An Objective Approach to Content-Based Audio Signal Classification

Diplomarbeit

eingereicht am 19. Mai 2003

Juan José Burred
215029

Betreuer:

Alexander Lerch, zplane.development
Dr. Nicolas Moreau, Technische Universität Berlin

Prof. Dr.-Ing. Thomas Sikora

Fachgebiet Nachrichtenübertragung
Institut für Telekommunikationssysteme
Fakultät IV Elektrotechnik und Informatik
Technische Universität Berlin

Die selbständige und eigenhändige Anfertigung versichere ich an Eides Statt.

Berlin, den 19.05.2003

Abstract

The present work describes the design, implementation and evaluation of a system for automatic audio signal classification. The signals are classified according to audio type, such as speech, background noise and several musical genres. The classification process is organized hierarchically, that is, as a succession of type decisions. At the highest level of the hierarchy, signals are recognized as speech, music or background noise. Speech signals are further divided into male speech, female speech and speech with background music or noise. Special focus is given to the distinction between music genres, where a total of 13 classical and non-classical genres have been considered. Also, an effort was made in selecting the music genres, aiming at simplicity and generality.

A large number of audio features are evaluated for their suitability in such a classification task, including well-known physical and perceptual features, audio descriptors defined in the MPEG-7 standard, as well as new features proposed in this work. They are selected with regard to their robustness to noise and bandwidth changes, and to their ability to distinguish a given set of audio types.

A density estimation classifier (Gaussian Mixture Model) and a nonparametric classifier (k-Nearest Neighbor) are thoroughly compared with respect to classification accuracy and computational performance. In contrast to previous systems, the feature selection and the classification process itself are carried out in a hierarchical way. This is motivated by the numerous advantages of such a tree-like structure, which include easy expansion capabilities, flexibility in the design of genre-dependent features and the ability to reduce the probability of costly errors. A special effort is made in comparing such a hierarchical approach with the more common direct, single-stage approach.

As a result of these considerations, a hierarchical Gaussian Mixture Model has been chosen to be implemented in a final prototype application for classifying audio files. The evaluation of this application, which can perform feature extraction in real-time, confirms the suitability of the hierarchical approach, both in computational and classification performance. In spite of the higher number of audio classes considered, the used approach achieves similar or higher classification rates than previous related systems, apart from offering the mentioned advantages. This makes the hierarchical approach a promising base for future developments.

Kurzfassung

Ein objektiver Ansatz zur inhaltsbasierten Klassifizierung von Audiosignalen

Die vorliegende Arbeit beschreibt den Entwurf, die Implementierung, sowie die Auswertung eines Systems für die automatische Klassifizierung von Audiosignalen. Die Signale werden den Kategorien Sprache, Hintergrund und verschiedenen Musikgenres zugeordnet. Die Klassifizierung ist hierarchisch organisiert, d.h., als eine Folge von Klassenentscheidungen. Auf der höchsten Stufe der Hierarchie werden Signale als Sprache, Musik oder Hintergrundgeräusch erkannt. Sprachsignale werden daraufhin in männliche Sprache, weibliche Sprache und Sprache mit Hintergrundmusik oder Geräusch getrennt. Auf die Unterscheidung zwischen insgesamt 13 klassischen und nichtklassischen Musikgenres wird ein besonderer Schwerpunkt gelegt.

Eine große Anzahl von Audiomerkmalen (*features*) wird auf ihre Eignung für eine solche Klassifizierungsaufgabe untersucht. Unter diesen Merkmalen befinden sich bereits vorhandene physikalische und wahrnehmungsangepasste Größen, im MPEG-7 Standard definierte Audio-Deskriptoren, sowie neue, in dieser Arbeit vorgeschlagene Merkmale. Sie werden mit Rücksicht auf ihre Klassentrennungsfähigkeit, sowie auf ihre Invarianz gegenüber hinzugefügtem Rauschen bzw. Veränderungen der Bandbreite ausgewählt.

Ein parametrischer Klassifikator (*Gaussian Mixture Model*) und ein nichtparametrischer Klassifikator (*k-Nearest Neighbor*) werden in Bezug auf ihre Klassifizierungsfähigkeit und rechnerische Effizienz umfassend verglichen. Im Gegensatz zu bisherigen vorgeschlagenen Systemen wird in dieser Arbeit sowohl die Auswahl der Merkmale als auch die Klassifizierung hierarchisch durchgeführt. Dies wird von den zahlreichen Vorteilen, die eine solche Baumstruktur aufweist, motiviert. Unter ihnen befinden sich leichte Erweiterungsmöglichkeiten, Flexibilität beim Entwurf genrespezifischer Merkmale und die Fähigkeit, die Wahrscheinlichkeit schwerwiegender Klassifikationsfehler zu reduzieren. Besonderer Wert wird somit auf den Vergleich einer hierarchischen Klassifizierung mit dem üblicheren direkten Ansatz gelegt.

In Folge dieser Betrachtungen wird ein hierarchischer *Gaussian Mixture Model* Klassifikator für die Implementierung einer Prototyp-Anwendung zur Klassi-

fizierung von Audiodateien gewählt. Die Anwendung kann die Merkmale in Echtzeit extrahieren. Die Auswertung der Ergebnisse bestätigt die Eignung des vorgeschlagenen hierarchischen Ansatzes, sowohl in Hinsicht auf Klassifizierungsgenauigkeit, als auch auf rechnerische Leistungsfähigkeit. Der hier verwendete Ansatz erreicht, trotz einer höheren Anzahl von Klassen, ähnliche oder höhere Klassifizierungsgenauigkeit wie bisherige Systeme, wobei er zusätzlich die genannten Vorteile bietet. Er erweist sich somit als eine vielversprechende Basis für zukünftige Entwicklungen.

Resumen

Procedimiento Objetivo para la Clasificación de Señales de Audio Basada en el Contenido

El crecimiento exponencial de Internet, así como los últimos avances en tecnologías de redes y de compresión de datos, han hecho posible el fácil acceso a grandes cantidades de información. Es más que probable que, en un futuro cercano, los servicios de música disponibles *on line* superen en importancia a la acostumbrada distribución de audio almacenado en soportes físicos, como los discos compactos o los DVDs. Actualmente, la exploración y administración de datos de audio está basada en breves informaciones textuales añadidas manualmente a los ficheros, lo cual es una tarea costosa que requiere altos recursos temporales y humanos. Más aún, esta información a menudo resulta incompleta, y en ocasiones ni siquiera está disponible.

Las técnicas de Análisis del Contenido tienen como objetivo extraer automáticamente de las señales información acerca de su contenido, y han hecho posible un gran número de nuevas aplicaciones, como clasificación y recuperación de información basada en el contenido, segmentación, tratamiento inteligente de señales, etc.

El presente Proyecto Fin de Carrera describe el diseño, implementación y evaluación de un sistema para la clasificación automática de ficheros de audio. El proceso de clasificación está organizado de forma jerárquica. En el primer nivel de la jerarquía, el audio es reconocido como habla, música o ruido de fondo. En los siguientes niveles, las señales de habla son divididas en habla masculina, femenina y habla con ruido o música de fondo. El proyecto está especialmente centrado en la distinción de géneros musicales, para lo cual han sido considerados un total de 13 géneros clásicos, como música sinfónica, coral, de cámara, etc. y no clásicos, como rock, pop, jazz, etc.

Las aplicaciones de un sistema de clasificación de estas características incluyen, por ejemplo, organización automática de archivos de sonido, tratamiento inteligente de señales, ecualización automática, asignación inteligente de ancho de banda, codificación inteligente de audio, segmentación de flujos de audio o tratamiento de señales de video basado en su banda sonora.

El sistema descrito se basa, al igual que todos los sistemas propuestos en

proyectos de investigación relacionados, en las técnicas proporcionadas por el campo del *Reconocimiento de Patrones*. De cada señal a clasificar se extrae una serie de características, que a su vez son tomadas como elementos del *vector de características* asociado a dicha señal. De esta forma, cada señal está representada por su vector asociado en el *espacio de características*, de tantas dimensiones como características extraídas. Estos vectores se emplean para entrenar a un clasificador, el cual infiere una regla de decisión que aplicará para asignar una clase determinada a un vector entrante de naturaleza desconocida.

En previas investigaciones relacionadas se han propuesto numerosas combinaciones de características a extraer (tímbricas, perceptivas, estadísticas) y clasificadores (estadísticos paramétricos, estadísticos no paramétricos, neuronales) para llevar a cabo detección de audio. Sin embargo, determinados aspectos inherentes al proceso de diseño de un clasificador han merecido sólo escasa atención hasta el momento. Estos son: la creación de una taxonomía adecuada, el estudio de los problemas que conlleva un alto número de dimensiones en el espacio de características y el estudio del diferente grado de adecuación de las características en función de las clases o géneros a clasificar. Parte de la motivación del presente proyecto ha sido la de investigar la influencia de estas cuestiones en la realización del sistema.

En primer lugar, se ha puesto especial interés en crear una taxonomía de clases de audio que sea a la vez simple y lo más completa posible. En el caso de los géneros musicales, se ha intentado definir clases que sean lo más consecuentes posible en cuanto a su significado musicológico generalmente aceptado.

Un total de 90 características sonoras han sido detalladamente examinadas respecto a su capacidad de representación y separación de señales pertenecientes a diferentes clases. Entre ellas se encuentran características ya conocidas y utilizadas en sistemas previos de clasificación de audio o de detección del habla, como centroide, *rolloff*, flujo espectral, envolvente temporal o MFCCs (*Mel Frequency Cepstral Coefficients*). Asimismo han sido implementados y examinados algunos de los descriptores de audio definidos dentro del reciente estándar MPEG-7. En particular, han sido encontradas algunas faltas de coherencia en la definición de la relación de armonicidad (*harmonic ratio*) perteneciente al citado estándar, razón por la cual se ha optado por la realización de una versión modificada de la misma. Por último, se proponen una serie de nuevas características, como los momentos centrales de tercer y cuarto orden de la señal en el dominio del tiempo, una medida simplificada de sonoridad y una medida de regularidad rítmica basada en la autocorrelación del histograma rítmico de la señal.

Como se ha mencionado, usar un alto número de características puede resultar contraproducente. La llamada *maldición de la dimensionalidad* es un problema clásico dentro del reconocimiento de patrones, e implica que el rendimiento de un clasificador deja de aumentar, o incluso puede disminuir, si se sobrepasa un cierto número de características observadas. Por ello, un reducido número de ellas han sido seleccionadas de entre las 90 iniciales. La selección ha sido llevada a

cabo de forma totalmente sistemática. En primer lugar, han sido descartadas las características más susceptibles a la adición de ruido y a la variación moderada del ancho de banda de la señal. De esta forma se garantiza un rendimiento similar del clasificador para diferentes calidades de audio. Las características restantes han sido objeto de un algoritmo de selección automático (Búsqueda Secuencial Progresiva) que da como resultado un subconjunto de características con propiedades óptimas en la separación de clases. Se ha obtenido que un número de 20 características conlleva una relación óptima entre rendimiento de clasificación y tiempo de cálculo.

Al contrario que en otros sistemas propuestos con anterioridad, la selección de características, así como el proceso de clasificación, han sido realizados en el presente proyecto de forma totalmente jerárquica, siguiendo un árbol de clasificación que se corresponde exactamente con la taxonomía de clases utilizada. El procedimiento común consiste en considerar el problema de clasificación como la toma de una única decisión entre todas las clases posibles, y no como una secuencia de decisiones. Sin embargo, el método jerárquico conlleva numerosas ventajas. En primer lugar, este procedimiento permite la obtención de aquéllas características que resultan más adecuadas a la hora de clasificar un subconjunto de clases dado. Por ejemplo, una característica que describa la fuerza de los pulsos rítmicos será probablemente más adecuada en la distinción entre música clásica y música pop que en la distinción entre géneros de música de cámara. Por otro lado, el método jerárquico permite que los errores obtenidos en la clasificación sean más aceptables que en el caso directo. Por ejemplo, que un fragmento de una sinfonía sea clasificado como perteneciente a un concierto para solista y orquesta es menos grave de cara al usuario que si fuera clasificado como rock. Otras ventajas incluyen la posibilidad de diseñar características que se adapten a las necesidades particulares de cada género, y la mayor facilidad de ampliación de la taxonomía.

Por todo ello, se ha puesto especial interés en comparar los métodos directo y jerárquico en cuanto a la calidad de clasificación y al rendimiento de cálculo. En cada uno de los métodos se ha comparado a su vez el empleo de un clasificador paramétrico (Modelo de Mezclas Gaussianas, *Gaussian Mixture Model, GMM*) y de un clasificador no paramétrico (Vecinos más Cercanos, *k-Nearest Neighbor, kNN*). La extracción de características, así como los experimentos de selección de las mismas y de evaluación de clasificadores, han sido implementados usando *MATLAB*.

El método finalmente elegido, tras considerar todo lo anterior, ha sido el de una clasificación jerárquica basada en GMM. En base a este algoritmo se ha implementado una aplicación prototipo para la clasificación de ficheros de audio en formato WAV. La herramienta ha sido programada en el lenguaje *C/C++*, y funciona como aplicación de línea de comando en sistemas operativos Windows. El programa es capaz de extraer las características en tiempo real y de diferenciar música, habla y ruido de fondo con una precisión del 95%, música clásica de música no clásica con un 96% o música de cámara de música orquestal con un

82%. Estos porcentajes indican la posibilidad de utilización práctica del sistema para los niveles más altos de la taxonomía. Sin embargo, las mayores dificultades se han encontrado en la diferenciación de géneros más específicos, como en la distinción entre géneros de música de cámara, donde sólo se ha obtenido una precisión del 55%. Para mejorar la clasificación en los géneros más específicos de la taxonomía, y permitir una futura realización práctica de todos los niveles de la jerarquía, se requiere el diseño de características más sofisticadas y adaptadas a su correspondiente clase o subclase.

Este Proyecto Fin de Carrera ha sido realizado en la empresa zplane.development, situada en Berlín y dedicada al desarrollo de software y hardware para el tratamiento de audio, en colaboración con el Instituto de Sistemas de Telecomunicación (Institut für Telekommunikationssysteme) de la Universidad Técnica de Berlín (Technische Universität Berlin), dentro del programa de intercambio entre dicho instituto y la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad Politécnica de Madrid.

Acknowledgments

I would like to express my gratitude to all who supported, directly or indirectly, the realization of this work.

First of all, I thank all the colleagues and friends who lent me the *hundreds* of CDs I needed to compile the audio database.

I received much more replies to the musical questionnaire than expected, many of them from people who also showed great interest and curiosity in my work. I express my appreciation to all of them.

I am also most thankful to the people at zplane.development, especially to my supervisor, Alexander Lerch, for their advice and for the great working atmosphere.

This project would have not been possible without the constant support of my exchange program coordinators, Ángel Álvarez from the Polytechnical University in Madrid and Werner Eschenberg from the Technical University in Berlin.

My special thanks to the people who patiently made the proof-readings of the manuscripts.

Finally, I would like to thank my parents and my family back in Spain for their unwavering encouragement during my stay in Berlin.

Contents

List of Abbreviations

Abbreviation	Meaning
ASR	Automatic Speech Recognition
bpm	Beats Per Minute
CA	Content Analysis
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
FDIS	Final Draft International Standard
FFT	Fast Fourier Transform
FSS	Feature Subset Selection
FST	Feature Space Transformation
GMM	Gaussian Mixture Model
GS	Simple Gaussian Classifier
IR	Information Retrieval
ISO/IEC	International Organisation for Standardization / International Electrotechnical Commission
kNN	k-Nearest Neighbor
LLD	MPEG-7 Low Level Descriptor
MAP	Maximum A Posteriori
MFCC	Mel Frequency Cepstral Coefficients
ML	Maximum Likelihood
MPEG	Moving Pictures Experts Group (Subgroup of the ISO/IEC)
NN	Nearest Neighbor
pdf	Probability Density Function
PR	Pattern Recognition
RMS	Root Mean Square
STFT	Short Time Fourier Transform

List of Symbols

Signal theory symbols and indices

Symbol	Meaning
$\lvert \cdot \rvert$	Absolute value
$x[n]$	Time-domain signal
$w[n]$	Window function
$x_r[n]$	Windowed signal
$\hat{x}[n]$	Linear prediction of $x[n]$
f	Frequency in Hertz
f_s	Sampling frequency in Hertz
ω	Circular frequency in rad/s
$X_r[k]$	STFT of frame r
$P_r[k]$	Power spectrum of frame r
N	DFT/FFT/STFT length
L	Window length
R	Step length
E	Energy
I	Sound intensity
AC_r	Autocorrelation of frame r
B	Total number of frequency subbands
$B[i]$	Beat histogram
T	Number of beat histogram bins

Index	Maximum range	Meaning
n	$(-\infty, \infty)$	Audio sample (time variable)
k	$[0, N-1]$	DFT coefficient (frequency bins)
r	$(-\infty, \infty)$	Frame
b	$[1, B]$	Frequency subband
τ	$(-\infty, \infty)$	Lag of the autocorrelation
i	$[1, T]$	Beat histogram bin

Pattern recognition symbols and indices

Symbol	Meaning
$\lvert \cdot \rvert$	Determinant
x	Random variable
$p(x)$	Probability density function of x
μ	Mean
$\hat{\mu}$	Sample mean
σ	Standard deviation
$\hat{\sigma}$	Sample standard deviation
ξ_3	Skewness
ξ_4	Kurtosis
$\mathbf{x}$	Random vector (feature vector)
$p(\mathbf{x})$	Joint probability density function of $\mathbf{x}$
$\boldsymbol{\mu}$	Mean vector
$\boldsymbol{\Sigma}$	Covariance matrix
ω_k	Class k
$g_k(\mathbf{x})$	Discriminant function
$P(\omega_k)$	A priori probability of class ω_k
$p(\mathbf{x}\|\omega_k)$	Likelihood of class ω_k
$P(\omega_k\|\mathbf{x})$	A posteriori probability of class ω_k
$\boldsymbol{\theta}_k$	Parameter vector of class ω_k
$\hat{\boldsymbol{\theta}}_k$	Maximum likelihood estimate of $\boldsymbol{\theta}_k$
$p(\mathcal{X}_k\|\boldsymbol{\theta}_k)$	Likelihood of $\boldsymbol{\theta}_k$ with respect to a set of samples from the class ω_k
D	Number of dimensions (features)
N	Total number of sample vectors
N_k	Number of sample vectors in class ω_k
$\mathbf{S}_W$	Within-class scatter matrix
$\mathbf{S}_B$	Between-class scatter matrix
J	Criterion function
$tr(\cdot)$	Trace
$p(\mathbf{x})_{km}$	mth component of mixture density k
w_{km}	mth weight (prior) of mixture density k

Index	Maximum range	Meaning
i	[1,D]	Vector element
j	[1,N]	Feature vector
k	[1,C]	Class
m	[1,M]	Component of a mixture density

Chapter 1

Introduction

Over the last decade, the exponential growth of the Internet has made huge amounts of information easily available to millions of people. Furthermore, advances in networking technologies, as well as in coding and compression algorithms, brought about increased bandwidth and allowed to make optimal use of it. As a result, content requiring high levels of bandwidth, such as video and audio streams, or any kind of multimedia documents, coexists nowadays with the traditional textual and graphical data. It is well known that the social, commercial, and even legal impacts of this new paradigm are changing the way people produce, share and store information.

In contrast to radio and television, two technologies also providing high bandwidth and real-time transmission, the Internet allows to retrieve content *on demand*. With regard to this particular characteristic, the Internet is not a direct competitor to radio or TV, but to what could be regarded as the present prevalent "on-demand" technology: the storage on physical media. It is not unlikely that in the near future, music available on-line will overtake the usual distribution of audio stored on CDs or DVDs. The record industry is well aware of this, as it is starting to set up on-line music services, as well as fighting against copyright infringements in music file sharing. In the next years, the video and movie industry should also begin to worry.

Now, although the variety and quantity of accessible data is enormous, the way in which we can manage and search for it is frustratingly limited. At this point, it is only possible to search the Internet using textual queries, for example using a web browser. This works well for written documents, since in this case text itself is the content. But when searching for an image, a video or an audio clip, one must rely on the textual information manually attached at one time to the corresponding file. This information is often inaccurate, and in most cases it consists only of the title and the author. In many cases, there is no such information at all, the file name being the only hint about its content. Today, it is not possible to perform such natural actions as to search for an image containing certain objects, a movie scene featuring a given actor, or organizing a collection

of music files according to the genre or the mood, unless this information has manually been attached to the file beforehand, which is in most cases a nearly unfeasible task.

Moreover, one can take a further step and imagine content-aware search engines which will not only be based on text queries, but will also allow to search for images that match a certain pattern or for music excerpts that contain a melody that has been played on a keyboard or hummed into a microphone.

The above considerations led in the past years to the fast growth of the *Content Analysis* (CA) or *Machine Perception* research field. Its goal is to make computers capable of automatically extracting information about the content from data. Although CA is not a completely new discipline (some CA technologies that have been widely used for many years include text analysis, computer vision and speech recognition systems), it is now being enriched to consider much wider types of signals, and thus it will provide the infrastructure needed to implement all the above mentioned retrieval and classification services as well as many others. It is likely that this will revolutionize our day-to-day use of media.

