MUSIC GENRE RECOGNITION

Karin Kosina

DIPLOMARBEIT

eingereicht am
Fachhochschul-Studiengang

MEDIENTECHNIK UND -DESIGN

in Hagenberg

im Juni 2002

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 19. Juni 2002

Karin Kosina

Contents

Preface

This work is dedicated to the music of the universe, its fractal beauty expressing itself in the miracle of life, the laws of nature, the songs of the wind, the rhythm of your heartbeat, the dance of the planets.

I would like to express my gratitude to:

My beloved mother, Marianne Kosina, for reminding me to stop hacking every once in a while to eat, drink, take a break, or sleep, for listening to all the geek talk about sampling rates and segmentation faults, and for believing in me; but most of all, for teaching me that questions are more important than answers.

Stephan Dreiseitl, my supervisor, for his constant encouragement, support and feedback, and especially for his enthusiasm for this project from the very beginning.

George Tzanetakis, for sharing his knowledge and vision in a truly inspiring exchange of thoughts.

My friends, especially Barbara, Eric, Lars, Niki, Petra, René, and Roland, who have supported this work through (in no particular order) proof-reading, encouragement, internet and printing services, suggestions, criticism, access to their non-artificial neural nets, access to their music collections, and blank CD-Rs.

The Free Software community, without whose amazing shared efforts all of this would be unthinkable.

Abstract

The present work describes a system for the automatic recognition of music genres, based exclusively on the audio content of the signal. A comprehensive overview of music genre classification is presented, including disquisitions on the required background from fields such as human sound perception, pattern classification, and signal processing. A critical evaluation of the tacit assumptions behind many existing approaches is made, resulting in a number of conclusions about the design of a music genre recognition system. A prototypical system (*MUGRAT*) that is based on these principles is presented.

MUGRAT is based on extracting various features from the input sound that are likely to also be important in human music genre recognition. These features can roughly be distinguished into two categories: (a) features related to the musical texture, and (b) features related to the rhythm/beatedness of the sound. A k-nearest-neighbour classifier is trained with a set of sample data consisting of randomly chosen musical excerpts of 3 seconds length. The system reaches a classification accuracy of 88 % for the three test genres Metal, Dance, and Classical.

Kurzfassung

Die vorliegende Arbeit stellt ein System zur automatischen Erkennung von Musikrichtungen vor, das ausschließlich auf den Audiodaten des Signals basiert. Sie bietet einen umfassenden Überblick über das Gebiet der automatischen Erkennung von Musikrichtungen, inklusive einer Erörterung des nötigen Hintergrundwissens aus Bereichen wie der menschlichen Wahrnehmung, Klassifizierung und Signalverarbeitung. Eine kritische Betrachtung der Annahmen, die vielen bestehenden Systemen zu Grunde liegen, wird präsentiert. Das Resultat ist eine Reihe von Schlußfolgerungen über das Design eines Systems zur automatischen Erkennung von Musikrichtungen. Ein prototypisches System (*MUGRAT*), das auf diesen Prinzipien basiert, wird vorgestellt.

Die Grundlage von MUGRAT ist, dass verschiedene Features, die vermutlich auch in der menschlichen Klassifizierung von Musik von Bedeutung sind, aus dem Musikstück extrahiert werden. Diese Features können grob in zwei Kategorien unterteilt werden: (a) Features, die die musikalische Textur betreffen, und (b) Features, die sich auf den Rhythmus/Beat des Songs beziehen. Zur Klassifizierung wird der k-Nearest-Neighbour Algorithmus eingesetzt; dabei werden zufällig ausgewählte Ausschnitte von 3 Sekunden Dauer für das Training verwendet. Das System erreicht eine Klassifizierungs-Genauigkeit von 88 % für die drei Test-Genres Metal, Dance und Klassik.

Chapter 1

Introduction

Distinguishing between music genres is a trivial task for human beings. A few seconds of music usually suffice to allow us to do a rough classification, such as identifying a song as rock or classical music. The question that this thesis attempts to address is whether it is also possible for a *machine* to make such a classification. I think that it is, within certain limits. The present work investigates ways to automatically classify music files according to genre, based exclusively on the audio content of the files.

1.1 Motivation

The idea for this work first crossed my mind when I issued the command `locate .mp3`[1], and it returned a listing of 3125 files scattered all across my hard drive, most of them having names such as `01A4029531-B.mp3`. Most of these files contained no indication whatsoever about artist, title, or at least general musical category. I realised that this shows a grave problem in the current handling of multimedia data in general, and music in particular[2]. *We do not deal with sound as sound;* we describe sound, and then use this meta-information as the basis for accessing, classifying, and sharing the original data.

New Media on the Internet

The availability of fast connectivity and large bandwidth for acceptable prices, together with advances in sound and video computer hardware, has fundamentally transformed the Internet. The net used to be based on plain ASCII text; other data such as images, sound, and movies were considered

[1] *Locate* is a UNIX command that lists all files matching a given pattern. In that case, I was using it to look up all the files on my hard drive that contain ".mp3", the suffix for the popular compressed audio format MP3.

[2] The specific discussion here is concerned with the domain of audio data; however note that most of the following issues apply to video and other multimedia data just as well.

and handled as add-ons. Nowadays, the Internet has evolved to be a serious exchange medium for multimedia data of all kinds. Experts believe that this trend will continue, and even accelerate: While the main purpose of having a personal computer used to be the ability to connect to the Internet, and exchange information, this is no longer enough. In the future, the computer will be regarded mainly as a *digital hub* that allows the user to browse, edit, share, and experience multimedia from multiple sources [Job01]. Unfortunately, the handling of this non-traditional content is still at a very primitive stage. This is especially true with regards to searching and automatic indexing and classification, which are essential capabilities for dealing with large amounts of data.

Meta-Data

At present, multimedia data are usually classified based on textual meta-information. The best-known example for this is probably ID3, an extension to the popular MP3 format[3]. ID3 allows the user to add information such as song title, artist, album name, release year, genre, etc. to the beginning or the end of the file. These data are commonly referred to as *tags*.

Meta-data can be very useful for sorting, comparing, and getting quick information. Unfortunately, this system has some important drawbacks:

1. The textual description of audio content is subjective and ambiguous. This is especially true for music.

2. The meta-data have to be entered and updated manually, which means a high degree of effort both in establishing and in maintaining a music database. Such data are very likely to become outdated.

3. The formulation of non-ambiguous and clear search criteria is very difficult. Due to the subjectivity mentioned above, search results will depend not only on the content, but also on its interpretation by the individuals that did the initial classification.

All of these problems are based on the fact that the meta-information is generated manually. Extracting the information from the actual data through an automated, deterministic process would overcome these problems.

Advantages of Automatic Music Genre Recognition

The automatic classification of audio data according to music genres will aid the creation of music databases. It will also allow users to generate personal playlists on the fly, where the user specifies a general description such as

[3]For more information about ID3, refer to the ID3 homepage [Nil01]. A good introduction can also be found in [Hac00].

80s Synth-Pop, and the software does the actual file selection. Furthermore, the features developed for automatic music genre recognition might also be useful in related fields such as similarity-based searching.

Apart from these practical considerations, music genre recognition is a very interesting field for other reasons as well. Our understanding of how humans perceive music is still very limited and incomplete. The results from research in computer-based music processing might well increase our knowledge about the principles of our own perception.

1.2 Scope of this work

Overview

The thesis is a critical evaluation of the current state of automatic content-based classification of music, and the principles, assumptions, and methods of the field, and presents a prototype based on the conclusions drawn from that evaluation. The work gives a general overview of music genre recognition as a special audio signal classification problem, and provides the necessary background from areas such as signal analysis and supervised machine learning. Special focus is put on discovering the features of a song that make it possible for a computer to recognise a music genre. A set of principles that should be kept in mind when designing a music genre recognitionn system is proposed, and a prototype based on these features is presented.

Structure

The thesis is organised as follows: Chapter 2 gives a general overview of the field, explaining the basic ideas, and giving some background in human music perception. It also lists related fields, and discusses their relevance for music genre recognition. Chapters 3 and 4 deal with the two major parts of the problem, respectively feature extraction and automatic classification. Chapter 3 contains a general overview of feature extraction, and a critical discussion of important features for music genre recognition. The necessary background from signal processing is presented as well. Chapter 4 gives an introduction to classification, describing both its theoretical background and the practical application to music classification. In Chapter 5, a detailed description of the prototype I implemented is presented, including the basic assumptions and hypotheses I made, and the limitations of the system. The results I obtained are discussed in Chapter 6. Chapter 7 concludes the thesis with suggestions for future work and a summary of the contributions made by this thesis. In Appendix A the songs that were used to train and test the prototype are listed. Appendix B shows how a classifier using the features extracted by the prototype described in Chapter 5 can be trained and evaluated. Appendix C lists the contents of the enclosed CD-ROM.

Chapter 2

Music Genre Recognition

2.1 Overview

2.1.1 Terminology

Like in every other field of science, a common definition of terms is necessary for music genre recognition research. Unfortunately, such a definition does not exist yet – the literature disagrees on even the most basic terminology, and often uses expressions without defining them properly. Two reasons for this can be identified:

1. Music is part of our daily life. Many terms have an intuitive meaning that seems to be obvious, such as *pitch* or *tempo* of a song.

2. Human perception of sound in general, and music in particular, is dependent on a variety of personal, cultural, and emotional aspects; therefore its description also eludes clear definition. An example for this is the notion of *musical texture*.

This lack of a common basis is one of the most serious obstacles that hinder progress in audio signal classification. Hopefully a conscious effort will be made in the future to establish a well-defined terminology. For this work, I will at least try to define all technical terms when they are mentioned for the first time, and be consistent throughout the thesis in how they are used.

Description of the Field

Music genre recognition is a subsection of the larger field of audio signal classification (ASC), which can be defined as extracting relevant features from a sound, and using these features to identify into which of a set of classes the sound is most likely to fit [Ger00]. Likewise, music genre recognition is the extraction of information from music, and its classification into musical classes (genres). *Automatic* music genre recognition refers to accomplishing

this task with the aid of machines. Since the subject of the present work is music genre recognition by computers, the expressions *automatic music genre recognition* and *music genre recognition* are used synonymously here.

Genres

The term *genre* comes from the Latin word *genus*, which means *kind* or *class*. A genre might be described as a type or category, defined by structural, thematic, or functional criteria [Bre01]. It is important to note that these criteria are not determined by objective facts but are part of a dynamic cultural process:

> A genre is a patterning of communication created by a combination of the individual (cognitive), social, and technical forces implicit in a recurring communicative situation. A genre structures communication by creating shared expectations about the form and content of the interaction, thus easing the burden of production and interpretation. [Eri99]

Genre classification is always subjective with regards to both the individual listener and the socio-cultural environment. For this discussion, a music genre can be considered as *a specific class of music with a set of common properties that in the perception of the average listener distinguish music in that category from other songs*. A specific genre is characterised by the instrumentation and rhythmic structure of the kind of music it describes, but there are many other factors that influence genre classification as done by humans. One of the main challenges of automatic music genre classification is to find out what these factors are.

Processing digital music with computers is easy. Music in a digital format is nothing but a sequence of bits whose values correspond to the sound-pressure levels in an analogue acoustic waveform. These bits can for instance easily be interpreted by a machine to find out certain facts, such as the overall amplitude of the signal at a given time. *Understanding* music, like humans do it all the time without effort, is far more complex. The recognition of music genres is one of these advanced tasks.

2.1.2 Schematic Model

As defined previously, music genre recognition is about classifying music into certain pre-defined categories. The goal is therefore to develop a system with the following properties: Input is a music signal[1] in digital form. Output

[1] I am using the term *music* in its common connotation. Music genre recognition systems try to imitate human skills, so if it *sounds* like music, then it *is* music.

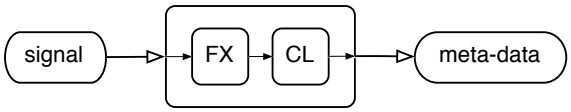


Figure 2.1: Schematic overview of music genre recognition system.

is information about the input signal, describing how the input pertains to the various music genre classes. The design of the *black box* that generates this output from the input signal is the subject of music genre recognition research.

When working on a complex problem, it is often desirable to see if it can be split into invidual parts that can be tackled independently. Music genre recognition is a classification problem, and as such consists of two basic steps that have to be performed:

1. Feature Extraction
2. Classification

The goal of the first step, *feature extraction*, is to get the essential information out of the input data. Feature extraction is discussed in detail in Chapter 3. The second step is to find what combinations of feature values correspond to what categories, which is done in the *classification* part. Classification is a relatively well understood problem; an overview is given in Chapter 4. The two steps can be clearly separated: The output of the feature extraction step is the input for the classification step. We can substitute this subsystem into the black box introduced above, resulting in the model shown in Figure 2.1. This is the basic music genre recognition system in its most simple representation.

2.1.3 Difficulties

Music genre recognition has turned out to be a difficult task for computers to perform. Several reasons for this can be identified.

First of all, music is generally very complex. An amazing amount of work has been done in the area of audio signal classification in the last ten years, and advances are impressive. Still, most of this work focuses on simple monophonic signals – sounds that do not appear in real-world situations, and most notably not in music. Only recently has the study of real-world polyphonic sounds become more active, and the field is still in its infancy.

Secondly, music is a perceptual phenomenon. The physical signals (periodic oscillations of air pressure) that hit our ears are well understood, but how we actually perceive them is still mostly in the dark. The knowledge that we have is not sufficient to explain many phenomena, and by far too

incomplete to allow us to model a computer system that perfectly simulates human music perception.

Finally, it has to be granted that the problem might be inherently difficult. Music genre recognition often seems trivial to humans, but this does not mean that the results are always correct, or even objective, as the following example shows. I obtained MP3 files containing music by the band *Sisters of Mercy* from three different sources. I myself would have labelled the songs as *Gothic Metal*. Checking the ID3 tags of the files revealed the classifications *Industrial*, *Heavy Metal*, and *Other/Unclassified*.

To evaluate a classification system, it might be interesting to see if it makes the *right mistakes* – errors that also a human in that situation would be likely to make, such as classifying slow parts of *Symphonic Rock* songs as classical music.

2.2 Human Sound Perception

2.2.1 The Human Ear

Perhaps the most fundamental thing that has to be understood is that *music is sound*. Music perception happens through the interaction of physical objects that produce sound (musical instruments) with our auditory system. Therefore, it has to be treated as a psychoacoustical phenomenon, not a purely physical one.

The way we hear sound determines how we perceive music, so in order to understand music-listening, it is important to know how human auditory perception works. A short overview of the setup of the human ear follows. More information can be found in any biology textbook; a good summary is also given in [Rub01].

The human ear consists of three main regions: the outer ear, the middle ear, and the inner ear. The outer ear collects sound waves in the air, and channels them to the inside. The middle ear transforms the acoustical vibrations of the sound wave into mechanical vibrations. It contains three interconnected movable bones called *the ossicles* – *hammer*, *anvil*, and *stirrup* – which act as a lever to bridge the eardrum with the *oval window* opening of the inner ear. In the fluid-filled inner ear, whose most important part is the snail-shaped *cochlea* containing the *organ of Corti* with the *basilar membrane*, a compressional wave is generated by the movements. As the fluid in the cochlea moves back and forth, it causes motions in the basilar membrane. This organ is lined with approximately twenty-thousand hair-like nerve cells, each of which has a natural sensitivity to a particular resonant frequency. When the hair cells are excited by vibration, a nerve impulse is generated, and transmitted to the brain. Only the hair cells that respond to the frequencies present at a given moment are activated. Thus, the cochlea basically performs a *frequency analysis* of sound.

The connection between the physical properties of the sound signal and our perception is still not totally clear. This will again be discussed in Section 3.3.

2.2.2 A Study of Human Music Genre Classification

Human listeners have remarkable music genre recognition abilities. This was shown in a study conducted by R.O. Gjerdigen and D. Perrot [PG99]. They used ten different genres, namely Blues, Classical, Country, Dance, Jazz, Latin, Pop, R&B, Rap, and Rock. Eight sample songs for each genre were downloaded from the web in MP3 format. The songs had been classified by leading web-based CD vendors. Half of the eight songs for each style contained vocals, and half of them contained instrumental music only. Five excerpts were taken from each song, with durations of 3000 ms, 475 ms, 400 ms, 325 ms, and 250 ms.

The subjects of the study were 52 college students enrolled in their first year of psychology. The average time per week that they spent listening to music was around 24 hours, according to their own statements. They were presented with the short excerpts in a random order, and asked to decide on one of the ten genres for each excerpt.

The accuracy of the genre prediction for the 3000 ms samples was around 70%, compared to the CD companies' original classification. Taking into account that music genres are a relatively fuzzy concept, and that even the music industry is sometimes contradicting in assigning genres, this percentage is unexpectedly high. The accuracy for the 250 ms samples still was around 40%, and the agreement between the 250 ms classification and the 3000 ms classification was around 44 %. For each time interval, the classification was slightly more accurate for the instruments-only excerpts.

