\gamma = 50\%$, B_{ef}, RC_{cpt}, 140 songs per genre.

Comparing Table 3.3 to Figures 3.2 and 3.3, we see that both the distinguishable genres and the number of genres alters the overall prediction accuracy. In Figure 3.2 we see an impressive classification accuracy. Note that this experiment has four distinguishable genres, so it is expected that the accuracies are higher. One stipulation is that all experiments using D_{CAL} are done only on MFCC features. This clearly shows the effectiveness of our approach on a small set of features.

Also, in Figures 3.2 and 3.3 we notice the trend that a lower support value generates more frequent fv-sets, which leads to a higher chance that a testing piece will be classified

correctly, thus yielding a higher average classification accuracy across several genres. We find this to be the general trend across all of the experiments, with relatively few exceptions. Furthermore, if too high of a m_s is used, it is possible that very little frequent fv-sets for a genre will be found, so some tuning of m_s is necessary to have comparable characteristic set sizes between the genres (this allows for CBC to perform fairly across the selected genres). With a high enough m_s it is also possible that a certain combination of features becomes absent in the characteristic fv-sets of a genre. In essence, these absences represent the notion that these feature combinations are not statistically significant, and will not assist in the classification. From this, we see that our approach offers further utility in empirically determining the features that cause a particular genre to become distinguishable. This will be explored in our future work.

Table 3.5: D_{GAD}, $\mu = 50\%$, $\gamma = 50\%$, B_{ef}, $m_s = 8\%$, $\alpha = 0.7$, 50 songs per genre.

	$\phi_s = 60\%$				$\phi_r = 60\%$			
	L	HH	R	RE	L	HH	R	RE
RC_n	0.43	0.67	0.52	0.41	0.43	0.62	0.43	0.54
$RC_{\alpha\beta}$	0.44	0.70	0.51	0.42	0.43	0.54	0.41	0.54
RC_{cpt}	0.35	0.82	0.50	0.52	0.27	0.74	0.42	0.62
RC_{ppt}	0.38	0.79	0.50	0.52	0.38	0.73	0.43	0.61

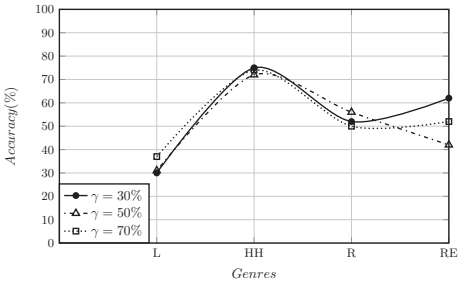


Figure 3.4: D_{GAD}, $\mu = 16\%$, $\phi_r = 0.6$, $m_s = 8$, B_{ef}, RC_{cpt}, 50 songs per genre.

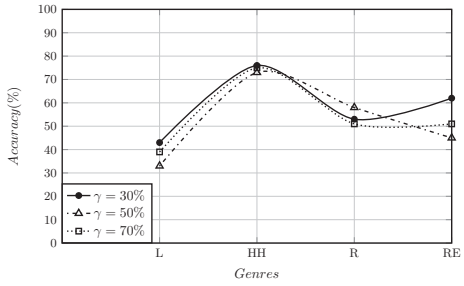


Figure 3.5: D_{GAD}, $\mu = 16\%$, $\phi_r = 0.6$, $m_s = 8$, B_{ef}, RC_{ppt}, 50 songs per genre.

In Table 3.5 we observe the effects of various ranking criteria on D_{GAD}. We notice that the ranking criteria does not have a stable impact on the prediction accuracies. Sometimes, however, the ranking criteria provides slightly higher accuracies, depending on the criteria chosen. We do notice a tendency for RC_n to be outperformed by more "sophisticated" scoring methods. This shows the importance of considering the size of the frequent fv-

sets (in a genre's characteristic set) when matching it to a testing piece. However, we still determine the impact of the ranking criteria to be unstable, since all criteria other than RC_n have similar chances of performing the best. In the aforementioned table, there are only 50 songs per genre, which means there is only a small amount of training music pieces that are utilized. While the genre prediction accuracies are mediocre, the training time is minuscule, and if anything, this shows that it is possible to scale our approach, such that it remains applicable on a smaller dataset. We have not done any further processing to D_{GAD}, like applying feature reduction, or some ensemble technique, which may further improve the classification accuracies.

In Figures 3.4 and 3.5, we observe the effect that γ has on prediction accuracy for the ranking criteria RC_{ppt} and RC_{cpt}. We notice that, for three of the four genres for RC_{ppt}, a lower γ threshold (i.e., a more lenient threshold) performs better. Whereas for RC_{cpt}, only two of the genres achieve an improved prediction with a lenient γ threshold. It may be the case that γ affects only some of the ranking approaches. With the case of RC_{ppt}, a more lenient ranking threshold does not necessarily mean the whole cardinality of the frequent fv-set will be rewarded, but with a low γ percentage, the number of matching feature-value pairs will still be considered (even for a small number of matched feature-value pairs in a frequent fv-set), so we believe that γ may alter the performance more for RC_{ppt}. There must be further work done to draw additional conclusions regarding the impact γ has on various ranking criteria.

Interestingly, an explanation for the lower genre prediction accuracies achieved using D_{GAD} for the experiment results on four genres, could be the fact that D_{LMD} and D_{CAL} have the genre labelling done by musicological experts, whereas D_{GAD} is a dataset with labelling done by the authors [56]. The correlations between the D_{CAL} and D_{GAD} experiments are that the prediction accuracy for hip hop is quite high in both and the prediction accuracy for rock is moderate; this demonstrates that our approach does not have any bias towards a particular dataset's set of genres. The prediction accuracies for hip hop and rock are

also intuitive as hip hop is somewhat distinguishable, and there are characteristics of rock present in other genres.

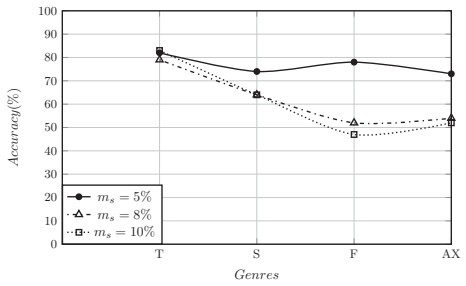


Figure 3.6: D^m_{LMD}, $\mu = 50\%$, $\phi_s = 0.6$, $\gamma = 50\%$, B_{ef}, RC_n, 300 songs per genre.

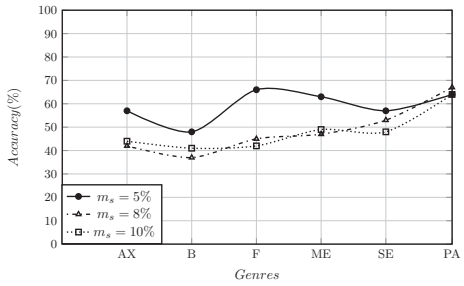


Figure 3.7: D^m_{LMD}, $\mu = 50\%$, $\phi_s = 0.6$, $\gamma = 50\%$, B_{ef}, RC_n, 300 songs per genre.

In Figure 3.7 we see the relationship between the support and the distinguishability of a genre. No matter the m_s, the distinguishable genres should always be more accurately predicted, however, with a smaller m_s, a complex variety of characteristics will be found, and less distinguishable genres will also have a chance of being classified more accurately. This can be seen with the genres fórro, axé, and merengue, but this is also found to be true across all of our experiments. Recall that D_{LMD} only has a moderate number of features extracted per piece, with no feature selection mechanism in place.

In Figure 3.6, we see the classification of four genres on D_{LMD}. As discussed previously, we expected tango to have higher accuracies, and this is found to be true, even for higher support values. However, we once again see the trend that a lower m_s percentage yields a higher classification accuracy. We notice that, even though RC_n is used, a smaller amount of genres leads to better performance, especially when a low m_s percentage is used.

Table 3.6: D_{LMD}, $\mu = 50\%$, $\gamma = 50\%$, B_{ef}, $m_s = 5\%$, $RC_{\alpha\beta}$, 300 songs per genre.

	$\phi_s = 60\%$						$\phi_r = 60\%$					
	AX	B	F	ME	SE	PA	AX	B	F	ME	SE	PA
Beginning	0.51	0.50	0.63	0.61	0.53	0.56	0.51	0.41	0.63	0.52	0.51	0.57
Middle	0.57	0.52	0.65	0.68	0.57	0.64	0.62	0.49	0.62	0.87	0.54	0.60
End	0.58	0.45	0.62	0.65	0.53	0.56	0.59	0.43	0.61	0.83	0.49	0.61

In Tables 3.6 and 3.7, we see that the middle segments of both D_{CAL} and D_{LMD} performing reasonably well, outscoring the results from the beginning and end segments in

Table 3.7: D_{CAL}, $\mu = 16\%$, $\gamma = 50\%$, B_{ef}, $m_s = 3\%$, RC_n, 140 songs per genre.

	$\phi_s = 60\%$						$\phi_r = 60\%$					
	C	CO	J	E	R	HH	C	CO	J	E	R	HH
Beginning	0.63	0.47	0.35	0.39	0.50	0.53	0.58	0.46	0.46	0.37	0.44	0.43
Middle	0.61	0.44	0.48	0.39	0.61	0.58	0.57	0.51	0.42	0.44	0.54	0.56
End	0.50	0.48	0.45	0.36	0.45	0.46	0.49	0.49	0.39	0.43	0.53	0.45

Table 3.8: D_{CAL}, $\mu = 16\%$, $\gamma = 50\%$, B_{ef}, $m_s = 3\%$, RC_n, 400 songs per genre.

	$\phi_s = 60\%$						$\phi_r = 60\%$					
	C	CO	J	E	R	HH	C	CO	J	E	R	HH
Beginning	0.72	0.48	0.41	0.42	0.55	0.54	0.70	0.53	0.43	0.44	0.48	0.56
Middle	0.74	0.50	0.39	0.37	0.62	0.63	0.70	0.51	0.43	0.40	0.56	0.62
End	0.72	0.47	0.41	0.45	0.57	0.61	0.67	0.48	0.40	0.48	0.56	0.57

both experiments, for half or more of the genres. D^m_{LMD} is expected to have higher accuracies, according to Silla et al.'s [41] findings. This entertains the idea that the final prediction accuracy can be related to song sections, with a greater prediction accuracy being related to those sections of pieces that contribute more substantial acoustic content to the classifier, such as the middle sections of songs, when a piece is less likely to offer sparse musical instrumentation.

With these results, we also see how our approach performs on two datasets with six genres, over three different sections. It is interesting to note that even though there is a smaller number of songs and fewer features, the MFCC features of D_{CAL} are just as useful for our classification approach. This may have to do with the features being extracted from a third of the whole piece, as compared to the D_{LMD}, where features are found from 30-second segments. It is possible that features representing longer segments have a more beneficial impact on the prediction accuracies, as they summarize acoustic content over longer periods. We will provide further experiments related to this notion in the future.

We further see that an increase in the number of songs does improve the classification accuracy. This is demonstrated in Table 3.7 and Table 3.8, but we found this to be true across all of our experiments. For nearly every genre, and for both ϕ_r and ϕ_s, the classification accuracy is higher for all segments when there are more songs present. This may be due to the greater variety of frequent fv-sets that are found when characterizing a genre. This

shows that our approach is scalable.

Table 3.9: Confusion matrix (10-fold cross validation) - D^{M}_{CAL}, $\mu = 16\%$, $\phi_s = 60\%$, $\gamma = 50\%$, B_{ef}, $m_s = 3\%$, RC_{ppt}, 400 songs per genre.

	C	CO	J	E	R	HH
C	**0.80**	0.06	0.08	0.00	0.04	0.02
CO	0.07	**0.48**	0.10	0.08	0.14	0.13
J	0.13	0.15	**0.37**	0.10	0.13	0.12
E	0.00	0.15	0.05	**0.34**	0.22	0.24
R	0.00	0.14	0.07	0.11	**0.64**	0.04
HH	0.01	0.06	0.04	0.18	0.06	**0.65**

Finally, we analyze the confusion matrix of D_{CAL} genres to determine if the misclassifications match our intuition. We note that classical and hip hop are quite distinguishable, with only some misclassifications of hip hop being caused by electronic music, which is an appropriate misclassification. Rock and country appropriately caused the most confusion for one another. Jazz is less discernible, with frequent misclassifications corresponding to classical and country, possibly due to a similarity in timbral characteristics. Finally, electronic music, like jazz, is also more vaguely classified. However, most of the misclassifications are due to confusions with hip hop, which is also reasonable. With Tables 3.4 and 3.9 we see appropriate and intuitive misclassifications. For instance, if the genre electronic is being misidentified as classical, then this would be quite problematic and unintuitive. This, however, is not the observed behaviour of our approach.

3.5 Summary

A novel approach to the problem of music genre classification has been given. The performance of this approach has been empirically evaluated, and various parameter combinations have been investigated. However, some aspects of this approach are not without some concerns.

One concern regarding this chapter's proposed approach is the usage of the parameter μ, as used in the Inter-Genre Removal procedure. Currently, this approach applies the inter-genre removal across all genres if the μ threshold is met. However, this removal is not

optimal. For instance, the genre jazz can contain both the musical elements of jazz and blues. Removing blues elements from jazz could inadvertently decrease the classification accuracy for some jazz pieces when comparing jazz to classical, especially if the more "bluesy" jazz pieces are distinguishable from the classical pieces. So, even though stricter μ thresholds yield higher accuracies, the removal of common frequent fv-sets in this way is not ideal, as we will demonstrate shortly. In the next chapter, we present the extension of this approach via a pairwise classification arrangement akin to a dichotomy structure, which addresses this concern.

In summary, we have presented an approach that handles the music genre classification problem by capturing genre-specific characteristics. Our approach characterizes music genres, in terms of distinguishable acoustic features and their values. Through this characterization of genres we are able to tell how, and why, genres are distinguishable from each other. The effectiveness of our approach is demonstrated through our experiments on D_{LMD}, D_{GAD}, and D_{CAL}. As shown, it classifies music genres with a practical accuracy. The raw features are binned into discrete values, and the feature-value pairs are used to represent various musical aspects of a genre (e.g., rhythmic, timbral, etc.) through frequent itemset mining.

Chapter 4

Genre-Specific Characterization and Pairwise Evaluation

In this chapter, we continue to explore the genre classification problem in MIR. We further make use of each genre's unique characteristics, which are represented via a set of features and their corresponding values. To further extend our findings, we present this chapter in light of the various concerns presented at the end of Chapter 3. That is, we now describe a way of improving genre predictions and characterizations via association analysis by introducing a pairwise dichotomy-like strategy.

This chapter is organized in a similar way as the previous chapter. After providing a brief introduction below, we provide a description of our approach in Section 4.2. In Section 4.3, we describe the music datasets, features, and songs per genre used in the empirically conducted experiments. We then discuss, with greater detail, those experiments in Section 4.4 in relation to the effectiveness of the proposed approach. Lastly, in Section 4.5, various related issues, such as potential future work along the same direction, are examined.

4.1 Introduction

As presented in Chapter 3, we begin by treating acoustic features and their values as data objects, by discretizing the features per piece. Again, we find the frequent co-occurrences of feature-value pairs. A critical addition to this chapter, however, is that we present a way to single out each candidate genre. To do so, we conduct genre classification based on a pairwise dichotomy-like strategy. We compare the differences of the characteristics

of two genres in a symmetric manner and use them to classify testing pieces into music genres. The reasoning for doing this is as follows. If we are considering a genre such as *jazz*, there are specific jazz pieces that have influences of *rock* (this is true for certain jazz subgenres like *jazz fusion*). So when we are comparing jazz to another genre like *classical* music, it makes sense to keep these jazz fusion qualities, however, when we are comparing jazz to rock, it makes sense to remove these qualities. By applying a removal method that removes characteristics from all potential genres (after some thresholds are met), as done in Chapter 3, we are removing potentially useful characteristics for genre comparisons. One of our main motivations is to show that excessive use of *Inter-Genre Removal* to remove common frequent fv-sets (i.e., using the μ parameter) may provide suboptimal genre characterizations, whereas improvements can be made by allowing for the characterization of genres (without Inter-Genre Removal), and by applying a pairwise dichotomy-like strategy.