Content Analysis versus Information Retrieval

The term *Information Retrieval (IR)* is often used as a synonym for *Content Analysis*. However, to retrieve information also means to search for a given entry in a database. This has led to certain confusion about what is really meant when speaking of IR. For example, one of the main subjects of classic IR is the study of text-based query techniques like the ones used to browse information on the Internet. On the other hand, retrieving information from a signal can also be viewed as an act of CA. In order to avoid misunderstandings, the clearly defined term of Content Analysis will be used in the present work instead of Information Retrieval.

1.1 Audio Content Analysis

Audio Content Analysis, also called *Computer Audition* or *Machine Listening*, deals with the extraction of information from sounds. It should be noted that, in order for an audio system to be regarded as performing Content Analysis, it should be able to provide "high-level" descriptions, such as spoken content for voice or melody and tempo for music, or even more abstract information like musical structure, genre or mood. Simple features like loudness or pitch provide "low-level" information and by themselves might not suffice for speaking of a true description of content. Nevertheless, they constitute the base upon which higher levels of abstraction are built.

Automatic Speech Recognition (ASR) is by far the field that has gathered the most attention of the audio analysis community over the last 20 years. Many

programs capable of transcribing voice into text with excellent performance have been on the market for a long time. Although ASR is by no means a trivial task, it has raised great interest among researchers and investors, because it offers a natural human-machine-interaction. In contrast, little interest has been paid to non-speech audio signals.

But, as mentioned, this situation is changing rapidly with the advent of on-line available audio material. Many research projects have started with the goal of developing tools that will allow the analysis, management and browsing of any kind of sound data [13]. Applications include sound archiving, retrieval, classification, segmentation, intelligent signal processing, musicological analysis, etc. The increased interest for content analysis has even led to the completion of an International Standard for multimedia content description, MPEG-7, whose audio part will be discussed in Sect. 3.2 (for a historical review of research in audio classification, see Sect. 3.1).

As shown in Fig. 1.1, there are two possible ways to extract content from an audio signal. One approach is to transcribe the sounds into some kind of symbolic representation. In the case of speech, this symbolic description is the text that has been spoken. The resulting text data (*Representation Stage*) can be forwarded to a text analyzer using one of the many available text-based content extraction techniques, which can analyze its syntactic or semantic contents at the final *Application Stage*. Of course, the transcription can also be viewed as an application itself. The musical equivalent of the transcription approach is to extract the musical score in first place, and then to analyze it to extract further content.

There are also a number of applications that cannot rely on transcription, such as speaker recognition or musical instrument detection. These are solely based on acoustical features and correspond to the lowest path on the figure.

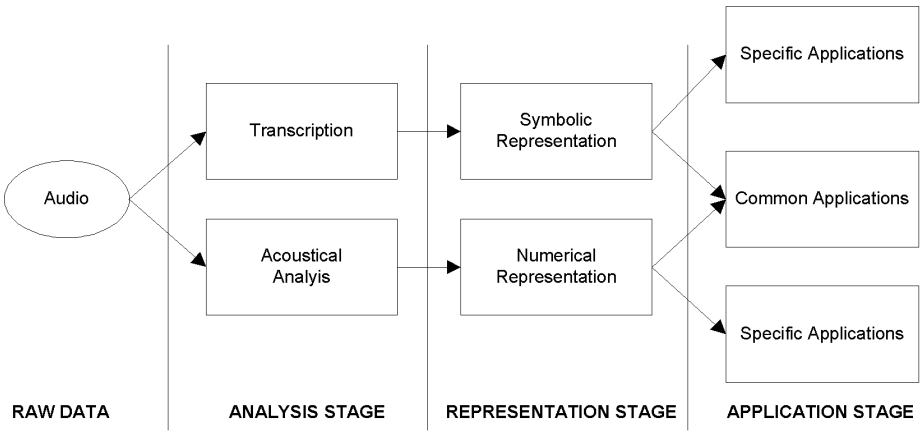


Figure 1.1: Overview of Audio Content Analysis.

Additionally, there is a third group of applications, labeled as *common applications* in the figure, that can be accomplished following either way. For example, a music archiving and retrieval system could be based on the musical scores, but also on some reduced set of acoustical measurements that could accurately describe each musical piece (a technique called *Audio Fingerprinting*). It is also possible to perform musical genre detection based on these characteristics; however, one can expect this classification to be less accurate than when the scores are available, because in the latter case we could easily extract information about harmony, melody, instrumentation and other factors contributing to describe a specific style.

Hence, if we are seeking to implement any of the common applications, why should we attempt to design them following the acoustic analysis approach, when the symbolic approach is likely to outperform it in all cases? The reason is that the transcription is a daunting, difficult task in most cases, and constitutes a "bottleneck" in the process of Audio Content Analysis. As we have seen, transcription has long been achieved successfully in ASR systems, but the mostly polyphonic nature of music[1] makes it considerably harder to discriminate and transcribe the constituent voices, or to extract one of them out of the polyphonic mixture. Today, there are no systems capable of reliably transcribing more than two simultaneous voices into a score, and it is a matter of controversy as to whether this will ever be achieved.

Therefore, most of the work in the field of symbolic music content analysis has been made assuming that the music is already available in symbolic form (e.g. in *MIDI*[2] files). Obviously, this is not the case in most situations, in which audio is available as a raw bit stream. Acoustic-based approaches allow us to avoid the transcription "bottleneck" and to implement a wide range of content-based applications based directly on the raw audio data.

1.2 Audio Classification

> *"Whoever tries to find useful information over the World Wide Web understands immediately that classification and similarity are his best allies, and the panic we are facing when exposed to such amounts of information is a good suggestion to understand not the semioticist's* [3] *but every man's and woman's anxiety to separate, define, classify."*[10]

[1] In its most precise sense, the term *polyphony* denotes multi-voiced music in which each voice develops an independent melodic line, as opposed to *homophony*, in which each voice is subordinated to a rigid vertical chordal structure. However, it will be used here in the broad sense, denoting any kind of multi-voiced music.

[2] Musical Instrument Data Interface.

[3] *Semiotics* is the study of *signs* and their interrelations. In its broadest sense, a *sign* is an element of language composed by a *signifier* (a sound or an image) and a *signified* (its content).

As mentioned, classification is one of the possible applications of Audio Content Analysis. In a classification system, the input signal is analyzed and a label describing that signal is delivered at the output. There are many possible criteria upon which the signals can be separated, such as timbral content, pitch or musical tempo. But in the context of multimedia data managing and browsing, the most useful classification is the one that assigns labels describing the *type* or *category* of the signal, that is, if the analyzed sound is speech, noise, if it belongs to a certain music genre, etc. This is the natural way to classify audio signals, and all common sound databases on the Internet, sound archives, as well as retail shops are organized this way.

Audio classification should not be confused with another closely related application: *audio identification*. In the latter, the exact identity of the signal is obtained, while in the first, a broad category is assigned to it. In a musical context, to identify means to retrieve the title and composer or performer of a piece, while classifying means to assign a genre to it. One of the most important applications of audio identification is the copyright control. Technically, the most important difference is that audio identification relies on a previously created database of descriptions of individual sounds (such as the *audio fingerprints* mentioned earlier), and it will not identify a new sound if it is not already contained in the database. In contrast, a classification system should be capable of assigning a correct class to any incoming, unknown signal.

1.2.1 Applications

The following possible applications of a category-based classification show why this research field is so attractive at present.

Audio Database Indexing. A stored audio collection is scanned, assigning an audio-class label to each file. This enables later audio type based browsing, for example in searching a database for a specific musical genre. This is specially useful for large audio and music collections, such as the audio archives in broadcasting facilities (radio, TV), in soundtrack studios for the movie industry or in music content providers on the Internet. Currently, this classification is done manually, which is an extremely time and human resource consuming task. Perhaps the most common example illustrating this case are the large collections of downloaded *MP3* files stored on hard disks, very often in a chaotic manner. A classifying program could create a directory tree to organize the music according to genre.

Intelligent Signal Processing. A possible example of intelligent signal processing is automatic equalization. Equalizers are a well-known feature in virtually all types of audio equipment, not only at the professional, but also at the consumer level, including HiFi systems, Walkmans, computer sound

cards, car audio systems, etc. If these systems were capable of detecting the kind of audio they are playing, the corresponding equalization presets could be automatically loaded and applied. Another possibility could be a content dependent automatic control of dynamics processing, for example in the adaptation of loudness in broadcasting.

Intelligent Signal Analysis. Classification would allow to construct an intelligent signal analysis system, which could send the input signal to an ASR transcriber if speech were detected, to an (hypothetical) polyphonic music transcriber if music were detected, or to an environmental sound classifier if other sounds were detected. For solo music, a musical instrument identifier could be activated, for rock or pop music, a beat tracking system, and so on.

Automatic Bandwidth Allocation. A telephone network with audio classification capabilities could dynamically allocate bandwidth for the signal being transmitted. More bandwidth would be allocated for music than for speech transmissions, and no bandwidth at all if only background noise is detected. This would help multiplexing systems to work more efficiently. The same applies to audio streams in data networks such as the Internet.

Segmentation. The signal to be classified contains often a sequence of different audio classes. Examples are commented music radio programs, or TV or movie soundtracks, where speech, music and background sections alternate. Segmentation is a closely related topic to classification, and consists of finding the exact transition moment between two consecutive audio types. This allows to extract desired parts from a stream and to discard undesired ones.

Intelligent Audio Coding. Some audio coding algorithms are designed to work in an optimal way with speech signals, others with music signals. As a particular case of Intelligent Signal Processing, audio classification would allow to switch automatically between coders, depending on the audio type at the input.

Broadcast Browsing. A classification system with real-time capabilities could allow to scan a radio or TV dial in order to find the desired music genre.

Audio Assisted Video Processing. It is usual to regard video and its corresponding audio as completely independent data sources. It is clear, however, that they contain a lot of information about each other. Soundtrack analysis can help to segment a video in scenes, and to find a certain kind of video content. For example, an audio-based TV-commercial detector could stop the video recorder to skip unpleasant commercial breaks during a movie.

1.3 Scope and Overview

The goal of this work is to find a meaningful set of acoustical and perceptual measurements, as well as the suitable algorithms to objectively classify an input raw audio signal as speech, background noise, or one of several musical genres. It especially focuses on the distinction between musical types. For this purpose, the audio features, as well as the selection algorithms and the evaluation experiments, are firstly implemented in *MATLAB*. The considerations resulting from this evaluation will led to the creation of a prototype command-line application, programmed in the *C/C++* language, for the classification of audio files. The preliminary assumption is that the analyzed files contain only one type of audio. Otherwise, segmentation techniques, which are out of the scope of the work, would have to be applied.

The first step in the design process of the classifier consists of selecting the audio classes into which the signals are to be classified. In the case of music, the genre structure should be very carefully created, taking into account generality and consistency.

The main part of the work consists of the thorough evaluation of the candidate sound measurements and classification algorithms, which will be compared with regard to classification accuracy, computational performance, flexibility and generality. The resulting software will serve to demonstrate the feasibility of the selected approach, and as a possible base for future developments.

The work is structured as follows:

In Chapter 2 the needed theoretical background from the fields of signal processing, psychoacoustics, statistics and pattern recognition is provided.

Previous research on audio classification is reviewed in Chapter 3. Also, the MPEG-7 standard is briefly introduced here.

Chapter 4 focuses on the selection of the audio classes. It especially addresses the inherent difficulty of establishing an objective taxonomy of musical genres.

Chapter 5 details the audio-specific part of the process: the choice and design of the features to be extracted from the audio signals.

As will be seen, it is worthwhile to use only a reduced set of features for the classification task. This feature selection process is outlined in Chapter 6.

Several classification algorithms are evaluated in detail in Chapter 7. Classifying performances and other implications concerning implementation are reviewed for each approach.

Following the considerations in Chapter 7, an algorithm was selected for the final application. This implementation is addressed in Chapter 8, and the results of its evaluation are given in Chapter 9.

In Chapter 10 some conclusions on the obtained results are drawn. The present work is also positioned in the context of the current state of the research by reviewing its new contributions. Finally, some thoughts on future research directions are outlined.

Chapter 2

Background

In this chapter, the theoretical background needed to follow the rest of the work is provided. Its goal is also to present the definitions and notational conventions that will be used.

2.1 Signal Theory

2.1.1 Signal Energy and Power

Digital audio signals are both discrete in time and in amplitude. A time-domain discrete signal is represented mathematically by the notation $x[n]$[1].

For a discrete signal $x[n]$, the *energy* within the interval $[0, N]$ is given by

$$E = \sum_{n=0}^{N} |x[n]|^2 \tag{2.1}$$

The *average power* within that interval is defined as the energy per sample:

$$P = \frac{1}{N+1} \sum_{n=0}^{N} |x[n]|^2 \tag{2.2}$$

The square root of the average power is often used, since it has the same unit as the signal amplitude. It is called the *root mean square (RMS)* value of the signal:

$$RMS = \sqrt{\frac{1}{N+1} \sum_{n=0}^{N} |x[n]|^2} \tag{2.3}$$

[1]The square brackets *[ ]* are used to denote a discrete variable, while the round brackets *( )* denote a continuous variable.

2.1.2 The Short Time Fourier Transform

In many signal processing applications it is most useful to represent a time-varying signal as a linear combination of sinusoidal (or, more generally, complex exponential) functions. This is called the representation of the signal in the *frequency domain*, or *spectrum*. The mathematical method that computes the spectrum from a signal is the *Fourier Transform* [29]. For a discrete signal $x[n]$, the Fourier Transform is defined by:

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-jn\omega} \tag{2.4}$$

$\omega = 2\pi\frac{f}{f_s}$**:** Normalized circular frequency

f**:** Frequency

f_s**:** Sampling frequency

The Fourier Transform of a discrete signal is always periodical in ω with period $\omega_0 = 2\pi$. If the signal $x[n]$ is real-valued (as in the case of an audio signal), its magnitude spectrum is evenly symmetric[2] about the zero axis.

As can be seen from the above definition, the Fourier Transform assumes that the input signal is infinitely long. Also, the resulting frequency-domain representation is continuous (since the frequency f is a continuous variable). However, none of these facts is desired if computer-based processing is desired, where quantities (that is, signals *and* their spectra) should have a discrete and finite nature.

The *Discrete Fourier Transform (DFT)* solves both problems at once. If we assume that $x[n]$ is only N samples long, we can define the *N-point DFT* as:

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn} \tag{2.5}$$

Thus, the finite signal can be represented as a discrete spectrum consisting of only N frequency coefficients placed at N evenly spaced *frequency bins*. The DFT can be viewed as a sampled period of the Fourier Transform of the signal, which indicates that, for real-valued signals, it is always evenly symmetric about its middle coefficient $k = N/2$. This coefficient corresponds to the highest frequency the DFT can represent, which is called the *Nyquist frequency*. In general, the frequency represented by each transform coefficient k is given by

$$f[k] = \frac{kf_s}{N}, \qquad 0 \leq k \leq N/2 \tag{2.6}$$

It can be seen from this equation that the Nyquist frequency equals half the sampling frequency f_s.

[2] A function f is said to be evenly symmetric if $f(x) = f(-x)$.

A highly efficient algorithm for computing the DFT, called the *Fast Fourier Transform (FFT)* has allowed the wide deployment of Digital Signal Processing technologies. In order for the FFT to be computed efficiently, N must be a power of two.

In this work, the discussion will be restricted to audio signals. Although all audio inputs are in fact finite, it is not possible to apply the DFT directly to them because of the following reasons:

- Usually, the length of the input signal is unknown, that is, we do not know the parameter N needed to compute the DFT.
- The computational costs for computing the DFT grow exponentially with N, becoming impractical for large values of N, even if we use the FFT.
- The audio signals are stochastic processes whose spectral characteristics vary in time.

Intuitively, the solution to this problem is to divide the input signals into blocks, and to compute the DFT of each block, thus obtaining a time-dependent DFT. More formally, breaking the input signal $x[n]$ into blocks can be regarded as multiplying it with a *window function* $w[n]$ that "jumps" along the signal every R samples. The window length L should be chosen to assure that the spectral characteristics of the signal remain reasonably stationary during that interval of L samples. The trade-off between temporal and spectral resolutions must also be regarded. In the general case, the window length L must not necessarily equal the step length R, and thus the windowed signal segments can be notated as:

$$x_r[n] = x[rR + n]w[n], \quad 0 \leq n \leq L - 1 \tag{2.7}$$

Throughout this work, the term *frame* will be used to denote a signal segment. We will use r as the frame index, whose range will depend on the total length of the analyzed signal.

The *Short Time Fourier Transform (STFT)*(in this case, it would be more accurate to speak about the Short Time DFT) is the time-dependent Fourier Transform resulting of computing the DFT of each signal frame. Using the above notation we can write the STFT as:

$$X_r[k] = \sum_{n=0}^{L-1} x[rR + n]w[n]e^{-j\frac{2\pi}{N}kn}, \quad 0 \leq k \leq N - 1 \tag{2.8}$$

N**:** DFT length

L**:** Window length

R**:** Step length

According to the modulation property of the Fourier transform, the transform of the product of two signals equals the convolution of their transforms. Therefore, when windowing a signal, the original spectrum will be altered as a result of

its convolution with the Fourier transform of the window function. Just dividing the signal into blocks corresponds to using a rectangular window. The Fourier transform of a rectangular window is a *sinc* function, which leads to high spectral deformations when convoluted with the original spectrum. To avoid this, many different window functions with transforms producing less degradations have been proposed in the literature [17]. One of the most widely used window functions in audio signal processing is the *Hamming* window, defined by:

$$w[n] = 0.54 - 0.46 \cos \frac{2\pi n}{L}, \quad 0 \leq n \leq L - 1 \tag{2.9}$$

N, L and R are the three parameters that determine the time and frequency accuracy and the computational performance of a Fourier Analysis using the Short Time DFT. The meaning of these parameters is shown graphically in Fig. 2.1. Throughout this work, we will refer to them as *DFT (or FFT) length* N, *window length* L and *step length* R, respectively.

To allow perfect reconstruction of one block from its transform, we must select $L \leq N$. When $L < N$, the last $N - L$ samples of the block must be *zero-padded* (replaced by zeros). Also, to avoid data loss, we must select $R \leq L$. If $R < L$, there is an overlapping of contiguous segments. These considerations lead to the following design constraint for the three parameters: $N \geq L \geq R$.

2.1.3 Power Spectrum and Parseval's Theorem

Parseval's theorem in the context of the discrete STFT states that the signal energy within an analysis window equals the scaled sum of the squared magnitude

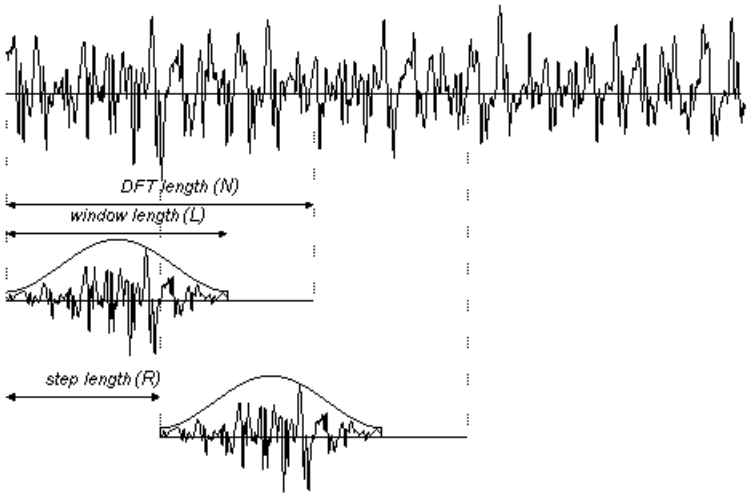


Figure 2.1: Signal windowing and Short Time Fourier Transform parameters.

of the DFT coefficients:

$$E_r = \sum_{n=0}^{L-1} |x_r[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X_r[k]|^2 \tag{2.10}$$

This allows the expression $\frac{1}{N}|X_r[k]|^2$ to be interpreted as the *energy density spectrum*, that is, the energy concentrated at each frequency bin. Similarly, the *power density spectrum* or, short, the *power spectrum* for frame r is given by:

$$P_r[k] = \frac{1}{LN}|X_r[k]|^2 \tag{2.11}$$

2.1.4 The Discrete Cosine Transform

In general, a *signal transformation* is the representation of a signal as a linear combination of a set of orthogonal *basis functions*. In the case of the DFT, these basis functions are periodic complex exponentials of the form $e^{-j\frac{2\pi}{N}kn}$ (see Eq. 2.5). One property of the DFT is that it yields complex coefficients, even when the signal $x[n]$ is real-valued. This motivated the search for other transforms that would yield only real coefficients.

One of these real-valued transforms is the *Discrete Cosine Transform (DCT)* [29], in which the basis functions are cosines. The N-point DCT of a sequence of N samples is defined as:

$$X[k] = \sqrt{\frac{2}{N}}\alpha[n] \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi k(2n+1)}{2N}\right), \quad 0 \leq k \leq N-1 \tag{2.12}$$

where $\alpha[n]$ is given by

$$\alpha[n] = \begin{cases} \frac{1}{\sqrt{2}} & if \quad k = 0 \\ 1 & if \quad 1 \leq k \leq N-1 \end{cases} \tag{2.13}$$

The DCT has the property that its energy is concentrated in its lowest coefficients. Discarding the higher coefficients does not significantly affect the reconstruction of the original signal by inverting its DCT. For this reason, the DCT has found its main application in the compression of data.