The results of the study are especially interesting, since they show that it is possible to accurately recognise music genres without using any higher-level abstractions. 250 ms are much too short to hear the rhythm, melody or conceptual structure of a song, so apparently classification is possible based on spectral and timbral[2] characteristics alone.

2.3 Related Fields

The following is a short overview of related research domains. This list is by no means complete, nor does it try to be. An outline of the general area of audio signal classification is presented here because many issues, problems, and ideas are also relevant for music genre recognition.

[2]Timbre is discussed in Section 3.4.6.

2.3.1 Speech Recognition

Speech recognition is perhaps the most fundamental audio classification problem: giving a computer the ability to analyse and understand speech. This task is generally difficult due to the large number of ambiguities in spoken language. It only seems easy to humans because we have years of practice in understanding phonetically identical utterances by deriving clues from environmental context, information about the speaker and subject, etc.

The basic functionality of speech recognition is as follows: The system has to decide what phoneme is being spoken at any time, which can be done through spectral analysis and the use of pattern matching algorithms. It has turned out that temporal shifting algorithms such as *Hidden Markov Models* (see also Section 4.1.5) produce the best results. After this, the phonemes are assembled into words based on likelihood estimations.

For further information, please refer to one of the many excellent books and publications on the subject, for instance [BR99]. Johnathan Foote discusses speech recognition as a specific audio classification problem in [Foo99].

2.3.2 Speaker Recognition

Speaker recognition is of special interest for security applications where access rights are granted by analysing a voice sample. This is used to determine whether the speaker belongs to a list of authorised people.

There are two problems that make speaker recognition difficult. (1) It is hard to find features that uniquely identify individual speakers, allowing for variations in the speech patterns, for instance due to a cold, but making voice imitation impossible. (2) Dealing with environmental noise is often a challenge.

2.3.3 Music vs. Speech Classification

Speech recognition systems work well on speech input, but do not provide usable results when music is fed into the system. Likewise, it does not make much sense to try and determine the music genre of a phone conversation. The ability to distinguish between music and speech is important as a front-end to a generic sound-analysis system that is able to process real-world input data, passing the signals on to the specific back-end audio signal classification program that can handle them appropriately. Recent work in that area includes [ZK98a] and [Foo97b]. Besides this, separating music and speech is also very useful in video content analysis.

2.3.4 Video Content Analysis

The audio part of a video signal can be a valuable element in video content analysis. Its importance has long been underestimated, but it has recently

started to get the attention of researchers. Often it is even easier to classify scenes by the audio component than by the visual part. For instance, there are many ways (elements in the take, perspective, camera angle, editing sequence, etc.) to visualise a shoot-out, but in all such scenes, the soundtrack will contain the noise of a shot.

Audio features were used in automatic violence detection by trying to recognise typical sounds, such as cries or shots in [PFE96]. [LWC98] uses audio features to discriminate five types of TV programmes, namely commercials, basketball games, football games, news reports, and weather forecasts, with an accuracy of about 75%. It is likely that by combining video and audio information even better results can be reached.

2.3.5 Beat Tracking and Rhythm Detection

Tapping their foot along with a musical performance is an easy thing to do even for non-musicians, but has been found to be a challenging problem for automatic systems. Rhythm seems to be one of the key elements of musical information. Beat tracking systems are therefore an important part in music genre recognition. A more detailed discussion of the subject can be found in Section 3.4.5.

2.3.6 Query By Example

The emergence of audio databases makes new query mechanisms necessary. One intuitive way to search for a specific tune is to hum the song, and have the system search for similar pieces of music. The classic description of this concept can be found in [GLCS95]. The system proposed in this paper uses approximate pattern matching to find similar tunes; the basic assumption is that *melodic contour*, defined as the sequence of relative differences in pitch between successive notes, can be used to discriminate between melodies. This work suffers from one important drawback. The audio data have to be in a parameterised format such as MIDI; the system cannot be used on a database of e.g. MP3 files. [SSNY97] solves the problem of similarity-based searching for audio data in WAVE format by using transform-based indexing. Each file is divided into small blocks, and a discrete cosine transform (DCT) is applied to each of those, yielding a set of coefficients in the frequency domain. These coefficients are used for matching against the (equally transformed) query data.

2.3.7 Automatic Music Transcription

The purpose of music transcription is to produce a fully notated musical score from audio input. This is trivial for monophonic signals, since here the quality of the solution depends only on the accuracy of the pitch detection algorithm. Polyphonic transcription is much more complicated, as

the system has to identify several concurrent pitches. To further complicate things, there is no clear one-dimensional sequence: a note might begin before a previous note finishes, making it hard to identify sequences [Pic01]. Most work in this area at the moment tries to simplify the problem by separating the audio stream into monophonic components, and then transcribing those.

Rodger J. McNab has developed a system that accepts acoustic input, typically sung by the user, and displays it in standard music notation [MSW96]. A system for the automatic transcription of polyphonic music is presented in Keith D. Martin's paper [Mar96], which also contains an excellent introduction into the subject.

2.3.8 Auditory Scene Analysis

At any given moment, we are surrounded by sound that generally contains more than just one auditory event. The physical signal that hits our ears does not contain any information on the individual parts that have contributed to it, but still we are able to distinguish and separate between sound events. For instance, it is easy for me to follow a conversation in a crowded room with a lot of background noise, even though the frequency and intensity of the voice I am listening to are changing.

The question how this is possible is addressed by the research field called *auditory scene analysis* (ASA). ASA usually needs to be performed as a first step in all computer-listening systems that work in a real-world environment in order to filter out the relevant part of the signal. Most existing ASA systems are limited by the fact that they use only data-driven processing, while the human ability to understand complex sound environment is based strongly on context-dependent inferences. A prediction-driven approach to ASA that is better able to cope with complex sound scenes is presented in [Ell96].

2.3.9 Image Recognition

Transforming the data from the auditory to the visual domain, e.g. by using spectrograms (plots of the spectral analysis of a sound signal) makes it possible to draw from the well-understood domain of image recognition, and use the techniques employed in that area. Promising results have been obtained by researchers from Stanford University [DNS01]. The use of spectrograms for music genre recognition is discussed further in Section 3.4.1.

Chapter 3

Feature Extraction

3.1 Overview

This chapter provides an introduction to the key concepts in feature extraction, gives some necessary signal processing background, and discusses features that are often used in music analysis. Please refer to Chapter 5, Section 5.2 for a detailed description of the actual features used in the prototype developed for this thesis.

3.1.1 Introduction

One of the challenges in music genre recognition is to find out what it is that allows us to differentiate between music styles. The problem is that we want to make observations about the similarity or dissimilarity of two objects (in our case: music clips) that are not directly comparable[1]. To make comparison (and therefore classification) possible, we must transform the data first in order to be able to access the essential information contained in them, a process referred to as *feature extraction*: computing a numerical representation that characterises a segment of audio.

Feature extraction is one of two commonly used preprocessing techniques in classification; it means that *new* features are generated from the raw data by applying one or more transformations. The other possible technique is *feature selection* – the process of identifying a subset of features within the input data that can be used for effective classification. Feature selection can be applied to the original data set or to the output of a feature extraction process. A classification system might use both or either of these techniques. Theoretically, it is also possible to use the raw data, if these are already in a format suitable for classification. In reality, this is hardly ever the case, though. The dimensionality of the datasets is often too high, the data

[1] This is not strictly true. It *is* possible to compare two music files, but such a comparison would be limited to observations that are not relevant to the problem at hand, such as the sampling rate or length of the files.

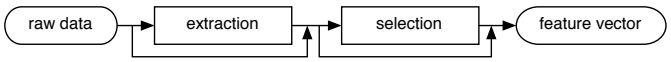


Figure 3.1: Generating a feature vector from an input data set.

contain a lot of redundancy, or are generally not suited for direct comparison. This is especially true in the area of audio signal classification, where we are dealing with long streams of redundant, noisy signals. A schematic overview of the connection between features selection and feature extraction is shown in Figure 3.1.

3.1.2 Formal Notation

A *feature vector* (also referred to as *pattern* or *observation*) $\mathbf{x}$ is a single data item used by the classification algorithm, constisting of d measurements: $\mathbf{x} = (x_1, \ldots, x_d)$. The individual scalar components x_i of the feature vector $\mathbf{x}$ are called *features* or *attributes*, and the dimensionality of the feature space is denoted by d. Each feature vector can be thought of as a point in the feature space. A *pattern set* containing n elements is denoted as

$$X = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$$

and the ith feature vector in X is written as

$$\mathbf{x_i} = (x_{i1}, \ldots, x_{id})$$

In most cases, a pattern set can be viewed as an $n \times d$ *pattern matrix* [JMF99].

3.1.3 Choosing the Right Features

Using the right features is crucial for the classification process. A classification algorithm will *always* come up with some kind of result, but a poor feature representation will lead to a result that does not reflect the true nature of the underlying data. The choice of features needs to fullfill two basic criteria:

1. Objects that we perceive as similar must map to nearby points in feature space, and the distance between the regions marked by the decision boundaries[2] should be as large as possible. This is referred to as *small intra-category scatter* vs. *high inter-category scatter* [DNS01]. Proximity is not necessary in all cases, though. Depending on the

[2]Decision boundaries are explained in Chapter 4, which gives an introduction into classification.

classifier, it might be enough that the points representing one category lie in a region of feature space that can be clearly separated from the regions for the other categories.

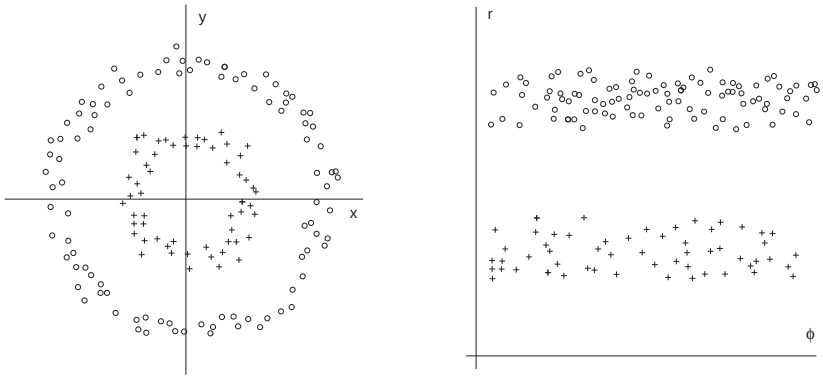
2. The features should preserve all important information that is contained in the data. This has strong implications on what transforms are allowed in the feature extraction process: the data should be simplified without losing information. In the best case, the transform is reversible, meaning that the original data can be reconstructed from the transformed data. One of the properties of the Fourier transform (described in Section 3.2) is full invertability, which makes it well-suited for the problem at hand.

A simple example that demonstrates the importance of choosing the right features can be seen in Figure 3.2. The data consists of points in 2D space, which all have approximately the same distance from the origin. If Cartesian coordinates – (x, y); see Figure 3.2(a) – are used as features, most classification algorithms will have problems finding the correct decision boundaries. On the other hand, if polar coordinates – (r, ϕ), where r is the radius and ϕ is the angle; see Figure 3.2(b) – are chosen as feature representation, it is easy to separate the two regions in feature space because of the radius coordinate. Note that the dimensionality of the feature space is $d = 2$ in both cases. This shows that even a simple transformation can often greatly improve classification results.

Finding the best features is a very difficult task, and it often can only be accomplished through a trial-and-error process. An overview of many of the associated problems, such as how to rule out irrelevant features, is given in [JKP94]. In general, the following can be said: Usually it is desirable to reduce the dimensionality of the input data to improve performance and increase computational efficiency, and in many cases this can be achieved by applying suitable transformations. In the audio domain, a well-known and useful technique is the *Fourier transform*, discussed in Section 3.2.

3.2 Background: Fourier Analysis

Fourier analysis is the generic name for a group of mathematical techniques that are used for decomposing signals into sine and cosine waves. Looking at the waveform (the time-amplitude plot) of a sound signal does not tell us much about the contents. The information is encoded in the frequency, phase and amplitude of the spectral components that make up the signal. To be able to examine this information, it is necessary to calculate the signal's *frequency spectrum*, similar to processes that happen in the human auditory system (see Section 2.2.1).



(a) Cartesian coordinates (b) polar coordinates

Figure 3.2: Example that illustrates how the choice of features influences classification. In these two representations of the same data set, polar coordinates are likely to yield better results than Cartesian coordinates. The data in each region have different x and y values, but share a similar radius, which makes it possible to separate the regions by a straight line in the polar representation.

The following section outlines in brief the principles behind one of the basic techniques to do this. For a more detailed discussion, consult any book on signal processing, for instance [Smi98], an excellent disquisition on digital signal processing that I recommend to everybody who seeks in-depth understanding of the subject. Extended information on the mathematical backgrounds can be found in [Wei02]. Unless stated otherwise, the information in the following section is taken from these references.

3.2.1 Sinusoid Superposition

Jean Baptiste Joseph Fourier (1768 – 1830), a French mathematician and physicist, was the first to discover that *any continuous periodic signal can be represented as the sum of properly chosen sinusoidal components*.

A *sinusoid* is a mathematical function that describes a simple harmonic oscillator: Let θ be an angle measured counterclockwise from the x-axis along the arc of the unit circle. Then $\sin(\theta)$ is the vertical coordinate of the arc endpoint. By definition, it is periodic with period 2π. θ can be expressed as $2\pi f$, resulting in the sine function:

$$y(t) = A\sin(2\pi ft + \phi) \tag{3.1}$$

where A is the amplitude of the wave, f is the frequency of the signal and ϕ is the phase angle of the wave. All signals, no matter how complex

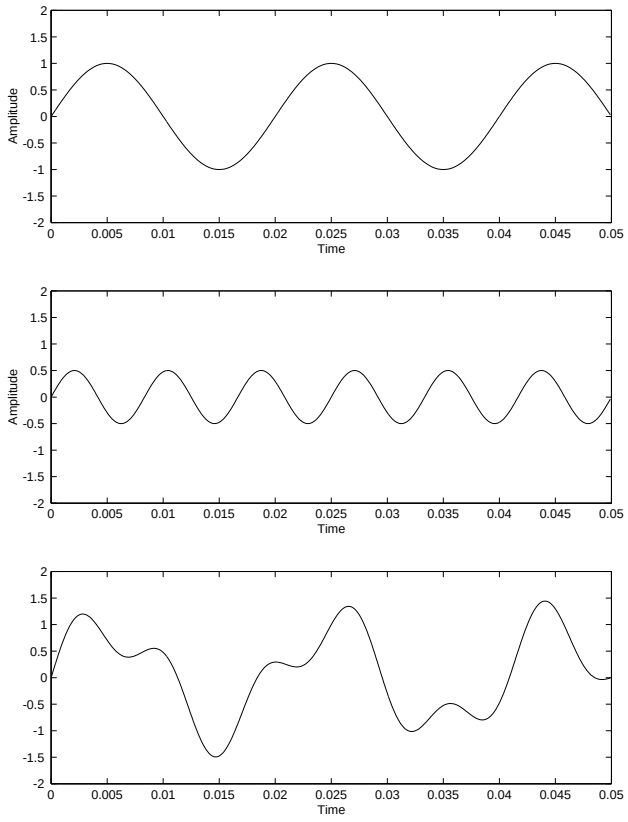


Figure 3.3: Superposition of sine waves: The topmost two plots show two sine waves with frequencies 50 Hz and 120 Hz, $a = \sin(2\pi 50t)$ and $b = 0.5\sin(2\pi 120t)$. Combining those two signals by addition, $c = a + b$, results in the signal displayed in the lowermost plot.

they appear, can be composed by adding such sine functions with different frequency, amplitude and phase. This also applies (without restrictions) to sound, which propagates through waves. The composition of a signal by superposition of sinusoids is called *Fourier synthesis*. A simple example for the generation of a more complex signals by adding sinusoids is shown in Figure 3.3.

3.2.2 Fourier Transform

The opposite of Fourier synthesis is to split a signal into its sinusoidal components; this is called *Fourier analysis*. Fourier analysis allows us to see what frequencies are present in the signal, and how strong the influence of each spectral component is. The mathematical transform[3] to calculate the Fourier analysis – called the *Fourier transform* – is given as:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft} dt, \tag{3.2}$$

and the inverse transform as

$$x(t) = \int_{-\infty}^{\infty} X(f) e^{j2\pi ft} df. \tag{3.3}$$

In this equation, $x(t)$ is the function in time, and $X(f)$ is the corresponding frequency function. j is the square root of -1 and e denotes the natural exponent

$$e^{j\phi} = \cos(\phi) + j\sin(\phi).$$

Applying the Fourier transform to a signal converts it from its *time domain representation* into the *frequency domain representation*. Note that these are equivalent alternative ways of representing the *same* signal; modifying a signal in one domain also changes it in the other. (For instance, performing a convolution on time domain signals results in a multiplication of their frequency spectra, and vice versa.) Figure 3.4 shows the frequency spectrum of the example signal from Figure 3.3, obtained by calculating the Fourier transform of the signal.