Our goals for this chapter can be further summarized. We aim to show how the evaluation of a new music piece through dichotomy-like pairwise comparisons is done, and that comparing genres in this manner yields higher genre classification accuracies than what is presented in the previous chapter's approach. This second goal can be partitioned into a further set of goals. That is, through our experiments, we aim to achieve competitive classification results, even with a small number of features, while showing that a higher strictness during the pairwise comparison yields higher classification accuracies.

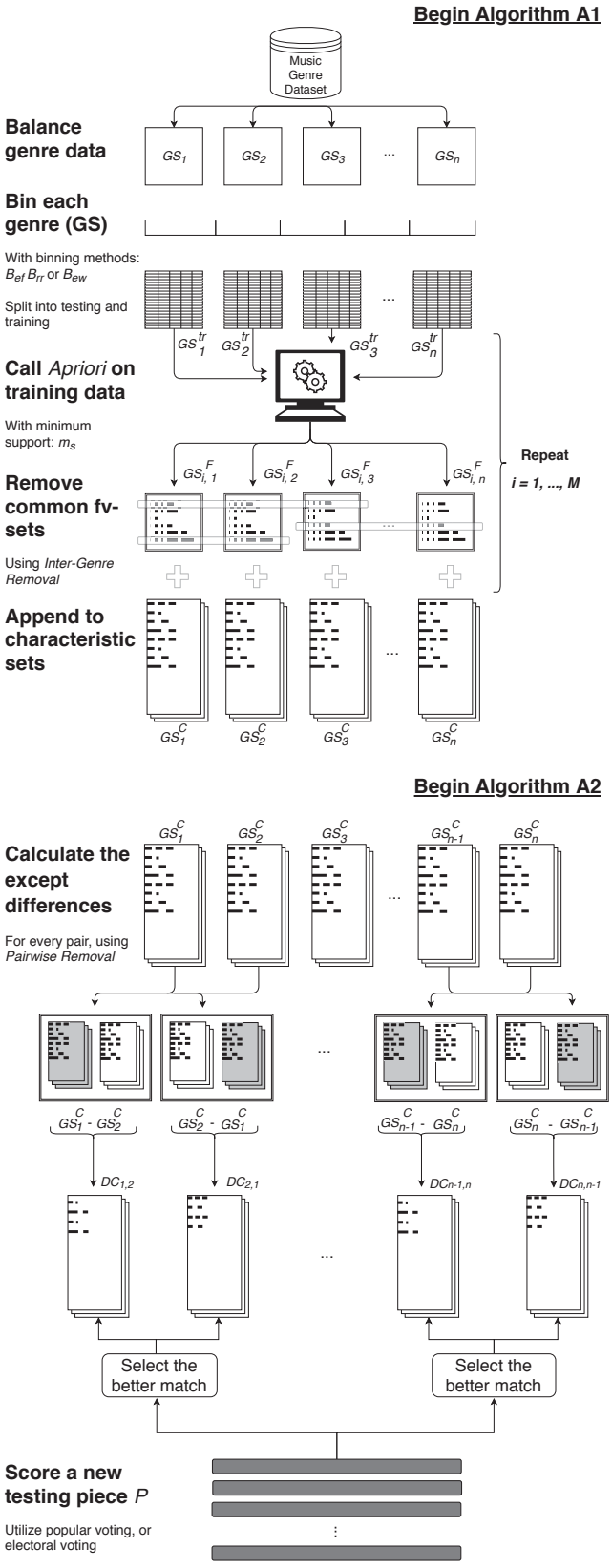
4.2 Genre-Specific Characterization and Pairwise Evaluation

The approach presented in this chapter can be broken down into two algorithms. The first algorithm generates the characteristics of the genres and the second performs the pairwise calculations on the characteristics of those genres. The first algorithm is similar to the approach in Chapter 3. The number of music pieces is balanced between the genres and the features of the music pieces are first discretized using a binning method. The resulting

discrete data objects are split into testing and training portions where the training data is sent to *Apriori* so that frequent fv-sets can be generated. The main difference between this Chapter's first algorithm and the approach found in Chapter 3, is that we take a subset of the training data each time and append the newly found frequent fv-sets to a characteristic set. We allow for the optional use of Inter-Genre Removal at every iteration.

In the second algorithm, we start with each genre's characteristic set and remove all of the characteristics from one genre that are found in another genre's characteristic set. We do this for each pair of genres which generates a number of differences for a new testing piece to match. When calculating the score of each genre for that testing piece, we match each pair against its opposite pair (i.e., jazz with classical characteristics removed versus classical with jazz characteristics removed) to see which genre better matches the testing piece. We provide the high-level depiction of both algorithms in Figure 4.1, to demonstrate how the frequent fv-sets are used to characterize genres, and how pairwise comparison is performed.

Figure 4.1: A high-level description of our second approach.



4.2.1 Generating Genres' Characteristic Sets

Association analysis, as presented in Section 2.3, and also utilized in Chapter 3, is used again to determine the data items that frequently occur together. With some minimum support value, we determine those frequent fv-sets used to characterize genres. Once again, we adapt the Apriori [1] association algorithm for our purposes.

A piece in a music dataset is represented in a similar way as the previous chapter. It is represented as a vector $P = \{p_1, p_2, \ldots, p_n\}$, where p_i is the value of the feature $f_i \in F$ and $F = \{f_1, f_2, \ldots, f_n\}$ is the acoustic feature set selected for classification.

A1: Characterizing music genres by feature-value pairs

1. For each binning method B
2. For each genre G's GS
3. Apply B to GS
4. Split GS into its training set GS^{tr} and testing set GS^{te}
5. For each G's GS^{tr}
6. Randomly generate M subsets of it (denoted as GS_i)
7. For each genre G and for each of its GS_i
8. GS_i^F = Call *Apriori* to GS_i with m_s
9. For $i = 1$ to M
10. Call procedure *Inter-Genre Removal* on all genres' GS_i^F
11. For each genre G
12. We append fv-sets from GS_i^F to GS^C and remove any duplications

This chapter's first step is detailed in the algorithm A1, and is similar to the previous chapter's approach. As before, the superscripts tr and te correspond to training and testing, a genre G has a dataset of pieces labelled G, a set GS is a randomly chosen subset taken from G (to balance the number of music pieces in each genre). We normalize the acoustic features' real values and then use a binning method to discretize these real values. The binning methods used are the same as in Chapter 3 (i.e., equal frequency, equal width, or Rice rule, denoted as B_{ef}, B_{ew}, B_{rr}, respectively). Again, we systematically encode each value for each feature, and represent a music piece as a set of feature-value pairs, called an *fv-set*. After this encoding, the frequent fv-sets for each genre's training pieces are found.

For this, we set the cardinality of each fv-set returned by Apriori to be at least 2, and utilize only maximal itemsets.

We still allow for all genres' corresponding GS_i^Fs to go through Inter-Genre Removal to remove the common fv-sets before performing the pairwise comparison. Inter-Genre Removal, in this case, is exactly the same procedure as used in the previous chapter's approach. That is, if a frequent fv-set from GS_i^F has matches from more than a certain μ of other genres, we delete it from every genre's GS_i^F. We use the parameter ϕ again, to check whether two fv-sets match each other. Except this time, we only use the relaxed version of ϕ (i.e., ϕ_r, as explained in Chapter 3). At this point, instead of scoring testing pieces using some ranking criterion, as in Chapter 3, we modify the algorithm, so that each genre's GS_i^F is appended to a characteristic set. That is, for the genre G we obtain its M sets of frequent fv-sets, where $i = 1, \ldots, i = M$ ($M = 10$ in our experiments). The purpose of doing this will now be described in further detail.

For each genre, we consolidate its M sets of frequent fv-sets (each GS_i^F), and generate its characteristic set, called GS^C. The major advantage of using the parameter M, is that a variety of possible frequent fv-sets are mined from the training data, thus producing a more complete and representative characteristic set. Furthermore, we can observe various metrics of frequent fv-sets across these M iterations. For instance, if a frequent fv-set occurs across all M iterations, we may view this fv-set as having greater importance than one that occurs in only one iteration. Furthermore, we can find the average m_s value that a frequent fv-set has or the number of times a frequent fv-set was found to be a subset of other frequent fv-sets, across all M iterations. After the characterization of genres is done, we continue to the pairwise comparison.

4.2.2 Evaluating Pairwise Music Genres

With the characteristic sets of individual genres ready, we classify an unseen music piece from a testing music dataset, which is represented as a vector of feature-value pairs

and scored against a pair of genres by comparing the differences between their respective characteristic sets to find the preferred genre. The steps, which should be done per binning method and m_s, are presented in the Algorithm A2. For a new music piece P from a subset of GS^{te} (we create 10 subsets of GS^{te} for each genre to average its experiment accuracies), we maintain a score vector $(S_{G_1}, S_{G_2}, \ldots, S_{G_n})$, where n is the number of genres under consideration and S_{G_i} is the "score" of G_i for P.

A2: Evaluating pairwise music genres by feature-value pairs

1. For a new music piece P (represented by feature-value pairs)
2. For the characteristic sets of two genres G_i and G_j, GS_i^C and GS_j^C
3. Calculate the except difference of GS_i^C and GS_j^C,
4. i.e., $DC_{ij} = GS_i^C - GS_j^C$ and $DC_{ji} = GS_j^C - GS_i^C$.
5. Score on P using DC_{ij} and DC_{ji}
6. s_i = Counting(P, DC_{ij})
7. s_j = Counting(P, DC_{ji})
8. if $s_i > s_j$ then
9. S_{GS_i} += $electoral$?1 : s_i
10. else
11. S_{GS_j} += $electoral$?1 : s_j
12. Set the genre of the highest score to be the one for P.

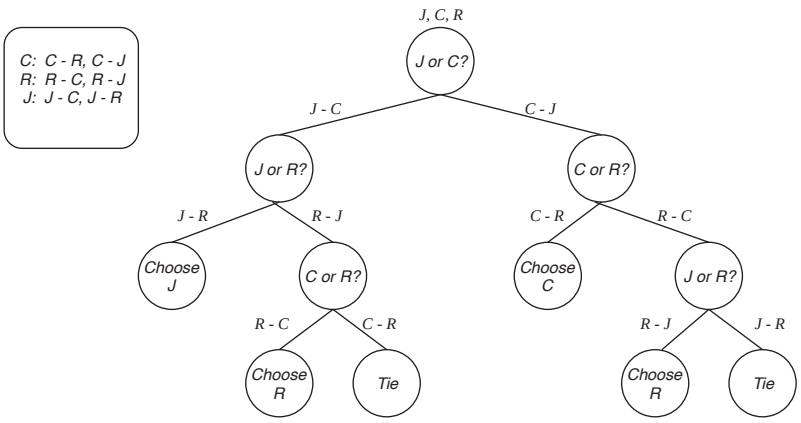
The except difference between the two characteristic sets, $GS_i^C - GS_j^C$, consists of those fv-sets that are present in GS_i^C but not in GS_j^C. We again utilize the same idea for the strictness factor ϕ in our evaluation when we conduct a "fuzzy" check for whether an fv-set from GS_i^C appears in GS_j^C, and vice versa. We label the ϕ value used during Inter-Genre Removal as ϕ_{ig} (recall that ϕ_{ig} is the same as ϕ_r as used in the previous chapter). We label the ϕ value used in the pairwise genre comparisons as ϕ_{pw}. We precalculate the except differences among all pairs of genres.

The procedure *Counting* counts how many fv-sets in the except difference are a subset of P's feature-value vector. Two mechanisms of Counting are implemented. The first is called *electoral voting* (i.e., winner takes all) and the second is called *popular voting*.

Electoral Voting

Electoral voting works as follows. It finds the number of times that the fv-sets in the except difference appear in P (normalized based on the size of the difference). This is then compared to the number of matches made with the other except difference (i.e., DC_{ij} versus DC_{ji}), and the except difference that provides the higher score determines what genre gets a point. The genre with the highest number of points is chosen. Therefore, if G^* is the set containing all of the genres, the number of except differences to be calculated are: $(|G^*|-1)\cdot(|G^*|)$, and the maximum number of votes that can be awarded towards a genre for any testing piece is $(|G^*|-1)$. The process of electoral voting may be understood as a type of decision tree, as shown in Figure 4.2. Let C, R, J represent the characteristic sets for classical, rock, and jazz, respectively. Let $J-C$ be the except difference representing all the characteristic frequent fv-sets in jazz that are not in classical (that is, i is jazz, and j is classical, and D_{ij} is $J-C$).

Figure 4.2: Electoral voting process as a decision tree.



Popular Voting

Popular voting sums up the number of matches to frequent fv-sets that the fv-set of a piece P has, across all of the except differences for a particular genre G_i. The number of matches found by Counting is normalized based on the number of frequent fv-sets in the

except difference (i.e., $|D_{ij}|$). Therefore, the genre that is chosen for a testing piece is given by the expression

$$\max_{G_i \in G^*} \sum_{j}^{|G^*|} \frac{Counting(P,\ DC_{ij})}{|DC_{ij}|}, \tag{4.1}$$

where $i \neq j$. We will see in Section 4.3 that the popular voting mechanism obscures the except differences and thus yields a lower classification accuracy.

4.3 Experiment Preparations

Now that the music genre characterization and pairwise comparison algorithms have been described in detail, the datasets, genres, and features used to determine the experiment results will now be discussed.

4.3.1 Datasets and Features

To verify our proposed approach, we apply it to D_{LMD} and D_{CAL}. We use the same sets of features found in Section 3.3.3 (Table 3.2) for both datasets. Recall that D_{CAL} only has 13 *Mel-frequency cepstral coefficients (MFCCs)* extracted over the entirety of each music piece, however in this chapter we create a subset of features taken from the middle segment only (each section is split evenly based on the music piece's length). For the extraction of MFCCs, the sampling rate is 22050 Hz, with a window size of 2048ms, and hop size of 1024ms. The standard deviation and mean across the analysis windows are calculated for each MFCC. Recall that D_{LMD} has 5 MFCCs extracted, as well as spectral-based, temporal-based, and pitch-based features. Each feature has its mean and variance calculated across the analysis windows.

4.3.2 Genres, Subset Selection, and Discretization Methods

Processing Details of D_{CAL}

In the D_{CAL} experiments we use six (6) genres to conduct two (2) sets of experiments, these genres include: *(C)lassical*, *(M)etal*, *(H)ip (H)op*, *(F)olk*, *(P)op*, and *(RE)ggae*. The m_s values that are used in the Apriori algorithm are 4%, 6%, and 8%. Three binning methods are used to preprocess music data, including: B_{ew}, B_{ef}, and B_{rr}. 10 bins are used for B_{ef} and 15 are used for B_{ew}.

The first set of experiments utilizes all six genres, and the second set excludes the genre pop, which is frequently misclassified. These experiments further break into two types: 1) using a hierarchical promotion of subgenres (at least two subgenres per their parent genre label, including the parent genre label as well), called *subgenre uplifting*, with 104 training pieces and 44 testing pieces; and 2) using pieces labelled only as a parent class in D_{CAL}, with 115 training pieces and 48 testing pieces, and with the genre *(R)ock* used in place of metal. The purpose of this is to investigate how the classification accuracy is affected by the uplifting of subgenres. Since there is more variation in the data when uplifting subgenres, we believe that these experiments should yield lower classification accuracies.

Processing Details of D_{LMD}

For D_{LMD} we use the entirety of the dataset, experimenting on each 30-second segment of the songs separately. The experiments are conducted using all of the songs possible per 10 genres after the subset equalization of songs. We utilize the same preprocessing strategy for D_{LMD}, as mentioned in Chapter 3 (i.e., we first collapse the folds, then remove duplicate feature vectors). Therefore, we use 2, 973 music pieces per genre for the beginning segment and 2, 988 music pieces per genre for the middle and end segments. The purpose for these experiments is to determine how well the classifier performs against other state-of-the-art classification approaches. We use all three binning methods (i.e., B_{ew}, B_{ef}, and B_{rr} with the same number of bins given above), and an 80%, 20% training testing split. m_s values of

3%, 4%, 5%, and 6% are used.