2.2 Psychoacoustics

The goal of *Psychoacoustics* is to analyze and model the behavior of the human hearing system [52]. It is well known that the ear perceives some acoustical

magnitudes in a nonlinear way, thus converting objective, physical measures in subjective, "aurally adequate" measures. Several psychoacoustical models have been proposed to adapt objective measures such as intensity and fundamental frequency to perceptual ones such as loudness and pitch. Many audio processing systems benefit enormously by making use of such a perceptual adaptation; the most representative example thereof are the perceptual audio encoders.

2.2.1 Loudness: Simple Model

The intensity I of a sound is a physical measure defined as the sound power transmitted through a given area in a sound field. Early psychoacoustical experiments showed that a sound with intensity I is not perceived as being twice as loud as a sound with intensity $I/2$. It was found that the perceived loudness, measured in *sones*, could be approximately related to the intensity following a simple exponential law [40]:

$$L = kI^{\alpha} \tag{2.14}$$

An exponent value of $\alpha = 0.23$ has proven to be adequate in the case of noise, and is also likely to be applicable to other broadband sounds such as music [52, 27]. Since intensity is linearly related to sound power P or energy E, any of these quantities can replace I in the above equation, the only difference being that the constant k will take a different value in each case.

The perceived loudness also depends on the frequency contents of the sound. This fact is taken into account in other more complex models that divide the spectrum in *critical bands* and analyze their individual influence on the final loudness by considering partial masking effects [52].

2.2.2 Pitch: The Mel Scale

The pitch of a sound depends closely on its fundamental frequency. However, as in the case of loudness, the two magnitudes are not related linearly. Several models have been proposed to describe this nonlinearity in pitch perception [41].

One of the most commonly used is the *mel scale*. It was the result of a set of experiments in which the test subjects were asked to judge when a pure tone seemed to have half the pitch of another previously played tone. As expected, at low frequencies, the relationship is linear, that is, a 220 Hz pure tone was perceived as having half the pitch of a 440 Hz tone. But at higher frequencies the relationship becomes more and more logarithmical. For example, a 1300 Hz pure tone was, in average across the subjects, perceived as having half the pitch of a 8000 Hz tone.

The mel scale transforms the frequencies in Hz into *mel frequencies* with constant "half pitch ratio". That is, a tone of M mel will always be perceived as

having half the pitch of a tone of $2M$ mel. The unit 1 mel is defined such that a 1000 Hz-tone has a pitch of 1000 mel. The relationship is approximately linear below 1kHz and logarithmic above. Pitch measured in mel is the perceptual counterpart of fundamental frequency measured in Hz.

Equation 2.15 shows one of the proposed analytical definitions for the mel scale, which is plotted in Fig. 2.2.

$$pitch(mel) = 2595 \log_{10} \left(1 + \frac{f(Hz)}{700}\right) \tag{2.15}$$

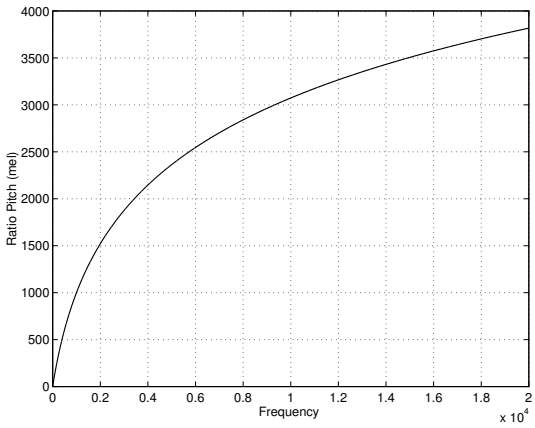


Figure 2.2: The mel scale.

2.3 Random Vectors

This section introduces the principles of mathematical statistics used throughout the present work. We begin by considering one-dimensional random variables and later generalizing them with a discussion of the multidimensional (or *multivariate*) case.

2.3.1 Random Variables and their Statistical Characterization

Although a random variable[3] can be fully characterized by its *probability density function (pdf)*, it is also possible to partially describe it using the so-called *moments*. This is particularly useful when the *pdf* is not known beforehand, or when it would be computationally costly to determine it, as it is often the case.

[3]Throughout this work, only continuous random variables will be considered.

Moments of a random variable

The nth *moment* α_n of the random variable x is defined by

$$\alpha_n = \int_{-\infty}^{\infty} x^n p(x) dx \tag{2.16}$$

where $p(x)$ is the *pdf* of x.

The first moment is called the *expectation* or *mean* of the random variable:

$$\mu = E\{x\} = \int_{-\infty}^{\infty} x p(x) dx \tag{2.17}$$

The *mean* is the most probable value of the random variable, and corresponds to the value x for which $p(x)$ is maximum.

The nth *central moment* of x can be defined in terms of the expectation as

$$\xi_n = E\{(x-\mu)^n\} = \int_{-\infty}^{\infty} (x-\mu)^n p(x) dx \tag{2.18}$$

The second central moment is called the *variance* of the random variable:

$$\sigma^2 = E\{(x-\mu)^2\} \tag{2.19}$$

The variance and its square root, the *standard deviation* σ, are measures of the spread of the possible values of x around its mean.

Estimation of Moments

The mean and the variance or standard deviation are the most commonly used parameters to describe a probability distribution. But, as stated before, the most common situation in statistics is when we do not know the exact *pdf*, but have instead a set of *samples*[4] from the random variable. In this case, the samples are used to estimate the moments of the distribution using the so called *moment estimators*. The estimator for the mean is called the *sample mean* and is simply the average value of the observed samples:

$$\hat{\mu} = \frac{1}{N} \sum_{j=1}^{N} x_j \tag{2.20}$$

where x_j are the observed samples and N is the total number of samples. The hat notation ($\hat{}$) denotes estimation. The estimator for the variance is called the *sample variance* and is defined by

[4]These samples should not be confused with the samples of an audio signal. Throughout this work, we will write "audio samples" every time the samples of a sound signal are meant, and just "samples" when values drawn from a random variable or random vector are meant.

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{j=1}^{N}(x_j - \hat{\mu})^2 \tag{2.21}$$

The *sample standard deviation* $\hat{\sigma}$ is defined as the square root of the sample variance.

In order to simplify terminology, it is common use to name the sample mean and the sample variance simply as *mean* or *variance*, although actually referring to its estimations. This simplification will be also adopted in the present work, since the exact meaning can always be extracted from the context. However, the subtle differences should be kept in mind.

High-order Statistics

In contrast to the first and second-order moments described above, there are several possible definitions for the higher order moments that follow. Although the statistical meaning of these measures is always the same across definitions, their value range and scaling can vary significantly. In the following, the definitions used in this work are presented.

The *skewness* ξ_3 of a random variable is defined here as its third central moment divided by the cube of its standard deviation:

$$\xi_3 = \frac{E\{(x-\mu)^3\}}{\sigma^3} \tag{2.22}$$

The skewness is a measure of the asymmetry of the *pdf*, and equals zero for densities that are symmetrical about their mean. It is negative if the data is spread out more to the left of the mean than to the right and positive if the data is spread out more to the right.

Similarly, the *kurtosis* ξ_4 is defined as the fourth central moment divided by the fourth power of the standard deviation:

$$\xi_4 = \frac{E\{(x-\mu)^4\}}{\sigma^4} \tag{2.23}$$

The kurtosis measures the "nongaussianity" of a random variable, that is, how far does the form of its *pdf* differ from that of a normal distribution (see Sect. 2.3.3). The kurtosis of the normal distribution is 3. If the kurtosis is greater than 3, the distribution is called *supergaussian* or *leptokurtic*. If it is less than 3, it is said to be *subgaussian* or *platykurtic*. Supergaussian distributions tend to have a sharper peak and longer tails than the normal *pdf*, while subgaussian distributions tend to be flatter.

As every statistical moment, skewness and kurtosis can be estimated from N samples by the *sample skewness* and *sample kurtosis* defined by:

$$\hat{\xi}_3 = \frac{1}{N\hat{\sigma}^3}\sum_{j=1}^{N}(x_j - \hat{\mu})^3 \tag{2.24}$$

$$\hat{\xi}_4 = \frac{1}{N\hat{\sigma}^4}\sum_{j=1}^{N}(x_j - \hat{\mu})^4 \tag{2.25}$$

2.3.2 Random Vectors and their Statistical Characterization

A D-dimensional random variable can be represented mathematically as a D-element *random vector*, that is, a vector whose elements are random variables. Random vectors are usually defined as column vectors:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{pmatrix} = (x_1, x_2, \ldots, x_D)^T$$

where T denotes the transpose.

A random vector is fully characterized by its *joint probability density function* $p(\mathbf{x})$. The term *joint pdf* emphasizes that a complete description of a multidimensional random variable must not only account for the statistical behavior of each of its elements, but also for the relationships between them. In contrast, the *marginal pdfs* describe each of the vector elements regarded independently, and correspond to the *pdf*s $p_i(x_i)$ of the constituent random variables.

Vector Moments and their Expectations

As in the one-dimensional case described in the preceding section, the joint *pdf* of a random vector can be described partially by its moments. The multidimensional version of the mean is the *mean vector*:

$$\boldsymbol{\mu} = E\{\mathbf{x}\} = (E\{x_1\}, E\{x_2\}, \ldots, E\{x_D\})^T \tag{2.26}$$

As can be seen, the components of the mean vector are the means of the components of the random vector, as defined in Eq. 2.17. Using the same analogy, if N *sample vectors* are available, its *sample mean vector* is given by[5]

[5]In this work, the following convention will be used: the total number of dimensions will be denoted by D, and dimensional indices for the elements of the vectors will be denoted by i. The total number of samples will be denoted by N, and the sample indices by j.

$$\hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{x}_j \tag{2.27}$$

The multidimensional counterpart of the variance is the *covariance matrix*, also called *autocovariance matrix*, given by

$$\boldsymbol{\Sigma} = E\{(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T\} = \begin{pmatrix} \sigma_1^2 & \dots & c_{1D} \\ \vdots & \ddots & \vdots \\ c_{D1} & \dots & \sigma_D^2 \end{pmatrix} \tag{2.28}$$

The diagonal components σ_i^2 are the variances of the components of the vector, while an off-diagonal element c_{ij} is called the *covariance* of variables x_i and x_j. All covariance matrices are symmetric, that is, $c_{ij} = c_{ji}$ for all i, j.

Covariances measure the degree of correlation between two random variables. Intuitively, two random variables x_1 and x_2 are correlated if it is possible to infer anything about the value of x_1 by observing x_2, or vice versa. If the variables are *uncorrelated*, $c_{ij} = 0$.

It is also possible to interpret the covariance matrix graphically: it describes the multidimensional spreading of the samples around the point defined by the mean vector.

The estimator of the covariance matrix is the *sample covariance matrix*:

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{j=1}^{N}(\mathbf{x}_j-\hat{\boldsymbol{\mu}})(\mathbf{x}_j-\hat{\boldsymbol{\mu}})^T \tag{2.29}$$

2.3.3 The Multivariate Normal Density

The most widely used joint probability function in multidimensional statistics is the *Normal* or *Gaussian Density*, not only because it models accurately many random experiments, but also because its mathematical simplicity. It generalizes the well-known one-dimensional normal density. For D dimensions, it is written as

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{2.30}$$

As it is completely determined by its mean vector and covariance matrix, it is common to use the notation $p(\mathbf{x}) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ (the number of dimensions is implicitly indicated by the sizes of the vector or the matrix). In this context, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are called the *parameters* of the distribution.

Figure 2.3 illustrates a two-dimensional normal density. The left figure is a three-dimensional representation ($p(\mathbf{x})$ as a function of the constituent random

variables x_1 and x_2) of the so-called *Gauss bell* described by the joint *pdf*. It can be seen that the mean vector $\boldsymbol{\mu}$ points to the center of the bell. The right figure shows the corresponding *contour plot*. Ellipses denote levels of equal probability density and are always concentric around the mean. All the points on an ellipse have the same so-called *Mahalanobis distance* to the center $\boldsymbol{\mu}$. The multidimensional counterparts of the ellipses are the *hyperellipsoids*. The form and orientation of the ellipses or hyperellipsoids with regard to the space axes is determined by the covariance matrix. A diagonal covariance matrix corresponds to ellipses or hyperellipsoids whose principal axes are parallel to the space axes. A diagonal covariance matrix with equal diagonal elements (that is, with equal variances) corresponds to circles in the two-dimensional case, and to *hyperspheres* in the multidimensional case.

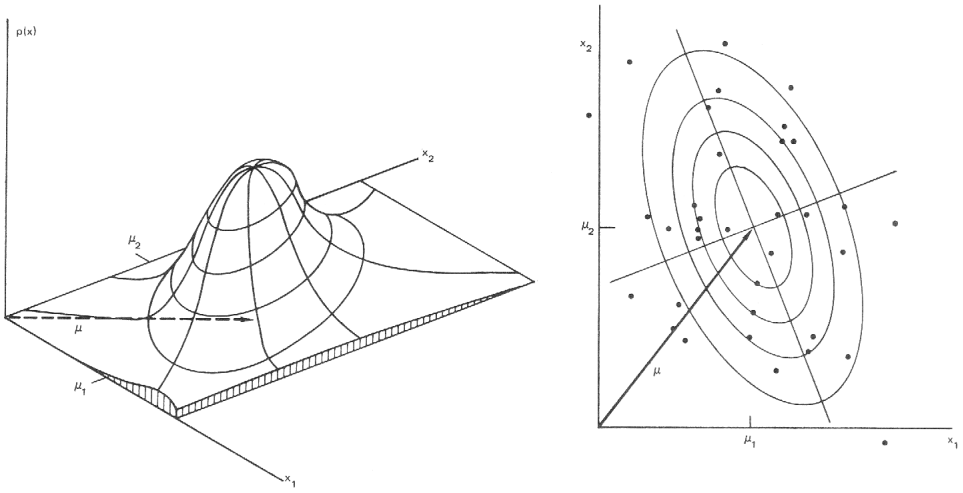


Figure 2.3: Three-dimensional (left) and contour plots from a two-dimensional normal density (from [8]).

2.4 Pattern Recognition

Pattern recognition (PR) or *pattern classification* can be defined in a broad sense as *"the assignment of a physical object or event to one of several pre-specified categories"*[7]. Human identification of known objects or sounds are examples of PR acts. The science of *mathematical pattern recognition* [8, 7, 16, 42, 26] seeks to formulate mathematically any classification process, its two main motivations being, first, to gain insight into the biological and psychological processes that take place in human or animal perception, and second, to allow an automated implementation of these procedures using computers.

Any classification application in the context of Content Analysis is based on the mathematical tools provided by PR theory. In the computer vision field, existing applications include scanner recognition of printed or handwritten characters and extraction of objects from complex images. Automatic Speech Recognition is also a classification process in which a given spoken fragment (be it a sentence, a word or a phoneme) is classified into a possible transcription. Other applications include financial predictions, weather forecasts and medical diagnosis. In this work, PR will be used to classify an audio signal into one of several categories.

One key idea underlying any human or computing PR process is that, in order to classify an object, there must be some kind of *previous knowledge* about it. Any classification relies on a certain *decision rule* that was derived from previous experience. For example, we are able to read a given word because at some point in time, we learned to recognize the patterns that form the characters. Similarly, a computer must be *trained* before it can perform any kind of recognition. Any PR algorithm has a *learning* or *training phase* and a *classification phase*. Regarding the way in which this training and classification is made, we can distinguish three different computing PR approaches:

Statistical Pattern Recognition. In this case, the previous knowledge consists of a set of observed samples from the observation object. In the training phase, a probabilistic decision rule is derived from the samples and is applied to an unknown object to be recognized in the classification phase. In a musical analogy, this corresponds to judging the genre of an unknown piece having heard similar pieces before, which we know belong to that specific genre. This is the most widely used approach of PR and, as it is also adopted in the present work, it will be presented in more detail in the next sections.

Syntactic Pattern Recognition. This approach is used if we have a complete knowledge about the syntactical or structural rules that form the objects. Following the same musical analogy, it corresponds to finding musical genre by detecting certain melodic, harmonic or structural characteristics that we know are typical for that genre, for example by reading the score or hearing *analytically* to the unknown piece. If we have enough previous knowledge of music theory, it would be theoretically possible to guess the genre correctly even if we had not listened to any other pieces of the same genre before. In this case, training consists of implementing these construction rules on the computer; no training samples are needed. Therefore, classification is performed using exactly defined heuristic rules, rather than probabilistic rules.

Neural Pattern Recognition. A set of samples is also available *a priori*, but rather than estimating densities from it, they are used as stimuli to train a *neural network*. The network is capable of dynamically adapt its decision

rules based on the samples, until it has reached an optimal separation of classes.

In the cases where the training is based on a set of samples, another classification of PR algorithms is possible, this time regarding the amount of prior knowledge. It can be distinguished between

Supervised Learning. In this case, the classes to which the training samples belong are known beforehand. In PR jargon it is said that the samples are *labeled*. Thus, the recognition algorithm is designed to perform classification into an initially desired class structure.

Unsupervised Learning is used if the training set is not labeled. In this case, the separation into classes is up to the algorithm, which groups the samples according to some measure of similarity. This process is called *clustering*.

Many possible combinations of the above described methods have been proposed to perform sound classification (see Chapter 3). In this work, a *statistical*, *supervised* approach is used. Unsupervised learning is discarded because the goal here is to implement a *predefined* class taxonomy (see Chapter 4). A syntactic approach is discarded because it would require the music to be in symbolic form, or to transcribe the raw music data into scores. The main drawback of the Neural Network approach is that long training time is required, the final performance being similar to that of statistic approaches [43].

The next sections introduce the principles of *statistical pattern recognition*.

2.4.1 Feature Vectors and Classifiers

In order to classify an incoming object, some measures or *features* must be extracted from it in first place. In pattern recognition, a set of D features extracted from a sample is represented as a D-dimensional random vector $\mathbf{x} = (x_1, x_2, \ldots, x_D)^T$ called *feature vector* or *pattern*, and the corresponding D-dimensional vector space is called the *feature space*. Each sample is represented as the point in the feature space defined by its feature vector. This is shown for a two-dimensional case in the *scatter plot* of Fig. 2.4, where samples belonging to different classes are marked by different symbols, and separated by *decision boundaries*. The regions between boundaries are called *decision regions*.

Graphically, classifying means to find in which decision region falls a given feature vector, and assigning to it the *class* ω_k corresponding to that region[6]. The goal of pattern recognition is to use the available training samples to find decision boundaries that separate the classes in an optimal way.

[6]We will use k for the class index and C for the total number of classes.

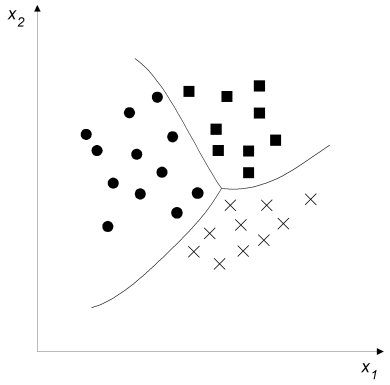


Figure 2.4: Two-dimensional feature space. Samples belonging to different classes are marked by different symbols. Curves separating classes are the decision boundaries, which are optimal in this ideal case.

This intuitive idea of classification can be expressed more formally as follows: given C classes, a set of C *discriminant functions* $g_k(\mathbf{x})$ is defined. Classification of a feature vector $\mathbf{x}$ consists to

$$\text{Decide } \omega_i \text{ if } \; g_i(\mathbf{x}) > g_j(\mathbf{x}) \quad \text{for all } j \neq i \tag{2.31}$$

Thus, a *classifier* can be viewed as an algorithm that evaluates all discriminant functions for a given feature vector and assigns the class corresponding to the largest discriminant. The goal of the training phase is to derive this set of discriminant functions. Many different classifiers have been proposed; some of them will be discussed later. Equation 2.31 defines the *decision rule* of a classifier in its most general form.

The relationship between graphical and analytical interpretations is the following: if the decision regions for classes ω_i and ω_j are contiguous, the decision boundary separating them is defined by the equation $g_i(\mathbf{x}) = g_j(\mathbf{x})$.

Figure 2.5 summarizes the concepts introduced in this section.

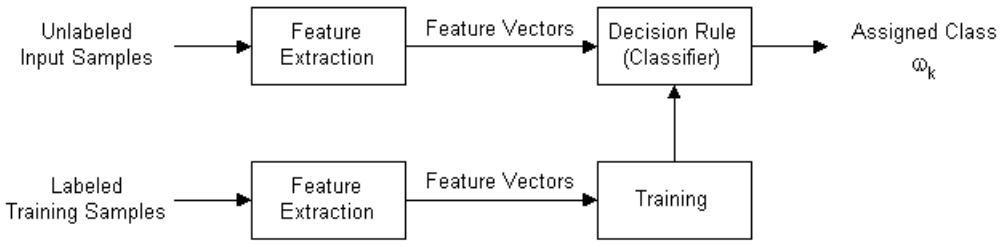


Figure 2.5: Overview of statistical Pattern Recognition.

2.4.2 Bayes Decision Theory

As mentioned above, the central problem in statistical pattern recognition is that of finding the set of discriminant functions for a classifier. The difficulty of this problem will depend on the amount of information about the classes known beforehand. The ideal case is that in which the probability structure of the problem (that is, the probability density function defining each class) is perfectly known. The *Bayes Decision Theory* describes the classification problem when the *pdf*s are known. Although in practice this is very rarely the case, it is useful to understand the principles of Bayes classification, as they provide the general background for the rest of the problems. Also, the Bayes classifier is an optimal classifier, which means that *no other classifier can perform better*. Therefore, it is often taken as a benchmark to compare performance.