The Fourier transform operates on a signal of theoretically infinite length. It provides information about the spectral content of the whole signal, and totally lacks time resolution. This is not a problem for periodic signals (such as the ones in the example above), but in the case of aperiodic signals (such as music), a different solution is needed.

3.2.3 Short-Time Fourier Transform

The *short-time Fourier transform (STFT)* breaks the input data into short sequential frames and performs the Fourier transform on each of them, analysing how the signal changes over time. While this solves the problem of taking temporal changes into account, it also generates new difficulties. Since the Fourier transform requires the input signal to be infinite, each

[3]The mathematical term *transform* describes a procedure that changes an input vector into an output vector, the difference to a *function* being that functions can have multiple input values but only one output value, whereas the number of input and output values of a transform are arbitrary.

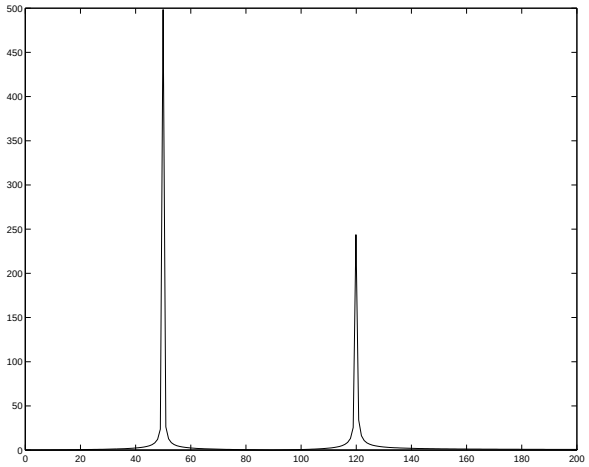


Figure 3.4: Fourier Analysis of the signal c in Figure 3.3. Two peaks at 50 Hz and 120 Hz can be observed, which correspond to the two sine waves that made up the signal: $a = \sin(2\pi 50t)$ and $b = 0.5\sin(2\pi 120t)$.

frame has to be expanded into a signal of infinite length. The sudden break at the frame boundaries introduces spectral components that are not present in the original signal.

A workaround for this problem is to multiply each frame with a *windowing function*, which scales the signal to zero at each end. The artefacts cannot be totally eliminated by this, but can be significantly reduced by choosing the right window function. A very common function is the *Hamming window* [Smi98, Ch. 16], which features a good trade-off between computing efficiency and good results. Note that windowing leads to loss of information near the window boundaries, which is why the windows have to overlap. The STFT can be calculated as follows[4]:

$$X(f,t) = \int_{-\infty}^{\infty} h(t'-t)x(t)e^{-j2\pi ft}dt \tag{3.4}$$

3.2.4 Discrete Fourier Transform

The techniques discussed so far (see Equation 3.2) work on continuous signals. The equivalent in the digital domain is the *discrete Fourier transform (DFT)*. Consider the case of a discrete function, $f(t) \rightarrow f(t_k)$, by letting

[4] $h(t)$ is the window function in this equation.

$f_k \equiv f(t_k)$, where $t_k \equiv k\Delta$, with $k = 0, \ldots, N-1$. Choose the frequency step such that

$$v_n = \frac{n}{N\Delta}$$

with $n = -N/2, \ldots, 0, \ldots, N/2$. The standard DFT is then given by

$$F_n = \frac{1}{N} \sum_{k=0}^{N-1} f_k e^{-j2\pi nk/N} \tag{3.5}$$

and the inverse transform by

$$f_k = \sum_{n=0}^{N-1} F_n e^{j2\pi nk/N}. \tag{3.6}$$

Note that the frequency response obtained from the DFT is often complex, even though the original signal is completely real.

There are several ways to calculate the DFT, such as solving simultaneous linear equations or correlation [Smi98, Ch. 8], a discussion of which would go beyond the scope of this thesis. The most efficient and most widely used of these algorithms is the *Fast Fourier Transform (FFT)*, described in detail in [Smi98, Ch. 12].

3.3 Physical Features vs. Perceptual Features

Usually, the features used in audio signal classification systems are divided into two categories: *physical* and *perceptual*. Physical features are based on a mathematical and statistical analysis of the properties of the sound signal. Examples for physical features are fundamental frequency, energy, zero-crossing rate etc. Perceptual features are based on how humans hear sound, for instance pitch, timbre and rhythm.

Obviously, all perceptual features are related to the physical features in some way, since our perception is based on the physical sound signal. Some of these connections are relatively straightforward: The amplitude of the signal corresponds to the perceived loudness, and the fundamental frequency is related to the pitch of a sound. However, taking a closer look at these connections reveals that they are only rough approximations. For instance, the subjective loudness depends on the spectral content of the signal. Many perceptual features are very hard to describe in mathematical terms altogether, since they are based on an intricate combination of physical properties.

I am, for two reasons, not going to follow the strict separation between physical and perceptual features in this thesis.

First of all, as stated above, all perceptual features are based on the physical aspects of the signal. Likewise, all physical properties of the sound signal are subjected to the human perceptual mechanism when we hear the sound, and are thus turned into perceptual features. We cannot hear "true" sounds, as little as we can see "true" colours. An interesting phenomenon that illustrates this fact is the reconstruction of the fundamental frequency if it is missing from a signal. For example, the fundamental frequency of a male adult's voice is around 120 Hz, but many phone systems transmit only frequencies between 300 and 3400 Hz. While in this case we do hear that the quality of the signal is bad, we do not realise that the fundamental frequency is lacking; the auditory system constructs and fills in the missing frequency from the frequencies that are present [PFE96].

Secondly, I am sceptical about using the term *perceptual*. There are many setups that try to integrate perceptual features, e.g. pitch, but do so by modelling the physical properties of the signal that are thought to be important in pitch perception. They are, hence, physical models using a specific set of physical features. A truly perceptual system would have to use a perceptual model for the feature extraction, i.e. model the way sounds are perceived by humans through their auditory system. Such an approach would probably produce the best results in audio signal classification, but so far our understanding of perceptual processes is too limited to make this work.

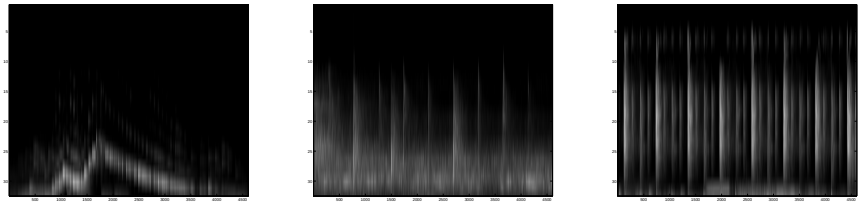
The distinction between physical and perceptual features can be useful in some contexts; but for the reasons given above I will avoid using it in the rest of this chapter.

3.4 Features Used in Music Genre Recognition

3.4.1 Frequency Spectrum

The frequency distribution of a signal can be calculated through the Fourier transform (described in Section 3.2). Similar processes also happen in the auditory perception system of humans and other vertebrates, which indicates that the spectral composition of a signal is indeed the primary carrier of information. The frequency spectrum is one of the essential features for music genre recognition, and is used as the basis for deriving many other features.

An interesting and intuitive way of working with spectral information is to transform the FFT output to the visual domain by using *spectrograms*. A spectrogram is a time-frequency plot of the FFT results. The x-axis shows increasing time t, while the y-axis displays the frequency f (low frequencies on top). The colour of a point (t, f) indicates the amplitude (black being zero, and using a linear scale for converting from amplitude values to greyscale values).



(a) Classical (b) Heavy Metal (c) Dance

Figure 3.5: Spectrograms for music samples from three genres. Time is shown on the x-axis, frequency on the y-axis (increasing from the top), grey values indicate power.

Figure 3.5 shows example spectrograms[5] for randomly chosen samples from three songs, all sampled at 22050 Hz with a time window of 64 samples. The songs can be considered typical for their respective genres: *Für Elise* by Ludwig van Beethoven (Classical, Figure 3.5(a)), *I just want you* by Ozzy Osbourne (Heavy Metal, Figure 3.5(b)), and *The X-Files (Remix)* by DJ Dado (Techno/Dance, Figure 3.5(c)). A number of important observations can be made by looking at the spectrograms: The Beethoven piece is distinctly different from the other two, since there is only one instrument (a piano). The Dance clip has a strong beat that is clearly visible: high energy in many frequencies for short periodic time intervals. The Metal song is characterised by the electrical guitar, which produces high energy in a certain frequency band over a more or less continuous time.

Spectrograms have been used in an attempt to solve the problem of music genre recognition with image analysis techniques. A group of researchers from Stanford University proposed a system that transforms the data from the audio domain (waveform) to the visual domain (spectrogram), and then applies the *Texture-of-Textures* algorithm, a fairly new technique for edge detection in images [DNS01]. They reached a classification accuracy of approximately 75% in distinguishing Rock, Classic and Jazz.

3.4.2 Mel-Frequency Cepstral Coefficients

Human perception of the frequency content of sounds does not follow a linear scale but uses a logarithmic distribution. Mel-frequency cepstral coefficients (MFCCs) are based on the spectral information of a sound, but are modelled

[5]The spectrograms were generated using the *Auditory Toolbox* for Matlab, written by Malcolm Slaney [Sla98].

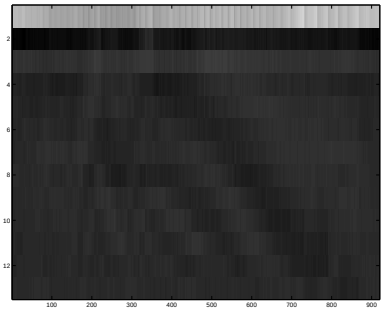


Figure 3.6: Plot of mel-frequency cepstral coefficients for the sample from Figure 3.5(a), approximately 9 seconds at 22050 Hz. The sound is sampled at 100 Hz in the MFCC calculation, resulting in the 910 samples on the x-axis. The y-axis shows the 13 MFCC features in increasing order, C_0 on top.

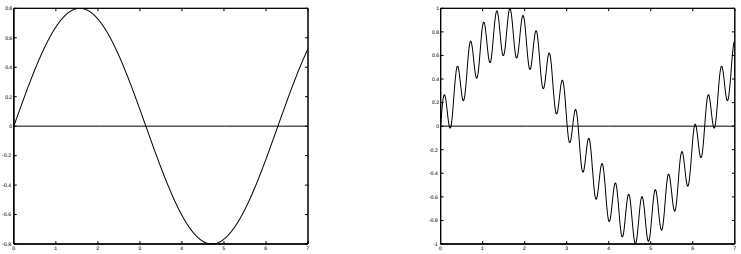
to capture the perceptually relevant parts of the auditory spectrum[6]. The sequence of processing is as follows [Sla98]:

1. Window the data (e.g. with a Hamming window);
2. calculate the magnitude of the FFT;
3. convert the FFT data into filter bank outputs;
4. calculate the log base 10;
5. calculate the cosine transform.

The filter bank is what makes MFCCs unique. It is constructed using 13 linearly spaced filters and 27 log-spaced filters, following a common model for human auditory perception. The distance between the centre frequencies of the linearly spaced filters is 133,33 Hz; the log-spaced filters are separated by a factor of 1.071 in frequency. The final cosine transform (step 5) is applied to reduce the dimensionality of the output, typically to the 12 most important coefficients. Additionally, the power of the signal for each frame is calculated, resulting in a feature vector of $d = 13$.

MFCCs are commonly used in speech recognition systems, and seem to capture the perceptually relevant part of the spectrum better than other techniques. They have successfully been applied to the content-based retrieval of audio samples [Foo97a] and also used in music genre recognition systems [Li00].

[6]The term *cepstrum* refers to a transform of the spectrum, and is a pun on itself: *cepstrum* is the word *spectrum* with the first syllable reversed.



(a) $0.8sin(x)$. (b) $0.8sin(x) + 0.2sin(20x)$.

Figure 3.7: Example showing that fundamental frequency determination is not always possible through the ZCR. Note how the signal on the right crosses the x-axis more than twice in one cycle (after [Roa96]).

The results of calculating the MFCCs for the music clip used in Figure 3.5(a) is shown in Figure 3.6. The MFCC plot is harder to interpret visually than the spectrogram, but has been found to yield better results in computer sound analysis.

3.4.3 Average Zero-Crossing Rate (ZCR)

A zero-crossing occurs when successive samples in a digital signal have different signs. Therefore, the rate of zero-crossings can be used as a simple measure of a signal's frequency content.

For simple signals, the ZCR is directly related to the fundamental frequency (f_0). A sinusoid will cross the zero line twice per cycle, and therefore its frequency can be determined by dividing the ZCR by 2. At first sight, this might be a very simple and effective method of finding the fundamental frequency of a sound signal. This is not possible in all cases, though: As shown by [Roa96], signals that contain partials with frequencies that are much higher than the fundamental frequency will cross the zero line many more times per cycle (see Figure 3.7).

The reason why finding the fundamental frequency is so important is that it is crucial for pitch detection, which in turn is essential in music transcription and melody recognition systems.

In music genre recognition, the fundamental frequency is not as important. This is due to several reasons. First of all, it is very hard to determine the fundamental frequency of a complex auditory scene containing several sound sources (instruments, voices, sound effects). Actually, it is not only hard, but it does not really make sense. The question is, what should be considered as the primary signal in such a scene? In most cases, the answer

would be arbitrary. Secondly, even if we overcome the problem of how to find a pitch in our music sample, what does this accomplish? We can derive the melody contour for a given song, but this as such does not tell us what genre this song belongs to. As an illustration, consider the song *Fade To Black* by *Metallica*, a fairly typical Heavy Metal song. The EBM/Synth remix by *Apoptygma Berzerk* still has the same main melody, but definitely sounds different, and no human would classify the EBM song as Metal based on that melody.

The observation that pitch- (and fundamental frequency-) based features are of limited relevance is also reported in [LWC98], where several audio features for video scene segmentation and classification[7] are evaluated.

Even though it cannot reliably be used for fundamental frequency determination, the zero-crossing rate is popular in audio signal classification. It is a time-domain feature, and as such very fast to calculate. The ZCR can been used as a statistical measure of spectral characteristics in music vs. speech determination by analysing the changes of ZCR over time. This feature is sometimes called the *ZCR contour* [Sau96]. The ZCR also makes it possible to differentiate between voiced and unvoiced speech components: voiced components have much smaller ZCR values than unvoiced ones [ZK98b]. The average short-time zero-crossing rate can also be useful in combination with other features in general audio signal classification systems. Tong Zhang and C.-C. Kuo used *ZCR curves* to distinguish environmental sounds based on regularity, periodicity, and range of amplitude [ZK98a]. The ZCR curves are calculated as follows:

$$Z_n = \sum_m |\text{sgn}[x(m)] - \text{sgn}[x(m-1)]| w(n-m), \tag{3.7}$$

where

$$\text{sgn}[x(n)] = \begin{cases} 1 & x(n) \geq 0 \\ -1 & x(n) < 0 \end{cases}$$

and

$$w(n) = \begin{cases} 1/2 & 0 \leq n \leq N-1 \\ 0 & \text{otherwise.} \end{cases}$$

m is the window size in this short-time function. The ZCR curve of music has a lower variance and average amplitude than that of speech, since music tends to be more "stable" during a certain period. Similar observations can be made about environmental sounds such as footsteps, chirping birds, etc. The zero-crossing rate is also used in the prototype described in Chapter 5.

[7] This field is comparable to music genre recognition since it also deals with complex non-speech audio signals.

3.4.4 Short-Time Energy

The short-time energy of an audio signal is defined as

$$E_n = \frac{1}{N} \sum_m [x(m)w(n-m)]^2, \tag{3.8}$$

where $x(m)$ is the discrete time audio signal, n is the time index of the short-time energy, and $w(m)$ is a rectangle window, i.e.

$$w(n) = \begin{cases} 1/2 & 0 \leq n \leq N-1 \\ 0 & \text{otherwise.} \end{cases}$$

The short-time energy function shows the amplitude variation over time. The energy value is usually calculated around every 100 samples. The short-time energy can be used to measure silence, and to distinguish between voiced and unvoiced speech components in speech recognition. (E_n values for unvoiced parts are significantly smaller [ZK98a].)

3.4.5 Rhythm and Beat

Tapping their foot along with a piece of music is a trivial task for human listeners, yet it has turned out to be remarkably difficult for automated systems. Beat tracking and rhythm detection is a large and rapidly evolving research area in itself. An in-depth discussion is not possible within the limitations of this work. Refer to [Dix99] for an overview of beat tracking methods; a system for tracking musical beats in real time is proposed in [AD90]. An introduction into the even more complex subject of musical rhythm can be found in [Bil92].