4.4 Experiment Results and Discussions

Not only are we concerned with demonstrating the classification accuracy of our genre characterization and pairwise evaluation algorithms, we also determine if sufficient accuracies can be achieved with a small number of content-based audio features. We further determine if a stricter ϕ_{pw} threshold will accomplish this. However, when applying ϕ_{ig} we execute classification experiments that demonstrate whether or not this produces less than ideal characteristic sets per genre, therefore producing a lower classification accuracy. For our purposes, we make the procedure Inter-Genre Removal an option in our algorithm. If we do not use it, we use n/a in the reports of our experiments.

For D_{CAL}, we notice that B_{ef} performs better, on average, for the six genre experiments without subgenre uplifting, meaning that this binning method is more useful when a greater variety of genre information is present. So in the next discussion, we fix the binning method to be B_{ef}. However, we do find that the trends present in the B_{ef} binning method also apply to other binning methods.

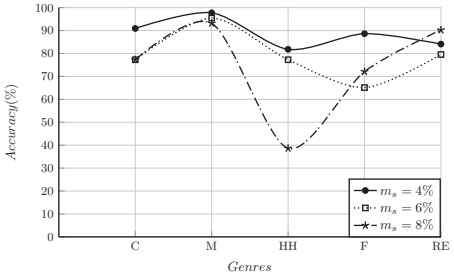


Figure 4.3: *With* subgenre uplift, B_{ef}, $\phi_{pw} = 0.3$, μ n/a, ϕ_{ig} n/a, Electoral, *pop* absent.

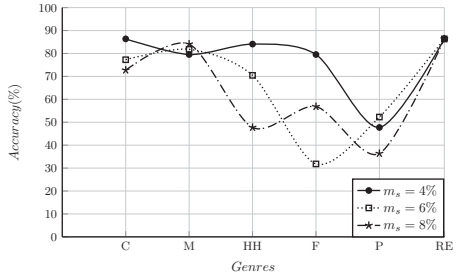


Figure 4.4: *With* subgenre uplift, B_{ef}, $\phi_{pw} = 0.3$, μ n/a, ϕ_{ig} n/a, Electoral, *pop* present.

In Figure 4.3 we first see the trend that we found in Chapter 3, that a lower m_s threshold yields higher classification accuracies. As before, it is believed that this is due to the greater volume of complex characteristic relationships that are found, since a lower m_s threshold is used to return a greater variety of frequent fv-sets. Comparable accuracies are also achieved

for subgenre uplift with smaller m_s values, but the stability between genres decreases as m_s increases. This is due to the introduction of testing pieces that do not explicitly represent that genre. However, we use a stricter ϕ_{pw} value, with a smaller m_s, which yields a higher accuracy even when other subgenres are introduced. In Figure 4.4, we notice that music pieces from the genre pop are misclassified, furthermore, pop is the only genre whose classification accuracy drops dramatically for a lower m_s value. This demonstrates that the impact of including a confusing genre can be softened using pairwise comparisons.

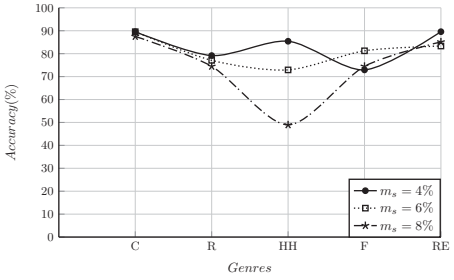


Figure 4.5: *Without* subgenre uplift, B_{ef}, $\phi_{pw} = 0.3$, μ n/a, ϕ_{ig} n/a, Electoral, *pop* absent.

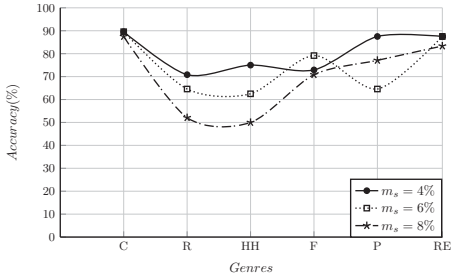


Figure 4.6: *Without* subgenre uplift, B_{ef}, $\phi_{pw} = 0.3$, μ n/a, ϕ_{ig} n/a, Electoral, *pop* present.

In Figures 4.4 and 4.6, the presence of pop does lower the classification accuracies of other genres. However, this effect is not as significant as we expected. Characteristics of pop can be found in various genres, so it is no surprise that the predictions suffer to some extent. Based on our experience with D_{CAL}, easily confused genres (i.e., reggae, hip hop, folk, etc.) are now often classified correctly. In Figures 4.4 and 4.6, we continue to see the trend that a lower m_s causes a higher prediction accuracy, and that a more strict dataset increases prediction accuracy.

When comparing those experiment results on the genre uplift and flat genre hierarchies, we make a handful of observations. First, we note that metal was more distinguishable, compared to rock. We also observe that when uplifting subgenres, there are significant "valleys" in both the five and six genre experiments, especially as the m_s thresholds are increased. Lastly, the accuracy of the music genre pop was affected more when subgenre uplift was performed. These observations match our intuitions. Fewer characteristics of

metal can be found in other genres when compared to rock, a higher m_s value reduces the frequent fv-sets that a testing piece can match against, and when subgenre uplift is performed there is a larger variety of data present for pop to be confused with.

The confusions for each genre in the above figures also match our intuitions. For instance, hip hop was confused with pop and reggae, rock was confused with pop, classical was confused with folk, and folk was sometimes confused with pop, hip hop, and reggae, and lastly, reggae was confused with pop and hip hop. Therefore, the confusions for D_{CAL} remain intuitive. Interestingly, pop and metal are both a source of confusion for one another, this may be due to the sometimes compressed production, and the rhythmical strictness inherent in both genres. However, MFCCs are the only features used, so it is possible that more features would differentiate these genres better.

In Tables 4.1 and 4.2, we see that a stricter ϕ_{pw} value yields higher prediction accuracies. The rows with a $*$ in Table 4.2 represents a μ value of 33% and a ϕ_{ig} value of 60%, which is used during Inter-Genre Removal. The average accuracies for ϕ_{pw} suffer when Inter-Genre Removal is called. This confirms our hypothesis that applying a strict μ value during Inter-Genre Removal creates a "non-optimal" characteristic set for some genres. We believe this is due to the removal of characteristics in a genre that could improve the classification when comparing against only one other genre (as done with a pairwise comparison). Recall from Chapter 3 that a stricter μ value normally yields higher accuracies, however, with this extra call to remove frequent overlapping fv-sets during Inter-Genre Removal, we notice that lower prediction accuracies do occur. The best result is obtained with a strict pairwise comparison threshold only, where we leave the accumulation of the characteristic sets (of each genre) intact until the pairwise comparison is done. In the future, we must experiment with even more μ and ϕ_{ig} values during the Inter-Genre Removal procedure, for now though, we do notice the aforementioned trend.

Table 4.1: *With* subgenre uplift, $m_s = 4\%$, B_{ef}, μ n/a, ϕ_{ig} n/a, Electoral, *pop* absent.

	C	M	HH	F	RE
$\phi_{pw} = 0.3$	0.91	0.98	0.82	0.89	0.84
$\phi_{pw} = 0.5$	0.77	0.93	0.41	0.75	0.59
$\phi_{pw} = 0.6$	0.50	0.93	0.34	0.78	0.59

Table 4.2: *With* subgenre uplift, $m_s = 4\%$, B_{ef}, $\mu = 0.33$* and ϕ_{ig} = 0.6*, otherwise μ and ϕ_{ig} n/a, Electoral, *pop* present.

	C	M	HH	F	P	RE
$\phi_{pw} = 0.3$	0.86	0.80	0.84	0.80	0.48	0.86
$\phi_{pw} = 0.5$	0.82	0.80	0.59	0.52	0.50	0.64
$\phi_{pw} = 0.6$	0.64	0.68	0.41	0.55	0.68	0.59
$\phi_{pw} = 0.3$*	0.75	0.89	0.68	0.68	0.61	0.77
$\phi_{pw} = 0.5$*	0.68	0.81	0.45	0.59	0.61	0.70
$\phi_{pw} = 0.6$*	0.75	0.68	0.39	0.68	0.52	0.59

In Tables 4.3 and 4.4, we notice that B_{ef} is the most stable binning method, with only a drop in accuracy for the genre pop, as in Table 4.4. When pop is reintroduced, as shown in Table 4.4, hip hop and folk's prediction accuracies are lower, but with B_{ef} the impact of introducing a confusing genre is softened.

Table 4.3: *With* subgenre uplift, $m_s = 4\%$, various binning methods, $\phi_{pw} = 0.3$, μ n/a, ϕ_{ig} n/a, Electoral, *pop* absent.

	C	M	HH	F	RE
B_{ef}	0.91	0.98	0.82	0.89	0.84
B_{ew}	0.93	0.91	0.86	0.89	0.91
B_{rr}	0.86	0.91	0.64	0.77	0.91

Table 4.4: *With* subgenre uplift, $m_s = 4\%$, various binning methods, $\phi_{pw} = 0.3$, μ n/a ϕ_{ig} n/a, Electoral, *pop* present.

	C	M	HH	F	P	RE
B_{ef}	0.86	0.80	0.84	0.80	0.48	0.86
B_{ew}	0.93	0.82	0.55	0.52	0.61	0.82
B_{rr}	0.91	0.75	0.64	0.41	0.43	0.73

Next, we investigate how the popular voting method performed. This is presented below in Tables 4.5 and 4.6.

Table 4.5: *With* subgenre uplift, $m_s = 4\%$, B_{ef}, μ n/a, ϕ_{ig} n/a, Popular, *pop* absent.

	C	M	HH	F	RE
$\phi_{pw} = 0.3$	0.55	0.41	0.70	0.90	0.68
$\phi_{pw} = 0.5$	0.57	0.86	0.39	0.80	0.50
$\phi_{pw} = 0.6$	0.48	0.93	0.45	0.80	0.55

Table 4.6: *With* subgenre uplift, $m_s = 4\%$, B_{ef}, μ n/a, ϕ_{ig} n/a, Popular, *pop* present.

	C	M	HH	F	P	RE
$\phi_{pw} = 0.3$	0.39	0.27	0.34	0.61	0.70	0.43
$\phi_{pw} = 0.5$	0.61	0.59	0.43	0.52	0.70	0.41
$\phi_{pw} = 0.6$	0.59	0.64	0.48	0.64	0.77	0.52

The most notable observation to be made is that the popular voting method underper-

forms for stricter ϕ_{pw} thresholds, and works opposite to the Electoral voting method (at least for the experiments with subgenre uplift). In the case of metal, which should be distinguishable, it becomes less recognizable for stricter ϕ_{pw} values. In the experiments with subgenre uplift more frequent fv-sets are generally removed for each genre when using a stricter ϕ_{pw} threshold, it is possible that these frequent fv-sets no longer have the capacity to optimally characterize the genre when popular voting is applied. That is, when comparing the sum of matches across all except differences for one genre, the score for that genre is solely dependent on the number of matches that occur, so with a strict removal threshold, genres that were once distinguishable may be less distinguishable, due to the decreased size of the except differences characterizing that genre (and therefore, the fewer chances a testing piece has of matching all of a genre's frequent fv-sets). Therefore, we determine that the popular voting method is less effective when using a pairwise comparison.

Next, we present our experiment results using D_{LMD}. In Figure 4.7, we find further confirmation that a lower support yields a higher classification accuracy. In Figure 4.8, the trend that stricter removal thresholds yield greater classification accuracies is also confirmed. We do see that those distinct genres (i.e., *tango* and *bachata*) benefit more from applying a stricter ϕ_{pw} value, and less distinguishable genres achieve lower accuracies. Some fine-tuning of this threshold may be needed, although the strictest ϕ_{pw} value generally produces a better average overall accuracy. In Figure 4.9, we see again that B_{ef} performs in a stable manner, and is less prone to accuracy drops for certain subgenres, which might occur with B_{ew}, or B_{rr}. Note that the binning method can exacerbate the difference between distinguishable and less distinguishable genres and make the classification less stable. At this point we have observed in several experiments that the most effective combination of parameters are a low minimum support (m_s) and a strict pairwise removal threshold (ϕ_{pw}), generally with equal frequency binning (B_{ef}) generating the most stable and greater classification accuracies. Finally, the segment of music pieces that achieves the highest accuracy for all of D_{LMD}'s ten (10) genres is the middle section, as we have seen in Chapter 3. This

matches other approaches [41], and is shown in Figure 4.10.

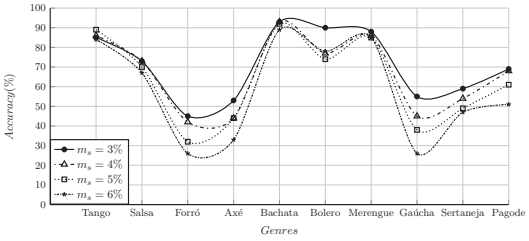


Figure 4.7: Analysis of m_s with parameters: D_{LMD}, $\phi_{pw} = 0.4$, μ n/a, ϕ_{ig} n/a, B_{ef}.

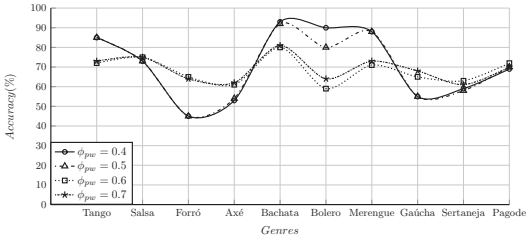


Figure 4.8: Analysis of ϕ_{pw} with parameters: D_{LMD}, $m_s = 3\%$, μ n/a, ϕ_{ig} n/a, B_{ef}.

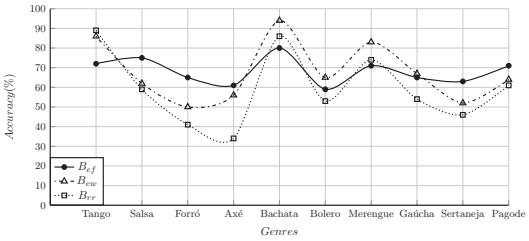


Figure 4.9: Analysis of binning with parameters: D_{LMD}, $m_s = 3\%$, μ n/a, ϕ_{ig} n/a, $\phi_{pw} = 0.6$.

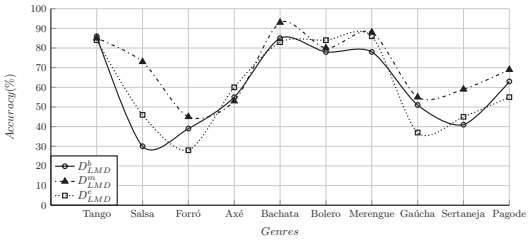


Figure 4.10: Analysis of segments with parameters: D_{LMD}, $m_s = 3\%$, $\phi_{pw} = 0.4$, μ n/a, ϕ_{ig} n/a, B_{ef}.

In Table 4.7, we see the confusion matrix for D^M_{LMD}. There is a noticeable improvement compared to the approach in Chapter 3 (Table 3.4, Figure 3.6, and Figure 3.7), especially since the experiment uses 10 genres. Several familiarities are noticed when comparing those experiment results in Chapter 3 to the confusion matrix for D^M_{LMD}. For example, we find that both *merengue* and tango are quite distinguishable, that *sertaneja* is a source of confusion for *pagode*, *bolero* is sometimes confused with sertaneja, and finally, *axé* and *fórro* are both less distinguishable than other genres. So it is still the case that many of the same confusions happen to occur, however, some confusions are softened with a strict ϕ_{pw} threshold, and no usage (i.e., n/a) of the ϕ_{ig} or μ thresholds. One further observation is that genres are often rarely misclassified as tango, which is to be expected since in D_{LMD}, there is the proclivity for tango to contain older and noisier recordings [90].