The *pdf* describing each class is written as $p(\mathbf{x}|\omega_k)$, that is, the conditional *pdf* of $\mathbf{x}$ given the class ω_k. This quantity is also called the *likelihood* of that class with respect to $\mathbf{x}$.

It should be noted that, in general, some classes in a classification problem are more probable than others (for example classical music signals are more probable than rock music signals if our input stream is a classical radio station). Therefore, each class is also associated with its *a priori probability* $P(\omega_k)$[7].

But what we want to know in order to make a classification is how probable a class ω_k is, given an observation $\mathbf{x}$. This is the so-called *a posteriori probability* $P(\omega_k|\mathbf{x})$, and its relationship to the class likelihood is provided by the *Bayes Rule*:

$$P(\omega_k|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_k)P(\omega_k)}{p(\mathbf{x})} \tag{2.32}$$

$p(\mathbf{x}|\omega_k)$**:** Likelihood

$P(\omega_k)$**:** A priori probability

$P(\omega_k|\mathbf{x})$**:** A posteriori probability

$p(\mathbf{x}) = \sum_{k=1}^{C} p(\mathbf{x}|\omega_k)P(\omega_k)$

As intuition suggests, we will assign to $\mathbf{x}$ the class ω_k for which the a posteriori probability is highest, thus obtaining the following decision rule:

$$\text{Decide } \omega_i \text{ if } \; P(\omega_i|\mathbf{x}) > P(\omega_j|\mathbf{x}) \quad \text{for all } j \neq i \tag{2.33}$$

That is, the selected class maximizes the a posteriori probability. Consequently, this decision rule is called the *Maximum A Posteriori (MAP)* criterion. The term $p(\mathbf{x})$ in Eq. 2.32 is only a scale factor and does not affect the decision.

[7]Lower case ps denote probability densities, while capital Ps denote probabilities, which always range from 0 to 1.

When comparing equations 2.31 and 2.33 we conclude that a MAP classifier is the one whose discriminant functions are given by $g_k(\mathbf{x}) = P(\omega_k|\mathbf{x})$.

In many problems, all classes are equally probable a priori. In this case, the $P(\omega_k)$ term in Eq. 2.32 is constant for all k and therefore it does not affect the decision. Thus, with equal priors, maximizing the posterior probabilities is the same as maximizing the likelihoods, obtaining following rule:

$$\text{Decide } \omega_i \text{ if } \; p(\mathbf{x}|\omega_i) > p(\mathbf{x}|\omega_j) \quad \text{for all } j \neq i \tag{2.34}$$

which is called the *Maximum Likelihood (ML)* criterion. To decide a class for a given feature vector, we evaluate each conditional *pdf* and select the one that provides a higher value. The discriminant functions of a ML classifier are $g_k(\mathbf{x}) = p(\mathbf{x}|\omega_k)$.

In this work it is assumed that all audio classes are equally probable, and therefore the ML criterion will be used. In this case, if we choose to model each class by a normal density, the discriminant functions $g_k(\mathbf{x})$ have the form of Eq. 2.30, obtaining a *simple Gaussian (GS)* classifier. A *Gaussian Mixture Model (GMM)* classifier models each class as a linear combination of normal densities, and it will be reviewed in more detail in Section 7.1.

2.4.3 Maximum Likelihood Estimation

When the exact densities of the classes are not known, which is the most common situation in pattern recognition, there are two possible ways to perform classification: one approach is to estimate the class densities from the training samples and then applying Bayes Theory to perform classification. The other possibility is to perform classification based directly on the observed samples without making any probabilistic estimation. We will address the first way (*density estimation*) in the present section, and the second (*nonparametric classification*) in the next one.

In the context of density estimation, there are furthermore two possible situations:

- We do not know the exact density functions, but we do know their form. For example, we know that the classes are described by normal densities but we do not know their parameters, that is, their means and covariances.
- We neither know the parameters nor the form of the densities.

In the first case we use one of the *parameter estimation* techniques to obtain an approximate model of the classes. The second situation requires the more difficult problem of estimating the whole density function, and makes use of *nonparametric estimation* techniques. Here it will be assumed that the density forms are known and the most common method for parameter estimation, the *Maximum Likelihood*

Estimation (not to be confused with the Maximum Likelihood criterion of the preceding section), will be introduced.

Formally, we have C different densities $p(\mathbf{x}|\omega_k)$ of a given form, but each of them having a set of unknown parameters which is represented as the *parameter vector* $\boldsymbol{\theta}_k$ for that class. For example, if the densities are gaussian, then $\boldsymbol{\theta}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

To denote that the densities are now also dependent on a given set of parameters, they are now written as $p(\mathbf{x}|\omega_k, \boldsymbol{\theta}_k)$. Assuming that a training set $\mathcal{X}_k$ consisting of N training samples $\mathbf{x}_j$ is available for a given class ω_k, the likelihood of $\boldsymbol{\theta}_k$ with respect to the set of samples can be written as

$$p(\mathcal{X}_k|\boldsymbol{\theta}_k) = \prod_{j=1}^{N} p(\mathbf{x}_j|\omega_k, \boldsymbol{\theta}_k) \tag{2.35}$$

This measures how likely it is that the given samples were drawn from a density function described by $\boldsymbol{\theta}_k$. Training by ML parameter estimation consists of finding the parameter vector $\hat{\boldsymbol{\theta}}_k$ that maximizes the previous expression for a given class.

In the particular case of a normal distribution, the ML estimation of its mean is none other than its sample mean $\hat{\boldsymbol{\mu}}$ (Eq. 2.27). The ML estimation of its covariance is a biased version of its sample covariance matrix $\hat{\boldsymbol{\Sigma}}$ (Eq. 2.29). ML estimation of a gaussian mixture density will be addressed in Sect. 7.1.

2.4.4 Nonparametric Classification

As mentioned in the preceding section, it is also possible to perform classification directly based on the training data, without previous density or parameter estimation. This is called *nonparametric classification* (not to be confused with *nonparametric density estimation*), and its two most important algorithms are the *Nearest Neighbor (NN)* classifier and the *k-Nearest Neighbor (kNN)* classifier.

The very intuitive NN rule consists of assigning to the unlabeled feature vector the label of the training feature vector that is nearest to it in the feature space. The distance can be measured using one of several existing metrics, but usually the *Euclidean Distance* is used, which is given by

$$d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{\sum_{i=1}^{D} (x_{1,i} - x_{2,i})^2} \tag{2.36}$$

A NN classifier partitions the feature space into decision regions formed by a set of so-called *Voronoi cells*. Each training feature vector is surrounded by its corresponding cell, which is formed by all the points of the feature space that are nearer to it than to any other feature vector. This is illustrated in Fig. 2.6 for a two-class and two-feature problem.

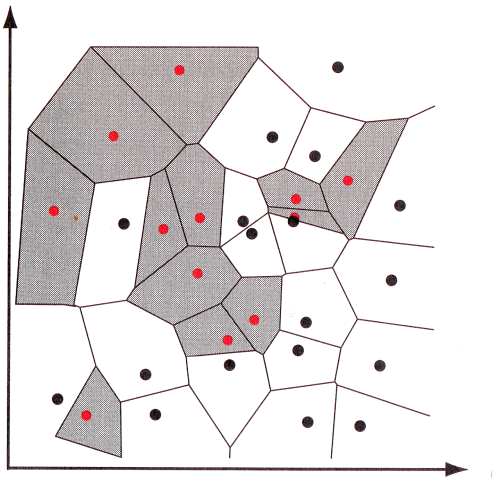


Figure 2.6: Decision regions of a Nearest Neighbor classifier (from [7]). The decision regions are formed by the *Voronoi cells* associated to each of the samples, which are represented by points in the figure.

The kNN rule extends the NN rule by examining the k nearest training samples to the observed vector. It assigns the label which is most frequent among these k samples. Usually, choosing moderate values for k improves performance in comparison with the NN rule, because it yields smoother decision boundaries and provides more probabilistic information. However, large values for k can be detrimental, not only because the increased computation complexity, but because it would destroy the locality of the estimation by considering samples that are too far away.

It can be shown that a kNN classifier will never perform worse than twice the error rate achieved by a Bayes classifier [8].

From a computational point of view, a kNN classifier[8] requires to store all feature vectors of the training database in order for the input vector to be compared with each of them. In contrast, density-estimation-based classifiers, such as a GS or a GMM, need only to compute and store the C parameter vectors $\boldsymbol{\theta}_k$, making them more computationally efficient at *classification time*. kNNs require no *training time*, as long as all feature vectors have alredy been computed and stored.

2.4.5 The Design Process of a Pattern Classifier

Figure 2.7 shows the whole design process of a classifier. As the first step, a database of training samples must be collected. Then, it must be decided which

[8]The NN classifier will always be considered as a particular case of the kNN classifier.

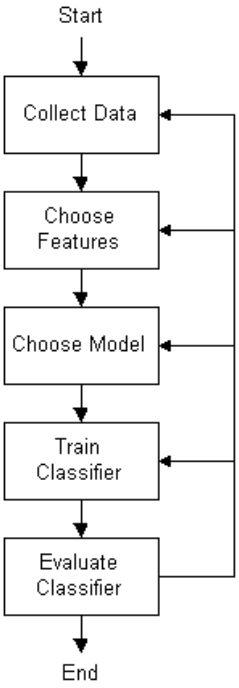


Figure 2.7: Design cycle of a classifier (adapted from [7]).

features will be extracted to perform the classification. Choosing appropriate features, or designing new ones must be made very carefully, as it will highly affect the overall performance.

Feature design is said to be the *domain-dependent* part of classifier design, meaning that it requires an in-depth knowledge of the specific field of application. On the other hand, classification algorithms regard input data as a set of numbers, without any knowledge about their true nature. Therefore, choosing the classifier model is the *domain-independent* part. Due to its generality and domain independence, classifier algorithms have been thoroughly discussed in the literature as general purpose, abstract tools that can be applied to any input data. In contrast, feature design, and especially audio feature design, has received little attention up to now.

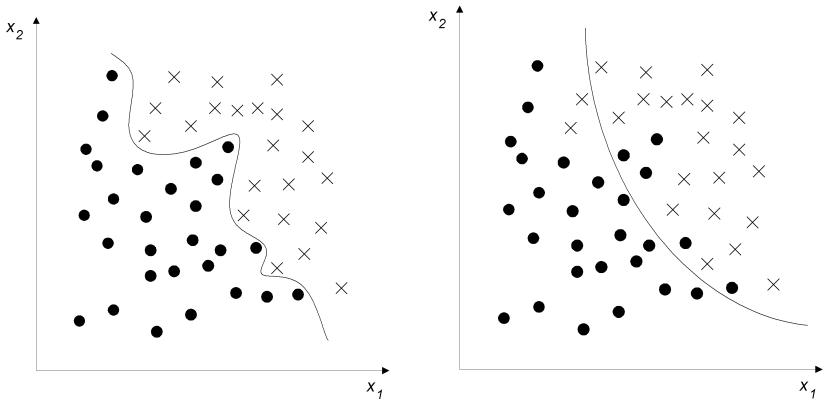
Once the features and the classifier have been chosen, the latter must be trained using the collected data. At this point we have reached a complete working system and we are ready to evaluate its performance. However, as it is shown in the figure, backward modifications on any of the previous stages can be made observing the feedback provided by the evaluation.

This outline is closely followed in this work. The selection of features, as well as the design of new ones will be described in Chapters 5 and 6. In Chapter 7, several classification algorithms will be evaluated, not only with regard to their

correct classification rate, but also considering other properties like computational performance and flexibility. These considerations led to the selection of one of the models, which was implemented in the final prototype tool.

The designer of a classifier must confront problems inherent to pattern recognition that often defy intuition. The most important of these issues are the problem of *overfitting* and the problem of *high dimensionality*.

Overfitting. Intuition suggests that it is desirable to have as many training samples available as possible. However, if we fit the decision rules too closely on the training data, the classifier will perform badly on new, unknown data. This idea is shown in Fig. 2.8. In Fig. 2.8(a), the obtained boundary perfectly separates the training samples, but will not work well on new samples. The boundary in Fig. 2.8(b) missclassifies some training data, but is expected to reflect the statistical behavior of data more accurately, and consequently will give a better overall performance. As a result, there is an optimal number of training samples upon which the performance will begin to decrease. The higher the underlying complexity of the data is, the higher is this optimal number of samples.



(a) Perfectly fit boundary (b) Generalized boundary

Figure 2.8: Illustration of the problem of overfitting. Figure 2.8(a) shows an overfitted decision boundary, while Fig. 2.8(b) shows a generalized decision boundary.

High Dimensionality. Similarly, one could expect that adding new features (that is, new dimensions) will always help improving the correct classification rate, even if the new features add few discriminating information. Practice has shown that this is not always the case. This apparent paradox, which has been called the *Curse of Dimensionality*, has attracted the attention of statisticians since many years, and there have been many attempts

to proof theoretically what has been observed in practice. This has important implications for the application-oriented designer, too, since reducing the number of features allows to reduce computational costs while keeping similar classification performance. In some cases, the classification rate can even benefit from the reduction in dimensionality. For these reasons, many *dimensionality reduction* methods have been proposed in the literature (see Sect. 6.2).

Furthermore, these two issues are closely related, as the optimal number of features closely depends on the number of available samples. Both can be regarded as particular implications of the general scientific and philosophical principle of *Occam´s razor*, which states that *one should not increase, beyond what is necessary, the number of entities required to explain anything.*

These problems and their implications in the specific audio application described in this work, will be addressed in Chapters 6 and 7.

Chapter 3

Related Work

Research projects about general audio classification have only begun in the last few years. Although ASR systems can also be viewed as classification systems, they will not be reviewed here, as we will restrict our discussion to general, categorical classification.

3.1 Overview of Audio Classification Systems

In the present overview, only systems based on *supervised learning* techniques will be considered. Such systems allow the implementation of a given taxonomy defined beforehand by the user or by the implementation itself, and therefore correspond to the scope of this work. Some music *clustering* systems based on unsupervised learning have also been proposed [11, 34, 15, 31], but will not be regarded here.

The following works will be reviewed in chronological order.

Wold, Blum, Keislar and Wheaton (1996)

One of the first general audio classification systems was developed by the Muscle Fish company in 1996 [48]. In this system there are no pre-specified classes, since the training is up to the user. Pitch, brightness modeled by the spectral centroid (Sect. 5.3.2), spread (Sect. 5.3.6.2) and mel frequency cepstral coefficients (MFCCs, Sect. 5.3.5) are used as the features. The system uses a simple gaussian (GS) classifier, and its main application is similarity retrieval. It has been tested using short, individual sound segments such as sound effects and musical instrument notes, which train very specific classes like laughter, animals, bells, crowds, water, etc. To our knowledge, is has not been tested using broader classes like musical genres.

Scheirer and Slaney (1997)

Another key application in audio classification is the discrimination between speech and music. A system proposed by Scheirer and Slaney [35] uses 13 features such as 4Hz modulation energy, rolloff (Sect. 5.3.3), centroid, flux (Sect. 5.3.4), zero crossings (Sect. 5.3.1) and beat features to distinguish between speech and music. Four different classifiers were evaluated: a GS classifier, a gaussian mixture model (GMM, see Sect. 7.1), a nearest neighbor (NN) classifier and a k-nearest neighbor (kNN) classifier. The four provided very similar classification performance. A final classification rate of 98.6% is reported. When using the system as a three-way classifier to separate music, speech and simultaneous speech and music, the rate drops to 65%.

Foote (1997)

Foote [12] proposed a tree-based quantizer as the classifier. This tree partitions the feature space into regions with feature vectors belonging to maximally different classes. MFCC coefficients are used as the features. A three-way classification between speech, music and non-vocal sound is implemented using this technique. No specific performance measures are provided in this paper.

Zhang and Kuo (1998)

Zhang and Kuo [50, 49] have developed a system that performs classification in a two-level hierarchy. At the first level, which they call the *coarse-level*, sound is classified into speech, music, environmental sound and silence using a model-free heuristic procedure. That is, there are no statistical rules derived from a density estimation, but a decision based on empirically obtained thresholds for each feature. The features used at this stage are the short-time energy, the zero crossings and the fundamental frequency. A classification accuracy of 90% is reported for the coarse level. At the second stage, or fine-level classification, sounds are further divided into finer classes. The work focuses on the classification of environmental sounds into 10 classes such as applause, birds, laugh, rain, etc. At this stage, the classification is based on modeling each class as a Hidden Markov Model (HMM). In contrast to the already reviewed statistical classifiers, such as the GS, GMM and kNN classifiers, a Hidden Markov Model accounts for the time-varying nature of features representing each class as a set of density functions with transition probabilities between them. Particularly, in this work, each state of the HMM is modelled by a GMM density. The spectrum coefficients are used directly as the fine-level feature, but its transitions across the HMM implicitly contain information about the rhythmic structure of the signal. At this level, an accuracy of 80% is reported.

Lambrou, Kudumakis, Speller, Sandler and Linney (1998)

In a work by Lambrou *et al.* [21] statistic measures such as mean, variance, skewness and kurtosis from the wavelet transform coefficients of the signal and the zero crossings are used as the features. Classification into three musical classes (rock, piano and jazz) is evaluated with a kNN classifier, and with three variations of the GS classifier. A classification accuracy of 91.67% is reported. However, since only 12 audio examples were used for training (4 for each class), it is questionable if this value is really representative for the overall performance of the system.

Soltau, Schultz, Westphal and Waibel (1998)

Soltau *et al.* [39] constructed a prototype system for classifying music into rock, pop, techno and classic. As it is stated in this paper, the major drawback of HMMs is their poor discriminative power, and therefore another temporal structure modeling technique based on neural networks, which achieve better discrimination, is presented. It is called Explicit Time Modeling with Neural Networks (ETM-NN). The features used are the activations of the hidden units of the neural networks. The ETM-NN approach achieved a classification rate of 86.1%, compared to the 79.2% achieved by the HMMs.

Pye (2000)

Pye [33] addresses classification of audio signals compressed in MP3 format. Two approaches are compared with respect to performance and computation cost. In the first, MFCCs are used as the features, which requires the previous MP3 decompression. The other proposed method consists of deriving a MFCC-like set of features performing only a partial decompression, and is called MP3CEP. GMM and a tree-based vector quantizer like the one used in [12] are the evaluated classifiers. Music classification is performed into the following 6 classes: blues, easy listening, classical, opera, dance (techno) and indie rock. The best classification rate for both the MFCC and MP3CEP was obtained using the GMM classifier. While MFCCs performed slightly better (92%) than MP3CEP (90.9%), in the latter case the computation was more than five times faster.

El-Maleh, Klein, Petrucci and Kabal (2000)

In a 2000 work by El-Maleh *et al.* [9] a speech/music discriminator is described using the Line Spectral Frequencies (LSF) in combination with zero-crossing based measures as the features. LSF is the result of applying a certain linear transformation to the Linear Prediction (LP) coefficients (see Sect. 5.5.6). A GS and a NN classifier are evaluated. The best classification rate (95.9%) was obtained with the GS classifier.

Deshpande, Nam and Singh (2001)

In Deshpande *et al.* [6], the MUGEC (Music Genre Classification) project is presented. Here, music is classified into rock, classical and jazz. The spectral coefficients and the MFCCs are chosen to describe the signals. But rather than directly using these values for the classification, the feature vectors are extracted from the two-dimensional graphical representations of the features by making use of an image classification technique called the *texture-of-textures approach*. Three different classifiers were evaluated: kNN, GS and a *Support Vector Machine (SVM)* algorithm (SVMs project the data into a higher dimensional feature space and try to find linear discriminant functions in that space). The best performance (75%) was achieved with the kNN classifier. Another interesting conclusion from this work is that a simple gaussian *pdf* seems to model the distribution of classical music feature vectors with a reasonable accuracy, while giving poor results in the case of rock and jazz.

Casey (2002)

The system described by Casey [5, 25] is intended for audio classification and retrieval in a generalized way. That is, similarly to the Muscle Fish system, the definition and training of all classes is made by the user. The feature vectors are obtained from a compact representation of the spectrum using *Singular Value Decomposition (SVD)*. The explicit time-modeling approach is also used here, in this case using HMMs like in Zhang and Kuo [50]. Each state is modeled as a GS distribution. This approach has mainly been tested in classifying environmental and single-instrument sounds. Also, a performance test in classifying music into Bluegrass, Reggae, Rap, Folk, Blues, Country, Gospel and New Age obtaining a classification accuracy of 95.4% is reported. Unfortunately, there is no indication of the number of training and test samples used for the evaluation, making a performance comparison impracticable. This system has been adopted by the MPEG-7 standard as one of its high-level audio tools: the *General Sound Recognition and Indexing* tool (see next section).

Tzanetakis and Cook (2002)

The most elaborate music hierarchy among the papers reviewed here can be found in Tzanetakis and Cook [45, 44]. Music signals are divided into classical, country, disco, Hip-Hop, jazz, rock, blues, reggae, pop and metal. Classical music is further divided into choir, orchestra, piano and string quartet. Jazz is further divided into Big Band, Cool, Fusion, piano, quartet and Swing. The musical hierarchy is extended by introducing a previous music/speech discriminator similar to the one implemented by Scheirer and Slaney [35]. Speech signals are further divided into male speech, female speech and speech with noisy background. The authors divide the used features into three groups: *timbral texture*, *rhythmic content*

and *pitch content* features. Timbral features include centroid, rolloff, flux, zero crossings and MFCCs. Rhythm and pitch features are based on the file-based computation of the beat and pitch histograms, respectively (see Sect. 5.4.1). The GS, GMM and kNN classifiers were evaluated. The performance percentages given in this work are independent of the previous classification stages in the musical tree. That is, the performance in classifying into the Jazz subgenres is evaluated assuming all the input signals are Jazz examples, the same applying to the speech, music and classical subdivisions. These rates are the following:

- Music vs. Speech: 86%
- Speech classes (3): 74%
- Music genres (10): 61%
- Classical subgenres (4): 88%
- Jazz subgenres (6): 68%

In the three musical classifications, the best performance was achieved using the GMM classifier with 3 gaussian components per mixture. The above rates correspond to a file-based classification in which one feature vector is computed for each file using mean and variance as subfeatures (see Sect. 5.1.1). The classification into the 10 musical genres was also tested in real-time mode, hence using only the timbral features, and yielding a 44% correct classification rate.