There are a number of factors that make beat tracking difficult, but that are no problem for the human auditory perception system. For instance, we can easily determine the beat if tempo and metrical structure are not explicitly specified in the beginning of the song, something that most commercially available beat tracking systems depend on. If the tempo changes throughout the song, we can easily adapt to this within seconds, while most current automatic systems are unable to adjust. Furthermore, beat tracking systems usually cope poorly with noise, i.e. deviations from the expected timing. Another important issue is dealing with *syncopation* – sections where salient events occur between the beats and not on the beat [Dix99]. Note that all of these factors are usually present in music and make up much of what we find interesting in it (improvisation, unexpected change, drama).

Fortunately, accurate beat detection is not necessary for music genre recognition. We are not interested in the exact beat, but more in the perceived *tempo* of the song and the *beatedness*. *Tempo* can be described as the perceptual sense that the sound is recurrent in time at regular intervals, with the length of this interval between 250 ms and 2s [MSV98]. It

should be noted that tempo is a subjective feature – a sound does not have a *real* tempo that can be measured exactly. *Beatedness* is a measure of the strength of the signal's beat, given by determining a periodic strong peak in the signal, and measuring the relative amplitude of this beat compared to the average amplitude of the signal. This feature is used in the prototype developed for this work and will be described in more detail in Section 5.2.

3.4.6 Timbre

Timbre is generally defined as "the quality which allows one to tell the difference between sounds of the same level and loudness when made by different musical instruments or voices" [ZK98b]. It depends on the spectrum, the sound pressure, the frequency location of the spectrum, and the temporal characteristics of the stimulus. In music, timbre is thought to be determined by the number and relative strengths of the instruments partials.

Timbre is one of the most subjective audio features, and so far no systems have been developed that model it in a satisfactory manner: It is common to use only the spectrograms of sounds for timbre information. More advanced and truly perceptual timbre modelling would be a significant improvement to many areas of audio signal classification.

Chapter 4

Classification

4.1 Overview

The feature extractor (see Chapter 3) computes feature vectors representing the data to be classified. These feature vectors are then used to assign each object to a specific category. This is the *classification* part, which constitutes the second basic building block of a music genre recognition system.

There is a large variety of classification techniques to choose from. The fact that many of these can be used as black-box algorithms makes it tempting to just apply one without understanding it. However, I believe that a basic understanding of the subject is essential in order to choose the right classifier, and I will provide a short overview in the next section, including the general concepts, mathematical background, and basic ideas behind the most widely used classifiers. The actual classification system used in the prototype developed for this thesis is described in detail in Chapter 5, Section 5.3.

It is of course impossible to cover all interesting aspects of the topic here. There are many excellent books on classification to which I would like to refer the interested reader, for instance [DHS01]. Unless stated otherwise, the information in the following section is taken from this publication. A good review of pattern recognition and classification techniques is also given in [JDM99].

4.1.1 Introduction

Classification is a subfield of decision theory. It relies on the basic assumption that each observed pattern belongs to a category, which can be thought of as a prototype for the pattern. Regardless of the differences between the individual patterns, there is a set of features that are similar in patterns belonging to the same class, and different between patterns from different classes. These features can be used to determine class membership.

Consider the example of music genre recognition. Music can be of arbitrary complexity, songs from one genre differ in many ways. Still, humans are able to categorise them easily. This seems to support our assumption that there are certain fundamental properties shared by pieces belonging to one genre.

Classification can also be understood by approaching it in geometrical terms. As stated before, the feature vectors can be regarded as points in feature space. The goal of the classifier is to find *decision boundaries* that partition the feature space into regions that correspond to the individual classes. New data items are then classified based on what region they lie in. This depends on a feature representation of the data in which feature vectors from the same category can easily be distinguished from feature vectors from other categories. (The importance of finding a good representation is discussed in Section 3.1.3, and illustrated in Figure 3.2 on page 15.)

Domain Independence

Finding a good feature representation requires in-depth knowledge of the data and context; feature extractors must be adapted to the specific problem and are highly domain-dependent. Classification techniques, on the other hand, are basically domain-independent. This can be explained by the fact that feature extraction is also an abstraction step, transforming domain-specific data into a more general numerical representation that can be processed by a generic classifier. Note that this is an important difference: The feature extraction part is where knowledge of music, psychoacoustics, signal processing, and many other fields is required; it is an area that has only recently started to receive the attention it deserves, and there is a limited basis of previous work to build on. Classification, on the other hand, is an advanced field that has been studied for many years, and that provides us with many fast, elegant and well-understood solutions that can be adopted for use in music genre recognition.

Design aspects

The most critical aspects in designing a classification system are the choice of features, the number of features used, the quantification of distance, the robustness of the system, the complexity in terms of speed and memory usage when implemented in software, and scalability.

Difficulties

The main difficulty in classification arises from the fact that in addition to the dissimilarities caused by the different underlying models, the feature values for objects belonging to the *same* category often also vary considerably. If all objects from one class were perfectly equal, classification would

be trivial. This is of course never the case; but it should be noted that the classifier never sees the actual data, only the feature vectors. Therefore, the following is equally true: A feature representation that extracts exactly the information that differentiates the categories would also eliminate the need for a complex classification step. (Likewise, a perfect classifier would not need any feature extraction at all, but would be able to uncover the true class membership from the raw data. In reality, neither feature extractors nor classifiers are perfect, but may be combined to produce working results.)

The variation in patterns belonging to the same category can be due to two factors: First, the underlying model might generate that complexity: A relatively simple model can create seemingly random output, which cannot trivially be detected by an observer who does not know the model[1]. Secondly, considerable variation can be caused by noise. *Noise* can be defined as *any property of the pattern that is not due to the true underlying model but instead to randomness in the world or the sensors*. As is obvious from this definition, noise is present in all objects in nature.

The challenge is to distinguish the two kinds of differences between feature values: Are they caused by different models, which means that the objects belong to different categories, or are they due to noise or the complexity of the model, meaning that the objects belong to the same category?

Training and Learning

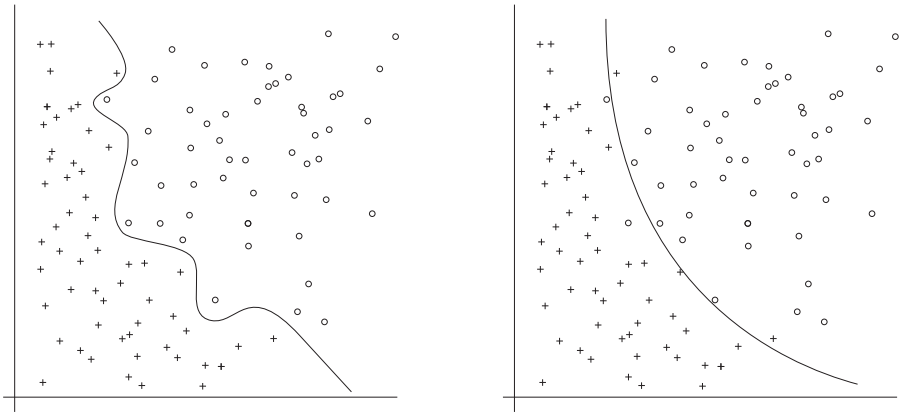
Creating a classifier usually means specifying its general form, and estimating its unknown parameters through training. *Training* can be defined as *the process of using sample data to determine the parameter settings of the classifier*, and is essential in virtually all real-world classification systems.

Classification is often called *supervised learning*. Supervised learning consists of using labelled feature vectors to train classifiers that automatically assign class labels to new feature vectors. Another variant of learning is *unsupervised learning* or *clustering*, which does not use any label information; the system tries to form "natural" groupings. *Reinforcement learning*, a third type, refers to a technique where the feedback to the system is only *right* or *wrong* – no information about the correct result is given in case of a wrong categorisation.

Generalisation and Overfitting

An important concept in connection with the training process is *overfitting*: If the system is tuned too closely to the training data, it might perform perfectly on these samples, but is unlikely to work well on new patterns.

[1] As a side note, this is also the basis of many cryptographical algorithms: Encrypted ciphertext contains all the information of the plain text message, but looks like random noise to the casual observer.



(a) Decision boundary fit too closely to the training data.

(b) More general decision boundary.

Figure 4.1: Example that demonstrates the problem of overfitting. In the left image, the system perfectly classifies all the training samples, but is unlikely to perform well on new data. The decision boundary in the right image is less complex and probably better reflects the true nature of the data. While some of the training data are misclassified by this system, the overall performance on unknown input is better.

Figure 4.1 illustrates this idea. If the decision boundary is fit too closely to the training data, the system will obtain 100% decision accuracy on these samples; but the decision boundaries in that case do not reflect the true model behind the data. New, unknown patterns are likely to be misclassified by such a system. A more general representation using less complex decision boundaries is often favourable. The challenge is to find the right trade-off for the generalisation: having a system that is complex enough to capture the differences in the underlying models, but simple enough to avoid overfitting.

4.1.2 Bayesian Decision Theory

Classification means taking a decision about the class membership of an object. Such a decision can be correct or incorrect; informally speaking, the goal is to maximise the chance of making the right decision. One of the formal approaches to this problem is Bayesian decision theory, which quantifies the tradeoffs between possible decisions by using probabilities. A short discussion of Bayesian decision theory follows.

Bayes Formula

The **state of nature** is denoted by ω. This is the variable that needs to be described probabilistically and describes the real state of an object. In our case, the music genres used by the system form a finite set of c states of nature $\{\omega_1, \ldots, \omega_c\}$.

The **prior probability** $P(\omega_j)$ describes the prior knowledge that we have before actually seeing any data. It is sometimes also called *a priori probability*. For instance, suppose that a user's online music collection has been encoded from 35 EBM CDs and 15 Metal CDs, each CD containing 10 songs. If we let ω_1 denote the genre EBM and ω_2 be Metal, the prior probility that a randomly chosen song will belong to the EBM category is then $P(\omega_1) = 0.7$.

The **class-conditional probability density function** $p(\mathbf{x}|\omega_j)$ denotes the probability that a feature vector $\mathbf{x}$ is observed given that the state of nature is ω_j.

Suppose that both the prior probilities $P(\omega_j)$ and the conditional densities $p(\mathbf{x}|\omega_j)$ for $j = 1, \ldots, n$ are known. If a value $\mathbf{x}$ is measured, we can calculate the probability of the state ω_j given $\mathbf{x}$ by using Bayes formula

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)P(\omega_j)}{p(\mathbf{x})}, \tag{4.1}$$

where

$$p(\mathbf{x}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j)P(\omega_j).$$

$P(\omega_j|x)$ is called the *posterior probability* and it denotes the probability that the state of nature is ω_j given that feature value $\mathbf{x}$ has been measured. Obviously, in order to minimise the probability of error, the system must decide on the ω_j for a given $\mathbf{x}$ for which $P(\omega_j|\mathbf{x})$ is highest. This is called the *Bayesian decision rule*, and can be expressed as

$$\text{decide } \omega_i \text{ if } P(\omega_i|\mathbf{x}) > P(\omega_j|\mathbf{x}) \text{ for all } i \neq j.$$

Risk

In many classification problems, there are different *losses* associated with taking certain decisions. In these cases, the posterior probabilities must be weighted according to the costs associated with taking that decision, the goal of a classifier being to minimise the expected loss. The rest of this discussion assumes that the *zero-one loss function* is used, which means that all errors are equally costly, i.e.

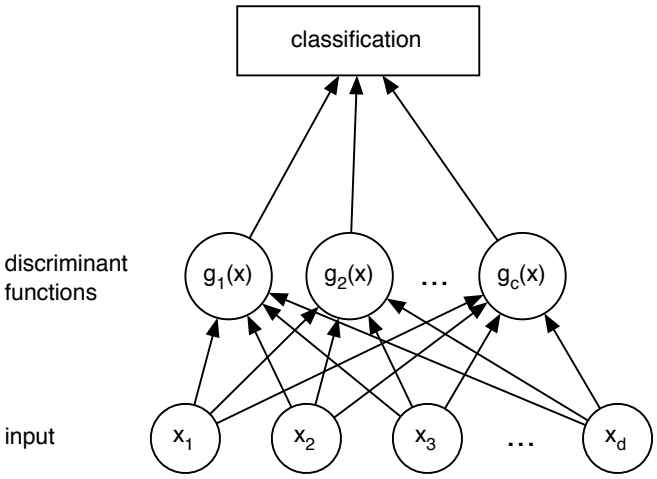


Figure 4.2: Structure of a classifier with d inputs $\mathbf{x_i}$ and c discriminant functions $g_i(\mathbf{x})$. For each input, the discriminant functions are evaluated. Each of the discriminant functions is associated with a category ω_i; the system chooses the category for which $g_i(\mathbf{x})$ is at a maximum.

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \quad i, j = 1, \ldots, c$$

where $\{\alpha_1, \ldots, \alpha_a\}$ is the finite set of a possible actions (i.e. classifications), and $\lambda(\alpha_i|\omega_j)$ is the *loss function* that describes the loss incurred for taking action α_i when the state of nature is ω_j. The *expected loss* is called *risk* in decision theory, and is calculated as

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j) P(\omega_j|\mathbf{x}). \tag{4.2}$$

The risk corresponding to the zero-one loss function is the average probability of error

$$R(\alpha_i|\mathbf{x}) = 1 - P(\omega_i|\mathbf{x}). \tag{4.3}$$

This is also referred to as *minimum error rate classification*.

4.1.3 Discriminant Functions

Bayes formula and the theoretical principles outlined in the previous section can be applied to many fields. The connection to classification can be described as follows:
A classifier can be represented as a network that computes c *discriminant functions* of the form $g_i(\mathbf{x})$ where $i = 1, \ldots, c$, and selects the category

corresponding to the largest discriminant. The classifier assigns a feature vector $\mathbf{x}$ to a class ω_i if $g_i(x) > g_j(x)$ for all $i \neq j$. A schematic overview of this functional structure is presented in Figure 4.2.

In the case of a minimum error rate Bayes classifier[2], the maximum discriminant function corresponds to the maximum posterior probability; we can therefore set $g_i(\mathbf{x}) = P(\omega_i|\mathbf{x})$. The discriminant functions are then given by

$$g_i(\mathbf{x}) = p(\mathbf{x}|\omega_i)P(\omega_i). \tag{4.4}$$

4.1.4 The Normal Density

As explained in previous sections, a Bayes classifier is determined by the prior probabilities $P(\omega_j)$ and the conditional probability density functions $p(\mathbf{x}|\omega_j)$. One of the most common density functions is the *normal density*. The normal density models the case when all feature vectors $\mathbf{x}$ for a given class ω_j are based on a prototype vector $\mu_\mathbf{j}$, but are corrupted by normally distributed noise. Many objects in nature can be described by this model. The normal density in d dimensions can be expressed as

$$p(x) = \frac{1}{(2\pi)^{\frac{d}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}, \tag{4.5}$$

where $\mathbf{x}$ is a d-component column vector, $\boldsymbol{\mu}$ is the d-component *mean vector*, $\mathbf{\Sigma}$ is the d-by-d *covariance matrix*, and $|\mathbf{\Sigma}|$ and $|\mathbf{\Sigma}^{-1}|$ are its determinant and inverse. $(\mathbf{x} - \boldsymbol{\mu})^T$ denotes the transpose of $(\mathbf{x} - \boldsymbol{\mu})$. Equation 4.5 is often written in the simplified form

$$p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \mathbf{\Sigma}). \tag{4.6}$$

The *expected value* $\mathcal{E}$ is given as

$$\mathcal{E}[\mathbf{x}] = \int \mathbf{x}p(\mathbf{x})\mathrm{d}\mathbf{x} \equiv \boldsymbol{\mu}, \tag{4.7}$$

and the covariance matrix as

$$\mathcal{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T] = \int (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T p(\mathbf{x})\mathrm{d}\mathbf{x} \equiv \mathbf{\Sigma}. \tag{4.8}$$

If the data are normally distributed and $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$ are known, the conditional density function $p(\mathbf{x}|\omega_i)$ in the classifier discriminant function (see Equation 4.4) can be substituted by the normal density as defined in Equation 4.5, resulting in a fully specified classifier.

The normal density is an elegant and powerful model. It can, of course, only be used if the data to be classified are actually normally distributed,

[2] A Bayes classifier is an ideal classifier that obeys the principles outlined in Section 4.1.2.

which is often not the case.

The typical situation in a real classification problem is one of the following:

1. The full probability structure is unknown, but the general form of the distribution can be assumed to follow one of the standard distributions, such as the normal density. In this case, we need to determine the unknown parameters of the conditional probability density function, for instance the mean vector $\boldsymbol{\mu}$. This is called *parameter estimation* and is described in Section 4.1.5.

2. There is no knowledge about the underlying structure, not even the general form of the probability densities. The classification has to be based on the information gained from the training data alone. Such *non-parametric techniques* are discussed in Section 4.1.6.

4.1.5 Parameter Estimation

In most classification problems, the conditional densities are not known. However, in many cases, a reasonable assumption can be made about their general form. This makes the problem significantly easier, since we need only estimate the parameters of the functions, not the functions themselves.

The unknown probability densities are usually estimated in a training process, using sample data. For instance it might be assumed that $p(\mathbf{x}|\omega_i)$ is a normal density. We then need to find the values of the mean $\boldsymbol{\mu}$ and the covariance $\boldsymbol{\Sigma}$.