This chapter's approach can obtain classification accuracies for D_{LMD} that outperform many existing approaches. For instance, when compared to Silla et al.'s work [41] we

Table 4.7: Confusion matrix ($M = 10$) - D^m_{LMD}, $m_s = 3\%$, B_{ef}, μ n/a, ϕ_{ig} n/a, $\phi_{pw} = 0.4$

	Tango	Salsa	Forró	Axé	Bachata	Bolero	Merengue	Gaúcha	Sertaneja	Pagode
Tango	**0.85**	0.00	0.00	0.00	0.00	0.14	0.0	0.01	0.00	0.00
Salsa	0.00	**0.73**	0.02	0.02	0.02	0.06	0.04	0.03	0.04	0.04
Forró	0.00	0.07	**0.45**	0.03	0.02	0.11	0.06	0.07	0.07	0.12
Axé	0.00	0.08	0.04	**0.53**	0.02	0.03	0.13	0.04	0.07	0.06
Bachata	0.00	0.02	0.01	0.00	**0.93**	0.02	0.02	0.00	0.00	0.01
Bolero	0.01	0.03	0.01	0.01	0.00	**0.80**	0.00	0.03	0.09	0.02
Merengue	0.00	0.03	0.01	0.01	0.05	0.00	**0.88**	0.01	0.01	0.00
Gaúcha	0.00	0.06	0.02	0.07	0.02	0.10	0.07	**0.55**	0.05	0.06
Sertaneja	0.00	0.06	0.03	0.08	0.00	0.14	0.03	0.01	**0.59**	0.06
Pagode	0.00	0.02	0.03	0.03	0.00	0.07	0.03	0.02	0.11	**0.69**

find that we are able to achieve higher classification accuracies than all of the methods proposed using the same set of features. In Table 4.8, we further compare this chapter's approach to other approaches using D_{LMD} by observing the best and worst raw classification accuracies (from 2015 to 2017) for the Music Information Retrieval Evaluation eXchange (MIREX) genre classification task using D_{LMD}. Recall that we do not use any feature selection approaches, and only use those features introduced by Silla et al. in [41].

Table 4.8: A comparison of approaches using the MIREX Audio Train/Test: Genre Classification (Latin) Task.

MIREX Year	Best Classification Accuracy % (Author)	Worst Classification Accuracy % (Author)
2015	66.9% (Cai et al. [14])	54.69% (Lidy [52])
2016	69.88% (Lidy and Schindler [53])	62.78% (Foleiss and Tavares [29])
2017	75.86% (Lee et al. [49])	61.48% (Xu et al. [97])
2018	70.0% **(Proposed Approach)** Parameters: $\phi_{pw} = 0.4$, ϕ_{ig} n/a, μ n/a, $m_s = 3\%$, B_{ef}, D^m_{LMD}	

For this chapter, we only conducted our experiments with $M = 10$. With a higher M value it is possible that even more characteristic frequent fv-sets can be found per genre, therefore, each genre's characteristic set will become even more representative of that genre. When M is smaller, less frequent fv-sets for each genre can be found; however, the amount of processing during the pairwise removal will be less. In these cases, leaving this parameter to the user provides a trade-off between running time and accuracy, which may assist with scalability issues in the future.

We can also fine-tune the parameters per pair, allowing for plenty of flexibility. That

is, each pair of genres could have a specific pairwise removal threshold, minimum support, binning method, etc. This would be especially useful for genres that do not follow the trends previously outlined, as we observe that a strict pairwise removal threshold does not always yield the highest accuracies for a genre such as pop (as in Table 4.2), and that some genres can benefit from higher m_s values (as in Figures 4.5 and 4.6), for instance.

As mentioned, it is easy to collect the frequency information, average support, or the number of times a frequent fv-set is a subset in a genre's M frequent sets. This criteria may be further used as a linear combination, such that each statistic for a frequent fv-set is given some weight and awarded to a testing piece during the Counting procedure. The use of these statistics may be explored further in future work, although, it was found that a naive method of counting the matching fv-sets performed successfully. The statistics for each frequent fv-set in a genre's characteristic set may be convenient for various feature selection techniques. One can determine the importance of a frequent fv-set based on the statistics described above, which may assist in the pruning of unnecessary features.

4.5 Summary

In this chapter, we have presented our second novel approach to the music genre classification problem. Like in Chapter 3, it captures our intuitions that each music genre has its own characteristics which can be extracted and represented. Moreover, we find that genres compete for their influence when classifying music data, and therefore a pairwise, dichotomy-like approach is appropriate for distinguishing genres. To our knowledge, this is the first of its kind in the literature, and the initial experiments show that our approach is promising.

Since the pairwise dichotomy-like strategy presented in this chapter is effective in differentiating the characteristics of genres, we will now conduct classifications on various sets of similar subgenres. This task is more difficult since the characteristics between any two subgenres of a parent genre can be harder to decouple than normal parent genres.

Chapter 5

Characterization and Classification of Music Subgenres

In this chapter, we discuss the difficult and often overlooked problem of subgenre classification. So far, we have only reported our experiments with genres. The experiments closest to using subgenres are those on D_{LMD}. However, Silla et al. [41, 83] state that this dataset is comprised of genres, and not subgenres. In some datasets, *Latin* music may be collapsed into one genre comprised of several subgenres (as in D_{CAL} [91]). So, from the data processing point of view, D_{LMD} may be viewed as either a group of genres or a group of subgenres. This sheds some light on why automatic subgenre classification is important; classifier confusions may easily arise from such broad groupings (e.g., representing all non-western music as "world" music). On the other hand, it is possible that the effectiveness of a classifier is not truly being tested since this broad grouping overlooks many important musical details (e.g., differences in instrumentation, rhythmic and harmonic structures). Thus, there are many essential motivations for this chapter that will be discussed shortly.

This chapter is organized as follows. In Section 5.1, we briefly detail what a music subgenre is, and relate it to the task of genre classification. We further state our goals for this chapter. In Section 5.2, we outline the motivations and difficulties of performing subgenre classification. We provide a discussion on why this task should not be overlooked and why we are motivated to perform our experiments. In Section 5.3, we describe which of the algorithms we use from the last two chapters, and why it is used. In Section 5.4, we discuss the datasets and features utilized, as well as the choices of subgenres for each

parent genre and why these subgenres are chosen. Lastly, in Section 5.5, we compare our experiment results to the goals outlined in the following section.

5.1 Introduction

The purpose of labelling pieces as music subgenres is not just to divide music that sounds different. There are also deep social, financial, and artistic implications for the emerging labels that describe these subgenres. For example, an artist may wish to describe their music as a new subgenre, a record company may present an artist under a new subgenre label to increase record sales, or a subgenre may be created as a reaction to political agendas, other music styles, or as a gatekeeping device to exclude (or include) other listeners. Any two subgenres may be quite different from one another (despite having the same parent genre label). In summary, a subgenre's label is assigned for a variety of deeply complex reasons [63]. This makes the hierarchical organization of music subgenres less straightforward than with genres.

Since we have found that there are not many works concerned with subgenre classification at large (i.e., for the most common parent genres), it is a primary goal to provide a transparent, and reproducible experiment design for the subgenre classification tasks. We also aim to demonstrate that the approach presented in Chapter 4 can report the distinguishable subgenres with high accuracies and can achieve comparable accuracies across different datasets. Finally, we use the following subgenre classification experiments to observe how parameters may need to be tuned differently when applying the selected approach.

5.2 Motivations and Difficulties

Some ambiguities may occur with defining subgenres, since some qualities of one *subsubgenre*, such as the production, musical performance, harmonic content, and timbre, may be quite similar to its parent subgenre in their parent genre. For instance some pieces of *shoegaze* can be very similar to *psychedelic rock* (both technically subgenres of *rock*) in

every musical way. Moreover, the labels are coined for various purposes that may be difficult for a simple content-based classifier to delineate, such as slight differences in musical performance style. For a successful classification approach, this should not be an issue. However, some subgenres are too generic to include in classification tasks and will cause excess confusion.

A major difficulty associated with the subgenre classification task is the limited access to data supporting this task. Mainly, there is a limited number of datasets supporting subgenre classifications. This shortage of appropriate data may be the cause of the lack of subgenre classification research in MIR.

Recall from Section 2.7.2, that there is somewhat of a void with respect to the current subgenre classification research in MIR. We have found that the classification of music into subgenres is often neglected. Many recent works do not address the problem of subgenre classification at large, though there are several works attempting to classify region-specific music [3, 44], ballroom music [64], and Latin music [23, 41, 76]. One of our primary motivations is to fill this void.

It should be the aim of any classification approach to classify music into the finest category possible. Intuitively, this is the next step for successful genre classifications, as it would benefit those that listen to particular subgenres. In a practical sense, many people may spend more time listening to one specific subgenre than any other subgenre, or even other genres as a whole. This is our second motivation; to show that our approach can provide a more detailed classification that is useful for listeners.

5.3 Approach

In this chapter, we make use of the approach presented in Chapter 4 exclusively. That is, we extract the distinguishable characteristics that makes each genre unique. We continue characterizing each music subgenre by a set of acoustic features through association analysis and then conduct the dichotomy-like pairwise comparisons. That is, we now consider

applying the approach presented in Chapter 4 to the problem of music subgenre classification. We review the proposed approach and discuss the reasons for selecting it.

The approach presented in Chapter 4 is chosen for several reasons. For instance, it performed well in the experiments with subgenre uplifting using six genres. This means that there is some chance that it is able to identify and group those subgenres that belong to their parent genres. There may be plenty of overlapping subgenre information, so by using the pairwise removal method on each subgenre's characteristic sets, the useful subgenre information will be kept, while removing only the necessary overlapping characteristics from the "rival" subgenre's characteristic set. A brief summary of this approach will now be given.

We select a random subset of pieces from a music genre dataset and represent each music piece in it as a discretized vector of features called an fv-set. To do so, we use a binning method, such as: B_{ef}, B_{ew}, or B_{rr}. Next, we split these pieces into testing and training components. We again use association analysis, and in particular, the *Apriori* algorithm, to mine frequent (maximal) fv-sets with some m_s threshold, disregarding those frequent fv-sets less than a size of 2. We then obtain M sets of frequent fv-sets per subgenre, again, $M = 10$ in our experiments. We perform the same "fuzzy" check for whether an fv-set appears in the characteristic sets of a pair of subgenres. We make use of ϕ_{pw} for this, as discussed in Section 4.2.2, however, we do not make use of the variable ϕ_{ig}, since we saw in Chapter 4 that it may create a less than optimal characteristic set for a genre. We then classify the pieces of music (now represented as a vector of feature-value pairs) from the testing music dataset and score it against each pair of subgenres. The except differences are calculated for each pair of subgenres, and a score vector per testing piece, with scores for each subgenre, is maintained. However, we no longer experiment using the popular voting mechanism, and only utilize electoral voting, since we found in Section 4.4 that it obscures the except differences. For a more detailed description of the selected approach, please see Section 4.2.

5.4 Experiment Preparations

For our purposes, we explicitly use the subgenres defined in the datasets that are used, even if the assigned labels are awkward from a musicological point of view. That is, we are reliant on those labelling the pieces in each dataset. The problems with relying on correct labelling are exacerbated when performing subgenre classification since subgenres can be even more difficult to distinguish than genres. For instance, if those labelling the data are unfamiliar with the subgenres' parent genre then the data may be labelled incorrectly. Therefore, we decide to use two frequently used datasets from the MIR literature, as described with greater detail in Section 5.4.1.

For simplicity, if a subgenre is a *child* of a *parent* genre in the dataset, then we claim that it is, in fact, a subgenre. We then continue to use this label for music subgenre classification tasks. Sometimes a subgenre may belong to multiple parent genres (e.g., *folk rock*), or numerous subgenres. We accept that a genre may belong to two or three similar parent genres, by including these pieces in the classification process. However, we reject pieces belonging to more than one subgenre, on the premise that we are performing subgenre classification. If a piece belongs to two subgenres, then this is a task for multi-label classification. Furthermore, we are able to keep the representative and "pure" subgenre pieces by avoiding subgenre overlappings. In order to have enough complexity in the data, and allow for practical classification scenarios, we select at least three (3) subgenres per parent genre, but less than or equal to six (6). We also select subgenres where the number of music pieces is 50 songs or greater. It may also be the case that a subgenre may have a parent that is also a subgenre (i.e., the subgenre is actually a sub-subgenre), even though pieces are labelled separately in the dataset with no reference to this. We attempt to avoid this scenario, as it may cause unnecessary confusions during music subgenre classification.

5.4.1 Datasets

We include two large benchmark music datasets from the MIR community in our experiments. They are the Cal10k (D_{CAL}) dataset, and the *Free Music Archive* dataset (D_{FMA}). We have discussed D_{CAL} previously, as it is used for the experiment sections in Chapters 3 and 4, and is introduced in Section 2.6. However, the new dataset that we include in the experiment portion of this chapter is D_{FMA} [9] (see Section 2.6.2 for more details). There are a number of reasons why we use these datasets. For one, they are large enough to support subgenre classification tasks, with an appropriate amount of songs per subgenre. Recall that, for D_{CAL}, genre labels are assigned for each music piece by the consensus of experts [91], and for D_{FMA}, each (sub)genre label is assigned by the artist that created each music piece, therefore with D_{FMA} some labelling noise is possible, as artists may have many motivations for labelling their music [9]. There is some trust given to the genre hierarchies and labels defined in these datasets. However, it is important to note that subgenres, by their nature, are also harder to define for many pieces, as the distinction between them can be even more blurry than with genres.

5.4.2 Selected Genres and Subgenres

For both D_{CAL} and D_{FMA} many standard parent genres are successfully incorporated into our experiments. In Table 5.1 and Table 5.2, we provide all of the genres and subgenres found in Section 5.5, and in Appendix A. We will now detail some important subgenres that are excluded from our experiments. We begin with D_{CAL}.

Table 5.1: D_{CAL} genre/subgenre hierarchy as used in Section 5.5.

Genres	Subgenres
blues	delta blues, chicago blues, texas blues
classical	piano concerti, symphonic, violin features, piano solo, choral
country	country rock, old time country, traditional country, country pop, western swing
electronic	drum and bass, trance, trip hop, techno, industrial, ambient
folk	traditional folk, contemporary folk, british folk, bluegrass
hip hop	underground hiphop, southern rap, classic hiphop, electro
jazz	smooth jazz, jazz fusion, avante garde jazz, bebop, swing
metal	heavy metal, doom metal, stoner rock/metal, deathcore metal
pop	new age pop, pop rock, dance pop, classic pop, teen pop
reggae	dub, reggaeton, dancehall, ska
rhythm and blues (rnb)	soul, funk, traditional gospel, contemporary gospel

Table 5.2: D_{FMA} genre/subgenre hierarchy as used in Section 5.5.

Genres	Subgenres
country	bluegrass, rockabilly, country and western
electronic	house, glitch, drum and bass, downtempo, dubstep
folk	british folk, free folk, freak folk
metal	thrash, black metal, death metal, sludge, grindcore
punk	hardcore, postpunk, electropunk, no wave
rock	krautrock, new wave, post rock, shoegaze, industrial, progressive

Genre and Subgenre Selection for D_{CAL}

For the genre *blues*, the subgenre *early blues* is unfortunately excluded, due to the small number of music pieces. However, we believe this subgenre would be quite distinguishable, as we found the genre *tango* (from D_{LMD}) to be distinguishable because of the number of older recordings present in the data. Several other subgenres are excluded for the genre *classical*. These classical subgenres are more "historical" in nature, with music from the Baroque, Classical, and Romantic eras, however only the *romantic* subgenre has more than 50 music pieces, so we focus primarily on subgenres relating to classical music's instrumentation, with our intuition being that *piano solo* can be frequently classified correctly. Some specific parent genres are excluded, including: *new age*, and Latin, for instance. There are only two subgenres for new age, and we have conducted experiments on D_{LMD} already. Some choices are made regarding which of the parent genres a possible piece can belong to. For instance, if a music piece is labelled as both *country* and *folk* we include this piece, as long as it is labelled as only one of the subgenres in Table 5.1. Subgenres that would have created excess confusions are also removed, that is, they are too similar,

or vague, when compared to the other subgenres (e.g., *electric blues*, *adult contemporary*, *contemporary country*, etc.). Other than the exclusions defined above, most of the subgenre information is present in the experiments for D_{CAL}, and allows for plenty of potential classification scenarios.