Lu, Zhang and Jiang (2002)

In Lu *et al.* [24], audio is labeled as speech, music, environment sound and silence, achieving an accuracy of 96.51%. Classification is made in two steps. At the first, it is distinguished between speech and nonspeech signals using a combination of a kNN classifier based on zero-crossings, flux and energy features, and a set of heuristic rules based on a linear-prediction-related feature called Linear Spectral Pairs (LSP). At the second stage, nonspeech is classified into music, environment or silence by a set of heuristic rules based on flux, the ratio of noisy frames, and a periodicity measure based on the autocorrelation, similar to the Harmonic Ratio defined in *MPEG-7* (see Sect. 5.3.6.4).

Jiang, Lu, Zhang, Tao and Cai (2002)

In this paper [20], a feature called *Spectral Contrast (SC)* is proposed as an improvement of the MFCCs. While the MFCCs represent the average form of the spectrum (see Sect. 5.3.5), SCF describes the relative strengths of its peaks and valleys within a set of subbands. An accuracy of 82.3% was achieved by using a GMM to classify into following 5 musical classes: baroque, romantic, pop songs, jazz and rock, compared with 78% reached with the MFCCs.

The systems reviewed in this section are summarized in the following two tables. Table 3.1 shows the speech/music and speech/music/background discriminators. Table 3.2 lists the systems with more classes, most of them musical genres. The given classification rates are not directly comparable since they closely depend on the number of classes considered in the system, the number of samples used for training and testing, and the particular evaluation method used. For this reason, all these values are also listed here.

Author	Year	Speech classes	Music classes	Other classes	Traing samples per class	Test samples	Classification rate
Scheirer *et al.*	1997	1	1	0	72	16	98.6%
Scheirer *et al.*	1997	1	1	1	72	24	65%
Zhang *et al.*	1998	1	1	1	6-8	50	90%
El-Maleh *et al.*	2000	1	1	0	?	?	95.9%
Lu *et al.*	2002	1	1	1	?	?	96.51%

Table 3.1: Overview of existing music/speech(/background) discriminators.

Author	Year	Speech classes	Music classes	Other classes	Traing samples per class	Test samples	Classification rate
Zhang *et al.*	1998	1	1	10	6-8	50	80%
Lambrou *et al.*	1998	0	3	0	4	?	91.67%
Soltau *et al.*	1998	0	4	0	60	18	86.1%
Pye	2000	0	6	0	?	175	92%
Deshpande *et al.*	2001	0	3	0	35	17	75%
Casey	2002	0	8	0	?	?	95.4%
Tzanetakis *et al.*	2002	0	10	0	100	?	61%
Jiang *et al.*	2002	0	5	0	1000	1250	82.3%

Table 3.2: Overview of existing multi-class audio classification systems. It should be noted that the given classification rates are not directly comparable because they highly depend on the number of classes and training samples. For these reasons, these numbers are also explicitly given here.

3.2 The MPEG-7 Standard

One proof of the increasing significance of information retrieval nowadays is that there already exists an international standard defining a set of techniques for analyzing and describing raw data. Unlike the previous published standards by the MPEG[1] Group (MPEG-1, MPEG-2 and MPEG-4), the new MPEG-7 standard

[1]Moving Picture Experts Group

[25] does not deal with the compression of bit streams, but with its content-based description. MPEG-7, officially called *ISO/IEC 15938 "Multimedia content description interface"*, started in 1997 as an attempt to facilitate the access and management of the huge amount of digital information that the previous compression standards had made available, most of all through the Internet.

The main objectives of MPEG-7 is to standardize feature extraction for any type of digital data and to define a language that describes this data in terms of its features. In MPEG-7 terminology, a feature is represented by a *descriptor*, and the language describing the data is called the *Descriptor Definition Language (DDL)*. A certain data object, such as an image, an audio file or a movie, is represented by a *Description Scheme*, which defines the relationships between the individual descriptors that make up the data, or between different description schemes. The DDL, which is an extension of the XML[2] language, also allows to create new description schemes and new descriptors, making *MPEG-7* a highly flexible and easily upgradable standard.

Of most interest for this work are the audio descriptors defined in Part 4 of the standard [1]. As it will be described in Chapter 5, a number of MPEG-7 descriptors have been evaluated as possible features for audio type classification. Here, the main organization of audio description tools within MPEG-7 will be reviewed, while in Chapter 5, the detailed definition and implementation of the selected descriptors will be addressed.

The standard distinguishes between low-level and high-level audio description tools. The low-level tools are called the *Audio Framework* and consist of generic descriptions applicable to any kind of audio data. The audio framework consists of the *Low-Level Descriptors (LLD)*, which define the extracted audio features, the *Scalable Series*, which define data types for performing downsampling of series of descriptors, and the *Silence Description*, which specifies the segment to represent silence.

The high-level tools are application-specific, and are implemented as description schemes using the low-level audio framework. Examples include instrumental timbre description, audio signature tools, melody description tools and general sound recognition and indexing tools.

The described audio tools hierarchy is summarized as follows:

- Audio Framework
  - Low-Level Descriptors
  - Scalable Series
  - Silence Segment
- High-Level Tools

[2]Extensible Markup Language

- Audio Signature
- Instrumental Timbre
- General Sound Recognition and Indexing
- Spoken Content
- Melody Description

3.2.1 MPEG-7 Low-Level Audio Descriptors

A total of 17 time and frequency-domain low-level audio descriptors are defined within the audio framework. They can be classified as follows [25]:

- *Basic:* Time-domain amplitude and power.
- *Basic Spectral:* Power spectrum and spectrum-related features, such as spectral centroid, spectral spread and spectral flatness.
- *Signal Parameters:* Fundamental frequency and harmonicity.
- *Temporal Timbral:* Attack time and temporal centroid.
- *Spectral Timbral:* Several spectral measures related to the harmonic components.
- *Spectral Basis Representations:* Compact representation of a spectrum as a set of basis functions and its projections.

The temporal timbral and spectral timbral descriptors, as well as the fundamental frequency, are features designed to be computed only on segments of monophonic[3] sounds, as needed for example in the high-level tools for instrumental timbre description. Particularly, the spectral timbral descriptors are based on a previous estimation of the harmonic components of the spectrum, which is in turn based on an estimation of the fundamental frequency. Therefore, this features are not suitable to describe complex sounds like polyphonic music.

In contrast, the rest of the descriptors are applicable to any kind of audio. The harmonicity descriptor measures the harmonic content of the spectrum without having estimated the fundamental frequency (see Sect. 5.3.6.4), and is therefore also appropriate for complex signals.

In this work, four features out of this set have been selected for performance evaluation in the classification task. The argumentation behind this decision will be discussed in Sect. 5.3.6.

[3] *Monophonic* in the musical sense (that is, *single-voiced*), not in the recording sense (*single-tracked*).

An audio descriptor can either be of scalar type (that is, one value per frame; for example, instantaneous power or spectral centroid) or of vector type (that is, more that one value per frame, such as spectra or spectral basis representations). In MPEG-7, two specific data types for audio descriptors are specified: *AudioLLDScalarType* for scalar values and *AudioLLDVectorType* for vector values, which are correspondingly inherited by the LLDs.

To complete this overview of MPEG-7 terminology, it will be mentioned that what in this work is called *step length*, as shown in Fig. 2.1, corresponds to the *hopSize* in the standard. It is consequently the rate at which a new value for a descriptor is computed. The default *hopSize* has been chosen to be 10ms. LLDs use either the *hopSize* or an integer multiple thereof as the update interval. The terms *analysis window* and *FFT length* are used in both the standard and this work with the same meaning.

Chapter 4

Creating the Audio Taxonomy

Although the main focus of the present work is the classification into musical subgenres, a preliminary classification into speech, music and background noise has been implemented and evaluated. By *background noise* we refer to any kind of non-speech and non-musical sound, such as environmental sounds (traffic, industry, country, forest, rain), animal sounds, or other non-speech human sounds (laughter, crowds, applause, etc.).

Speech is further divided into male speech, female speech and speech with background or noise. This last class is intended to comprise any kind of male or female speech mixed with music, such as a radio or TV commercial or a commentated music program; or with background noise, such as a radio play, a movie soundtrack or any kind of sports program. Children speech is not explicitly considered, although the system is likely to classify it as female speech.

The above class definitions are straightforward, since they rely on completely objective criteria. However, when it comes to music, many problems arise. Rather than consisting on a set of objective rules, musical genres are the result of a combination of not only musicological, but also social, historical, cultural and even economical factors. One of the main difficulties in designing an automated music classifier is to find to what extent is it possible to reduce all these aspects to a set of objective descriptions which can be interpreted by a computer. This issue is discussed in the present Chapter.

4.1 Musicological Considerations

In the audio classification systems described in the preceding section, many different music taxonomies[1] have been used. Examples of them are:

- Rock / Piano / Jazz (Lambrou *et al.* [21])

[1] *Taxonomy* is the branch of science concerned with classification, or a scheme of classification (definition of the *Oxford English Dictionary*).

- Rock / Pop / Techno / Classical (Soltau *et al.* [39])
- Blues / Easy Listening / Classical / Opera / Dance / Indie Rock (Pye [33])
- Bluegrass / Reggae / Rap / Folk / Blues / Country / Gospel / New Age (Casey [5])
- Baroque / Romantic / Pop Songs / Jazz / Rock (Jiang *et al.* [20])
- Classical / Country / Disco / Hip-Hop / Jazz / Rock / Blues / Reggae / Pop / Metal (Tzanetakis *et al.* [44])

Some of them may serve well in demonstrating the system capabilities, but are not likely to be useful in a final user-oriented application, since they are often incomplete or musicologically inconsistent [3]. An example of incompleteness is the absence of classical music in Lambrou [21] or Casey [5]. Musicologically inconsistent is, for example, the distinction between classical and opera or the choice of Indie Rock or Metal instead of the more general category of Rock in Pye [33] and Tzanetakis [44], respectively. Another incoherence is the fact that, at the same classification level, some classes are defined according to their musical genre and others according to their instrumentation, such as the Piano class in Lambrou [21] or the Quartet and Piano jazz subgenres in Tzanetakis [44].

For an automatic classification system, taxonomies should be constructed according to criteria as objective as possible. In Pachet and Cazaly [30] an attempt on building a universal, objective musical taxonomy is presented. Each of the 378 genres is differentiated from the others by a set of objective properties such as tempo, rhythm and instrumentation. For example, Funky Music is defined as the particular subgenre of Rhythm and Blues with massive presence of funk guitar and bass, and Rhythm and Blues is in turn defined as the particular subgenre of Blues with massive presence of brass instruments.

Although such objective, reduced descriptions may work well in defining broad music categories, one can figure out the difficulties they will present in the context of the most specific subgenres of the class tree. In fact, the same authors admit in a later paper the huge complications of such an approach, reducing their ambitions to a much simpler taxonomy [3].

But even in the case in which such a sophisticated set of purely objective rules would succeed in describing the differences between genres, it could not be directly implemented to perform automatic classification, since it would require the extraction of very complex and high-level features, a task that is still unfeasible.

For all the above reasons, a special effort was made in this work to find an audio taxonomy that would be at the same time:

- simple enough to allow class separation by feasible features

- complete enough to allow an acceptable classification of as much input signal types as possible, and
- musicological consistent to avoid ill-defined classes.

The first question that arises in creating the musical taxonomy is how to define the highest genres in the hierarchy. Perhaps the broadest genres that can be defined would correspond to the music of the different cultures, that is, western music, Chinese music, Indian music, etc. Our preliminary assumption is that only western music will be considered[2].

At the next stage, it is common use to distinguish between classical and popular music, in our case between western classical and western popular music. Although the term "classical music" is in its origin rather inappropriate, there is a generalized consensus about its meaning. However, the term "popular music" is much more polemical. On the one hand, it can be confused with the term "pop music", which designates a more specific subgenre. On the other hand, there are several genres that do not fit into the idea of popular music as music intended for massive consume; the most paradigmatic of these genres is jazz. In other languages there are other solutions to defining all that is not classical, like the terms "U-Musik" (entertainment music) in German or "música ligera" (light music) in Spanish, which are also at least as much questionable. We considered the term "drum-set-based music", since virtually all popular and jazz music is based on a rhythmic section, but discarded it to allow a complete generality. In the lack of better terms, we opted for the most diplomatic solution and called it just "non-classical music".

In contrast to non-classical music, which is usually classified according to stylistic genres that are nearly parallel in history (like rock, pop, rap, jazz, etc.), western classical music is often separated in sequential historical periods (medieval, renaissance, baroque, classic, romantic and contemporary). This would constitute a serious problem for an audio classifier, since it would require high-level features capable of distinguishing a classical string quartet by Haydn from a romantic string quartet by Brahms, or of classifying an early symphony by Beethoven as classical even if it was conducted by a romantic-inclined conductor.

For these reasons, it was decided to divide classical music according to instrumentation, not to historical period, which is also common practice in many market-related taxonomies. Following this approach, it was straightforward to divide chamber music[3] and orchestral music at the next level. Orchestral music

[2]However, it should be kept in mind that western classical genres like baroque, romantic, etc. and western popular genres like rock, pop, etc. are only a special case among many other types of classical and popular music in the world, which is often unfairly ignored as a consequence of their ubiquitous presence in the media.

[3]*Chamber music* is classical music written for a reduced number of performers in which each voice or group of voices is played by a single instrument. In contrast, in orchestral music, some voices are played by a group of instruments, like the string section of an orchestra.

comprises symphonic music (music with "only" orchestra), choral music (orchestra with choir) and music with orchestra and soloist (for example, concertos). Opera can fall into any of these three classes, depending on if the corresponding excerpt is instrumental (overtures, interludes), choral, or a soloist part (arias, duets, etc.).

Although there is a huge variety of chamber music ensembles, it is possible to group them into a few genres by considering the most important chamber ensembles (all chamber ensembles with piano and string quartets) as independent classes and defining an additional "other chamber ensembles" group. Due to its similarity, unusual variations to the string quartet such as string trios or string quintets are also intended to be a part of the string quartet class. Solo music[4] has been considered here as a chamber subgenre as well, since it fits into the above definition of chamber music.

In contrast to classical music, instrumentation is not a good criterium to separate non-classical genres like rock, pop and jazz, since they almost invariably follow the same scheme consisting of a rhythm section (drums and bass), an accompaniment section (usually keyboards or guitars) and a lead section (vocals and solos). Differences point mainly to rhythm structure, timbral variations of the instruments and singing character. Besides, non-classical music is organized attending to style in virtually all music labels, shops or on-line taxonomies. The main problem in this case consists of finding the broadest classes for non-classical music. We have chosen a three-way partition into rock, pop and jazz, believing that, in a broad view, they can comprise the vast majority of non-classical varieties. The definitions of rock, pop and jazz used here should be understood in their broadest possible interpretations.

While jazz is easier to separate, distinguishing between pop and rock is a far more problematic issue. Many audio taxonomies take the short way and avoid this problem by simply defining a pop/rock class. However, we believe that there are enough differentiating properties to separate them.

We understand rock as having a tendency to the ample use of guitars, both distorted or clean, and to be related to blues in musical form and in character. We divide rock into soft rock and hard rock. Soft rock tends to the use of undistorted guitars and soft vocals. Hard rock is characterized by high distorted guitars, more powerful drums and a more energetic way of singing.

In contrast, pop tends to the use of synthetic sounds, relegating guitars to a secondary role and resulting often in a brighter timbre. The vast majority of pop subgenres are decidedly dance-related. To emphasize that synthetic sounds play a key role in our definition of pop, and to make a wide interpretation of this class possible, we have called it "electronic/pop". We define as subclasses techno/dance, rap/hip-hop and a class comprising other pop subgenres, labeled

[4]Solo music is played by a single instrument, which can be monophonic, like a flute or a clarinet, or polyphonic, like a piano or a guitar.

just as "pop".

The resulting audio taxonomy is illustrated in Fig. 4.1.

4.2 Questionnaire about Musical Genres

The taxonomy described above certainly fulfills the condition of simplicity, but it is questionable to what extent it also accounts for generality. Separation of classical types posed no serious problems, as we based our classification on the instrumentation. However, non-classical genres are inherently defined in a much more fuzzy manner.

In spite of our belief that a very high number of musical examples could fit into one of the classes of our non-classical taxonomy, there are some particular genres for which this assumption was difficult to hold. This is the case, for example, of reggae, funk, country, folk, etc. To examine to what extent this dubious classes could fit into the proposed taxonomy, a small field study was carried out as follows:

17 musical examples belonging to one of the dubious classes were selected and reduced to 15-second excerpts. They were collected in a web page and subjected to an on-line questionnaire consisting of the following two questions:

1. Into which of the following genres would you classify the current excerpt: Rock, Pop or Jazz? Even if you think the example does not fit at all into one of these three categories, try to decide in which of them would the classification be more acceptable.

2. Without taking into account the previous constraint, how would you classify the excerpt? Write in the text field a genre or subgenre that you think is appropriate.

A total number of 75 replies were received. The results confirm the very fuzzy nature and the many different interpretations of non-classical music genres. We initially intended to consider country as a special case of soft rock, and ska, reggae and funk as special cases of pop. The results of the study showed this was a rather inappropriate generalization, and suggested to regard the mentioned genres as independent classes in future expansions of the taxonomy. As a consequence, all the funk, ska, reggae and country examples that were initially present in the training database were discarded and replaced with examples belonging to one of the more undisputed genres. Some remarkable results are outlined in Table 4.1. They show how different the criteria are when it comes to assign a broad class to a well-defined genre such as reggae or country.

Another interesting conclusion was the inconsistency of some commonly used genres such as "alternative" or "independent", which were often proposed to describe very different kinds of examples. The complete results of the test are

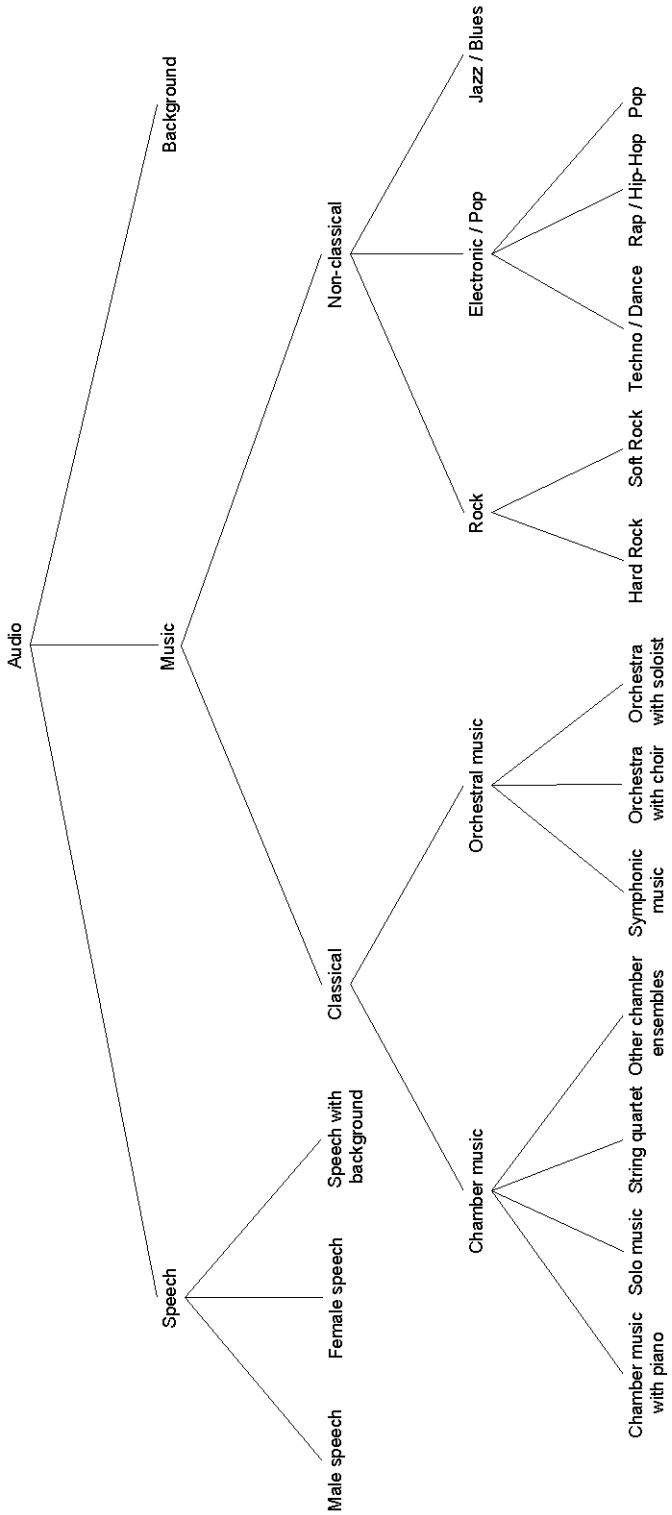


Figure 4.1: Audio taxonomy.