Maximum-Likelihood Parameter Estimation

There are several approaches to parameter estimation. One of the most common ones is *maximum-likelihood parameter estimation*, which views the parameters as quantities that are fixed but unknown. The best estimate is then the one that maximises the probability of obtaining the training samples. For instance, the maximum-likelihood estimate for the mean $\boldsymbol{\mu}$ is the *sample mean* $\hat{\boldsymbol{\mu}}$, calculated as

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{k=1}^{n} \mathbf{x_k}, \tag{4.9}$$

i.e. the arithmetic average of the training samples. It can be shown that the accuracy of the estimate will increase with the number samples. Recalling that the samples can be thought of as a cloud of points in feature space, the sample mean corresponds to the centroid of that cloud.

Hidden Markov Models (HMMs)

Hidden Markov Models [RJ86] solve the problem of making a *sequence* of decisions, instead of just a single one. In many classification problems, we are dealing with a series of states, where the state at time t is influenced by previous states. A simplifying assumption (called *Markov assumption*) is that previous states influence the state at time t only via the state at time $t-1$. An example is speech recognition, where words must be generated from a sequence of spoken sounds. Speech recognition is the main application domain of HMMs.

One of the main concepts in Markov Models are *transition probabilities*. The transition probability $a_{ij} = P(\omega_j(t+i)|\omega_i(t))$ is the probability that the state at time $t+1$ is ω_j given that the state at t was ω_i. The transition probabilites need not be symmetric, and states may be visited more than once. In many cases, the states $\omega(t)$ are not accessible directly; only *visible symbols* can be measured. This leads to the concept of *Hidden Markov Models*, where at each time t the system is in state $\omega(t)$ and emits a symbol $v(t)$. Therefore, we have a probabily $b_{jk} = P(v_k(t)|\omega_j(t))$, which is the probability that symbol $v_k(t)$ is emitted given that the system is in state $\omega_j(t)$. The term *hidden* refers to the fact that the true states ω_j cannot be accessed directly and are hidden from observation.

It is possible to determine the probabilities a_{ij} and b_{jk} given the structure of a HMM (i.e. the number of states and visible symbols) and a number of training samples by using an iterative approach that updates the estimates for the probabilites so that they fit the training samples more closely (i.e. the training samples are more likely to have been generated by the HMM). A detailed explanation of Hidden Markov Model training can be found in [DHS01, pp. 137–138].

4.1.6 Nonparametric Methods

Parameter estimation techniques rely on a knowledge or reasonable assumption of the underlying probability structure of the conditional probability densities. There are at least two possible cases where this does not work:

1. There might be no knowledge about the form of the probability density functions at all.

2. The densities might be multimodal (i.e. have more than one local maximum), while most theoretical parametric densities are unimodal. The normal density discussed in Section 4.1.4 has its single local maximum at the mean $\boldsymbol{\mu}$, for instance.

In these cases, maximum-likelihood estimation as described above does not work. There are several alternatives, depending on the structure of the

data and our knowledge of that structure. For instance, if we know that the distribution is a combination of several unimodal densities for which we know the functional form, the parameters can be estimated by using a technique called *Expectation-Maximisation* (see [DHS01, pp. 124–128]).

If we have do not know anything about the model behind the data, the classifier must be trained based on the information from the training samples alone. Such nonparametric estimation techniques can be divided into two groups, (1) estimating the density functions $p(\mathbf{x}|\omega_i)$[3], and (2) estimating the posterior probabilities $P(\omega_j|\mathbf{x})$ directly.

The Nearest-Neighbour Rule

One of the most widely used approaches to estimate the posterior probability is the *nearest-neighbour rule*. The labelled training data form a set of prototypes $\mathcal{D} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$. If we want to classify a point $\mathbf{x}$, the nearest-neighbour rule states that it is assigned the same class label as $\mathbf{x}' \in \mathcal{D}$, where $\mathbf{x}'$ represents the prototype which is nearest to $\mathbf{x}$.

The k-Nearest-Neighbour Rule

The *k-nearest-neighbour rule* is an extension of the nearest-neighbour rule outlined above. According to this rule, the labels associated with the k nearest prototypes of a point $\mathbf{x}$ are examined, and the relative frequencies of the labels are taken as $P(\omega_i|\mathbf{x})$. This is necessary since the neighbour points may not all have the same label. $\mathbf{x}$ is then assigned the label associated with the class ω_i with the greatest $P(\omega_i|\mathbf{x})$, which is the label that is most frequent among its k neighbours. The challenge here is to find the best value for k: The neighbours that are used for the estimate should be close to $\mathbf{x}$ so that $P(\omega_i|\mathbf{x}') \approx P(\omega_i|\mathbf{x})$, but the value of k must be large enough to get a reliable estimate.

4.1.7 Distance Metrics

Algorithms such as the nearest-neighbour rule rely on computing the distance of feature vectors in feature space. A common distance measure is the Minkowski metric

$$d_p(\mathbf{x_i}, \mathbf{x_j}) = \sqrt[p]{\sum_{k=1}^{d} (|x_{i,k} - x_{j,k}|)^p} = \|\mathbf{x_i} - \mathbf{x_j}\|_p. \tag{4.10}$$

The most widely used distance metric for continuous features is the *Euclidean distance*, which is commonly used to calculate the distance between

[3]This method is not discussed here any further; refer to [DHS01, pp. 161–177] for details.

objects in 2D or 3D space. The Euclidean distance is a special case of the Minkowski metric with $p = 2$:

$$d_2(\mathbf{x_i}, \mathbf{x_j}) = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2} = \|\mathbf{x_i} - \mathbf{x_j}\|_2 \tag{4.11}$$

There are many other ways to calculate the similarity between feature vectors, a discussion of which would be beyond the scope of this thesis.

4.2 Audio Classification Examples

The following section gives some examples from the audio signal classification domain; the purpose is to show how different classifiers are used for different classification problems.

4.2.1 Nearest Feature Line

An audio classification and retrieval system that uses the nearest feature line (NFL) method was described by Stan Z. Li in [Li00]. NFL is a nonparametric classification technique that works similar to the nearest-neighbour rule, with one important difference: Instead of comparing a test vector $\mathbf{x}$ with each of the prototypes individually, NFL uses information from multiple prototypes at once. This is achieved as follows: A *feature line* is the line passing through two prototypes $\mathbf{x_i}, \mathbf{x_j}$, $i \neq j$, for a class. The feature lines for all possible pairs of prototypes for each class are generated. The classification is then done by using the minimum distance between the point $\mathbf{x}$ and the feature lines.

Nearest feature line classification has been found to perform significantly better than nearest-neighbour classification. Li obtained an error rate of 9.78% for NFL and 13.94% for NN, using the same set of features. This can be explained by the fact that the feature line can be understood as a generalisation of the two prototypes it connects, representing an unlimited number of possible prototypes that lie on that line.

4.2.2 Tree-based MMI Vector Quantiser

Jonathan T. Foote has developed a system for the content based retrieval of audio [Foo97a] that uses tree-based quantisation[4]. The same classifier has also been used to distinguish between speech and music [Foo97b].

The system proposed by Foote uses a supervised tree-based vector quantiser trained to maximise mutual information (MMI). The feature space is

[4]Tree-based quantisation is related to k-means vector quantisation, which is an unsupervised learning technique. Tree-based quantisation is supervised, and it scales better to high dimensionality.

automatically partitioned into discrete regions by a decision tree; the decisions are taken according to a threshold that maximises the mutual information $I(\mathcal{X};\mathcal{C})$ between the training data set $\mathcal{X} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ and the associated classes $\mathcal{C} = \{\omega_1, \ldots, \omega_c\}$. Mutual information is defined as the difference in entropy

$$I(\mathcal{X};\mathcal{C}) = H(\mathcal{C}) - H(\mathcal{C}|\mathcal{D}). \tag{4.12}$$

The splitting process is repeated recursively until a stop criterion is met. At this point, the feature space is divided into a number of regions, each of which corresponds to a leaf of the tree. The tree is then used to generate a template using a histogram of leaf probabilities. This is necessary since one second of audio might be split up into n sections by appropriate windowing, meaning that n feature vectors are generated, resulting in n leaf labels. These are tracked in a histogram: if i of the vectors are classified as leaf j, then leaf j is given a value of i/n in the histogram. The resulting histogram can be used as a reference template for the probability distribution, against which unknown data can be matched.

4.2.3 Hidden Markov Models

A hierarchical system for the audio classification and retrieval using Hidden Markov Models was proposed by Tong Zhang and C.-C. Jay Kuo in [ZK98b]. Their system consists of a coarse-level and a fine-level classification step; HMMs are used in the latter to represent timbre and rhythm. Timbre is expressed by the states of a Hidden Markov Model, while the rhythm information is denoted by the duration and transition parameters.

4.2.4 Others

Many other classifiers have been applied to audio signal classification problems, and all of them have their advantages and disadvantages. In many cases, a classifier that is well-suited to one specific problem will not perform well in another. For instance, Hidden Markov Models are used in virtually all speech recognition systems (see Section 2.3.1), but some researchers doubt their usefulness in music classification. Other, perhaps more flexible techniques, are often suggested, such as neural networks [SSW98] or the Fisher Kernel method [MR00, MK98]. Another interesting approach is to use unsupervised learning: To just state one example, Andreas Rauber and Markus Fruhwirth developed a system for music style recognition that uses a self-organizing map [RF01].

Chapter 5

Prototype

Based upon my evaluation of various approaches to music classification, I developed a prototypical music genre recognition system, drawing conclusions from what I consider to be the respective flaws and merits of these approaches. The software is called MUGRAT[1], and it is available as Free Software under the GNU General Public License (GPL)[2]. The full source code for MUGRAT can be obtained from the official webpage at http://kyrah.net/mugrat/. This site will also host the full API and user documentation and precompiled binaries for various platforms as soon as the system reaches a stable 1.0 version. I intend to continue working on MUGRAT, finally developing it into a fully functional end-user system. More information about the future of MUGRAT is given in Section 7.1.

5.1 Underlying Principles

MUGRAT is based on a number of principles. In some respects, these ideas are in contrast to common approaches in machine listening. Nevertheless, I have reason to believe that the principles described below are a good foundation for a music genre recognition system. To my mind, there are too many tacit assumptions in current music content analysis systems, and in a way MUGRAT is also a criticism of these assumptions. A paper of great interest in this context is [MSV98], which also questions many of the current approaches, and presents a number of case studies showing the importance of working with complex real audio signals.

All of the assumptions I made are somehow connected, and may be summarised informally as *assume as little as possible, and deal with the real situation*. Since each of them represents a contrast to an idea generally used in music classification, I will discuss them separately in the following section.

[1] MUGRAT is an acronym for **Mu**sic **G**enre **R**ecognition by **A**nalysis of **T**exture.
[2] http://www.fsf.org/licenses/gpl.txt

The system has to work with real-world data.

Many systems are designed to work with extremely simplified test cases that do not resemble the real situation. However, music is very complex. A system that is meant to be actually usable therefore *must* work with real data, i.e. arbitrary polyphonic musical signals. I do not doubt that test systems designed for an idealised laboratory setup can be very useful to explore one specific aspect of a problem. However, I am convinced that the complexity of real music cannot be tackled by using overly simplified systems, not even by a combination of such systems. This complexity is a fundamental and important property *in itself*, and the system must be designed from ground up to deal with it. Therefore, instead of using "toy data", the MUGRAT prototype works on real music, i.e. samples obtained from the author's regular music collection.

The system must not rely on the skills of expert listeners.

Some parts of music psychology are based on features that can only be perceived by expert listeners, who have been trained in music perception for years, such as musicians and graduate music students. It is not valid to assume these abilities in average humans. For instance, most untrained listeners do not perceive the advanced musical structures that are often used in music classification systems. They cannot determine the key of a music piece, or clearly perceive pitch similarity. This does not mean that these people have no musical abilities. On the contrary, naive listeners can extract a lot of information from music. Apparently, music understanding is possible without the use of advanced concepts.

I believe that a music genre recognition system must not rely on features that are not accessible to average human listeners. Even if such information was useful (and I am not convinced that this is the case), it would distort the results obtained by the system in comparison with the average listener's perspective[3].

Music theory is not necessary for music genre recognition.

Music theory is a scientific field that studies the structure of music. It might seem logical to use the findings from this field for music genre recognition. After all, if there is a theoretical way to describe music, why not use it? Initially, I intended to do this. There are two reasons why I changed my mind about the importance of music theory. Both of them are related to how human listeners understand music.

Firstly, in music theory, high-level information such as the sequence of repeating patterns, the overall structure of the song, etc. is very important.

[3]It has to be kept in mind that these naive listeners are going to be the majority of the user base for a music genre recognition system.

A model is abstracted from the actual sound data, and this model is used instead. Human listeners do not rely on this kind of information for classification. Most of us do not have any theoretical knowledge about music, nor do we need it to recognise genres.

Secondly, most parts of music theory are concerned with the piece of music as a whole, or definable parts of it. Humans do not need to listen to a whole song in order to classify it. We do not need to split it into segments, either. A short excerpt is enough for a human listener to accurately determine the genre of a song[4]. This implies that the information can be extracted from no more than a short sample, which is what the MUGRAT system tries to do as well.

Transcription is not necessary for music genre recognition.

According to our knowledge about the workings of the human sound perception system (see also Section 2.2.1), we do not hear individual notes. On the contrary, we often hear groups of notes (chords) as single entitites. Music transcription is a hard and unsolved problem for precisely the same reason: If several notes are played at once, it is difficult to find the individual pitches. I believe that there is no point in trying to transcribe the music signal before analysis. Instead, the sound signal itself should be used as the input to the feature extraction system.

Our understanding of human perception is very limited.

The arguments listed above seem to favour a system that tries to model the human perception system. In a way, this is true: I want to avoid as many abstract concepts as possible, and use the same input that is also used by humans, namely only the musical sound signal itself.

I am, however, aware that our present understanding of the human perceptual system is very limited, that our knowledge of how we perceive sound can at best be regarded as a working model, and that it is dangerous and error-prone to make assumptions based on this model. As explained in Section 3.3, I am very suspicious of existing so-called *perceptual* systems. I believe that we do not have enough understanding of the human perceptual system in general, and the sound perception mechanisms in particular, to model them accurately.

Therefore, MUGRAT does not concentrate on specific details that might, or might not, be important for certain aspects of our perception. It tries to use features that are likely to be important in human music genre recognition, but it does not try to simulate *how* these features are important to us.

[4]Refer to Section 2.2.2 for a discussion of a study about human music genre classification accuracy.

Genres can best be defined in terms of typical members.

Music genres are hard to define. The easiest way to think of a music genre is in terms of a number of typical songs or artists. For instance, I would have trouble describing what *EBM*[5] is. If asked to describe it, I would probably say, *"music that sounds like Nitzer Ebb, DAF, Dupont and so on"*, and provide example songs. I believe that this is not an individual thinking pattern of mine, but has its roots in the nature of music genres itself. Music genres are so hard to describe because there *is* no simple set of features that define each genre.

Making any assumptions about the probability distribution of the genres would be akin to trying to describe the genres, and I doubt the viability of this approach. Instead, I think that the best we can do is to find a number of representative examples for each genre, and use these for comparison. Classification then means looking for songs that sound like the song to be classified, just like human listeners do it. The genre label that is associated with the majority of these songs should be attributed to the new song. This is why I chose a non-parametric classification method for MUGRAT, namely k-nearest-neighbour classification. (The classification subsystem is described in Section 5.3.)

5.2 Feature Extraction Subsystem

The MUGRAT feature extractor reads input music files and calculates a number of features that allow the classification subsystem to distinguish the genres. The choice of features follows the principles outlined above, and will be explained in detail later in this section. Before doing this, I would like to make a couple of general comments about the history of the feature extraction system used in MUGRAT.

5.2.1 History/Background

Following the idea that all information needed for music genre recognition is contained in the sound signal itself, I initially intended to base MUGRAT on spectral attributes alone. My naive approach was to calculate the FFT coefficients for a number of frequency bands, and use these for the classification. I soon realised that this idea is bound to fail, since (1) the amount of data per song is too large and (2) the temporal influence is too strong. A salient event (such as a downbeat) occuring in the beginning of the sample would be represented differently in this system than if it occurred in the middle or the end. This is clearly not a good solution.

The next idea that occurred to me was to use the arithmetic mean over all windows for each frequency bin. This reduced the dimensionality to the

[5] *Electronic Body Music*

number of frequency bins, but it also blurred the results. Assuming that essential information is contained in short-time spectral change, simply using average energy values does not seem like a satisfying solution.

I began to look for alternative features, and discovered the set of features proposed by George Tzanetakis in [TEC01]. The assumptions underlying George Tzanetakis' work[6] are very similar to mine, and the final feature extractor used in MUGRAT is based on the features described in the cited paper.

There are two types of features used in MUGRAT: features related to the musical surface of the song, and features related to the beat of the song.