Genre and Subgenre Selection for D_{FMA}

Although D_{FMA} is a larger dataset than D_{CAL}, it is still quite unbalanced (i.e., a significant contrast in the number of music pieces from genre to genre) [9]. We only used a subset of the possible genres and subgenres, mostly due to the stipulations for data selection described above. For D_{FMA}, the stipulations that eliminated many pieces from the classification process are: a music piece can only belong to one subgenre, and there has to be more than 50 songs per subgenre. This removes parent genres such as *jazz*, and classical, from consideration. Non-musical genres are also excluded, which may still technically be labelled as genres (e.g., spoken). Genres such as *experimental*, or *instrumental* are also excluded, since they can cause unnecessary confusions; even though they are technically genres, they describe vast amounts of music outside of those genres.

5.4.3 Features

We use the standard approach for those features extracted from D_{CAL}, that is, 13 MFCC features are extracted from the middle segment via MARSYAS [94]. Recall that with D_{CAL}, the song segments are divided evenly. We will refer to this MFCC feature set as F^{M}_{CAL}. For D_{FMA}, we are using the "large" version of the dataset (106, 574 songs total) with two acoustic feature sets extracted using LibROSA [60] from the middle 30 seconds per song, as presented by Defferrard et al. [9]. We denote these feature sets as F^{L1}_{FMA} and F^{L2}_{FMA}, respectively. To be even more complete, we also create a less sizeable but more diverse set of features extracted using jAudio [59], with similar parameters to F^{M}_{CAL}, and similar features used in the experiments on D_{LMD}, as found in previous chapters. We will refer to this feature set as F^{J}_{FMA}. We use various feature sets for D_{FMA} because it is comprised of "raw"

data, in that its songs are directly dumped from the original repository (see Section 2.6), so extracting a variety of feature sets is beneficial for our experiments, since the performance of our approach with respect to feature sets of various sizes is of interest to us. Furthermore, a classifier should perform better, on average, for datasets with a greater variety and number of features [2]. The summary of these features is presented below in Table 5.3.

Table 5.3: A summary of the features used in the subgenre experiments.

Dataset	Feature Sets	Features	Parameters	Calculations
D_{CAL}	F^{M}_{CAL}	13 MFCCs (MARSYAS)	window size: 2048ms hop size: 1024ms sampling rate: 22050Hz	mean, std. dev.
D_{FMA}	F^{L1}_{FMA}	20 MFCCs (LibROSA)	window size: 2048ms hop size: 512ms sampling rates: unchanged	mean, std. dev., skew, kurtosis, median, min, max
	F^{L2}_{FMA}	20 MFCCs, spectral contrast and centroid (LibROSA)	window size: 2048ms hop size: 512ms sampling rates: unchanged	mean, std. dev., skew, kurtosis, median, min, max
	F^{J}_{FMA}	13 MFCCs, spectral centroid, zero crossings, strongest beat overall, beat sum overall, strength of strongest beat overall, strongest frequency via zero crossings (jAudio)	window size: 2048ms hopsize: 1024ms sampling rate: 22050Hz	mean, std. dev.

5.4.4 Further Experiment Details

For all experiments, we equalize the number of songs per subgenre. The three binning methods used in Chapters 3 and 4 are used to preprocess music data, including: B_{ew}, B_{ef}, and B_{rr}. We set the number of bins to be 10 and 15, for B_{ew} and B_{ef}, respectively. We then use a 80% 20% split for training and testing, with $M = 10$. 80% of the testing songs are then used to generate 10 random subsets for each average accuracy calculation. We fix ϕ_{ig} and μ to be n/a. Each m_s percentage is chosen so as not to allow the explosion of frequent fv-sets returned by Apriori. We follow the stipulations for selecting pieces per subgenre, as outlined above, and we provide the number of music pieces per subgenre in each experiment.

5.5 Experiment Results and Discussions

We only present a handful of the subgenre experiments conducted. Those included are chosen with respect to achieving the goals outlined in Section 5.1, and offer some additional observations worth mentioning, such as success in other facets of MIR (e.g., instrument recognition). We include additional experiments on D_{CAL} in Appendix A, as well as those confusion matrices for each experiment shown here. The subgenre experiments on D_{CAL} in Table 5.4 will now be examined first, followed by an analysis of those subgenre experiments on D_{FMA} in Table 5.5.

5.5.1 On D_{CAL}

We can easily see the effectiveness of our approach in Table 5.4(a), and the same trend that is found in the experiments from Chapter 4 (i.e., a more strict ϕ_{pw} threshold produces better classification accuracies). We see that *reggaeton* and *dancehall* are both more distinguishable than *dub* and *ska*, with dub being confused with ska, ska being confused with dancehall, and reggaeton and dancehall being confused with each other; the latter confusion is intuitive since reggaeton can be considered as a predecessor of dancehall. It is intuitive that dub be confused with ska, since the instrumentation and rhythmic syncopations may be similar. However, the rhythmic speed and production styles could be a source of differentiation between them.

In Table 5.4(b) we can see that our approach is effective in differentiating between two types of *gospel* music (i.e., far above a chance percentage of 25%). With this differentiation, we see the ϕ_{pw} threshold eliminating overlapping frequent fv-sets for gospel music, as *traditional gospel* performed with greater accuracy for a stricter threshold. The subgenres most confused are *soul* and *funk*. Soul was especially confused with traditional gospel and funk was especially confused with soul. Since many of these subgenres may have similar musical characteristics (e.g., rhythm, instrumentation, etc.) the results are still quite practical.

Next, in Table 5.4(c), for $\phi_{pw} = 30\%$ we note that *piano solo* pieces are classified correctly 100% of the time, and those pieces with heavy violin instrumentation are often classified correctly. This tells us that our selected approach is well-suited for instrument recognition tasks. The confusions for *piano concerti* are also very intuitive since some confusion is due to other subgenres having orchestral qualities, (e.g., *symphony*), with most of the confusion being caused by the piano solo subgenre, as one would expect.

We see that jazz has effective classification accuracies as well, as the subgenre *smooth jazz* achieves classification accuracies in the 80% range, and the classification on *bebop* approaches 70% accuracy for $\phi_{pw} = 40\%$. *Jazz fusion* is expected to have higher classification accuracies, as elements of rock should make this subgenre distinguishable. However, jazz fusion is confused with *swing*. *Bebop* is not confused with smooth jazz, which we believe to be a musically successful classification, since the performances, and musical complexity of these subgenres varies a substantial amount. Other subgenres had an even distribution of confusion, which is to be expected. With jazz, it is noticed that the strictest pairwise removal threshold does not necessarily yield the highest classification accuracies; some relaxation of this strictness may be necessary for certain subgenres.

We predicted the confusion of the *hip hop* subgenres *classic hip hop* with *electro*, as the production and recording of the music can be somewhat similar with respect to timbre. At the same time, we did expect electro to be quite distinguishable compared to the other subgenres of hip hop, due to its distinct rhythmic qualities. It was also found, however, that *underground hip hop* was a source of confusion for all of the other subgenres. One source of distinguishability in the subgenres of hip hop is the lyrical content. Lyricism is not directly accounted for by our approach, so the fact that we provide useful hip hop classifications using MFCCs is noteworthy. Overall, we continue to see the performance improvement caused by stricter values of ϕ_{pw}.

The most notable results in Table 5.4, after the impressive classification of *reggae*, is the classification of *pop*, as shown in Table 5.4(f). Pop is one of the least suspected genres

to have its subgenres be classified with a high accuracy, given the similarity of its subgenres, and the genre's tendency to cause confusions in regular genre classifications (e.g., see Section 4.4). However, the majority of the subgenres (especially for $\phi_{pw} = 40\%$) are classified successfully. In the case of this pairwise removal threshold value, our intuitions are matched, since *classic pop* pieces labelled in D_{CAL} seems to include a wide variety of artists (ranging from *rnb* to rock, and spanning decades), therefore classic pop is not expected to have been classified successfully, however, it is still classified higher than by chance. The presence of *pop rock* provided confusions for classic pop, and the opposite is also true. However, *dance pop* is also a source of confusion for pop rock, which is not an ideal misclassification. This misclassification may be due to a similarity in production styles as well as timbral similarities, and may be rectified with the inclusion of further features. *Teen pop* includes a slightly more coherent selection of pieces for D_{CAL}, with more emphasis on polished "modern" production styles, so this was expected to provide a higher classification as well.

Table 5.4: Selected results for D^m_{CAL}.

ϕ_{pw}	dub	reggaeton	dancehall	ska	avg
0.3	0.77	0.88	0.86	0.78	0.823
0.4	0.66	0.80	0.72	0.66	0.710
0.5	0.66	0.78	0.75	0.65	0.710
0.6	0.43	0.84	0.63	0.76	0.665
0.7	0.41	0.79	0.65	0.79	0.660

(a) Reggae, F^M_{CAL}, B_{ef}, $m_s = 5\%$, 71 songs per subgenre.

ϕ_{pw}	soul	funk	traditional gospel	contemporary gospel	avg
0.3	0.49	0.49	0.74	0.60	0.580
0.4	0.30	0.53	0.69	0.35	0.468
0.5	0.43	0.43	0.70	0.41	0.493
0.6	0.40	0.39	0.51	0.54	0.460
0.7	0.41	0.43	0.48	0.59	0.478

(b) Rhythm and Blues (RnB), F^M_{CAL}, B_{ef}, $m_s = 7\%$, 54 songs per subgenre.

ϕ_{pw}	piano concerti	symphonic	violin features	piano solo	choral	avg
0.3	0.49	0.53	0.63	1.00	0.31	0.592
0.4	0.61	0.55	0.20	0.95	0.20	0.502
0.5	0.61	0.58	0.31	0.90	0.20	0.520
0.6	0.59	0.30	0.48	0.50	0.49	0.472
0.7	0.59	0.30	0.41	0.89	0.51	0.540

(c) Classical, D_{CAL}, F^M_{CAL}, B_{rr}, $m_s = 6\%$, 53 songs per subgenre.

ϕ_{pw}	smooth jazz	jazz fusion	avant garde jazz	bebop	swing	avg
0.3	0.66	0.36	0.41	0.57	0.58	0.516
0.4	0.85	0.39	0.41	0.67	0.49	0.562
0.5	0.86	0.33	0.45	0.68	0.49	0.562
0.6	0.56	0.24	0.34	0.52	0.58	0.448
0.7	0.48	0.24	0.37	0.43	0.60	0.424

(d) Jazz, F^M_{CAL}, B_{ef}, $m_s = 6\%$, 60 songs per subgenre.

ϕ_{pw}	underground hiphop	southern rap	classic hiphop	electro	avg
0.3	0.57	0.38	0.47	0.83	0.563
0.4	0.38	0.43	0.47	0.82	0.525
0.5	0.38	0.46	0.43	0.83	0.525
0.6	0.40	0.66	0.24	0.81	0.528
0.7	0.33	0.60	0.27	0.89	0.523

(e) Hip Hop, F^M_{CAL}, B_{ef}, $m_s = 6\%$, 57 songs per subgenre.

ϕ_{pw}	new age pop	pop rock	dance pop	classic pop	teen pop	avg
0.3	0.80	0.29	0.70	0.55	0.53	0.574
0.4	0.78	0.31	0.71	0.53	0.67	0.600
0.5	0.76	0.32	0.66	0.55	0.63	0.584
0.6	0.78	0.37	0.46	0.25	0.73	0.518
0.7	0.78	0.41	0.47	0.23	0.75	0.528

(f) Pop, F^M_{CAL}, B_{ef}, $m_s = 5\%$, 67 songs per subgenre.

Note that the trend, of the strictest ϕ_{pw} value yielding the best average classification accuracy, is still present, except for jazz and pop, where the highest accuracies are achieved with a moderate strictness. However, in Tables 5.4(d) and 5.4(f) we observe a subtle tendency. We believe that with too strict of a removal threshold many intersecting frequent

fv-sets may be removed, leaving a genre's characteristic set sparse, so that the fv-set of a testing music piece has less of a chance matching the correct genre's characteristic set. This may be exacerbated by the small amount of training data too. Truncating the characteristic set of every genre (to be the same size as the smallest characteristic set) was attempted; however, this did not yield better results. This demonstrates that with some "hard" classifications, where the subgenres are similar, the ϕ_{pw} threshold may need fine-tuning.

Since we experiment on m_s values from 3% to 7%, we find that m_s values of 3% and 4% achieve lower classification accuracies the majority of the time. Since subgenres can be very similar, lower support frequent fv-sets will occur across multiple subgenres, and more combinations of them may need to be removed by using a higher m_s value, thereby leaving only the subgenre-specific frequent fv-sets. We can see here that for close subgenres, and with a small amount of data, having the lowest possible m_s value does not always provide the best result. Using parameter combinations that are successful for genre classifications, based on the assumption that they will also be successful for subgenre classification tasks, can sometimes cause unsatisfactory classifications. Fine-tuning is needed for the parameter m_s, and it may also be needed for the parameter ϕ_{pw}.

5.5.2 On D_{FMA}

Next, the experiment results shown in Table 5.5 are examined. Recall that the confusion matrices corresponding to Table 5.5 can be found in Appendix A. To begin, we notice that successful classification accuracies on subgenres are found with MFCCs for just one genre (i.e., *metal*), and that all other genres' subgenres have improved accuracies after including further features. The experiments on country demonstrate how stricter ϕ_{pw} values are achievable with a greater number of features. That is, with a small number of features and a strict enough ϕ_{pw} value, some characteristic sets returned no frequent fv-sets when calculating certain except differences. This is why, in Table 5.5 there are some experiments conducted with a very strict ϕ_{pw} value of 20%, and others are only as low as 30%.

By continuing the experiment analysis of *country*, we see an average accuracy of 85.67% for three subgenres. This is what is expected, since these subgenres are quite dissimilar, and the number of subgenres for the classification experiment is small. It may be argued that *bluegrass* and *rockabilly* are related to country, and not subgenres of country. However, as stated in Section 5.4, we are reliant on the labelling of a given dataset, and the parent genres that are defined in it. One peculiarity is that B_{ew} provides very successful predictions for country, unlike the other genres, this does not follow the normal trend where B_{ef} outperforms other binning methods. It is possible that the number of bins for B_{ew} (15), is more suitable for discretizing the data. More experiments must be conducted to determine that this is indeed the case for B_{ew}'s success. We do notice that *rockabilly* is the most indistinct genre, even with a very strict ϕ_{pw} threshold, this is due to the subgenre's similarity with *country and western* which may share similar instrumentation (i.e., more "twangy" electric guitar than acoustic guitar, and the use of a drumkit). *Bluegrass* has distinct acoustic instrumentation from the other subgenres. For example, a drumkit will sometimes be excluded and the use of banjos will provide distinct timbral characteristics. Therefore, it is intuitive that it be classified successfully.

Given that there are six (6) overlapping subgenres for rock, as shown in Table 5.5(b), we do not deem this as an unsuccessful classification since three (3) subgenres are classified at around 50%. Furthermore, *progressive rock* and *new wave* are classified more distinctly, however, *post rock* and *shoegaze* are confused. New wave is a source of many of the confusions, as it is quite diverse with certain pieces having elements belonging to various other subgenres (e.g., elements of *electronic* music may be found in both *industrial* and new wave).