Example	Rock	Pop	Jazz
Reggae	25.4 %	50.7 %	23.9%
Ska	46.6 %	49.3 %	4.1%
Country	30.1 %	43.8 %	26.1%
Funk	41.9 %	16.2 %	41.9%

Table 4.1: Selected results from the musical questionnaire. Rows correspond to individual reggae, ska, country and funk examples. Columns indicate the precentage of test subjects that proposed one of the rock, pop or jazz classes as possible father genres. For the complete results, see Appendix B.

given in Appendix B, together with the results of classifying these ambiguous examples with the end application that resulted from this work. The music examples used in the questionnaire are included on the enclosed CD-ROM (see Appendix C).

Following the considerations discussed in the preceding section, and the results of the questionnaire, we can outline the intended contents of each class of the taxonomy as indicated in table 4.2.

It should be noted that the most specific subgenres listed in this table reflect the line of argument in the creation of the taxonomy, and indicate the classes in which an input signal of that subgenre should ideally fall. They do not necessarily mean that all of them are represented in our audio training database.

Some examples of genres that are not intended to fit into this work's taxonomy are the following:

- *A capella* choral music (only choir)
- Wind bands (concert bands, marching bands)
- Electronic contemporary classical music
- Non-western classical music (such as Indian or Chinese classical music)
- Traditional and ethnic music (traditional Celtic music, flamenco, gamelan)
- Tradition and ethnic-related popular music (folk, salsa, tango, *world music*)
- Other popular genres: funk, reggae, ska, country, soul

Of course, the above genres can constitute the basis for future expansions of the taxonomy.

Class	Comprised subclasses	Representative examples
Speech		
Male speech	male speech	
Female speech	female speech	
Speech with background	speech with background noise, speech with background music	movie soundtracks, TV commercials, sports programms, radio plays
Chamber music		
Chamber music with piano	piano trios, piano quartets, piano quintets, sonatas with piano, lieder	
Solo music	solo sonatas, suites, partitas, miscellaneous solo pieces	
String quartet	string quartet, string trio, string quintet, string sextet	
Other chamber ensembles	triosonatas, clarinet quintet, wind quartet, septet, octet, opera recitatives, miscellaneous chamber groups	
Orchestral music		
Symphonic music	symphonies, overtures, symphonic poems, suites	
Choral music	choral passages in operas, oratorios, symphonies	
Orchestra with soloist	concertos, orchestral lieder, opera arias, opera duets	
Rock		
Hard Rock	Hard Rock, Heavy Metal, Punk, Grunge, Hardcore	Led Zeppelin, Metallica, Nirvana, Korn
Soft Rock	Rock 'n' Roll, Rhythm 'n' Blues, Pop-Rock	Chuck Berry, The Beatles, Dire Straits, Oasis
Electronic/Pop		
Techno / Dance	Techno, Dance, House, Disco	
Rap / Hip-Hop	Rap, Hip-Hop	M.C. Hammer, Cypress Hill, Eminem
Pop	Pop	ABBA, Michael Jackson, Madonna, Prince
Jazz/Blues		
Jazz/Blues	Jazz, Blues	Duke Ellington, Miles Davis, B.B. King
Background		
Background	environmental sounds, animal sounds, human non-speech sounds	

Table 4.2: Taxonomy classes and intended comprising subclasses.

4.3 Human Performance in Classifying Music

The observations in this Chapter have made clear that musical genres are of an inherently fuzzy nature, and that they are understood in many different ways across individuals. The difficulties in designing an automatic music classification system lie not only in the subjective characterization of the genres, but also in how it can meet the expectations of a wide range of users with different opinions about musical categories.

Human classification of music has been very little studied. Its in-depth investigation can provide useful results in finding to what extent is automatic classification possible. In a work by Soltau [38], a group of 37 test subjects are asked to classify a number of 3-second audio examples into rock, pop, techno and classic. The same samples were used to test the automatic classification system based on neural networks that was shortly described in the previous Chapter [39]. Both results were similar, not only in the total classification rate (86.1%), but also in the individual confusions between genres. It was found that both the test subjects and the system found most difficult to separate rock from pop. Only around 72% of the rock samples were correctly classified by both the system and by the test subjects.

In Perrot and Gjerdigen [32], college students were asked to classify short excerpts into one of the following ten classes: blues, country, classical, dance, jazz, latin, pop, rhythm 'n' blues, rap and rock. Two experiments were carried out, one using excerpts only 250ms long, and the other with 3-second excerpts. Reported final accuracies are 53% for the 250ms samples and 70% for the 3s samples. Listening to more than 3 seconds did not improve the performance.

Results such as these have led researchers to put in question the feasibility of a high-accuracy automatic classification system. After all, any automatic classification system relies on a set of training examples that were previously labeled and classified by a human. Consequently, can the resulting system classify better than its human trainer?

Perhaps the only possible answer to this is in turn another question: what does "classify better" exactly mean? It has been stated that current audio classification technologies have reached the performance levels of human music classification [44]. But, should the systems really be compared with experiments in which the test subjects were randomly selected, with probably very different musical habits and knowledge? This does not seem to be a plausible approach. An automatic classification tool should be rather regarded as an *expert system* in which a knowledge database has been carefully created using as many information sources as possible. The key point is to ensure that the human-supervised process of implementing this knowledge (in our case, collecting the audio database) is carried out following purely objective criteria. The only reliable way to measure the system performance is to compare its results with the selection criteria used in creating its underlying knowledge base. Comparing it with any kind of

general human performance is likely to be ill-founded.

4.4 Audio Samples Database

Once the audio class structure was decided, 50 audio examples were collected for each class, resulting in a 850 file database. Each sample is approximately 30 seconds long, resulting in over 7 hours of audio. All examples are sampled at 44.1kHz and stored as WAV files. Virtually all are stereo, and some few are mono. They were collected from CDs, TV, radio, and uncompressed MP3 files, thus representing a wide range of audio qualities.

Chapter 5

Feature Extraction

The present chapter addresses the audio-specific part in the design process of the classifier. As shown in Chapter 3, many different features have been proposed to recognize audio signals. Some of them were initially proposed for speech recognition applications and have been largely used in that field. Other music-specific features, such as beat features, have been proposed more recently, when music signals started to be considered.

In the present work, the selection of features has been done in a highly systematical way. As the first step, a large list of features was elaborated. These are introduced in this Chapter and include features proposed in previous research works for music and speech classification, features defined in the MPEG-7 standard, as well as new features proposed in this work. All of them were implemented and evaluated with several representative audio samples[1]. In the second step, a subset of these features was selected for the final implementation making use of a feature selection algorithm that will be discussed in detail in the next Chapter.

5.1 Audio Feature Extraction

In the previous introductory discussion of Pattern Recognition, we have assumed that a feature vector $\mathbf{x}$ represents the object to be classified *as a whole*. However, audio signals are time-varying objects, and thus special attention must be payed when defining what exactly does an audio feature vector represent. In this context, we can distinguish between two different approaches: *frame-based feature vectors* and *texture-based feature vectors*.

[1] All the features and its evaluations were implemented with MATLAB.

5.1.1 Frame-based and texture-based audio classification

Frame-based Feature Vectors

In the frame-based approach, the input audio signal is broken into small (possibly overlapping) blocks, and a feature vector is computed for each block. In this context, the blocks are called *analysis windows*. This analysis window corresponds to the *window length* depicted in Fig. 2.1.

In this case, a new feature vector is obtained in intervals that usually range from 10 to 40 ms [35, 33, 9, 44]. This can also be viewed as a *time-varying feature vector* that describes a trajectory in the feature space as the signal progresses.

Texture-based Feature Vectors

The frame-based approach is useful when real-time classification is desired. But its major drawback is that it does not allow to take into account other long-term characteristics of the signal that can be very useful for the classification. For example, one can expect that certain measures of the rhythmical structure of a music signal will be most useful to detect genre. Clearly, it is not possible to infer something about rhythmical structure when observing only 40 ms of music. The same applies to dynamic structure or envelope form. For this kind of structural descriptions, feature vectors that are more separated in time are required.

Furthermore, a recent study [44, 43] has shown that not only structural measures but also spectral and other timbral measures benefit from lower feature vector rates, especially when music signals are considered. This indicates that very often not the feature itself, but its variation in time provides a better description of the signal. As a result, the concept of *texture window* was introduced[2]. A texture window is a long-term segment (in the range of seconds) containing a number of analysis windows. In the texture-based approach, only one feature vector for each texture window is generated [35, 44, 48]. The features are not directly the values obtained in each analysis window, but statistical measures of all the values obtained for each analysis window within the current texture window.

The system described here follows the texture-window approach. More specifically, in this work, and for each underlying frame-based feature, the four following texture-based *subfeatures* are computed:

- Mean
- Standard deviation

[2]The term "texture window" originates from the notion of musical texture, which refers to the character of the constituent voices and to the relationships among them, and is in turn reflected in the medium or long-term timbral characteristics.

- Mean of the derivative[3]
- Standard deviation of the derivative

To be more precise, we should write *sample mean* instead of *mean* and *sample standard deviation* instead of *standard deviation*, since this values are in fact the estimated parameters of an unknown distribution and are given by Eq. 2.27 and by the square root of Eq. 2.21. In this case, the samples x_j are the values of a specific feature in each analysis window and N is the number of analysis windows within each texture window.

5.1.2 Stream-based and file-based audio classification

The sound input to a classification system can be either an audio stream of unspecified length (such as a radio program) or a stored audio file of known length. In both cases, any of the two feature-vector extraction methods described above can be used.

If we use frame-based extraction, we can obtain classification in real time. Using the texture approach, a classification result can be available at intervals defined by the texture window. These two procedures are useful if it is possible that the stream or the file contains more than one type of audio. Classification is constantly updated, thus allowing to detect changes in the audio content. The segmentation resolution will be highest for real-time classification and will depend on the texture window size for texture classification.

However, if the input signals are files, and we assume that each file contains only one type of audio, we can make use of the fact that the input length is known to implement a *file-based* classification. Also, in this case, it is possible to seek the whole file for the maximum amplitude in order to perform normalization of the signal (see next section), allowing the features to be independent of the amplitude-scaling. File-based classification can be done in two ways:

The first method consists of taking the whole length of the file as the texture window. Thus, classification is based on a single feature vector that represents the whole file. The other method consists of using either the frame-based or the texture-based methods and to assign to the file the most frequent class among the individual classifications. These two approaches to file-based classification will be denoted as *single vector approach* and *texture window approach*, respectively.

The system described here is intended for file-based operation. The two above methods for file-based classification are evaluated and compared (see Sect. 9.2 and 9.3). A system of this type could be also modified to perform stream-mode classification, but is expected to perform better on files than on streams, be-

[3]The term *first difference* would be more accurate than *derivative*, as we are working with discrete signals.

cause in the first case much more data has been analyzed and therefore more information is available before a class decision is met.

5.2 Signal Preprocessing

Before proceeding with the feature extraction, the input signal is subjected to following preprocessing steps, in this order:

Downmixing to mono. If the signal has more than one channel, it is reduced to a monophonic signal simply by adding its channels. The amplitude range of the sum signal will depend on the number of added channels, but it will be normalized to one in the next step in any case. Although the software implemented for this work is able to read signals with more than two channels, only stereo and mono signals have been considered for the training and testing. It has not been tested how signals with more than two channels could affect classification accuracy.

Normalizing. The scaling of the amplitude level of a signal should not influence the description of its content, and thus it must be regarded as irrelevant. To ensure this, it is extremely important to normalize the input signal before extracting features. The normalization is done with respect to the absolute valued signal, according to:

$$x_{norm}[n] = \frac{x[n]}{max\{|x[n]|\}} \tag{5.1}$$

It should be noted that, since this is a file-based system, the normalization is always done by searching the whole file for the maximum, even in the case where smaller texture windows are used for the feature extraction.

Elimination of zeros. To avoid possible overflows or divisions by zero, all the samples with zero amplitude values are substituted with a value of 10^{-9}.

5.3 Timbral Features

In this and the next sections, the features that were evaluated in this work are described in detail. A graphical plot of the variation of each feature in time will be given. In order to facilitate comparison and to allow insight into the meaning of the features, always two graphical trajectories of representative examples belonging to two different audio types will be superimposed.

As described above, each feature is evaluated in an analysis window. This fact will be emphasized by writing X_r for a feature X, where r is the window or frame index. We will follow the notational conventions of Sect. 2.1 and thus, the

audio signal will be denoted by $x[n]$, the frame index by r, the total number of audio samples in each frame (also the number of points of the FFT) by N, and the frequency bin index by k.[4]

It should always be kept in mind that in the system described here, the features are not directly the quantities described by the graphical trajectories, but their underlying statistical measures (mean, standard deviation and so forth) across a certain texture window, as described above.

Features are classified in this work according to the signal properties they describe. In the present section, the timbre-related features are overviewed. The next section presents the features related to rhythmical properties of the signal, and in Sect. 5.5 we focus on features describing dynamic, statistical and predictivity properties.

It is also possible to classify features according to whether they are computed in the time domain or in the frequency domain, as follows:

- *Time domain features:* Zero Crossings, Central Moments, Harmonic Ratio, Root Mean Square, Envelope, Low Energy Rate, Loudness, Predictivity Ratio.
- *Frequency domain Features:* Centroid, Rolloff, Flux, MFCCs, Spread, Flatness and all Beat Histogram features.

Except for the MFCCs, the MPEG-7 features and the Rhythm features, all the other features in this work were computed using a window length of $N = 1024$ and a step length of $R = 512$ audio samples, which means that there is an overlapping of 50% and no *zero padding* (see Fig. 2.1). For the frequency domain features, the window length corresponds to the FFT length. In their case, a *Hamming* window was used (Eq. 2.9).

5.3.1 Zero Crossings

The *Zero Crossings* feature [44, 35] counts the number of times that the signal amplitude changes signs in the time domain during one analysis window:

$$ZC_r = \frac{1}{2}\sum_{n=1}^{N} |sign(x[n]) - sign(x[n-1])| \tag{5.2}$$

where the *sign* function is defined by

$$sign(x) = \begin{cases} 1 & if \quad x > 0 \\ 0 & if \quad x = 0 \\ -1 & if \quad x < 0 \end{cases}$$

[4]The index notation used in this Chapter should not be confused with the one used in pattern recognition-related equations that appear throughout the rest of the work.

For single-voiced signals, zero crossings have been used in the literature to make a rough estimation of the fundamental frequency. For complex signals it is a measure of noisiness.

In Fig. 5.1(c), the Zero Crossings curves of two 5-seconds-long classical[5] (black curve) and pop music[6] (gray curve) excerpts are shown. These audio examples are included on the enclosed CD-ROM. In this example of pop music, sudden changes of the curve correspond to the onsets of the drum set. Particularly, the valleys in the curve correspond to the main beats given by the bass drum, whose predominant low frequencies make the Zero Crossings rate to decrease.

5.3.2 Centroid

In an 1974 study by von Bismarck [47], several commonly used verbal attributes describing timbre were collected and subjected to a statistical study. The goal was to find which of them could better describe variations in timbre. 30 different scales whose endpoints were opposite pairs of verbal attributes, such as dark-bright, smooth-rough, etc., were used by a group of subjects to rate a set of sounds. It was found that the scale dull-sharp, that is, the scale describing the attribute of *sharpness* was the one that carried most of the variance, and thus most of the information.

This encouraged to find an analytical model which could measure the very abstract attribute of sharpness. It was found that a good approximation to it was the center of gravity, or *centroid* of the spectrum of the signal [46, 52]. Sharpness is related to the high frequency content of the spectrum, since higher centroid values correspond to spectra skewed to the range of high frequencies.

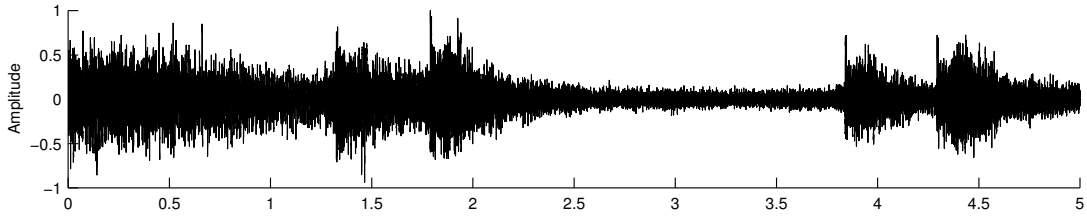
Due to its effectiveness to describe spectral shape, centroid measures have often been used in audio classification tasks [44, 35, 48]. Several variations in the definition of spectral centroid have been proposed in the literature [51], including centroids based on the amplitude spectrum, the power spectrum, the logarithmical power spectrum and other, more sophisticated definitions that take into account perceptual scalings. In this work, two of them have been evaluated. The first one is an amplitude spectrum centroid and is given by

$$C_r = \frac{\sum_{k=1}^{N/2} f[k]\ |X_r[k]|}{\sum_{k=1}^{N/2} |X_r[k]|} \tag{5.3}$$

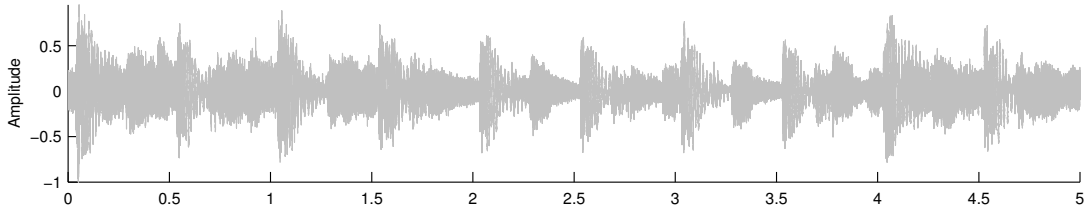
where N is the number of FFT points, $X_r[k]$ is the STFT of frame r as given by Eq. 2.8, and $f[k]$ is the frequency at bin k, as given by Eq. 2.6.

[5]Excerpt from *Siegfried´s Funeral March*, from Wagner´s *Götterdämmerung*.

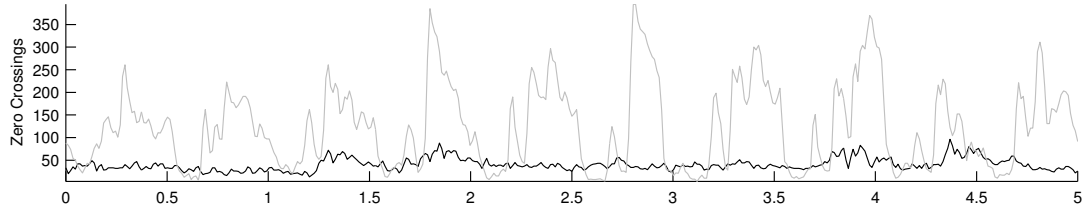
[6]Excerpt from Björk´s *Violently Happy*.



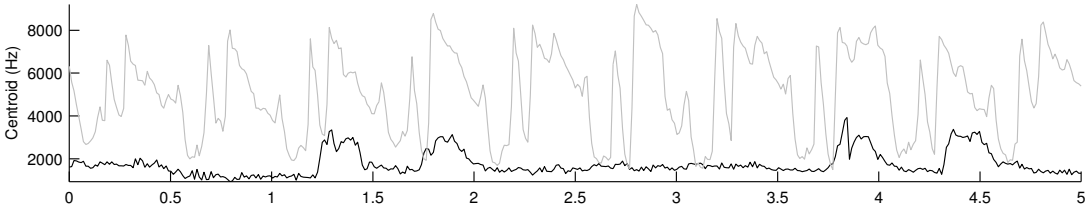
(a) Classical music example



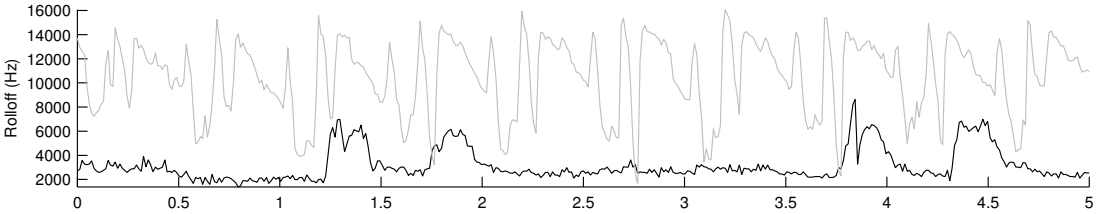
(b) Pop music example



(c) Zero Crossings



(d) Centroid



(e) Rolloff

Figure 5.1: Sound examples and timbral features (continued in Fig. 5.2).

The other definition of centroid that will be regarded is one of the three included in the *MPEG-7* standard, and will be discussed in Sect. 5.3.6.1.

5.3.3 Rolloff

The spectral *rolloff* point was first proposed as a feature to distinguish between voiced and unvoiced speech [35]. Like the centroid, it is also a measure of spectral shape, and yields higher values for right-skewed spectra. As it will be seen, both features are strongly correlated.

There are also slight variations in the definition of *rolloff*. Here, it is defined as the frequency below which 85% of the accumulated magnitudes of the spectrum is concentrated [44]. That is, if K is the bin that fulfils

$$\sum_{k=0}^{K} |X_r[k]| = 0.85 \sum_{k=1}^{N/2} |X_r[k]| \tag{5.4}$$

then, the rolloff is $R_r = f[K]$.