5.2.2 Musical Surface Features

Humans can determine the music genre of a song from a very short sample, as was shown in the study discussed in Section 2.2.2. In such a short time, the only possible indication about the music genre has to be derived from the spectral distribution over time, from the instrumentation, the timbre, and the musical texture. A term often used in connection with these features is *musical surface*, and even though there is no good definition for it, I will – for lack of a better term – use it too. What exactly is meant by musical surface in the current context will become clearer through an examination of the actual features used in MUGRAT.

Feature 1: Spectral Centroid

The spectral centroid is the balancing point of the spectrum. It is a measure of spectral shape, and is often associated with the notion of *spectral brightness*. The spectral centroid can be calculated as

$$C = \frac{\sum_{n=1}^{N} M_t[n] \cdot n}{\sum_{n=1}^{N} M_t[n]}, \tag{5.1}$$

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

Feature 2: Rolloff

Like the centroid, rolloff is also a measure of spectral shape. It is defined as the frequency R corresponding to $r\%$ of the magnitude distribution, so that

[6] George Tzanetakis has released a general and very powerful Free Software framework for music analysis, which is called MARSYAS. It is available online at http://www.cs.princeton.edu/~gtzan/marsyas.html.

$$\sum_{n=1}^{R} M_t[n] = r \cdot \sum_{n=1}^{N} M_t[n]. \tag{5.2}$$

It can be seen as a generalisation of the spectral centroid; the spectral centroid is the rolloff for $r = 50\%$. In the MUGRAT prototype, a rolloff value of $r = 80\%$ is used.

Feature 3: Flux

Flux is a measure of local spectral change, and it is defined as

$$F = \sum_{n=1}^{N} (N_t[n] - N_{t-1}[n])^2, \tag{5.3}$$

where $N_t[n]$ is the normalised magnitude of the Fourier transform at window t.

Feature 4: Zero-crossing Rate

A zero-crossing occurs when successive samples in a digital signal have different signs. The zero-crossing rate is a simple measure of the noisiness of a signal. It can be calculated as

$$Z = \sum_{n=1}^{N} |s(x[n]) - s(x[n-1])|, \tag{5.4}$$

where $x[n]$ is the time domain signal, and s is a function that is 1 for positive arguments and 0 for negative arguments. Unlike spectral centroid, rolloff, and flux, which are frequency-domain features, the zero-crossing rate is a time-domain feature.

Mean and Variance

Following the suggestions by George Tzanetakis, these feature values are not used directly. The notion of musical surface is created by a series of spectral states that vary slightly over a short period of time. To capture this important information, the mean and variance of the features listed above are calculated over a number of windows. A value between 35 and 45 windows[7] turned out to provide the best results here.

[7]The final prototype uses 40 windows.

Feature 5: Low Energy

There is one additional feature, called *low energy*. This, too, is calculated over a number of windows like the mean and variance, and not separately for each window like the other features. The low energy feature is defined as the percentage of windows that have less energy than the average energy of all 40 windows. Music that contains silent parts will have a larger low energy value than continuous sounds.

This results in a nine-dimensional feature vector consisting of (1) mean-centroid, (2) std-centroid, (3) mean-rolloff, (4) std-rolloff, (5) mean-flux, (6) std-flux, (7) mean-zerocrossing, (8) std-zerocrossings, (9) low-energy.

5.2.3 Beat-Related Features

The beat and rhythmic structure of a song is often a good indication of the genre. For instance, Dance songs tend to have a very strong and distinctive main beat. Classical music, on the other hand, generally does not have a clear dominant and regular beat, due to the complexity of the arrangement. Like all beat detection systems, the MUGRAT beat feature extractor tries to find the main beat of the song and its period in BPM (beats-per-minute). In addition to this, MUGRAT also calculates the second-strongest beat, and a number of features concerning the relationship between the first and second beat. The feature set used in MUGRAT was originally proposed by George Tzanetakis, so more detailed information can also be found in [Tza02].

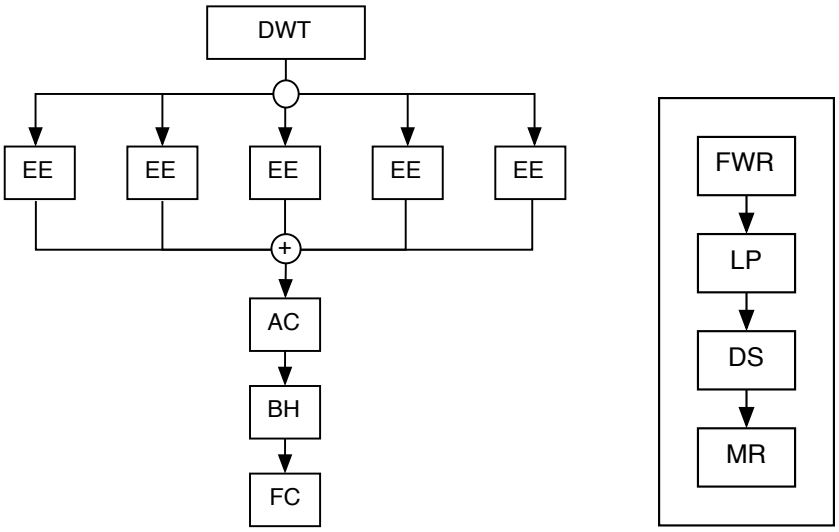
The MUGRAT beat extraction system is explained in the rest of this section. An overview of the individual steps is shown in Figure 5.1.

Wavelet Decomposition

First, the signal is decomposed into a number of frequency bands using the discrete wavelet transform (DWT). The wavelet transform is an alternative to the short-time Fourier transform. It was developed as a solution to the STFT's resolution problems[8], and can be used for decomposing a signal into a number of octave-spaced frequency bands.

A full discussion of wavelet techniques would be beyond the scope of this thesis. Implementation details can be found in the MUGRAT source code. For general information about the DWT, refer to [SP93] and [Coi90]. A more practical tutorial on how to use wavelet techniques is given in [Ngu95].

[8] It provides high time resolution and low frequency resolution for high frequencies, and low time resolution and high frequency resolution for low frequencies.



(a) Beat feature extraction module (b) `EE`

Figure 5.1: An overview of the steps necessary to extract the beat-related features in MUGRAT. The individual steps are: discrete wavelet transform (`DWT`), envelope extraction (`EE`), autocorrelation (`AC`), beat histogram calculation (`BH`), feature calculation (`FC`). The envelope extraction (`EE`) step, shown in detail in the figure on the right, consists of full wave rectification (`FWR`), low-pass filtering (`LP`), downsampling (`DS`), and mean removal (`MR`).

Envelope Extraction

After this decomposition, a series of steps for the extraction of the time domain amplitude envelope is applied to each band. These steps are full wave rectification, low pass filtering, downsampling, and mean removal.

Full Wave Rectification

$$y[n] = |x[n]| \tag{5.5}$$

Full wave rectification is a standard envelope extraction step, usually followed by a low pass filter. It is used to convert the time domain signal into its temporal envelope.

Low Pass Filtering

$$y[n] = (1 - \alpha)x[n] + \alpha y[n - 1] \tag{5.6}$$

For low pass filtering, a One Pole filter[9] with an α value of 0.99 is used. It re-

[9] A One Pole filter is a filter with the characteristics given by Equation 5.6.

sults in a smoother envelope, and is also a standard envelope extraction step.

Downsampling

$$y[n] = x[kn] \tag{5.7}$$

Downsampling is not an algorithmic requirement. It is applied to reduce the computation time for the autocorrelation step. The loss of precision is presumed to be negligible due to the large periodicities used in beat analyis. MUGRAT uses a value of $k = 16$.

Mean Removal

$$y[n] = x[n] - E[x[n]], \tag{5.8}$$

where $E[x[n]]$ is the expected value of $x[n]]$. This step centers the signal to zero. It is necessary for the autocorrelation step.

Autocorrelation and Beat Histogram Generation

After the envelope extraction step, the envelopes of each band are summed together, and the autocorrelation of the resulting envelope is calculated.

$$y[k] = \frac{1}{N} \sum_{n} x[n]x[n-k] \tag{5.9}$$

The result is an autocorrelation function where the dominant peaks correspond to the time lags where the signal has the strongest self-similarity. The actual autocorrelation function used in MUGRAT is an enhanced version of the standard autocorrelation function, following the ideas proposed in [TK00].

The first three peaks of the autocorrelation function are added to a beat histogram. Each bin in the histogram corresponds to a beat period in BPM. For each of the three selected peaks, the peak amplitude is added to the histogram. This is repeated for each analysis window. The strongest peaks in the final histogram correspond to the strongest beats in the signal.

Feature Calculation

MUGRAT uses a six-dimensional feature vector to model various features that are calculated from the beat histogram. The six features are:

- The relative amplitude (i.e. the amplitude divided by the sum of amplitudes) of the first and second peak in the beat histogram. This a measurement of how distinctive the beat is compared to the rest of the signal.

- The ratio of the amplitude of the second peak divided by the amplitude of the first peak. It expresses the relation between the main beat and the first subbeat. In songs with a strong, regular rhythm, this feature will be close to 2.0 or 0.5.
- The period of the first and second peak in BPM, indicating how fast the song is.
- The sum of the histogram, which is an indication of beat strength. Since the autocorrelation peaks are added by amplitude, and the amplitude is dependent on the self-similarity, the sum of the histogram bins is a measure of the strength of self-similarity between the beats, which in turn is a factor in how rhythmic a song feels.

5.2.4 Design Overview

Figure 5.2 gives an overview of MUGRAT's structure. The main classes are `FeatureExtractor`, `Signal` and `System`, and classes that are derived from these classes. A short discussion of these main classes follows. Please refer to the MUGRAT source code[10] for implementation details.

Signal

The actual data in MUGRAT are stored as vectors of floating point numbers (`vFloat` data type). The `Signal` class takes input in a specific format and passes it, converted to `vFloat[]`, to other classes that call `Signal`'s `get()` method. The classes for reading data from file are all derived from `FileReader`, which in turn is derived from `Signal`. Classes derived from `Signal` provide a number of utility methods for checking the plausibility of the data. As an example, `WavFileReader` – the class for reading WAVE input – checks if the file exists, if it is not empty, if the magic number in the header is correct, if the sampling rate is 8 bit or 16 bit, etc.

System

`System` is a very general class that takes an input vector and produces an output vector. The dimensionality of input vector and output vector may differ. The `System` can be applied by calling its `void process(vFloat& in, vFloat& out)` method. Most classes in the MUGRAT feature extraction module inherit from the `System` class. Fast Fourier Transform (`FFT`), Hamming windowing (`Hamming`), Discrete Wavelet Transform (`WaveletPyramid`, `WaveletStep`, `D4`), and all the individual features are all implemented as subclasses of `System`.

[10]The full source code is available for download at the official MUGRAT website http://kyrah.net/mugrat/download.html.

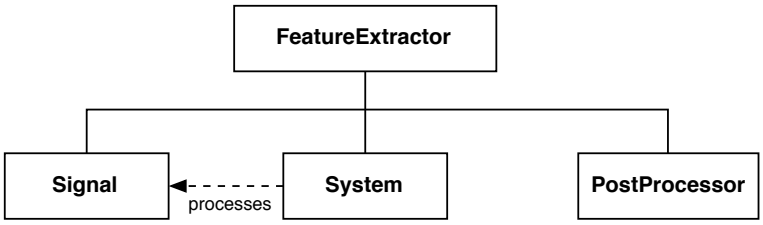


Figure 5.2: MUGRAT feature extraction module design overview.

The most interesting property of `Systems` (i.e. classes derived from the `System` class) is that they can be regarded and used as black boxes that transform an input vector into an output vector. Therefore, `Systems` can be arbitrarily combined into larger `Systems`. For instance, there is a class called `TextureFeatures`, derived from `System`. Calling the `process()` method of this class calculates a number of features related to the musical surface. It accomplishes this by calling `process()` on its data members, which are other `Systems` that represent the actual features.

FeatureExtractor

`FeatureExtractor` is a generic class that is constructed with a `Signal`, a `System` and a `PostProcessor`. The `System` and `PostProcessor` define the feature set that is examined by the `FeatureExtractor`, and the `Signal` contains the input data.

The `FeatureExtractor` iterates over the `Signal`'s windows, applying the `System` to each frame. The `Postprocessor` is then applied to the output of this process, resulting in the final feature vector. This feature vector can then be written to a disk file by using the `<<` operator of the `vFloat` class.

5.3 Classification Subsystem

The actual classification is done using WEKA, the *Waikato Environment for Knowledge Analysis*. WEKA is an excellent software framework developed by a project team from the University of Waikato, New Zealand. It contains a collection of standard machine learning techniques, and is released under the GNU General Public License (GPL). The official website for WEKA, which contains the full source code for the library and extensive documentation, is http://www.cs.waikato.ac.nz/~ml/weka/index.html.

WEKA expects the input data to be in ARFF (*Attribute-Relation File Format*) files. ARFF files are ASCII text files that contain two distinct sections: a Header section and a Data section. The Header contains the name of the relation, a list of the attributes, and their types. The list of attributes

is ordered, and indicates the attribute's position (column number) in the Data section of the file. The type can be either numeric (real or integer), an arbitrary string, or a nominal value (one of a list of possible values). More information about ARFF files can be found in [Tea02].

The ARFF definition for MUGRAT is as follows:

```
@RELATION genres

@ATTRIBUTE MeanCentroid REAL
@ATTRIBUTE MeanRolloff REAL
@ATTRIBUTE MeanFlux REAL
@ATTRIBUTE MeanZerocrossings REAL
@ATTRIBUTE StdCentroid REAL
@ATTRIBUTE StdRolloff REAL
@ATTRIBUTE StdFlux REAL
@ATTRIBUTE StdZerocrossings REAL
@ATTRIBUTE LowEnergy REAL
@ATTRIBUTE RelAmplitude1 REAL
@ATTRIBUTE RelAmplitude2 REAL
@ATTRIBUTE AmplitudeRatio REAL
@ATTRIBUTE BPM1 REAL
@ATTRIBUTE BPM2 REAL
@ATTRIBUTE Sum REAL
@ATTRIBUTE class {Metal,Dance,Classical}

@DATA
```

The ARFF Data section is generated by the MUGRAT feature extractor. It consists of one line per song that contains the extracted features in the order specified in the ARFF header, separated by ','. A typical entry for a song would be for instance (all in one line):

```
15.0061,17.9659,157.074,17.9604,2.65913,3.1901,111.073,4.88643,
0.0700521,0.137994,0.123342,0.893821,234,159,1.65178,Classical
```

WEKA contains a k-nearest-neighbour classifier implementation called `IBk` in the `weka.classifiers` package. The system performs automatic input normalisation, and can try a range of nearest neighbours k in order to determine the value which yields the best classification result. A detailed discussion of the results I obtained is provided in Chapter 6.

5.4 Limitations

The prototype is intended to serve as a *proof of concept*, showing that it is possible to automatically determine the music genre of a song according to the principles listed in Section 5.1. As a demonstration implementation, the system naturally has several limitations. The most notable ones are listed in the rest of this chapter.

Limited Number Of Genres

Firstly, and perhaps most imporantly, the MUGRAT prototype uses only three genres: Classical, Metal and Dance. I chose these three genres because they are relatively well-defined and in most (though not all) cases, a human listener can easily categorise a typical song from one of them.

A real system will need to use a much more sophisticated genre topology, probably using a hierarchical structure of genres. In this context, it may be advisable to have specialised feature extraction subsystems for the various genre subtrees. The classification is then done in two steps, (1) performing a coarse-level classification into rough categories, and (2) doing a fine-grained classification. The first part is comparable to the functionality of the MUGRAT prototype. The second step uses methods that are specifically designed for and adapted to the genre subset. Presumably, this approach makes a very detailed and accurate classification possible, since sub-genre-specific features can be taken into account.

No Special Handling of Ambiguous Input

The prototype does not handle *borderline cases* well. For instance, if the system tries to classify a certain song and finds that the probability that it belongs to the Metal genre is 50.1%, and the probability that it is a Classical song is 49.9%, the system will label the song as Metal.

An easy solution to this problem is to do a fuzzy classification instead of deciding on a single label. The output in the example above would then be *Metal:* 50.1%, *Classical:* 49.9%, clearly telling the user that the song in question contains strong elements from both genres. Unfortunately, this approach also has an important drawback. ID3 and other systems for handling meta-data expect a decision on one genre label. Therefore, not deciding on a genre (but using relative percentages) makes some input from the user – i.e. a final decision – necessary to be able to use MUGRAT for automatic classification in connection with such systems. In the MUGRAT prototype, a decision on one genre label is taken, but as additional information, the probability associated with it (i.e. the system's confidence in the predicition) is output as well.

There is another interesting aspect to this problem: Should it be possible for the system to not make a classification at all? This would be a typical human behaviour. If we are asked a question that we are unable to answer, we say that we are not sure, rather than just making the most likely guess. It might make sense to allow the system to leave songs unclassified, if the probability values resulting from evaluating the features are below a certain threshold.