Next, *metal*'s subgenres are classified quite successfully. We expect that *black metal* would provide higher accuracies compared to other metal subgenres, due to its often "lo-fi" production characteristics and distinct vocal stylizations. *Death metal* is similar to both *thrash* and *grindcore*; however, it is mostly confused with *sludge*. Grindcore is confused

more evenly between the other subgenres, with a tendency to be confused with black metal. Sludge is responsible for many of the misclassifications for the subgenres thrash, black metal, and death metal. This is not unreasonable, since characteristics (e.g., timbral, rhythmic, etc.) of sludge, at least for a number of the sludge pieces in D_{FMA}, can be found in other subgenres. Another interesting explanation as to why sludge may be a source of confusions is as follows. D_{FMA} uses subgenre labels provided by the artists of music pieces, and sludge is a subgenre that is not as strongly defined as, say, traditional black metal. Therefore, sludge may be more liberally used as a label for a music piece's subgenre. This is reflected in D_{FMA} at large, as more music pieces do have the sludge label.

Most of the classifications for *punk* are acceptable, with *electropunk* and *hardcore* being more successfully classified. Electropunk is a source of confusion for both *no wave* and *post punk*. We believe this is, again, due to electropunk's characteristics being found in these other subgenres. Since such a strict ϕ_{pw} value is used, we believe these characteristics are found in the fv-sets of the testing pieces, for example, both post punk and electropunk may contain electronic elements. We do note that the high classification accuracy of hardcore is intuitive, as pieces in this subgenre often have a more compressed and "aggressive" music production style.

Electronic is a broad genre with a notorious amount of subgenres. So the subgenres that were selected in the D_{CAL} and D_{FMA} experiments may not overlap. Electronic is classified as well as the other genres. However, we see in Table 5.6, that a higher classification accuracy is found for more subgenres of electronic music in the D_{CAL} dataset. This may be due to the similarity of subgenres chosen in the D_{FMA} dataset, as well as the subset of songs chosen for experiments. For Table 5.4(e), the majority of confusions are caused by either *downtempo*, or *drum and bass*, with *glitch* being confused with *house* and drum and bass. It is quite reasonable that drum and bass and *dubstep* to be confused with each other because of their instrumentation and production styles. Furthermore, it is intuitive that every subgenre is responsible for some misclassifications due to the feature sets not

having a large number of rhythmic-based features. Even a feature like the estimated *beats per minute* (see Section 2.1.2) could potentially improve the distinguishability between electronic subgenres.

Finally, we see practical classification accuracies for folk, as shown in Table 5.5(f), despite the overlap between its subgenres. *Freak folk* is a source of confusion for both *British folk* and *free folk*, and free folk is a source of confusion for freak folk. This is intuitive, as freak folk seeks to expand the boundaries of the folk genre (in a similar way that psychedelic rock does to rock), and as such, may still have similar characteristics found in both British, or free folk.

Table 5.5: Selected results for D^m_{FMA}.

ϕ_{pw}	bluegrass	rockabilly	country and western	avg
0.2	0.91	0.75	0.91	0.857
0.3	0.86	0.60	1.00	0.820
0.4	0.83	0.63	0.92	0.793
0.5	0.93	0.69	0.82	0.813
0.6	0.93	0.60	0.59	0.707
0.7	0.91	0.63	0.58	0.707

(a) Country, F^{L2}_{FMA}, B_{ew}, $m_s = 15\%$, 64 songs per subgenre.

ϕ_{pw}	krautrock	new wave	post rock	shoe-gaze	industrial	prog-ressive	avg
0.2	0.49	0.64	0.31	0.35	0.28	0.57	0.440
0.3	0.45	0.64	0.31	0.34	0.31	0.53	0.430
0.4	0.48	0.64	0.35	0.30	0.34	0.47	0.430
0.5	0.49	0.65	0.35	0.31	0.35	0.48	0.438
0.6	0.39	0.74	0.28	0.28	0.33	0.39	0.402
0.7	0.39	0.68	0.30	0.29	0.35	0.30	0.385

(b) Rock, F^{L2}_{FMA}, B_{ef}, $m_s = 7\%$, 232 songs per subgenre.

ϕ_{pw}	thrash	black metal	death metal	sludge	grindcore	avg
0.2	0.41	0.80	0.35	0.93	0.46	0.590
0.3	0.49	0.82	0.41	0.95	0.56	0.646
0.4	0.49	0.82	0.41	0.95	0.56	0.646
0.5	0.35	0.93	0.39	0.95	0.58	0.640
0.6	0.34	0.82	0.32	0.82	0.46	0.552
0.7	0.35	0.82	0.30	0.82	0.45	0.548

(c) Metal, F^{L1}_{FMA}, B_{ef}, $m_s = 12\%$, 82 songs per subgenre.

ϕ_{pw}	hardcore	post punk	electro punk	no wave	avg
0.2	0.81	0.50	0.85	0.63	0.698
0.3	0.79	0.43	0.88	0.54	0.660
0.4	0.75	0.43	0.88	0.53	0.648
0.5	0.80	0.43	0.89	0.55	0.668
0.6	0.61	0.41	0.91	0.51	0.610
0.7	0.58	0.39	0.89	0.49	0.588

(d) Punk, F^{L2}_{FMA}, B_{ef}, $m_s = 8\%$, 211 songs per subgenre.

ϕ_{pw}	house	glitch	drum and bass	downtempo	dubstep	avg
0.3	0.47	0.41	0.47	0.60	0.61	0.512
0.4	0.43	0.37	0.45	0.57	0.54	0.472
0.5	0.44	0.41	0.46	0.58	0.53	0.484
0.6	0.32	0.34	0.44	0.34	0.44	0.376
0.7	0.33	0.30	0.33	0.38	0.42	0.352

(e) Electronic, F^{J}_{FMA}, B_{ef}, $m_s = 7\%$, 189 songs per subgenre.

ϕ_{pw}	British folk	free folk	freak folk	avg
0.3	0.71	0.29	0.50	0.500
0.4	0.67	0.53	0.55	0.583
0.5	0.69	0.59	0.54	0.607
0.6	0.63	0.36	0.59	0.527
0.7	0.60	0.51	0.47	0.527

(f) Folk, F^{J}_{FMA}, B_{ef}, $m_s = 8\%$, 89 songs per subgenre.

Table 5.6: Electronic, D_{CAL}, F^{M}_{CAL}, B_{ew}, $m_s = 6\%$, 56 songs per subgenre.

ϕ_{pw}	drum and bass	trance	triphop	techno	industrial	ambient	avg
0.3	0.17	0.39	0.43	0.66	0.72	0.80	0.528

In the experiment results given above, we continue to observe that a lower ϕ_{pw} value yields higher classification accuracies. This is a trend we found in Chapter 4, and we further see it in Table 5.5. However, ϕ_{pw} must still be tuned to achieve optimal classifications as the strictest threshold value does not always yield the best classification accuracies. One property of the ϕ_{pw} threshold is that with a greater number of features a very strict threshold can be applied. This is indicated by $\phi_{pw} = 20\%$ in Table 5.5. The ability to use this amount

of strictness may also be due to the longer frequent fv-sets returned by Apriori. For the experiments on D_{FMA}, we see that a low m_s value is somewhat less responsible for inaccurate classifications when there are more music pieces per subgenre, especially when compared to D_{CAL}.

With the similarity of some subgenres we found that most confusions are intuitive, and with closer inspection, those confusions that seem unintuitive have reasonable explanations for their occurrence. In general, there are enough pieces in D_{FMA} to offer experiments on subgenre classification with more music pieces per subgenre than with D_{CAL}. Furthermore, D_{FMA} provides an "in-the-wild" scenario, where features are derived from raw audio files without audio quality taken into account, and with music piece labelling done by each pieces' respective artist. One cannot expect all pieces in a large music database to be of equal length, sampling rate, audio quality, etc. For these reasons, we believe the classifications for D_{FMA} may be useful for music listeners.

General remarks

With the experiment results given above, many observations can be made regarding various parameter combinations, subgenres, and the average effectiveness of the selected approach across the two selected subgenre datasets (given the experiment design proposed). We may begin with a discussion on the observed properties of each binning method. The binning method B_{ef} is able to withstand very strict ϕ_{pw} percentages, whereas B_{ew} and B_{rr} sometimes return empty sets after the pairwise comparison for the same strictness threshold. This is due to the same number of values being present in every bin with B_{ef}. Having the same number of values in each bin avoids the possibility that a certain set of bins will contain most of the values, and will prevent bins from having little to no values. However, this possibility still remains for B_{ew} and B_{rr}. So after training, plenty of pairwise removal can be done for both of them. The number of bins is not further explored, this is left to future work. The number of bins and the binning types become sensitive for subgenre

classification. With a smaller number of bins it is possible that less diverse frequent fv-sets will be returned by Apriori, and may need a less strict pairwise comparison threshold to produce successful results. Further investigation of this is needed. In general, it is found that B_{ef} still produced practical prediction accuracies for most of the experiments in Tables 5.4 and 5.5. However, the specific binning method for a subgenre classification task may need more consideration than regular genre classification tasks, and some experimentation may be required to achieve optimal performance.

As found in Section 5.5.1, choosing the lowest possible m_s value for subgenre classification tasks is found to be undesirable in some situations. One situation is when the reduced size of each subgenre's characteristic set becomes important for the computational complexity. Another situation is when frequently occuring characteristics for a subgenre are found, given a low m_s threshold, that are not removed after the calculation of except differences. In the latter situation, a higher m_s value is beneficial.

The quality of the data, although not a parameter that is controllable, may have made an impact on the classification accuracies, and even though we experimented on various feature sets for D_{FMA}, we found that the MFCCs for D_{CAL}, by comparison, are still quite effective at producing practical subgenre classification accuracies. This may be due to the subgenre labelling method (artist versus expert consensus), or the fact that audio quality is less accounted for with D_{FMA} (for more detail, see Section 2.6).

The limitations of the selected pairwise dichotomy-like approach are the same as those found in Chapter 4, as the problem of subgenre classification readily demonstrates. For example, the complexity of the pairwise removal tasks becomes burdened, given a small enough m_s and a larger M. The trade-off, as discussed in Chapter 4, can be made between complexity and classification accuracy. Another limitation, described above, is that after a strict pairwise removal is used to create the except differences a genre's characteristic set may become empty, although this depends on the m_s parameter and the binning method used as well.

5.6 Summary

Our clear experiment design provides a reasonable set of experiments to analyze our approach. The selected pairwise dichotomy-like approach determines the distinguishable subgenres with high accuracies. Furthermore, most confusions occurred with subgenres that had many musical characteristics in common with other subgenres. Lastly, we have determined that various parameter combinations that are assumed to be successful in Chapter 4 for the genre classification problem, may need more attention and fine-tuning for the subgenre classification problem. One example of this is that one cannot assume that the lowest possible m_s value will automatically yield the highest classification accuracies.

In this chapter, we selected the approach found in Chapter 4 and have shown that our approach can classify subgenres effectively (greater than by random choice). Our experiment results are easily reproducible, and can be used to compare our approach with others. In Section 2.7, we saw that there is a void of approaches attempting to classify subgenres at such a detailed level. We believe this is the first work of its kind in MIR.

We have now seen that, not only is the pairwise dichotomy-like approach able to effectively store and compare the characteristics of genres, it is also able to do so for subgenres. We have also observed that our approach can recognize various instruments, and may even be able to determine various styles of music production, but we need further experiments to support this. This will be left to our future work.

Chapter 6

Conclusion

There are several core tasks in MIR and one of the most fundamental tasks is music genre classification. Music genre classification is the process of automatically categorizing music pieces, first, by training a classification approach with music pieces given their genre label, and second, by classifying new incoming music pieces (referred to as testing music pieces), so that they are correctly labelled by the approach. Creating a genre classification approach that is both effective and useful is quite difficult, as music genres are notoriously ambiguous.

Nevertheless, the genre classification problem is exceedingly important, especially with the current amounts of digitized music that can be found through music streaming services, and through other music libraries that may be accessed online. Ultimately, it is beneficial to have an autonomous approach to curate these large volumes of music so that a costly human classification effort is avoided. As well, a successful genre classification approach may be useful for a variety of further applications. One such example is recommending new music to a listener that they might enjoy (i.e., playlist recommendation). Other closely related tasks are: archive querying, similar artist search, instrument recognition, mood or emotion classification, etc.

In Chapter 1, we describe what music genre classification is, and detail the difficulties and benefits of performing this task. In Chapter 2, we provide a firm background, so that the reader contextualizes our approaches within the field of MIR. We describe the concepts in sections, based on the order by which they appear in our approaches, that is: extracting content-based features from audio, binning these features into discrete data objects, and

performing association analysis to characterize genres and classify new music pieces. Popular off-the-shelf single-classifier approaches, and other fundamental ensemble techniques are also described. The datasets used in other chapters and their high level-descriptions are detailed, and lastly, we provide an in-depth review of the works related to our approach, especially those works concerned with association analysis, subgenre classification, and ensemble-based music genre classification.

After this, we propose two novel, content-based music genre classification approaches which both utilize association analysis. The first approach, presented in Chapter 3, provides a fundamental structure for characterizing genres using association analysis. To our knowledge, there has not been an approach that uses association analysis to handle the problem of genre classification as it is proposed in Chapter 3. The use of association analysis in this way is not entirely new [6], however, we are able to both characterize each genre in a novel way, and provide a novel method of classifying incoming music pieces. Furthermore, each genre's parameters can be fine-tuned, so we also contribute a genre-dependent classification approach. We validate this approach using empirical experiments on several benchmark datasets, and notice that practical accuracies are achieved.

The approach in Chapter 4 seeks to further improve the original classification performances by utilizing a dichotomy-like pairwise ensemble technique. The intuition for providing this approach is to better characterize, and allow for optimal comparisons of genres. To our knowledge, an ensemble technique has not yet been applied to association analysis as it is proposed in Chapter 4, especially when utilized for the task of music genre classification. We are also able to better characterize genres, and classify them with successful accuracies.

In addition to our attempts at handling the genre classification task, we curiously examine the performance of the pairwise dichotomy-like approach, presented in Chapter 4, on a set of subgenre problems in Chapter 5. Since there is limited amounts of MIR research focusing on the subgenre task at large, we propose a clear experiment design, so that other

genre classification tasks can be easily tested and compared to our selected approach. Although the subgenre classification task is markedly more difficult than the problem of genre classification, we are still able to provide practical classification accuracies. Furthermore, we observe that the fine-tuning of various parameters needs to occur in order to maintain high classification accuracies on various subgenre experiments. This fine-tuning contrasts the trends we saw in Chapter 3 and Chapter 4. If there are definitive rules made for certain parameters during genre classification, they may be bent (or broken) during subgenre classification.

6.1 Limitations of Our Approaches

The contributions made by this thesis have been enumerated, however, there are still certain limitations that need to be addressed, as all classification methods have pros and cons. These limitations fall into two categories: 1) those imposed by the proposed approaches, and 2) those imposed by the experiment designs which are used to validate these approaches.

The first limitation of both of the approaches is that genres' characteristic sets are sometimes left empty after Inter-Genre Removal or Pairwise Removal is applied. This occurs for less distinguishable genres, and for specific binning methods (i.e., equal width binning) when the minimum support is set high and the removal threshold is set to be more strict. All of this relates to characteristic sets with a smaller number of frequent fv-sets. The less the amount of frequent fv-sets means that a fewer number of matches can be made to an incoming testing piece, and therefore, the potential for missclassifications becomes greater. Furthermore, classifications cannot be made with empty characteristic sets. This is why we saw, at least in the case of genre (non-subgenre) classification, that a low minimum support performed best – there were simply more complete characteristic sets per genre. Related to this is our approaches' step to create equal subsets of data for each genre. Without this step, the size of the genres' characteristic sets can be greatly skewed, and further empty

characteristic sets can be found for those genres with less music pieces.

Another limitation of the approaches defined in Chapters 3 and 4 is the space and time complexity that necessarily occurs when utilizing association analysis. Even though maximal frequent fv-sets are returned, there is still the pervasive "explosion" of frequent fv-sets. This limits how low the minimum support can be set, and how long the inter-genre and pairwise removal steps take. With a basic genre classification task, lower minimum support values are found to increase the classification accuracy, but there is a tendency to have an exceedingly large search space because of this.