Centroid and rolloff curves for the above musical excerpts are given in Fig. 5.1. It is especially noticeable that these definitions of centroid and rolloff are strongly correlated.

5.3.4 Flux

Spectral *flux*, also called *Delta Spectrum Magnitude*, is a measure of the rate of change of the spectral shape [35, 44] and is given by the sum across one analysis window of the squared difference between the magnitude spectra corresponding to successive frames of the STFT:

$$F_r = \sum_{k=1}^{N/2} (|X_r[k]| - |X_{r-1}[k]|)^2 \tag{5.5}$$

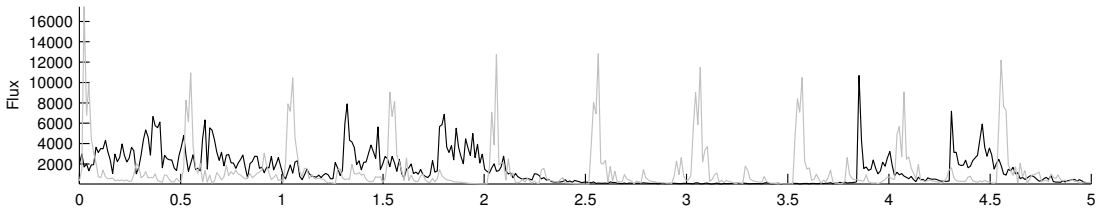


Figure 5.2: Spectral flux from the examples plotted in Figs. 5.1(a) and 5.1(b).

Flux has proven to be a suitable feature for the separation of music from speech, yielding higher values for music examples [35]. A plot of the flux curves for the above examples is given in Fig. 5.2.

5.3.5 Mel Frequency Cepstral Coefficients

Mel Frequency Cepstral Coefficients (MFCC) are a compact representation of the spectrum of an audio signal that takes into account the nonlinear human perception of pitch (see Sect. 2.2). They are one of the most used features in speech recognition, and have recently been proposed to analyze musical signals as well [12, 44]. A recent study [23] confirmed that MFCCs are appropriate to describe music.

A usual algorithm for the extraction of MFCCs, which has also been adopted here, consists of the following steps:

1. First, the FFT of each analysis window is computed using a Hamming window.

2. The FFT bins are combined according to a set of triangular weighting functions that approximate the human pitch perception as described by the mel scale. This can be viewed as filtering the spectrum with a filterbank of triangular bandpass filters, and then integrating the output of each filter over the frequency. The filterbank consists of 40 filters and is defined as follows:

   - The 13 first filters (low frequencies) have triangular frequency responses whose center frequencies are linearly spaced by a difference of 133.33 Hz.
   - The 27 last filters (high frequencies) have triangular frequency responses whose center frequencies are logarithmically spaced by a factor of 1.0711703.

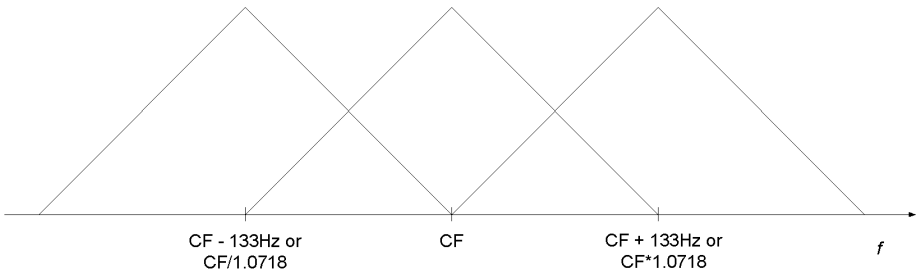


Figure 5.3: Definition of the MFCC filterbank. CF = Center Frequency (from [37]).

- For all the filters, the lowest frequency of the current filter corresponds to the center frequency of the previous filter, and the highest frequency to the center frequency of the next filter (see Fig. 5.3).

3. The base 10 logarithms of the 40 filterbank output coefficients are computed.

4. A Discrete Cosine Transform (DCT) is applied to decorrelate the coefficients (see Sect. 2.1.4).

It has been found [44] that using only the five first MFCCs provides an optimal audio classification performance. Therefore, in this work, a total number of 20 MFCC-related features are implemented: that is, the variance, standard deviation, mean of the derivative and standard deviation of the derivative of the first five MFCCs. The curves of the first five MFCCs from the previous music excerpts are shown in Fig. 5.4.

The publicly available Auditory Toolbox for MATLAB [37] was used in this work to compute the MFCCs. A 512-point FFT with no *zero padding* and 50% overlapping were chosen as the STFT parameters in this case.

5.3.6 MPEG-7 Features

In the design process outlined in this work, it has been always assumed that no a priori knowledge about the nature of the incoming signals is available. As a result, input sounds must be treated in a general way, and only features that

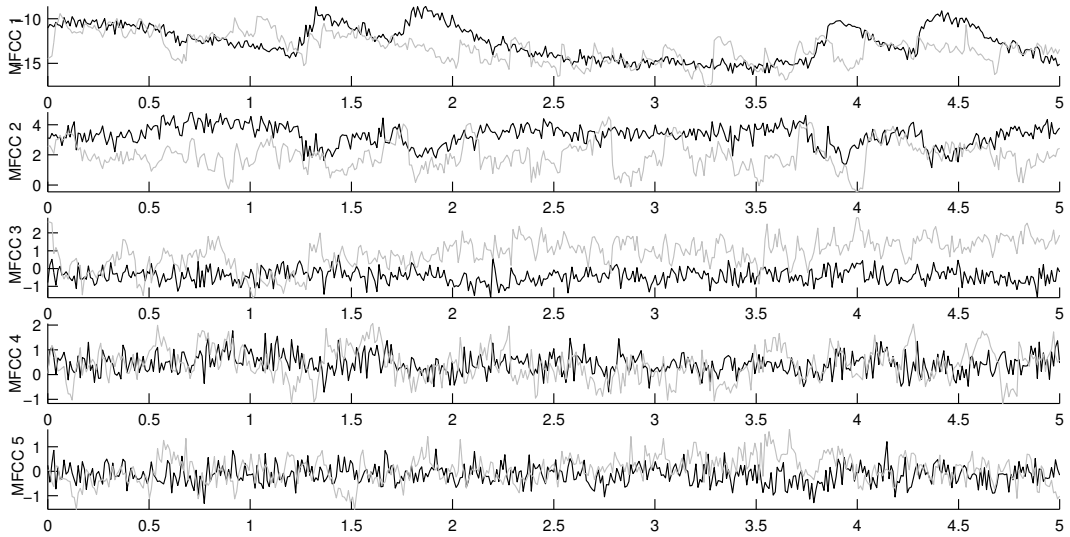


Figure 5.4: First five MFCCs from the examples plotted in Figs. 5.1(a) and 5.1(b).

are applicable to any audio type have been taken into consideration. For this reason, the timbral descriptors belonging to the MPEG-7 standard have not been regarded, since, as discussed in Sect. 3.2, they are intended for single-voiced, quasi-periodic audio segments.

For example, the *LogAttackTimeType* descriptor measures the time from the starting point of a signal to the sustain section of its envelope. Its motivation is to describe the onsets of single sound samples from different musical instruments. Clearly, this measure makes no sense in the context of this work, where complex and unsegmented excerpts are considered.

The same applies to all features that are based on an estimated fundamental frequency, that is, the spectral timbral features (*HarmonicSpectralCentroidType*, *HarmonicSpectralDeviationType*, *HarmonicSpectralSpreadType* and *HarmonicSpectralVariationType*). They measure different aspects of the harmonic components of the spectrum and rely on a previous estimation of the fundamental frequency. An accurate estimation of the fundamental frequency is only possible if the signal is single-voiced, and it will yield unpredictable results for the majority of signals used here. Therefore, these features have also been discarded.

The selected features (centroid, spread, flatness and harmonic ratio) are also timbral measures, but are applicable to any audio signal. The first three are frequency-domain features and, as such, they rely on the previously computed spectrum. Of course, the extraction of the spectrum is also normalized in the standard, and is part of the definition of the *AudioSpectrumEnvelopeType* descriptor. We will describe this common procedure here and will go into detail for each of the descriptors in the next four subsections.

The MPEG-7 descriptors outlined here were implemented following the *Final Draft International Standard (FDIS)* document (ISO/IEC FDIS 15938-4) [1].

MPEG-7 Spectrum Extraction

In MPEG-7, the spectral resolution and the frequency limits are parameters that can be set up by the user. However, in this work the recommended default values for them will always be adopted. Also, a sampling frequency of 44.1kHz is always assumed.

As mentioned above, the default step length, or *hopSize*, is 10ms (or, if f_s is the sampling rate, $0.01 f_s$ samples). The default analysis window length is chosen to be 3 default *hopSizes*, that is, 30ms or $0.03 f_s$ audio samples. The FFT length is then the next-larger power of two number of audio samples from the analysis window; the samples lying between the window end and the FFT end are zero-padded (see Fig. 2.1). The STFT is computed using a Hamming window (Eq. 2.9), and the resulting magnitude coefficients are converted to power coefficients $P_r[k]$ using Eq. 2.11.

5.3.6.1 Audio Spectrum Centroid

It has already been mentioned that spectral centroid measures are motivated by the fact that they provide an approximate model of timbral sharpness. No less than three different definitions of centroid descriptors are included in the audio part of MPEG-7. These are the following:

- *AudioSpectrumCentroidType:* based on a linear power spectrum and a logarithmical frequency scaling centered around 1kHz. It belongs to the group of *basic spectral* descriptors.
- *HarmonicSpectralCentroidType:* based on the amplitude of the harmonic components of the spectrum. It belongs to the group of *spectral timbral* descriptors.
- *SpectralCentroidType:* based on a linear power spectrum and a linear frequency scaling. It belongs to the group of *spectral timbral* descriptors.

The *HarmonicSpectralCentroidType* has been discarded for it requires an estimation of the fundamental frequency. The *SpectralCentroidType* corresponds to the centroid described in Sect. 5.3.2, the only difference being that it is based on the power spectrum, rather than the amplitude spectrum. Therefore, its behavior is expected to be very similar and has not been implemented.

In consequence, the *AudioSpectrumCentroidType* definition was selected for implementation. The steps for its extraction are the following:

1. The power spectrum is computed as described in the previous section. Since audio signals are real-valued signals, their spectra are evenly symmetric about the Nyquist frequency $f_s/2$ and therefore, only the first $N/2$ power coefficients $P_r[k]$ must be retained.
2. Before the computation of the centroid, the following intermediate step is taken to avoid that spurious very-low frequency coefficients have a disproportionally high weight. All coefficients below 62.5Hz are replaced by a coefficient at the nominal frequency of 31.25Hz and with power equal to the sum of the replaced coefficients. Formally:

$$P'_r[m] = \begin{cases} \sum_{k=0}^{N_{bound}} P_r[k] & if \quad m = 0 \\ P_r[m + N_{bound}] & if \quad 1 < m < N/2 - N_{bound} \end{cases} \tag{5.6}$$

$$f'[m] = \begin{cases} 31.25 & if \quad m = 0 \\ f[m + N_{bound}] & if \quad 1 < m < N/2 - N_{bound} \end{cases} \tag{5.7}$$

where $N_{bound} = floor(\frac{62.5N}{f_s})$ and $f[k] = \frac{kf_s}{N}$ as given by Eq. 2.6. The function $floor$ rounds towards $-\infty$.

3. From the obtained, modified power coefficients $P'_r[m]$ and their associated frequencies $f'[m]$, the centroid is defined as

$$C_r = \frac{\sum_{m=0}^{N/2-N_{bound}} \log_2(\frac{f'[m]}{1000}) P'_r[m]}{\sum_{m=0}^{N/2-N_{bound}} P'_r[m]} \tag{5.8}$$

As it can be seen, this definition uses a logarithmical frequency scaling centered at 1kHz. This log-frequency scaling approximates the perception of frequencies in the human hearing system.

A plot of this *MPEG-7* centroid is given in Fig. 5.5(a).

5.3.6.2 Audio Spectrum Spread

The spectral *spread*, also called *instantaneous bandwidth*, is another measure of spectral shape. More precisely, it describes how the spectrum is concentrated around the centroid. Low spread values indicate that the spectrum is highly concentrated near the centroid; high values mean that it is distributed across a wider range at both sides of the centroid.

Several definitions have also been proposed for the spread [22]. In MPEG-7 it is defined by the *AudioSpectrumSpreadType* descriptor. As previous steps, the modified power spectrum $P'_r[m]$ and its associated frequencies $f'[m]$ must be computed like in the computation of the centroid. Then, the spread is defined as

$$S_r = \sqrt{\frac{\sum_{m=0}^{N/2-N_{bound}} \left(\log_2(\frac{f'[m]}{1000}) - C_r\right)^2 P'_r[m]}{\sum_{m=0}^{N/2-N_{bound}} P'_r[m]}} \tag{5.9}$$

where C_r is the centroid defined by Eq. 5.8.

5.3.6.3 Audio Spectrum Flatness

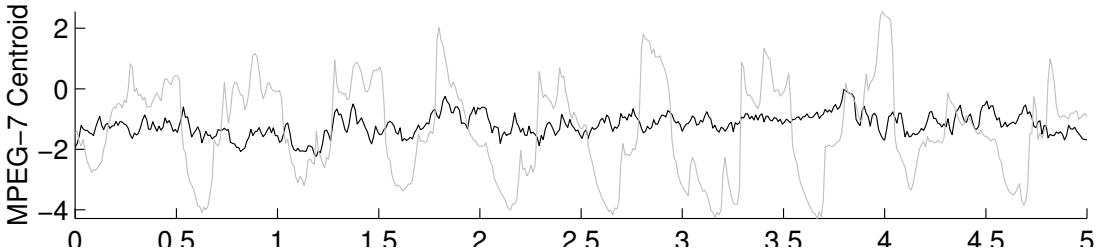
Spectral flatness is a measure of the deviation of the spectral form from that of a flat spectrum. Flat spectra correspond to noise or impulse-like signals, thus high flatness values indicate noisiness. Low flatness values generally indicate the presence of harmonic components.

The flatness feature is defined in MPEG-7 by the *AudioSpectrumFlatnessType* descriptor. Instead of calculating one flatness value for the whole spectrum, a separation in frequency bands is performed, resulting in one vector of flatness values per time frame. Thus, the flatness LLD is a two-dimensional descriptor that inherits the *AudioLLDVectorType*, in contrast to the rest of descriptors described here, which are scalar and hence of the *AudioLLDScalarType* type.

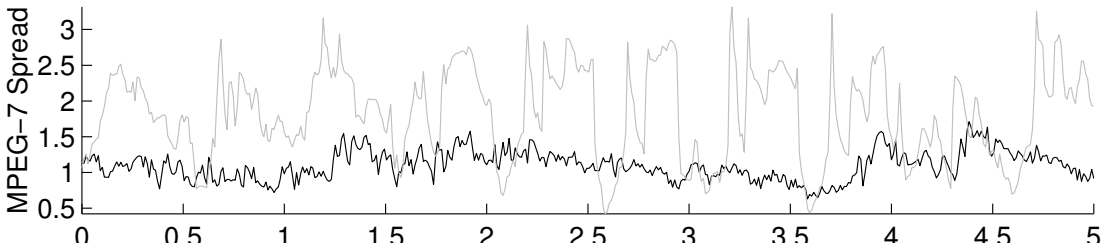
The extraction is normalized in the standard as follows:

1. The power spectrum is computed in first place as described in the preceding sections. The only difference is that, in this case, a *hopSize* of 30ms and an equal window length are recommended (which means that there is no overlapping).

2. The resulting power spectrum $P'_r[m]$ is then divided into frequency bands. The number of bands B and their bandwidth will depend on the low frequency limit (*loEdge*), the high frequency limit (*hiEdge*) and the logarithmic octave *resolution* according to the formula



(a) MPEG-7 Audio Spectrum Centroid



(b) MPEG-7 Audio Spectrum Spread

Figure 5.5: MPEG-7 Centroid and Spread from the examples plotted in Figs. 5.1(a) and 5.1(b).

$$hiEdge = 2^{resolution \cdot B} loEdge \tag{5.10}$$

A 1/4 octave resolution is used for the flatness. Also, the default values for *loEdge* and *hiEdge*, which will be used in this work, are 250Hz and 16kHz, respectively. With these values, a total number of $B = 24$ bands are obtained. Starting at *loEdge*, the *nominal* edge frequencies f_b for the bands are obtained by

$$f_b = 2^{b/4} loEdge, \quad 0 \leq b \leq B \tag{5.11}$$

3. However, defining the bands in such a non-overlapping manner can result in losses of frequency bins when different sampling frequencies are considered. Hence, the nominal edge frequencies are modified into the *actual* edge frequencies by introducing a 10% overlapping: the nominal low edge for each band is multiplied by 0.95 and the nominal high edge for each band is multiplied by 1.05.

4. Once the frequency boundaries have been obtained, we must find the corresponding edge bins by inverting Eq. 2.6. It should be noted that a rounding operation must be considered, since $k \in \mathbb{Z}$:

$$k = round(\frac{fN}{f_s}) \tag{5.12}$$

where *round* denotes rounding to the nearest integer. For a band b, its corresponding low and high edge bins will be denoted by $il(b)$ and $ih(b)$, respectively.

5. At this point, the standard specifies a grouping algorithm for the power coefficients in order to reduce computational costs. In this work the focus is to evaluate the behavior of the descriptors rather than to find computationally optimal implementations. Hence, this step has not been implemented here.

6. The flatness of a band b is defined as the ratio of the geometric and the arithmetic means of the power spectrum coefficients within that band:

$$SF_r[b] = \frac{\sqrt[ih(b)-il(b)+1]{\prod_{m=il(b)}^{ih(b)} P'_r[m]}}{\frac{1}{ih(b)-il(b)+1} \sum_{m=il(b)}^{ih(b)} P'_r[m]} \tag{5.13}$$

As noticed, flatness is a vector feature that produces a set of values per frame, one value for each spectral band. In this work, each vector has been reduced to a scalar by computing the mean value across the bands for each given frame, thus obtaining a scalar feature describing the overall flatness. Therefore, we will call it *mean flatness* rather than simply flatness:

$$MSF_r = \frac{1}{B}\sum_{b=1}^{B} SF_r[b] \tag{5.14}$$

Figures 5.6(a) and 5.6(b) show the flatness descriptor of the two above examples in its original vectorial form, which results in a spectrogram-like representation. The bright areas denote high flatness values, that is, noisiness. In the pop example, it is possible to appreciate the beat onsets as bright vertical strips. Figure 5.6(c) shows the frame-based mean values of both examples superimposed, which constitute our final feature.

5.3.6.4 Harmonic Ratio

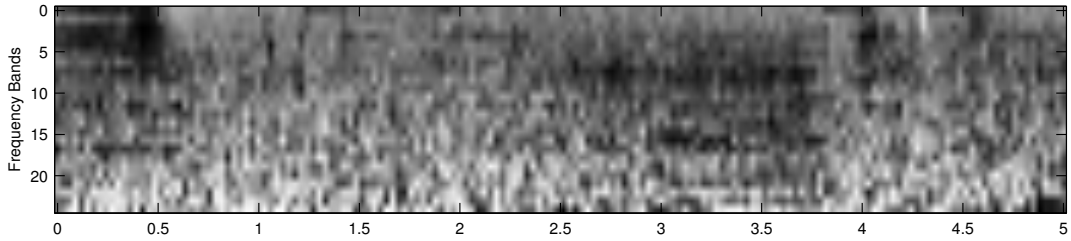
The *AudioHarmonicityType* is a dual descriptor that contains two measures of the harmonic properties of a spectrum: the *HarmonicRatio* and the *UpperLimitOfHarmonicity*. For the evaluations carried out in this work, the *HarmonicRatio* has been implemented. It is a measure of the proportion of harmonic components within the spectrum and, since it is not based on an estimation of the fundamental frequency, it can be reliably applied to any kind of input signal. Its extraction is defined in the standard as follows:

1. For each signal frame $x_r[n]$ of N samples, the normalized *autocorrelation (AC)* of the input signal at lags τ ranging from 1 to K is computed:

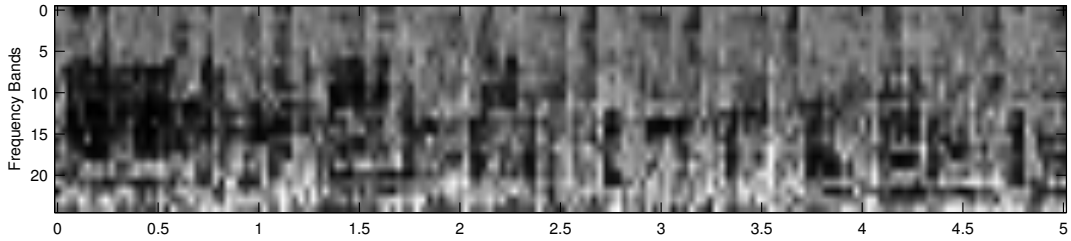
$$AC_r[\tau] = \frac{\sum_{n=0}^{N-1} x_r[n]x_r[n-\tau]}{\sqrt{\sum_{n=0}^{N-1} x_r[n]^2 \sum_{n=0}^{N-1} x_r[n-\tau]^2}}, \quad 1 \le \tau \le K \tag{5.15}$$

   For a purely periodic signal, the maximum autocorrelation values will be at lags corresponding to multiples of its fundamental period T_0[7]. In the above definition, all AC values over a range of K different lags are computed, starting at $\tau = 1$ and ending at the maximum lag K. In order to detect the first peak of any periodic or nearly-periodic signal, this maximum lag must have at least the same length as the maximum fundamental period expected $T_{0,max}$, and thus is defined by $K = T_{0,max} f_s = f_s / f_{0,min}$.