File Format and Platform Support

Currently, MUGRAT can read only 8 bit and 16 bit WAVE Files. Adding support for other input formats such as Ogg Vorbis or MP3 should be straightforward, though, since the input to the MUGRAT feature extractor are just windows of floating point vectors. For instance, implementing an MP3 reader will consist in extending the `FileReader` class, so that the subclass decodes the MP3 file and passes the output to the feature extractor as floating point values when requested.

MUGRAT currently works only on little-endian architectures. This is not a limitation of the feature extraction or classification system itself, but only of the WAVE file input class. Adding big-endian support will be trivial to do. MUGRAT has been developed on GNU/Linux, but a port to Windows and Mac OS X is planned.

Chapter 6

Experimental Results

6.1 Test Data

General Information

The MUGRAT prototype uses three different genres: Metal, Dance, and Classical[1]. I collected 189 test songs from the three genres: 63 Metal songs, 65 Dance tracks, and 61 Classical pieces[2]. The songs were all taken from regular music collections, using a sample rate of 44.1 kHz and a bit rate of 128 or 160 kbps. These are common settings for non-professional purposes. They were chosen according to the principle that MUGRAT should work with real music data.

Song Selection

Special attention was paid to having enough variance in the selection. For instance, the Metal collection contains songs from the Trash Metal, Speed Metal, Classic Metal, Hardrock, Gothic Metal, Black Metal, Punk, Grunge, and Death Metal categories. The same is true for the Dance genre, which covers a spectrum from Eurodance to Rave. The input data for the classical genre is somewhat more limited. This is due to the fact that classical music is based on different concepts than popular music. It requires the listener to actively pay attention in order to fully appreciate a piece. A four-year-old child without any understanding of the musical structures used in classical music can only make a rough distinction between *violin*, *piano*, *orchestral*, or *vocal music*. This highly oversimplifying viewpoint is also taken by MUGRAT – the system is not able to hear the difference between e.g. a piano piece by Frédéric Chopin and one by Robert Schumann.

[1] These genre names are intentionally unspecific. They pertain to the ideas that average music listeners have about the respective kind of music, rather than the definitions used by the music industry.

[2] A listing of the songs can be found in Appendix A.

Influence of Lossy Compression

The input files were all in MP3 format[3]. One of the main applications for automatic music genre classification will be to structure music collections obtained through the Internet. It is common to use compressed file formats for online distribution of audio data, such as Ogg Vorbis or MP3. In most of these cases, the compression is lossy; the goal of such compression algorithms is to make it impossible for a human listener to hear the difference. Nevertheless, the sound signal obtained by decoding an MP3 file is not exactly the same as the original signal. To music genre recognition software, which does not follow the perceptual criteria that such encodings take advantage of, the two signals do sound different. By using compressed input to train the classifier, it can be made sure that the system does not rely on features that are only present in the original, uncompressed sound file.

6.2 Test Procedure

Feature Extraction

From each file, an randomly chosen sample of three seconds length was used as input to the feature extraction module. The idea that three seconds are enough for accurate classification is based on a study of human music genre classification[4], and is one of the key principles of MUGRAT. A perl script was used to automatically process all input files.

Classification

The ARFF file that was generated by the feature extractor, and contained the feature representation for all music files, was used to train a k-nearest-neighbour classifier[5], weighting the influence of the neighbours by the inverse of their distance to the test instance. This classifier was evaluated using stratified ten-fold cross-validation. *Cross-validation* is a standard evaluation technique in pattern classification, in which the dataset is split into n parts (*folds*) of equal size. $n-1$ folds are used to train the classifier. The n^{th} fold that was held out is then used to test it. This is done n times, once for each fold. The figures are added to generate the final statistics. *Stratified* means that it is guaranteed that the different classes are equally distributed in the test and training sets. Stratified cross-validation is useful for getting reasonable estimates on future performance of the classifier, while only using the available training data.

[3] Note: Since MUGRAT currently supports only WAVE input, the actual input was generated by decoding the MP3 files.

[4] See Section 2.2.2 for a discussion of the study in question.

[5] A complete listing can be found in Appendix B.

6.3 Test Results

Using 3 nearest neighbours for the classification ($k = 3$), a classification accuracy of 88.3598% was reached in ten-fold stratified cross-validation. This is comparable to human music genre recognition capabilities. The confusion matrix for $k = 3$ is shown in figure 6.3.

The performance of the classification was slightly worse for other values of k. $k = 4$ resulted in 86.2434% and $k = 2$ in 84.127%.

Metal	Dance	Classical	
56	6	1	Metal
13	50	2	Dance
0	0	61	Classical

Figure 6.1: MUGRAT genre classification confusion matrix

6.4 Usage Example

The following example assumes that a classifier has been trained as shown in Appendix B. The working directory is */Users/kyrah/mugrat/*. The example song is recorded in mono with a bit rate of 128 kbps and a sampling rate of 44.1 kHz.

The system can be used to classify a new song *Shalala (La La)* as follows:

(1) Extracting the Features

Run the feature extractor:

```
mugrat "Shalala (La La).wav"
```

This generates a file called `Shalala (La La).arff`, which contains the feature representation of the song *Shalala (La La)*.

(2) Classifying the Song

Determine the genre of *Shalala (La La)*:

```
> java weka.classifiers.IBk -p -l /Users/kyrah/mugrat/genre_knn_classifier
-T "/Users/kyrah/mugrat/Shalala (La La).arff"

0 Dance 0.724637682530215 Unclassified
```

This command reconstructs a KNN classifier from the model saved in the file `genre_knn_classifier`. It uses the ARFF file generated in the previous step as test data, specified by the `-T` option. The option `-p` tells WEKA to output information about the individual data items – in our case, the single data item *Shalala (La La)* – instead of displaying only statistics about the classification performance, which is the default.

The first entry in the output is the song number. (MUGRAT can also be used for batch processing.) The second entry is the system's prediction of the genre, in this case *Dance*, and the third entry is the confidence of the classification. This value reflects the posterior probability for the predicted genre. The fourth entry is the known genre of the song, and is mainly for testing and debugging purposes. In the case of a new song, it is set to *Unclassified*.

Chapter 7

Conclusions

7.1 Where Do We Go From Here?

The prototype described in Chapter 5 will be used as the basis for a complete implementation of an automatic music genre recognition system. The goal of this project is to develop a production-level system for end-users, preferably with an easy-to-use graphical user interface in addition to the command-line version. The system will be fully modular, so that the genre hierarchy can be easily expanded in width and depth. It will also be possible to use alternative feature extraction methods and classifiers through plug-ins. More information about this system can be found on the official project website http://kyrah.net/mugrat/. Some of the most interesting things to look at in the near future are:

Extend the Genre Topology

The genre topology should be extended both in width and depth. This is discussed in more detail in Section 5.4.

Additional Features

In addition to the features used by MUGRAT, there are many other possible feature combinations that might improve the classification results, such as pitch-based features. It might also be interesting to use MFCCs (see Section 3.4.2) instead of the STFT-based spectral features.

Different Classifiers

MUGRAT uses KNN classification. While it is one of my fundamental assumptions that non-parametric methods are superior to parametric approaches in music genre recognition, it might be interesting to try out other, more advanced classification techniques.

Segmentation

At present, randomly chosen excerpts from the songs are used for the classification. It might be useful to segment the songs to make sure that the sample does not include a significant change in musical texture.

Real Time Classification

Some features, such as the features based on the music surface, can probably also be used for the classification of music genres in real time.

Direct Handling of Ogg/MP3 Input

Some popular formats for the distribution of music files are based on spectral transforms that are similar to the steps used in music genre recognition. Using e.g. MP3-encoded files directly could greatly increase the feature extraction speed.

7.2 Contributions to the State of the Art

The work presents a comprehensive critical overview of music genre recognition, including disquisitions on the necessary background. It fills a gap in the existing literature, which either focusses on signal processing and music perception, or on the pattern classification aspect of the problem. Readers with a background in sound analysis get a thorough introduction into classification theory and practice, and researchers from the field of pattern recognition find a discussion of music perception, Fourier analysis, and features that are important for music classification.

The state of the art in music genre recognition, and machine listening in general, is evaluated in a critical way. Widely-used assumptions and foundations are challenged, such as the use of concepts from music theory and features that can only be perceived by music experts, and the distinction between physical and perceptual features. It is explained why non-parametric classification methods can be presumed to be more suitable for use in music genre recognition than parameterised techniques. A set of foundations that a music genre recognition system should be based on is proposed. These foundations include the demand for working with arbitrary polyphonic signals, and using only the information available from the sound signal.

Finally, a prototype based on these principles is presented. While limited in functionality, it shows that these principles are indeed a good foundation for a music genre recognition system.

7.3 Final Remarks

At present, the musical capabilities of computers are not even at the level of a four-year-old child. There is much work to be done, many mistakes to be made, many things to be learnt.

Music is a central element in many people's lives, including mine. Teaching our computers to understand music, maybe we will come closer to understanding the magic, power, and beauty of music ourselves.

Ohne Musik wäre das Leben ein Irrtum. (Friedrich Nietzsche)

Bibliography

[AD90] Paul E. Allen and Roger B. Dannenberg. Tracking musical beats in real time. In *S. Arnold and G. Hair (editor), ICMC Glasgow 1990 Proceedings*, pages 140–143. International Computer Music Association, 1990.

[Bil92] Jeff Bilmes. A model for musical rhythm. In *ICMC Proceedings*, pages 207–210. Computer Music Association, 1992.

[BR99] Claudio Becchetti and Lucio Prina Ricotti. *Speech Recognition, Theory and C++ Implementation.* John Wiley & Sons, 1999.

[Bre01] Leen Breure. Development of the genre concept. URL, http://www.cs.uu.nl/people/leen/GenreDev/GenreDevelopment.htm, August 2001.

[Coi90] R. R. Coifman. Wavelet analysis and signal processing. In Louis Auslander, Tom Kailath, and Sanjoy K. Mitter, editors, *Signal Processing, Part I: Signal Processing Theory*, pages 59–68. Springer-Verlag, New York, NY, 1990.

[DHS01] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* John Wiley & Sons, Inc., New York, 2001.

[Dix99] Simon Dixon. A beat tracking system for audio signals. In *Proceedings of the Diderot Forum on Mathematics and Music*, pages 101–110. Austrian Computer Society, 1999.

[DNS01] Hrishikesh Deshpande, Unjung Nam, and Rohit Singh. Mugec: Automatic music genre classification. Technical report, Stanford University, June 2001.

[Ell96] Daniel P. W. Ellis. Prediction-driven computational auditory scene analysis for dense sound mixtures. Technical report, International Computer Science Institute, Berkeley CA, 1996.

[Eri99] T. Erickson. Rhyme and punishment: The creation and enforcement of conventions in an on-line participatory limerick genre.

In *Proceedings of the 32nd Hawaii International Conference on System Sciences (HICSS '99)*, 1999.

[Foo97a] Jonathan Foote. Content-based retrieval of music and audio. In *Multimedia Storage and Archiving Systems II, Proceedings of SPIE Corpora*, pages 138–147, 1997.

[Foo97b] Jonathan Foote. A similarity measure for automatic audio classification. In *Proc. AAAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora*, March 1997.

[Foo99] Jonathan Foote. An overview of audio information retrieval. *Multimedia Systems*, 7(1):2–10, 1999.

[Ger00] David B. Gerhard. Audio signal classification. Technical report, School of Computing Science, Simon Fraser University, February 2000.

[GLCS95] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. Query by humming: Musical information retrieval in an audio database. In *ACM Multimedia*, pages 231–236, 1995.

[Hac00] Scot Hacker. *MP3: The Definitive Guide*. O'Reilly & Associates, Inc., Sebastopol, 2000.

[JDM99] Anil K. Jain, Robert P.W. Duin, and Jianchang Mao. Statistical pattern recognition: A review. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, 22(1):4–37, 1999.

[JKP94] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994.

[JMF99] Anil K. Jain, M. Narasimha Murty, and Patrick J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[Job01] Steve Jobs. Macworld keynote. Keynote at the Macworld Expo, QuickTime movie available at http://www.apple.com/quicktime/qtv/mwsf01/, 2001.

[Li00] Stan Z. Li. Content-based classification and retrieval of audio using the nearest feature line method. *IEEE Transactions on Speech and Audio Processing*, 9(5):619–615, September 2000.

[LWC98] Z. Liu, Y. Wang, and T. Chen. Audio feature extraction and analysis for scene segmentation and classification. *Journal of VLSI Signal Processing System*, 20:61–79, 1998.

[Mar96] Keith D. Martin. Automatic transcription of simple polyphonic music: Robust front end processing. Technical Report 399, Massachusetts Institute of Technology, The Media Laboratory, December 1996.

[MK98] Keith D. Martin and Y. E. Kim. Musical instrument identification: A pattern-recognition approach. In *Proc. of 136th meeting of the Acoustical Society of America*, 1998.

[MR00] Pedro J. Moreno and Ryan Rifkin. Using the fisher kernel method for web audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2000.

[MSV98] Keith D. Martin, Eric D. Scheirer, and Barry L. Vercoe. Music content analysis through models of audition. In *Proc. 1998 ACM Multimedia Workshop on Content Processing of Music for Multimedia Applications*, Bristol, UK, September 1998.

[MSW96] R. McNab, L. Smith, and I. Witten. Signal processing for melody transcription. In *Proc. 19th Australasian Computer Science Conf.*, pages 301–307, Melbourne, January 1996.

[Ngu95] Truong Q. Nguyen. A tutorial on filter banks and wavelets. Technical report, University of Wisconsin, ECE Dptm., 1995.

[Nil01] Martin Nilsson. ID3v2. URL, http://www.id3.org/, December 2001.

[PFE96] Silvia Pfeiffer, Stephan Fischer, and Wolfgang Effelsberg. Automatic audio content analysis. In *ACM Multimedia 96*, pages 21–30, Boston, November 1996.

[PG99] D. Perrot and R. O. Gjerdigen. Scanning the dial: An exploration of factors in the identification of musical style. In *Proceedings of the 1999 Society for Music Perception and Cognition*, 1999.

[Pic01] Jeremy Pickens. A survey of feature selection techniques for music information retrieval. Technical report, Center for Intelligent Information Retrieval, University of Massachusetts, Amherst, MA, 2001.

[RF01] Andreas Rauber and Markus Fruhwirth. Automatically analyzing and organizing music archives. In *European Conference on Digital Libraries*, pages 402–414, 2001.

[RJ86] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

[Roa96] Curtis Roads. *The Computer Music Tutorial*. MIT Press, 1996.

[Rub01] Wilfried Rubin. Automatisierte Suche nach Audiosignalen in großen Datenbeständen. Master's thesis, Fachhochschule Hagenberg (Medientechnik und -design), Juli 2001.

[Sau96] John Saunders. Real-time discrimination of broadcast speech/music. In *International Conference on Acoustics, Speech and Signal Processing, vol. II*, pages 993–996, 1996.

[Sla98] Malcolm Slaney. Auditory toolbox, version 2. Technical Report 1998-010, Interval Research Corporation, 1998.

[Smi98] Steven W. Smith. *Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, San Diego, CA, 1998.

[SP93] W. Sweldens and R. Piessens. Wavelet sampling techniques. In *1993 Proceedings of the Statistical Computing Section*, pages 20–29. American Statistical Association, 1993.

[SSNY97] S. R. Subramanya, Rahul Simha, Bhagirath Narahari, and Abdou Youssef. Transform-based indexing of audio data for multimedia databases. In *International Conference on Multimedia Computing and Systems*, pages 211–218, 1997.

[SSW98] Hagen Soltau, Tanja Schultz, and Martin Westphal. Recognition of music types. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, Seattle, WA.*, 1998.

[Tea02] WEKA Project Team. ARFF. URL, http://www.cs.waikato.ac.nz/~ml/weka/arff.html, April 2002.

[TEC01] George Tzanetakis, Georg Essl, and Perry Cook. Automatic musical genre classification of audio signals. In *Proceedings International Symposium for Audio Information Retrieval (ISMIR)*, Princeton, NJ, October 2001.

[TK00] T. Tolonen and M. Karjalainen. A computationally efficient multipitch model. *IEEE Transactions on Speech and Audio Processing*, 8(6):708–716, November 2000.

[Tza02] George Tzanetakis. *Manipulation, Analysis and Retrieval Systems for Audio Signals*. PhD thesis, Princeton University, June 2002.

[Wei02] Eric Weisstein. World of mathematics. URL, http://mathworld.wolfram.com/, March 2002.

[ZK98a] T. Zhang and C. Kuo. Content-based classification and retrieval of audio. In *SPIE's 43rd Annual Meeting - Conference on Advanced Signal Processing Algorithms, Architectures, and Implementations VIII*, San Diego, July 1998.

[ZK98b] T. Zhang and C. Kuo. Hierarchical system for content-based audio classification and retrieval. Technical report, University of Southern California, Los Angeles, CA, 1998.