The experiment design for the (sub)genre classification tasks impose certain limitations that will now be discussed. Firstly, we analyze our experiments on only one dataset that approaches a moderate size (it certainly is not large when compared to the Million Song Dataset [10], for instance). It is necessary to evaluate our results on more massive datasets, so that we can further determine if the approach is both scalable and effective. Another experiment limitation is that we are reliant on the labelling of the pieces in each dataset to give a proper indication of the genre. This limitation is especially found in Chapters 3 and 5, with the Greek Audio Dataset, and the Free Music Archive dataset, where labelling is done by non-musicologists, and the artist of every music piece, respectively. We are further limited by the quality of the audio and genre-representative music pieces found in these datasets, and this is primarily the case with the Free Music Archive.

There is one important shortcoming of the subgenre experiment design that will now be addressed. The stipulations for defining each music subgenre with respect to the chosen datasets in Chapter 5 does not always allow for necessary musical ambiguities. In any music library there are certain (sub)genre ambiguities, that is, a piece may belong to many (sub)genres. This resulted in a limited usage of the Free Music Archive when many songs could have been incorporated into the experiments. If these musically ambiguous pieces were included, then a multi-label classification approach could be used. We believe our proposed approaches can also be restructured to handle this task, as we do store a ranking

vector, comprised of scores for each (sub)genre.

6.2 Future Work

As mentioned above, one can immediately notice a possible extension of our proposed approaches. The ranking vector that we store per testing music piece is comprised of scores for each (sub)genre, therefore assigning the next best (sub)genre would be a trivial task. For instance, we can determine the mixture of genres that a music piece belongs to given the highest scores in the ranking vector. This demonstrates that our approaches can be readily applied to the multi-label music genre classification problem.

A merit that has been discussed in the previous chapters is the ability to store the characteristics of the (sub)genres that are distinguishable. This would provide a way of handling other tasks, such as feature reduction or selection. Verifying the consistency of tag annotations is another task we wish to explore since the characteristics of each genre can easily be used to provide some confidence for a music piece's designated tags. We have also observed in Chapter 5 that the pairwise dichotomy-like approach can recognize various instruments with a high prediction accuracy, we will provide experiments on this in the future. The extraction and modelling of a group of music piece's characteristics further lends itself well to mood classification tasks.

Various parameters must be investigated further. Since the number of bins is fixed for two of the three binning methods used, we plan on conducting experiments that demonstrate the correlation between classification accuracy, and the number of bins. Some parameters that are less frequently studied can be found in Chapter 3. Examples are: the strictness of Inter-Genre Removal (i.e., ϕ_{ig}), and the ranking criterion $RC_{\alpha\beta}$. We have in no way provided exhaustive experiments for these parameters, as α, β, and ϕ_{ig} are fixed. It is possible that the approach proposed in Chapter 3 could surpass the approach proposed in Chapter 4, given the fine-tuning of these parameters. In Chapter 4, we mention the statistics that are gathered across a genre's M training iterations (i.e., the average support or number of occurrences of

an fv-set). Further work is needed to verify that the linear combination of frequent fv-set statistics (via the M training iterations) used during the scoring of a new music pieces does not improve the classification performance of the pairwise dichotomy-like approach, when compared to a naive scoring method.

Various subgenre experiments in Chapter 5 had subgenres that contributed to the overall confusions of an experiment. For example, *new wave* was a source of confusion for most subgenres in the *rock* experiment and *sludge* was a source of confusion for most subgenres in the *metal* experiment, as in Tables 5.5 and A.2. It would be interesting to remove these subgenres and determine if the classification accuracies improve.

In the experiments discussed, we do not use any datasets with a vast number of music pieces. It would be desirable to analyze the scalability and the general effects that the number of pieces plays in genre classification. The largest number of music pieces (per genre) presented in the experiments were those on D_{LMD} in Chapter 4, where successful classification accuracies were achieved. The number of pieces per genre has some effect in Chapters 3 and 4. However, more experiments need to be done to validate the fact that a greater number of songs will improve the classification accuracies. Determining if our approaches are scalable to extremely large repositories is of interest, as we have found practical classification accuracies on a small to moderate number of music pieces in Chapters 3 and 4.

In the future, we may characterize genres using multimodal feature sets, as we only use content-based audio features. We believe that with additional, diverse features extracted from each music piece (e.g., content-based, visual, lyrical, etc.), the classification accuracies could be even more useful for music data curation tasks. These will be our very next tasks in the future.

Bibliography

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *SIGMOD Record*, 22(2):207–216, 1993.

[2] Ritesh Ajoodha, Richard Klein, and Benjamin Rosman. Single-labelled music genre classification using content-based features. In *Proceedings Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech)*, pages 66–71, 2015.

[3] Pedro Girão Antunes, David Martins de Matos, Ricardo Ribeiro, and Isabel Trancoso. Automatic fado music classification. *Computing Research Repository*, abs/1406.4447, 2014.

[4] Hasitha Bimsara Ariyaratne, Dengsheng Zhang, and Guojun Lu. A class centric feature and classifier ensemble selection approach for music genre classification. In *Proceedings Structural, Syntactic, and Statistical Pattern Recognition*, pages 666–674. Springer Berlin Heidelberg, 2012.

[5] Tom Arjannikov, Chris Sanden, and John Z. Zhang. Verifying music tag annotation via association analysis. In *Proceedings International Society for Music Information Retrieval*, page 195–200, 2013.

[6] Tom Arjannikov and John Zhang. An association-based approach to genre classification in music. In *Proceedings International Society for Music Information Retrieval*, pages 95–100, 2014.

[7] Tom Arjannikov and John Zhang. An empirical study on structured dichotomies in music genre classification. In *Proceedings International Conference on Machine Learning and Applications*, pages 493–496, 2015.

[8] Jayme Garcia Arnal Barbedo and Amauri Lopes. Automatic genre classification of musical signals. *EURASIP Journal on Advances in Signal Processing*, (1), 2007.

[9] Kirell Benzi, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Fma: A dataset for music analysis. In *Proceedings International Society for Music Information Retrieval*, pages 316–323, 2017.

[10] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings International Society for Music Information Retrieval*, pages 591–596, 2011.

[11] Christian Borgelt. Frequent item set mining. *Wiley Interdisciplinary Reviews: Data*

Mining and Knowledge Discovery, 2(6):437–456.

[12] Judith C. Brown. Determination of the meter of musical scores by autocorrelation. *The Journal of the Acoustical Society of America*, 94(4):1953–1957, 1993.

[13] Donald Byrd and Tim Crawford. Problems of music information retrieval in the real world. *Information Processing and Management: An International Journal*, 38(2):249–272, 2002.

[14] Kang Cai, Deshun Yang, and Xiaoou Chen. Audio classification tasks: Mirex 2015 submissions. In *Proceedings Music Information Retrieval Evaluation eXchange (MIREX) in the 16th International Society for Musical Information Retrieval Conference (ISMIR)*, 2015.

[15] Michael Chanan. *Repeated takes: a short history of recording and its effects on music*. Verso, London, England; New York, NY, USA, 1995.

[16] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[17] Dhanith Chathuranga and Lakshman Jayaratne. Musical genre classification using ensemble of classifiers. In *Proceedings International Conference on Computational Intelligence, Modelling and Simulation*, pages 237–242, 2012.

[18] Austin C. Chen. Automatic classification of electronic music and speech/music audio content. Master's thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA, 2014.

[19] Darrell Conklin and Christina Anagnostopoulou. Comparative pattern analysis of cretan folk songs. *Journal of New Music Research*, 40:119–125, 2011.

[20] Yandre Costa, Luiz Oliveira, Alessandro Koerich, and Fabien Gouyon. Music genre recognition using gabor filters and lpq texture descriptors. In *Iberoamerican Congress on Pattern Recognition*, 2013.

[21] Hannes Datta, George Knox, and Bart J. Bronnenberg. Changing their tune: How consumers' adoption of online streaming affects music consumption and discovery. *Marketing Science*, 37(1):5–21, 2018.

[22] Paulo Ricardo Lisboa de Almeida, Eunelson José da Silva Júnior, Tatiana Montes Celinski, Alceu de Souza Britto, Luis Eduardo Soares de Oliveira, and Alessandro Lameiras Koerich. Music genre classification using dynamic selection of ensemble of classifiers. In *Proceedings IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2700–2705, 2012.

[23] J. Martins de Sousa, E. Torres Pereira, and L. Ribeiro Veloso. A robust music genre classification approach for global and regional music datasets evaluation. In *Proceedings IEEE International Conference on Digital Signal Processing*, pages 109–113, 2016.

[24] Mehdi Banitalebi Dehkordi and Amin Banitalebi-Dehkordi. Music genre classification using spectral analysis and sparse representation of the signals. *Signal Processing Systems*, 74:273–280, 2014.

[25] Simon Dixon, Fabien Gouyon, and Gerhard Widmer. Towards characterisation of music via rhythmic patterns. In *Proceedings International Conference on Music Information Retrieval*, page 509–517, 2004.

[26] Ke-Lin Du and M. N. S. Swamy. *Recurrent Neural Networks*, pages 337–353. Springer, London, England, 2014.

[27] Lin Feng, Sheng-lan Liu, and Jianing Yao. Music genre classification with paralleling recurrent convolutional neural network. *Computing Research Repository*, abs/1712.08370, 2017.

[28] Harvey Fletcher and Wilden A. Munson. Loudness, its definition, measurement and calculation. *The Bell System Technical Journal*, 12(4):377–430, 1933.

[29] Juliano Henreique Foleiss and Tiago Fernandes Tavares. A spectral bandwise feature-based system for the mirex 2016 train/test task. In *Proceedings Music Information Retrieval Evaluation eXchange (MIREX) in the 17th International Society for Musical Information Retrieval Conference (ISMIR)*, 2016.

[30] Eibe Frank, Mark. A Hall, and Ian H Witten. *The WEKA Workbench. Online Appendix for Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., 4 edition, 2016.

[31] Eibe Frank and Stefan Kramer. Ensemble of nested dichotomies for multi-class problems. In *Proceedings International Conference on Machine Learning*, pages 305–312, 2004.

[32] Ichiro Fujinaga. *Adaptive Optical Music Recognition*. PhD dissertation, McGill University, Montreal, Quebec, Canada, 1996.

[33] Johannes Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

[34] Miguel Alexandre Gaspar Lopes. Automatic classification of latin music: Some experiments on musical genre classification. Master's dissertation, University of Porto, Porto, Portugal, 2009.

[35] Bart Goethals and Mohammed J. Zaki. Advances in frequent itemset mining implementations: Report on fimi'03. *SIGKDD Exploration Newsletter*, 6(1):109–117, 2004.

[36] Tim Haifley. Linear logistic regression: An introduction. In *IEEE Integrated Reliability Workshop Final Report*, 2002.

[37] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, Waltham, MA, USA, 3 edition, 2012.

[38] G. E. Hinton. Deep belief networks. *Scholarpedia*, 4(5):5947, 2009. Revision 91189.

[39] Piotr Hoffmann, Andrzej Kaczmarek, Paweł Spaleniak, and Bożena Kostek. Music recommendation system. *Journal of Telecommunications and Information Technology*, 2014:59–69, 2014.

[40] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. In *Proceedings IEEE International Conference on Multimedia and Expo*, volume 1, pages 113–116, 2002.

[41] Carlos N. Silla Jr., Celso A. A. Kaestner, and Alessandro L. Koerich. Automatic music genre classification using ensemble of classifiers. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 1697–1692, 2007.

[42] Michael Kassler. Towards musical information retrieval. *Perspectives of New Music*, 4(2):59–67, 1966.

[43] Priit Kirss. Audio based genre classification of electronic music. Master's thesis, University of Jyväskylä, Jyväskylä, Finland, 2007.

[44] Merve Ayyüce Kizrak and Bülent Bolat. A musical information retrieval system for classical turkish music makams. *Simulation*, 93(9):749–757, 2017.

[45] Dave M. Lane. Online statistics education: An interactive multimedia course of study. Available at `http://onlinestatbook.com/`, 2017.

[46] Hugo Larochelle and Yoshua Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings International Conference on Machine Learning*, pages 536–543, 2008.

[47] Edith Law, Kris West, Michael Mandel, Mert Bay, and J. Stephen Downie. Evaluation of algorithms using games: the case of music annotation. In *Proceedings International Conference on Music Information Retrieval*, pages 387–392, 2009.

[48] Jin Ha Lee and J. Stephen Downie. Survey of music information needs, uses, and seeking behaviours: Preliminary findings. In *Proceedings International Conference on Music Information Retrieval*, 2004.

[49] Jongpil Lee, Jiyoung Park, Juhan Nam, Chanju Kim, Adrian Kim, Jangyeon Park, and Jung-Woo Ha. Cross-cultural transfer learning using sample-level deep convolutional neural networks. In *Proceedings Music Information Retrieval Evaluation eXchange (MIREX) in the 18th International Society for Musical Information Retrieval Conference (ISMIR)*, 2017.

[50] Tao Li, Mitsunori Ogihara, and Qi Li. A comparative study on content-based music genre classification. In *Proceedings Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 282–289, 2003.

[51] Tao Li, Mitsunori Ogihara, and George Tzanetakis. *Music Data Mining*. CRC Press Wadsworth, Belmont, CA, USA, 2011.

[52] Thomas Lidy. Spectral convolutional neural network for music classification. In *Proceedings Music Information Retrieval Evaluation eXchange (MIREX) in the 16th International Society for Musical Information Retrieval Conference (ISMIR)*, 2015.

[53] Thomas Lidy and Alexander Schindler. Parallel convolutional neural networks for music genre and mood classification. In *Proceedings Music Information Retrieval Evaluation eXchange (MIREX) in the 17th International Society for Musical Information Retrieval Conference (ISMIR)*, 2016.

[54] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *Proceedings International Conference on Knowledge Discovery and Data Mining*, pages 80–86, 1998.

[55] Richard Lyons. *Understanding Digital Signal Processing*. Prentice Hall, Boston, MA, USA, 3 edition, 2011.

[56] Dimos Makris, Katia Lida Kermanidis, and Ioannis Karydis. The greek audio dataset. In *Proceedings Artificial Intelligence Applications and Innovations: AIAI 2014 Workshops*, pages 165–173, 2014.

[57] Rudolf Mayer, Robert Neumayer, and Andreas Rauber. Combination of audio and lyrics features for genre classification in digital audio collections. In *Proceedings ACM International Conference on Multimedia, with co-located Symposium and Workshops*, pages 159–168, 2008.

[58] Stephen McAdams, James W. Beauchamp, and Suzanna Meneguzzi. Discrimination of musical instrument sounds resynthesized with spectrotemporal parameters. *The Journal of the Acoustical Society of America*, 105(2):882–897, 1999.

[59] Daniel McEnnis, Ichiro Fujinaga, Cory McKay, and Philippe Depalle. Jaudio: A feature extraction library. In *Proceedings International Society for Music Information Retrieval*, pages 600–603, 2005.

[60] Brian McFee, Colin Raffel, Dawen Liang, Daniel Patrick Whittlesey Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. Librosa : Audio and music signal analysis in python. In *Proceedings Python in Science Conference*, pages 18–25, 2015.

[61] Cory McKay. *Automatic Music Classification with jMIR*. PhD dissertation, McGill University, Montreal, Quebec, Canada, 2010.

[62] Cory McKay and Ichiro Fujinaga. Musical genre classification: Is it worth pursuing and how can it be improved? In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 101–106, 2006.

[63] Kembrew McLeod. Genres, subgenres, sub-subgenres and more: Musical and social differentiation within electronic/dance music communities. *Journal of Popular Music Studies*, 13(1):59–75.

[64] Fady Medhat, David Chesmore, and John Robinson. Automatic classification of music genre using masked conditional neural networks. In *Proceedings IEEE International Conference on Data Mining*, pages 979–984, 2017.

[65] Fady Medhat, David Chesmore, and John Robinson. Masked conditional neural networks for environmental sound classification. *Computing Research Repository*, abs/1805.10004, 2018.