Appendix A

Playlists

A.1 Rock

1. Paradise Lost – *Say Just Words*
2. Blind Guardian – *Into The Storm*
3. Nightwish – *Elvenpath*
4. Rage Against The Machine – *Killing In The Name Of*
5. HIM – *Poison Girl*
6. Metallica – *The Unforgiven II*
7. Sepultura – *Subtraction*
8. White Zombie – *El Phantasmo And The Chicken Run Blast-o-Rama*
9. Slipknot – *Purity*
10. Nirvana – *Smells Like Teen Spirit*
11. Bad Religion – *Punk Rock Song*
12. The Sisters Of Mercy – *Vision Thing*
13. Offspring – *Get A Job*
14. The Bates – *Independent Love Song*
15. Blink 182 – *Mutt*
16. Nightwish – *Sacrament Of Wilderness*
17. Slipknot – *Me Inside*
18. Guano Apes – *Lords Of The Boards*
19. Marilyn Manson – *I Don't Like The Drugs (But The Drugs Like Me)*
20. Samael – *Together*
21. White Zombie – *I Zombie*
22. Amorphis – *Divinity*

23. Lacuna Coil – *Circle*
24. Samsas Traum – *Aber die Liebe stirbt nie*
25. Metallica – *Prince Charming*
26. White Zombie – *Super-Charger Heaven*
27. Die Boehsen Onkelz – *Falsche Propheten*
28. Rammstein – *Sonne*
29. Slipknot – *(sic)*
30. The Atomic Fireballs – *Man With The Hex*
31. AC/DC – *Highway To Hell*
32. Blind Guardian – *Mirror Mirror*
33. Iron Maiden – *Blood On The World's Hand*
34. HIM – *Bury Me Deep Inside Your Heart*
35. Dream Theatre – *Metropolis II, Chapter 2*
36. Motorhead – *Killed By Death*
37. Accept – *Restless And Wild*
38. Tristania – *Angellore*
39. Bloodhound Gang – *Along Comes Mary*
40. The Sisters Of Mercy – *Doctor Jeep*
41. Nightwish – *Wishmaster*
42. Metallica – *Carpe Diem Baby*
43. Antichrisis – *Her Orphaned Throne*
44. Weeser – *My Name Is Jonas*
45. Ministry – *N.W.O.*
46. Skinny Puppy – *Cult*
47. Ecstatic Fear – *A Sombre Dance, Chapter 3*
48. The Sisters Of Mercy – *Dominion*
49. Green Day – *Basket Case*
50. U2 – *Who's Gonna Ride Your Wild Horses*
51. Die Ärzte – *Alleine In Der Nacht*
52. Bloodhound Gang – *Firewaterburn*
53. Tristania – *Wasteland's Caress*
54. Rammstein – *Links 2 3 4*
55. Paradise Lost – *Blood Of Another*

56. Ozzy Osbourne – *I Just Want You*
57. Darkness – *Dead Squad*
58. Tyrant – *Up The Hammer*
59. Stiltskin – *Inside*
60. Slipknot – *Prosthetics*
61. Nightwish – *The Riddler*
62. Blind Guardian – *Nightfall*
63. Molotov – *Volatino*

A.2 Dance/Techno

1. RMB – *Experience*
2. Capella – *U Got 2 Let The Music*
3. 2 Unlimited – *Let The Beat Control Your Body*
4. BBE – *7 Days And 1 Week*
5. Chicane – *Offshore*
6. Eiffel 65 – *Blue*
7. Space – *Magic Fly*
8. General Base – *Poison*
9. Future Breeze – *Keep The Fire Burning*
10. Dance 2 Trance – *Power Of American Natives*
11. Cult Of Bass – *Endless Love*
12. Chemical Brothers – *Hey Boys Hey Girls*
13. Bedrock Band – *(Meet) the Flintstones*
14. Forward – *You Make Me Feel*
15. DJ Sammy & Yanou Feat. DO – *Heaven*
16. Eagle J. & Played P.N.M. – *Living for The Night*
17. Lovers – *7 Seconds 136 bpm*
18. ATC – *I'm In Heaven (When You Kiss Me)*
19. Mascara – *Dance Faith*
20. Lasgo – *Something*
21. Gigi D'Agostino – *L'Amour Toujours*
22. Barthezz – *Infected*
23. Brooklyn Bounce – *Club Bizarre*

24. Charly Lownoise & Mental Theo – *Wonderful Days (2001 Remix)*
25. Mark'Oh – *Never Stop That Feeling 2001*
26. Dj Bobo – *Take Control*
27. Jam & Spoon – *Right In The Night*
28. Tag Team – *Whoomp!*
29. Culture Beat – *Anything*
30. M.C. Sar & The Real McCoy – *Automatic Lover*
31. Loft – *Hold On*
32. Magic Affair – *Omen III*
33. Twenty 4 Seven – *Is It Love?*
34. Technotronic – *Hey Yoh, Here We Go*
35. Maxx – *Get-A-Way*
36. 2 Unlimited – *Let The Beat Control Your Body*
37. Ann Lee – *2 Times*
38. Mellow Trax – *Outa Space*
39. Passion Fruit – *The Rigga-Ding-Dong-Song*
40. Vengaboys – *We Are Going To Ibiza*
41. T Seven – *Hey Mr. President*
42. Pfaffendorf – *Rhythm and Sex*
43. R.B.A. – *No Alternative*
44. Cosmic Gate – *Exploration of Space*
45. Storm – *Stormanimal*
46. Masterboy – *Porque Te Vas*
47. Music Instructor – *Electric City*
48. Wamdue Project – *King Of My Castle*
49. Mark'Oh Vs. John Davies – *The Sparrows And The Nightingales*
50. Lightforce – *Take Your Time (The Riddle '99)*
51. Central Seven – *Missing*
52. Martha Williams – *Don't Leave Me This Way*
53. Quik Feat. Charlotte – *Need You Tonite*
54. Loona – *Latin Lover*
55. Gigi D'Agostino – *The Riddle*
56. Suco E Sol – *Ylarie*

57. M.A.D.R.A.S. – *Woodoorave*
58. DJ Red 5 Vs. DJs @ Work – *Rhythm & Drums 2001*
59. Dj Andrea Boari – *Da Da Da*
60. Melba – *Mabel*
61. Reactor Bass – *My Name Is Bass*
62. DJ R.P.M. – *Pussy Pussy*
63. Voodoo & Serano – *Blood Is Pumpin*
64. Axel Konrad – *R.U.F.F. Cuts*
65. Azzido Da Bass – *Dooms Night*

A.3 Classical

1. Johann S. Bach – *Brandenburgisches Konzert Nr. 1 F-Dur (BWV 1046), 1. Satz (Allegro)*
2. Johann S. Bach – *Brandenburgisches Konzert Nr. 1 F-Dur (BWV 1046), 2. Satz (Adagio)*
3. Johann S. Bach – *Brandenburgisches Konzert Nr. 1 F-Dur (BWV 1046), 3. Satz (Allegro)*
4. Johann S. Bach – *Brandenburgisches Konzert Nr. 1 F-Dur (BWV 1046), 4. Satz (Menuetto)*
5. Johann S. Bach – *Brandenburgisches Konzert Nr. 5 D-Dur (BWV 1050), 1. Satz (Allegro)*
6. Johann S. Bach – *Brandenburgisches Konzert Nr. 5 D-Dur (BWV 1050), 2. Satz (Afettuoso)*
7. Johann S. Bach – *Brandenburgisches Konzert Nr. 5 D-Dur (BWV 1050), 3. Satz (Allegro)*
8. Johann S. Bach – *Brandenburgisches Konzert Nr. 6 B-Dur (BWV 1051), 1. Satz (Allegro)*
9. Johann S. Bach – *Brandenburgisches Konzert Nr. 6 B-Dur (BWV 1051), 2. Satz (Adagio ma non tanto)*
10. Johann S. Bach – *Brandenburgisches Konzert Nr. 6 B-Dur (BWV 1051), 3. Satz (Allegro)*
11. Ludwig van Beethoven – *Violinkonzert D-Dur (op. 61), 1. Satz (Allegro ma non troppo)*
12. Ludwig van Beethoven – *Violinkonzert D-Dur (op. 61), 2. Satz (Larghetto)*

13. Ludwig van Beethoven – *Violinkonzert D-Dur (op. 61), 3. Satz (Rondo, Allegro)*
14. Ludwig van Beethoven – *Romanze für Violine und Orchester Nr. 1 G-Dur (op. 40)*
15. Ludwig van Beethoven – *Romanze für Violine und Orchester Nr. 2 F-Dur (op. 50)*
16. Ludwig van Beethoven – *Konzert für Klavier und Orchester Nr. 3 C-Moll (op. 37), 1. Satz (Allegro con brio)*
17. Ludwig van Beethoven – *Konzert für Klavier und Orchester Nr. 3 C-Moll (op. 37), 3. Satz (Rondo: Allegro)*
18. Ludwig van Beethoven – *Fantasie für Klavier, Chor und Orchester, C-Moll (op. 80)*
19. Robert Schumann – *Konzert für Klavier und Orchester, A-Moll (op. 54), 1. Satz (Allegro affettuoso)*
20. Robert Schumann – *Konzert für Klavier und Orchester, A-Moll (op. 54), 2. Satz (Andantino grazioso)*
21. Robert Schumann – *Konzert für Klavier und Orchester, A-Moll (op. 54), 3. Satz (Allegro vivace)*
22. Frédéric Chopin – *Fantasie F-Moll (Op. 49)*
23. Frédéric Chopin – *Impromptu As-Dur (Op. 29)*
24. Frédéric Chopin – *Impromptu Fis-Dur (Op. 36)*
25. Frédéric Chopin – *Impromptu Ges-Dur (Op. 51)*
26. Frédéric Chopin – *Impromptu Cis-Moll (Op. 66)*
27. Frédéric Chopin – *Nocturne Cis-Moll (Op. 27.1)*
28. Frédéric Chopin – *Nocturne Des-Dur (Op. 27.2)*
29. Frédéric Chopin – *Nocturne G-Moll (Op. 37.1)*
30. Frédéric Chopin – *Nocturne G-Dur (Op. 37.2)*
31. Frédéric Chopin – *Polonaise Cis-Moll (Op. 36.1)*
32. Frédéric Chopin – *Walzer A-Moll (Op. 34.2)*
33. Frédéric Chopin – *Walzer Des-Dur (Op. 64.1)*
34. Frédéric Chopin – *Walzer F-Moll (Op. 70.2)*
35. Frédéric Chopin – *Nocturne B-Moll (Op. 9.1)*
36. Frédéric Chopin – *Prélude G-Dur (Op. 28.3)*
37. Frédéric Chopin – *Prélude Fis-Moll (Op. 28.8)*
38. Frédéric Chopin – *Prélude Des-Dur (Op. 28.15)*

39. Frédéric Chopin – *Prélude As-Dur (Op. 28.17)*
40. Frédéric Chopin – *Mazurka A-Moll (Op. 68.2)*
41. Frédéric Chopin – *Mazurka F-Dur (Op. 68.3)*
42. Frédéric Chopin – *Ballade G-Moll (Op. 23)*
43. Johann S. Bach – *Ouverture No. 1 C-Dur (BWV 1066), 1. Ouverture*
44. Johann S. Bach – *Ouverture No. 1 C-Dur (BWV 1066), 2. Courante*
45. Johann S. Bach – *Ouverture No. 1 C-Dur (BWV 1066), 3. Gavotte*
46. Johann S. Bach – *Ouverture No. 1 C-Dur (BWV 1066), 4. Forlane*
47. Johann S. Bach – *Ouverture No. 1 C-Dur (BWV 1066), 5. Menuet*
48. Johann S. Bach – *Ouverture No. 1 C-Dur (BWV 1066), 6. Bourrée*
49. Johann S. Bach – *Ouverture No. 1 C-Dur (BWV 1066), 7. Passepied*
50. Johann S. Bach – *Ouverture No. 2 H-Moll (BWV 1067), 1. Ouverture*
51. Johann S. Bach – *Ouverture No. 2 H-Moll (BWV 1067), 2. Rondeau*
52. Johann S. Bach – *Ouverture No. 2 H-Moll (BWV 1067), 3. Sarabande*
53. Johann S. Bach – *Ouverture No. 2 H-Moll (BWV 1067), 4. Bourrée*
54. Johann S. Bach – *Ouverture No. 2 H-Moll (BWV 1067), 5. Polonaise – Double*
55. Johann S. Bach – *Ouverture No. 2 H-Moll (BWV 1067), 6. Menuet*
56. Johann S. Bach – *Ouverture No. 2 H-Moll (BWV 1067), 7. Badinerie*
57. Johann S. Bach – *Ouverture No. 4 D-Dur (BWV 1069), 1. Ouverture*
58. Johann S. Bach – *Ouverture No. 4 D-Dur (BWV 1069), 2. Bourrée*
59. Johann S. Bach – *Ouverture No. 4 D-Dur (BWV 1069), 3. Gavotte*
60. Johann S. Bach – *Ouverture No. 4 D-Dur (BWV 1069), 4. Menuet*
61. Johann S. Bach – *Ouverture No. 4 D-Dur (BWV 1069), 5. Rejouissance*

Appendix B

Listing of Classifier Training

The following listing shows how to train a MUGRAT classifier. The results presented in Chapter 6 were obtained using this classifier. As described in Section 5.3, the WEKA framework is used for the classification subsystem. More information about WEKA can be found at http://www.cs.waikato.ac.nz/~ml/.

Options

The training dataset is specified by the `-t` option. The feature representations for the example songs are contained in the ARFF file `out.arff` in the directory `/Users/kyrah/mugrat/`. `IBk` is WEKA's implementation of a k-nearest-neighbour classifier. The option `-K` specifies the number of nearest neighbours to use. In the example, the labels of the three nearest neighbours are taken into account. The location where the trained classifier is stored is given by the `-d` option.

Explanation of the Output

The system evaluates the classifier's performance in two ways. First, a prediction based on the training data is made. This is an optimistic estimation, and shows the upper bound of the classifier's performance. Secondly, stratified ten-fold cross-validation is performed. For both cases, the confusion matrices are displayed.

Listing

```
> java weka.classifiers.IBk -K 3 -d /Users/kyrah/mugrat/genre_knn_classifier
-t /Users/kyrah/mugrat/out.arff

Options: -K 3

IB1 instance-based classifier
using 3 nearest neighbour(s) for classification
```

```
=== Error on training data ===

Correctly Classified Instances         167               91.2568 %
Incorrectly Classified Instances        16                8.7432 %
Mean absolute error                      0.0737
Root mean squared error                  0.1919
Total Number of Instances              183

=== Confusion Matrix ===

  a  b  c   <-- classified as
 56  5  0 |  a = Metal
  9 52  2 |  b = Dance
  0  0 59 |  c = Classical

=== Stratified cross-validation ===

Correctly Classified Instances         161               87.9781 %
Incorrectly Classified Instances        22               12.0219 %
Mean absolute error                      0.1186
Root mean squared error                  0.2834
Total Number of Instances              183

=== Confusion Matrix ===

  a  b  c   <-- classified as
 54  7  0 |  a = Metal
 13 48  2 |  b = Dance
  0  0 59 |  c = Classical
```

Refer to Section 6.4 for a description of how to use MUGRAT to classify a new song.

Appendix C

Contents Of The CD-ROM

File System: Joliet/Rock-Ridge, *Mode:* Single-Session

C.1 Thesis

Path: /

thesis.pdf	This document as *PDF*

Path: /tex/

thesis.tex	LaTeX main document
appendix_a.tex	Appendix A: Playlists
appendix_b.tex	Appendix B: Listing of Classifier Training
appendix_c.tex	Appendix C: Contents of the CD
ch1.tex	Chapter 1: Introduction
ch2.tex	Chapter 2: Music Genre Recognition
ch3.tex	Chapter 3: Feature Extraction
ch4.tex	Chapter 4: Classification
ch5.tex	Chapter 5: Prototype
ch6.tex	Chapter 6: Experimental Results
ch7.tex	Chapter 7: Conclusion
pre_abstract.tex	Abstract
pre_kurzfassung.tex	Kurzfassung (german abstract)
pre_preface.tex	Preface
literature.bib	Collection of relevant literature
*.sty	LaTeX style files (© FHS Hagenberg)

Path: /tex/images/

*.pdf	graphics and images

C.2 MUGRAT Source Code

The source code on the CD-ROM is a snapshot from 19th June 2002. The latest version can be downloaded from the official MUGRAT website http://kyrah.net/mugrat/.

Path: /src/

Makefile	GNU Makefile for GNU/Linux i386
defs.h	common definitions
main.cpp	`mugrat` feature extraction program
d_*.[h|cpp]	data types used in MUGRAT
io_*.[h|cpp]	input/output classes
x_*.[h|cpp]	classes concerning feature extraction
readme	README file
copying	GNU General Public License, version 2

Path: /src/scripts/

run_mugrat.pl	script to run `mugrat` in batch mode
cmd.txt	list of command lines used to generate the example samples