[66] Anders Meng, Peter Ahrendt, Jan Larsen, and Lars Kai Hansen. Temporal feature integration for music genre classification. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(5):1654–1664, 2007.

[67] David Moffat, David Ronan, and Joshua Reiss. An evaluation of audio feature extraction toolboxes. In *Proceedings International Conference on Digital Audio Effects (DAFx-15)*, 2015.

[68] D. G. J. Mulder. Automatic classification of heavy metal music. Master's thesis, University van Amsterdam, Amsterdam, Netherlands, 2014.

[69] Achmad Benny Mutiara, Rina Refianti, and N.R.A. Mukarromah. Musical genre classification using support vector machines and audio features. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 14(3):1024–1034, 2016.

[70] Kerstin Neubarth, Izaro Goienetxea, Colin Johnson, and Darrell Conklin. Association mining of folk music genres and toponyms. In *Proceedings International Society for Music Information Retrieval*, pages 7–12, 2012.

[71] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2018. Available at `http://neuralnetworksanddeeplearning.com/`.

[72] Aaron M. Oirere, Ganesh B. Janvale, and Ratnadeep R. Deshmukh. Automatic speech recognition and verification using lpc, mfcc and svm. *Proceedings International Journal of Computer Applications*, 127(8):47–52, 2015.

[73] Douglas O'Shaughnessy. *Speech Communication: Human and Machine*. Addison-Wesley, Boston, Massachusetts, USA, 1987.

[74] Haukur Pálmason, Björn Þor Jónsson, Markus Schedl, and Peter Knees. Music genre classification revisited: An in-depth examination guided by music experts. In *Proceedings International Symposium on Computer Music Multidisciplinary Research*, pages 45–56, 2017.

[75] Carlos Pérez-Sancho, David Rizo, José M. Iñesta, Pedro J. Ponce de León, Stefan Kersten, and Rafael Ramirez. Genre classification of music by tonal harmony. *Intelligent Data Analysis - Machine Learning and Music*, 14(5):533–545, 2010.

[76] Rene Josiah M. Quinto, Rowel O. Atienza, and Nestor Michael C. Tiglao. Jazz music sub-genre classification using deep learning. In *Proceedings IEEE Region 10 Conference*, pages 3111–3116, 2017.

[77] Louis Rompré, Ismaïl Biskri, and Jean Meunier. Using association rules mining for retrieving genre-specific music files. In *Proceedings International Flairs Conference*, pages 706–711, 2017.

[78] Aldona Rosner and Bozena Kostek. Automatic music genre classification based on musical instrument track separation. *Journal of Intelligent Information Systems*, 50(2):363–384, 2018.

[79] Bertrand Scherrer. Gaussian mixture model classifiers, 2007. Available at `http://www.medialab.bme.hu/medialabAdmin/uploads/VITMM225/GMMScherrer07.pdf`.

[80] Jay Schulkin and Greta B. Raglan. The evolution of music and human social capability. *Frontiers in Neuroscience*, 8(292):1–13, 2014.

[81] Bjoern Schuller, Alexander Lehmann, Felix Weninnger, Florian Eyben, and Gerhard Rigoll. Blind enhancement of the rhythmic and harmonic sections by nmf: Does it help? In *Proceedings German Annual Conference on Acoustics*, pages 361–364, 2009.

[82] Man-Kwan Shan, Fang-Fei Kuo, and Mao-Fu Chen. Music style mining and classification by melody. In *Proceedings International Conference on Multimedia and Expo*, volume 1, pages 97–100, 2002.

[83] Carlos N. Silla Jr., Alessandro L. Koerich, and Celso A. A. Kaestner. The latin music database. In *Proceedings International Conference on Music Information Retrieval*, pages 451–456, 2008.

[84] Julius O. Smith. *Mathematics of the discrete Fourier transform (DFT): with audio applications*. BookSurge Publishing, Seattle, WA, USA, 2 edition, 2007.

[85] Lama Soboh, Islam Elkabani, and Ziad Osman. Arabic cultural style based music classification. In *International Conference on New Trends in Computing Sciences*, pages 6–11, 2017.

[86] Mettu Srinivas, Debaditya Roy, and Krishna Mohan Chalavadi. Music genre classification using on-line dictionary learning. In *Proceedings International Joint Conference on Neural Networks (IJCNN)*, pages 1937–1941, 2014.

[87] Bob L. Sturm. Classification accuracy is not enough: On the evaluation of music genre recognition systems. *Journal of Intelligent Information Systems*, 41(5):371–406, 2013.

[88] Bob L. Sturm. The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. *Computing Research Repository*, abs/1306.1461, 2013.

[89] Bob L. Sturm. A simple method to determine if a music information retieval system is a "horse". *IEEE Transactions on Multimedia*, 16(6):1636–1644, 2014.

[90] Bob L. Sturm. Faults in the latin music database and with its use. In *Extended Ab-*

stracts for the Late-Breaking Demo Session of Proceedings International Society for Music Information Retrieval, 2015.

[91] Derek Tingle, Youngmoo E. Kim, and Douglas Turnbull. Exploring automatic music annotation with acoustically-objective tags. In *Proceedings International Conference on Multimedia Information Retrieval*, pages 55–62, 2010.

[92] Valeri Tsatsishvili. Automatic subgenre classification of heavy metal music. Master's thesis, University of Jyväskylä, Jyväskylä, Finland, 2011.

[93] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):467–476, 2008.

[94] George Tzanetakis and Perry Cook. Marsyas: A framework for audio analysis. *Organized Sound*, 4(3):169–175, 1999.

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

[96] George Tzanetakis, Andrey Ermolinskyi, and Perry R. Cook. Pitch histograms in audio and symbolic music information retrieval. *Journal of New Music Research*, 32(2), 2002.

[97] Huanmin Xu, Lv Jiancheng, and Bijue Jia. Convolutional neural networks system for the mirex 2017 audio classification (train/test) tasks. In *Proceedings Music Information Retrieval Evaluation eXchange (MIREX) in the 18th International Society for Musical Information Retrieval Conference (ISMIR)*, 2017.

[98] Xiaohong Yang, Qingcai Chen, Shusen Zhou, and Xiaolong Wang. Deep belief networks for automatic music genre classification. In *Proceedings Annual Conference of the International Speech Communication Association*, pages 2433–2436, 2011.

Appendix A

Confusion Matrices and Additional Subgenre Experiments from Chapter 5

Table A.1: Corresponding confusion matrices for Table 5.4, $M = 10$.

	dub	reggaeton	dancehall	ska
dub	**0.77**	0.06	0.00	0.17
reggaeton	0.00	**0.88**	0.12	0.00
dancehall	0.02	0.12	**0.86**	0.00
ska	0.04	0.05	0.13	**0.78**

(a) Reggae, F^{M}_{CAL}, B_{ef}, $m_s = 5\%$, 71 songs per subgenre, $\phi_{pw} = 0.3$.

	soul	funk	traditional gospel	contemporary gospel
soul	**0.49**	0.10	0.32	0.09
funk	0.27	**0.49**	0.18	0.06
t. gospel	0.09	0.13	**0.74**	0.04
c. gospel	0.03	0.28	0.09	**0.60**

(b) RnB, F^{M}_{CAL}, B_{ef}, $m_s = 7\%$, 54 songs per subgenre, $\phi_{pw} = 0.3$.

	piano concerti	symphonic	violin features	piano solo	choral
p. concerti	**0.49**	0.15	0.00	0.29	0.07
symphonic	0.11	**0.53**	0.16	0.15	0.05
v. features	0.19	0.08	**0.63**	0.10	0.00
p. solo	0.00	0.00	0.00	**1.00**	0.00
choral	0.21	0.09	0.21	0.18	**0.31**

(c) Classical, D_{CAL}, F^{M}_{CAL}, B_{rr}, $m_s = 6\%$, 53 songs per subgenre, $\phi_{pw} = 0.3$.

	smooth jazz	jazz fusion	avant-garde jazz	bebop	swing
s. jazz	**0.85**	0.15	0.00	0.00	0.00
j. fusion	0.14	**0.39**	0.17	0.05	0.25
a.g. jazz	0.29	0.18	**0.41**	0.04	0.08
bebop	0.00	0.15	0.13	**0.67**	0.05
swing	0.17	0.13	0.16	0.05	**0.49**

(d) Jazz, F^{M}_{CAL}, B_{ef}, $m_s = 6\%$, 60 songs per subgenre, $\phi_{pw} = 0.4$.

	underground hiphop	southern rap	classic hiphop	electro
u. hip hop	**0.57**	0.10	0.16	0.17
c. hip hop	0.36	**0.38**	0.09	0.17
s. rap	0.37	0.00	**0.47**	0.16
electro	0.15	0.02	0.00	**0.83**

(e) Hip Hop, F^{M}_{CAL}, B_{ef}, $m_s = 6\%$, 57 songs per subgenre, $\phi_{pw} = 0.3$.

	new age pop	pop rock	dance pop	classic pop	teen pop
n.a. pop	**0.78**	0.00	0.07	0.08	0.07
p. rock	0.00	**0.31**	0.48	0.16	0.05
d. pop	0.00	0.10	**0.71**	0.09	0.10
c. pop	0.08	0.25	0.06	**0.53**	0.08
t. pop	0.00	0.00	0.23	0.10	**0.67**

(f) Pop, F^{M}_{CAL}, B_{ef}, $m_s = 5\%$, 67 songs per subgenre, $\phi_{pw} = 0.4$.

Table A.2: Corresponding confusion matrices for Table 5.5, $M = 10$.

	bluegrass	rockabilly	country and western
bluegrass	**0.91**	0.00	0.09
rockabilly	0.08	**0.75**	0.17
c.a. western	0.09	0.00	**0.91**

(a) Country, F^{L2}_{FMA}, B_{ew}, $m_s = 15\%$, 64 songs per subgenre, $\phi_{pw} = 0.2$.

	krautrock	new wave	post rock	shoegaze	industrial	progressive
krautrock	**0.49**	0.21	0.18	0.02	0.01	0.09
n. wave	0.05	**0.65**	0.11	0.09	0.00	0.10
p. rock	0.08	0.41	**0.35**	0.06	0.00	0.10
shoegaze	0.11	0.40	0.09	**0.31**	0.04	0.05
industrial	0.15	0.26	0.05	0.06	**0.35**	0.13
progressive	0.11	0.22	0.04	0.06	0.09	**0.48**

(b) Rock, F^{L2}_{FMA}, B_{ef}, $m_s = 7\%$, 232 songs per subgenre, $\phi_{pw} = 0.5$.

	thrash	black metal	death metal	sludge	grindcore
thrash	**0.49**	0.11	0.05	0.28	0.07
b. metal	0.00	**0.82**	0.00	0.18	0.00
d. metal	0.00	0.04	**0.41**	0.49	0.06
sludge	0.00	0.05	0.00	**0.95**	0.00
grindcore	0.08	0.25	0.02	0.09	**0.56**

(c) Metal, F^{L1}_{FMA}, B_{ef}, $m_s = 12\%$, 82 songs per subgenre, $\phi_{pw} = 0.3$.

	hardcore	post punk	electro punk	no wave
hardcore	**0.81**	0.04	0.14	0.0
p. punk	0.04	**0.50**	0.35	0.11
e. punk	0.02	0.05	**0.85**	0.08
n. wave	0.06	0.10	0.21	**0.63**

(d) Punk, F^{L2}_{FMA}, B_{ef}, $m_s = 8\%$, 211 songs per subgenre, $\phi_{pw} = 0.2$.

	house	glitch	drum and bass	downtempo	dubstep
house	**0.47**	0.06	0.08	0.20	0.19
glitch	0.21	**0.41**	0.16	0.07	0.15
d.a. bass	0.11	0.04	**0.47**	0.22	0.16
downtempo	0.09	0.10	0.15	**0.60**	0.06
dubstep	0.06	0.03	0.20	0.10	**0.61**

(e) Electronic, F^{J}_{FMA}, B_{ef}, $m_s = 7\%$, 189 songs per subgenre, $\phi_{pw} = 0.3$.

	British folk	free folk	freak folk
B. folk	**0.69**	0.07	0.24
free folk	0.13	**0.59**	0.28
freak folk	0.12	0.34	**0.54**

(f) Folk, F^{J}_{FMA}, B_{ef}, $m_s = 8\%$, 89 songs per subgenre, $\phi_{pw} = 0.5$.

Table A.3: Additional subgenre experiments on D_{CAL}.

ϕ_{pw}	delta blues	chicago blues	texas blues
0.3	0.92	0.56	0.59
0.4	0.90	0.42	0.43
0.5	0.90	0.47	0.46
0.6	1.00	0.60	0.43
0.7	1.00	0.54	0.47

(a) Blues, F^M_{CAL}, B_{ef}, $m_s = 7\%$, 58 songs per subgenre

ϕ_{pw}	country rock	old time country	traditional country	country pop	western swing
0.3	0.35	0.41	0.57	0.80	0.41
0.4	0.40	0.50	0.26	0.71	0.35
0.5	0.43	0.50	0.28	0.66	0.35
0.6	0.32	0.54	0.36	0.38	0.25
0.7	0.49	0.58	0.35	0.52	0.25

(b) Country, F^M_{CAL}, B_{ew}, $m_s = 7\%$, 63 songs per subgenre

ϕ_{pw}	traditional folk	contemporary folk	british folk	bluegrass
0.3	0.44	0.68	0.76	0.71
0.4	0.33	0.84	0.65	0.64
0.5	0.34	0.83	0.63	0.66
0.6	0.30	0.59	0.67	0.35
0.7	0.33	0.65	0.69	0.36

(c) Folk, F^M_{CAL}, B_{rr}, $m_s = 7\%$, 60 songs per subgenre

ϕ_{pw}	heavy metal	doom metal	stoner rock/metal	deathcore metal
0.3	0.60	0.56	0.25	0.86
0.4	0.72	0.54	0.50	0.87
0.5	0.72	0.53	0.48	0.85
0.6	0.70	0.46	0.25	0.68
0.7	0.70	0.45	0.24	0.74

(d) Metal, F^M_{CAL}, B_{ef}, $m_s = 6\%$, 68 songs per subgenre

Table A.4: Corresponding confusion matrices for Table A.3, $M = 10$.

	delta blues	chicago blues	texas blues
d. blues	**0.92**	0.06	0.02
c. blues	0.18	**0.56**	0.26
t. blues	0.11	0.30	**0.59**

(a) Blues, F^M_{CAL}, B_{ef}, $m_s = 7\%$, 58 songs per subgenre, $\phi_{pw} = 0.3$

	country rock	old time country	traditional country	country pop	western swing
c. rock	**0.35**	0.15	0.05	0.30	0.15
o.t. country	0.12	**0.41**	0.17	0.03	0.27
t. country	0.14	0.06	**0.57**	0.13	0.10
c. pop	0.10	0.00	0.10	**0.80**	0.00
w. swing	0.16	0.00	0.19	0.24	**0.41**

(b) Country, F^M_{CAL}, B_{ew}, $m_s = 7\%$, 63 songs per subgenre, $\phi_{pw} = 0.3$

	traditional folk	contemporary folk	British folk	bluegrass
t. folk	**0.44**	0.26	0.18	0.12
c. folk	0.24	**0.68**	0.08	0.00
B. folk	0.19	0.01	**0.76**	0.04
bluegrass	0.09	0.18	0.02	**0.71**

(c) Folk, F^M_{CAL}, B_{rr}, $m_s = 7\%$, 60 songs per subgenre, $\phi_{pw} = 0.3$

	heavy metal	doom metal	stoner rock/metal	deathcore metal
h. metal	**0.60**	0.13	0.23	0.04
d. metal	0.15	**0.56**	0.21	0.08
s.r. metal	0.55	0.08	**0.25**	0.12
d. metal	0.07	0.00	0.07	**0.86**

(d) Metal, F^M_{CAL}, B_{ef}, $m_s = 6\%$, 68 songs per subgenre, $\phi_{pw} = 0.3$