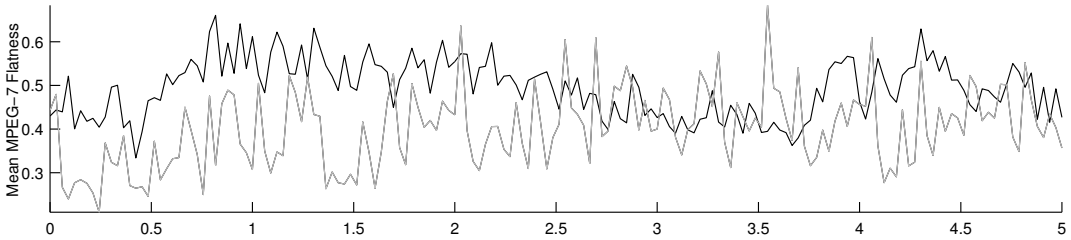
[7]Many fundamental frequency estimation techniques rely on this fact.



(a) MPEG-7 flatness (classical example)



(b) MPEG-7 flatness (pop example)



(c) Mean MPEG-7 Flatness

Figure 5.6: MPEG-7 flatness and mean flatness from the examples plotted in Figs. 5.1(a) and 5.1(b).

The default $T_{0,max}$ is 40ms, corresponding to a minimum fundamental frequency expected of 25Hz. It should be noted that K can be greater that the number of samples per frame N. This is in fact the case here, where a default *hopSize* of 10ms is used. This means that for the computation of the AC for one frame, samples ranging out of that particular frame must also be taken into consideration[8].

[8]This has to be especially kept in mind when designing the buffer processing in the implementation.

2. The Harmonic Ratio of one frame r is defined as the maximum AC value for that frame. This is written in the standard as:

$$HR_r = \max_{\tau=1,N-1} \{AC_r[\tau]\} \tag{5.16}$$

We used a computation of the autocorrelation in the frequency domain [29] to improve computational performance.

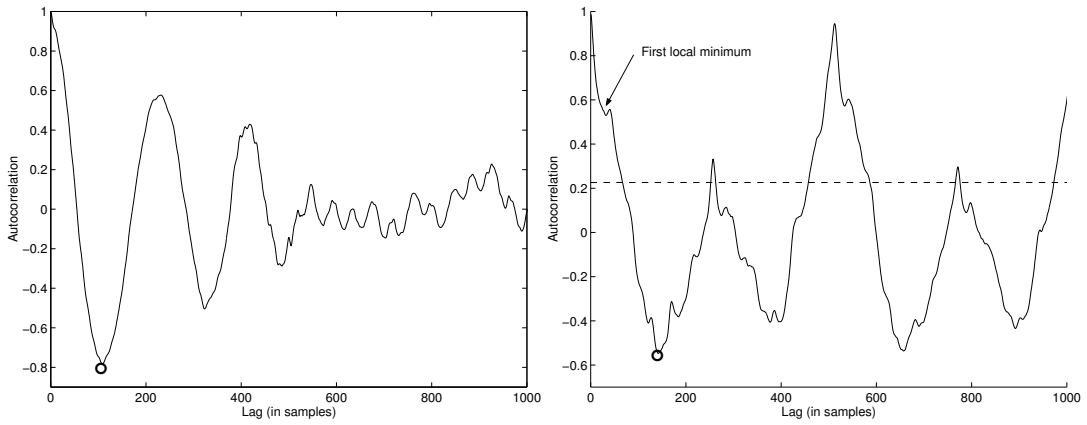
5.3.6.5 Modifications to the Harmonic Ratio

When implementing the above algorithm as defined in the FDIS document, two inconsistencies regarding the definition of indices for the maximum search in step 2 were observed:

- In the standard, the index range is written as $\tau = [1, N-1]$ (see Eq. 5.16), where N is the number of samples in each frame. But, as noted in step 1 of the algorithm, all the AC values for lags τ ranging from 1 to K must be computed, even in the case when $K > N$, to ensure peak detection with the lowest expected fundamental frequencies. Also, if the maximum had to be chosen only between 1 and $N-1$, it would not be necessary to compute the AC for the lags ranging from N to K. For these reasons, we believe that the rightmost limit of the search index should be K instead of $N-1$.

- In nearly all cases, the HR value is very close to 1, and it also corresponds to lag $\tau = 1$. This fact is illustrated in Fig. 5.7, where it can be observed that the first peak of the AC will most likely be the highest in the range $1, K$. This is specially the case for complex signals where, although there can be other peaks denoting periodicities, they are most likely to have less amplitude that the peak corresponding to little lags. In other words, for general, complex signals, adjacent samples are more correlated than samples separated by possible, detected periodicity intervals.

  Therefore, in this work the leftmost limit of the range is modified in order to skip the first peak of the AC when searching for the maximum. The goal is to find the lag corresponding to the first minimum of the AC after the first peak, marked by the circles in the figure. This is straightforward in the common situation described in Fig. 5.7(a), where this minimum equals the first local minimum of the AC function. However, this first peak often contains some ripping, as in Fig. 5.7(b). In this second situation, the first local minimum would correspond to the first minimum in the ripping, pointed by the arrow in the figure, and the main peak would not be correctly skipped. In order to avoid this, only AC minima with values below a certain limit l have been considered. This limit is defined by



(a) Smooth main peak

(b) Ripped main peak

Figure 5.7: Examples of two common autocorrelation forms of a frame (see text for details).

$$limit = \frac{1 + \min\limits_{\tau=1,K}\{AC_r[\tau]\}}{2} \tag{5.17}$$

and is shown in Fig. 5.7(b) by the horizontal dotted line.

Following the above considerations, the following modified algorithm for the extraction of *HarmonicRatio* has been implemented in this work:

1. Corresponds to the first step of the standard.

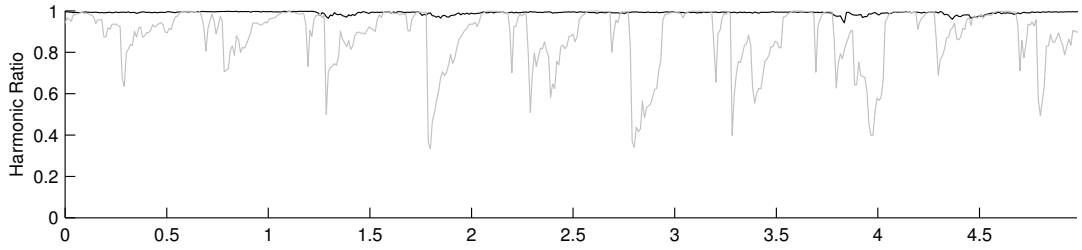
2. The Harmonic Ratio of one frame r is defined as:

$$HR_r = \max_{\tau=\tau_1,K}\{AC_r[\tau]\} \tag{5.18}$$

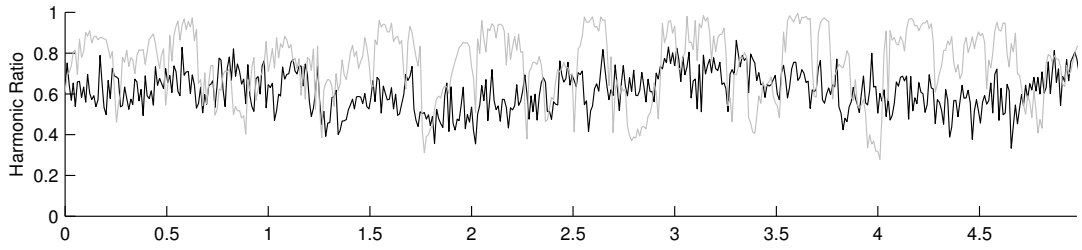
where τ_1 is the lag of the first minimum of $AC_r[\tau]$ in the range $\tau = [\tau_{limit}, K]$, and τ_{limit} is the lag of the first value of $AC_r[\tau]$ below the $limit$ value given by Eq. 5.17.

Figure 5.8 shows the two definitions of the Harmonic Ratio used in this work. It can be seen that, in the case of the classical example, the MPEG-7 definition yields always values very close to one. Although Fig. 5.8(a) seems to indicate that the standard definition could work well in separating classical from pop music, the fact is that its value is always near one for many other types of audio signals, including speech, background and all classical subgenres. This is shown in Fig.

5.9, where the harmonic ratios of a speech, a music and a background examples are compared. It can be noted that, on the one hand, the vertical scaling of the standard definition is much smaller than that of the modified version. On the other hand, the signals are more easily separable in Fig. 5.9(b). This will be confirmed in Chapter 6, where it will be shown that the modified version has proven to perform better in the classification task.



(a) MPEG-7 Harmonic Ratio



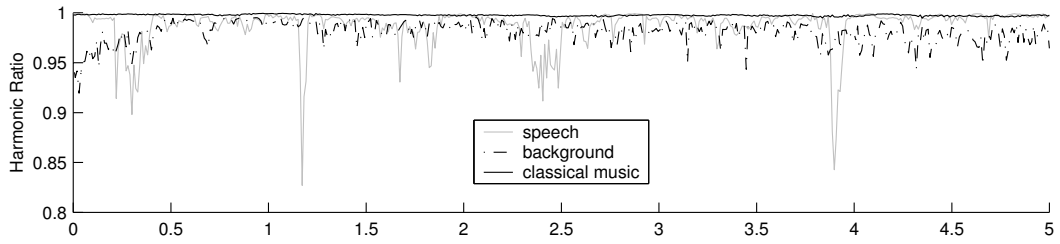
(b) Modified Harmonic Ratio

Figure 5.8: The two definitions of the Harmonic Ratio used in this work from the examples plotted in Figs. 5.1(a) and 5.1(b).

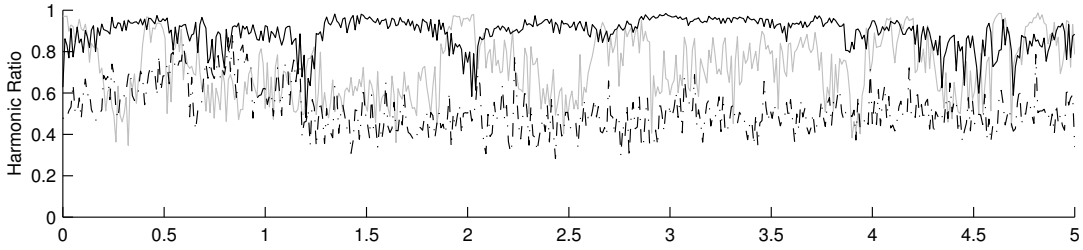
5.4 Rhythm Features

Tempo and rhythm are music-specific properties that were proposed as features for signal analysis when non-speech audio signals began to be taken into consideration. However, just measuring the tempo in bpm[9] is not interesting in the context of genre classification, since different pieces belonging to the same genre can have very different tempo properties (for example, two movements belonging to a symphony, or a rock ballad compared to faster rock songs), and pieces from different genres can have the same tempo.

[9]beats per minute



(a) MPEG-7 Harmonic Ratio



(b) Modified Harmonic Ratio

Figure 5.9: The two definitions of the Harmonic Ratio used in this work for speech, music and background examples.

In contrast, it is most interesting to extract information about the rhythmical structure. For example, about the regularity of the beats, which is expected to be much higher in the case of rock and pop music than in the case of classical music. Beat strength seems to be also a valuable feature. For example, at a more precise level, it is likely to be higher in techno music than in jazz.

5.4.1 Beat Histograms

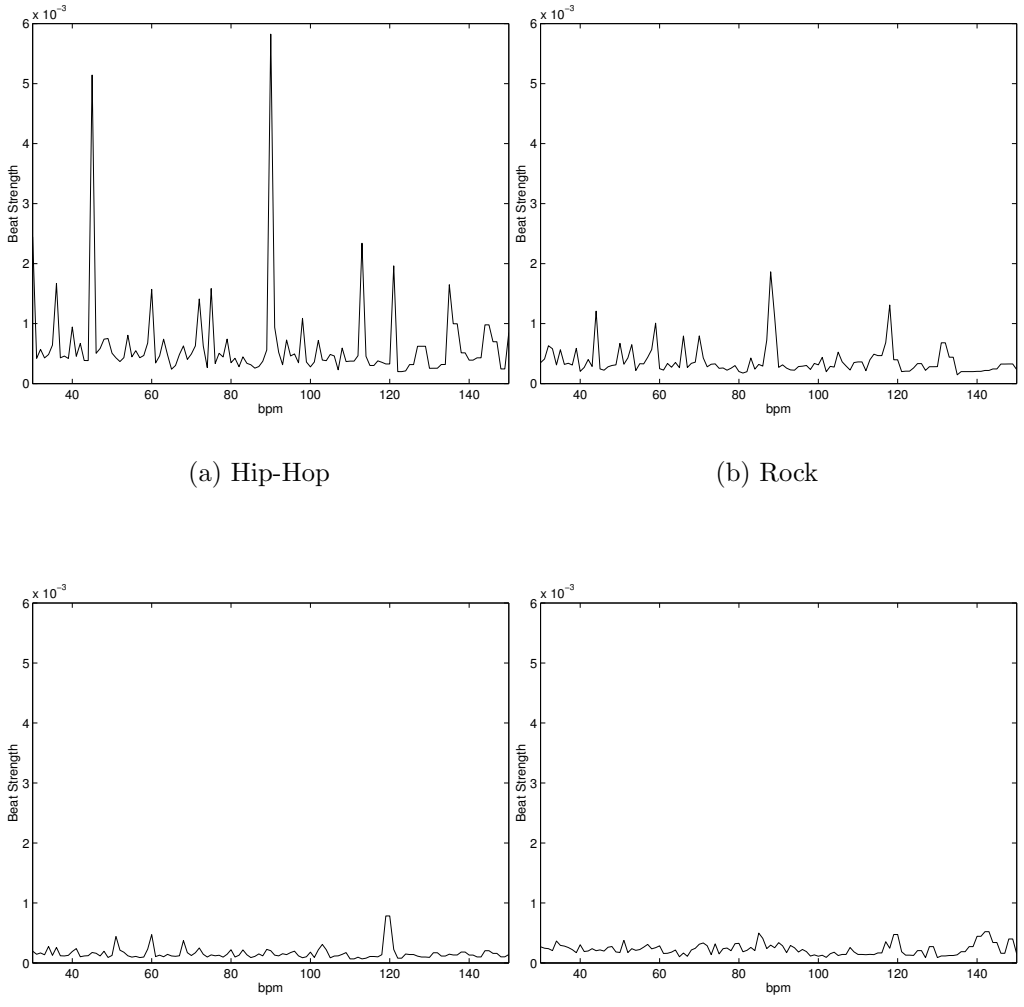
Beat histograms allow the extraction of such properties by calculating the beat strength of a signal along a wide range of bpm values. In this way, peaks on the histogram correspond to the main beat (most probably the highest peak) and other subbeats (multiples or divisors of the main beat). Several methods have been proposed for its computation [14, 44, 36]. In this work, an implementation provided by zplane.development, which is in turn based on the method described in [36], has been used. This implementation performs the following steps to obtain the histogram:

1. First, the 64-point STFT is computed for each frame.

2. The absolute difference between each FFT coefficient and its corresponding coefficient from the previous frame is computed.

3. The resulting difference values are added, obtaining a single value that measures the amount of spectral change for the current frame (similar to the flux feature described above).

4. The number of time frames is reduced by an integer downsampling factor D. In this work, a factor of $D = 4$ is used.



(a) Hip-Hop (b) Rock

(c) Jazz (d) Classical

Figure 5.10: Beat histogram examples. High peaks correspond to high beat strength. Peaks separated by integer bpm multiples denote rhythm regularity.

5. The resulting frame values are filtered by a resonator filterbank consisting of comb filters tuned to the frequencies corresponding to all possible bpm values. In this work, a range of 30 to 150 bpm is used.

6. The value of the histogram at a given number of bpm is taken as the mean power of the output of its corresponding comb filter.

The result is a curve describing beat strength as a function of the bpm values. In Fig. 5.10 some examples of beat histograms of representative signals belonging to different classes are shown. Figure 5.10(a) shows the beat histogram of a Hip-Hop excerpt from Cypress Hill. The high peaks denote a high overall beat strength. The main peak is at 90 bpm and the second at 45 bpm, denoting a high regularity corresponding to a *2/4* time signature. In contrast, Fig. 5.10(b), shows less beat strength in a rock example by Green Day. The jazz example by Kenny Burrell (Fig. 5.10(c)) has a significantly lower beat strength, but it is still possible to distinguish the main peak at 120 bpm. Finally, the classical example shown in Fig. 5.10(d) (excerpt from a string quartet by Dvořák) has an irregular beat histogram in which no clear main peaks can be detected, and no clear regularity factors, neither.

It should be noted that, although the computation is performed on a frame-by-frame basis, histograms are obtained in long-term intervals given by the texture windows. For this reason, all of the features related to the beat histogram are single-valued features to which the time-domain mean and standard deviation subfeatures are not applicable.

Once the histogram has been obtained, the next step consists of obtaining meaningful features from it. As mentioned above, beat strength (or *beatedness*) and rhythmic regularity are the desired properties to be extracted for the classification task. They will be treated separately in the next two sections.

5.4.2 Beat Strength

The amplitudes of the peaks at the different bpm values in the histogram measure the beat strength at the main tempo and its metric factors and subdivisions. To obtain an overall measure of beat strength, the following statistical measures of the histogram have been evaluated in this work:

- mean
- standard deviation
- mean of the derivative
- standard deviation of the derivative
- skewness

- kurtosis
- entropy

These measures are computed in the "beat domain", and should not be confused with the time-based statistical measurements mentioned throughout the rest of the work (including the subfeatures introduced in Sect. 5.1.1). They have been introduced in Chapter 2 except for the *entropy*, which measures the unpredictability (interpreted as amount of information) of a signal, and is given in this case by:

$$H(\mathcal{B}) = -\sum_{i=bpm_{min}}^{bpm_{max}} B[i]\log_2(B[i]) \tag{5.19}$$

where $\mathcal{B}$ denotes the beat histogram normalized to its sum and $B(i)$ the beat strength at i bpm.

5.4.3 Rhythmic Regularity

As mentioned in Sect. 5.3.6.4, autocorrelations (ACs) can be used to measure the periodicity of signals. Intervals between peaks on the autocorrelation function correspond to periods of the signal. To measure Rhythmic Regularity (RR), we explore here two features based on the periodicity of the beat histogram as given by its normalized autocorrelation:

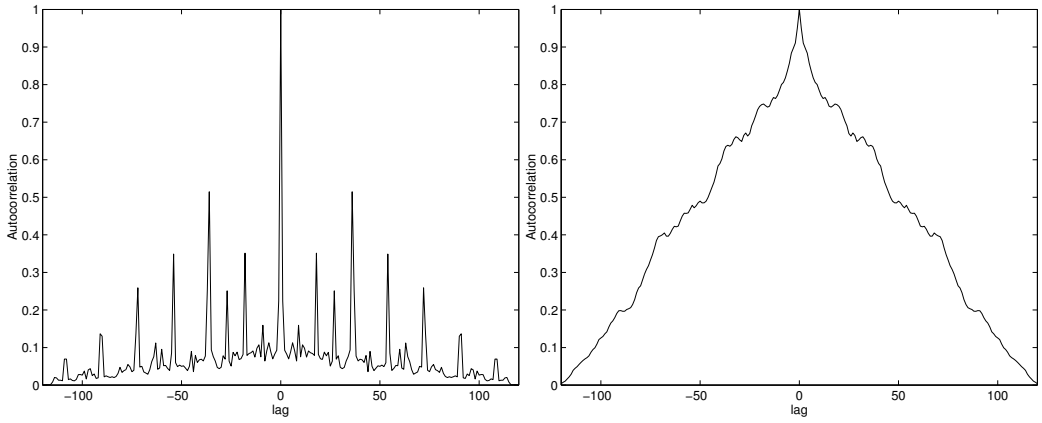
$$AC(\mathcal{B},\tau) = \frac{\sum_{n=0}^{N_b-1} B[n]B[n-\tau]}{\sqrt{\sum_{n=0}^{N_b-1} B[n]^2}}, \quad -N_b < \tau < N_b \tag{5.20}$$

where n is the element index of the histogram (not the bpm values). N_b is the total number of elements of the histogram, that is, $N_b = bpm_{max} - bpm_{min}$.

There are several possibilities to normalize an autocorrelation (see Eq. 5.15). In this case, the AC is normalized so that its value at lag $\tau = 0$ equals unity.

Figure 5.11 plots the autocorrelations of the file-based beat histograms of a pop example and of a string quartet example. High peaks on these plots (like in the pop example) denote high periodicity, that is, high regularity between the rhythmic subdivisions. Since ACs are symmetric about their zero lag, we only need to retain a half of their values. To obtain a single number describing this periodicity, the following two values have been evaluated as possible features:

- *standard deviation of the derivative of the autocorrelation:* Figure 5.12 shows the derivatives of the second halves of the autocorrelations of the two above samples. High changes in the derivatives correspond to high peaks in the AC, and will be reflected in high values of its standard deviation.



(a) Pop (b) String Quartet

Figure 5.11: Beat histogram autocorrelation examples. Music with regular beats, like pop, shows high peaks in the autocorrelation.

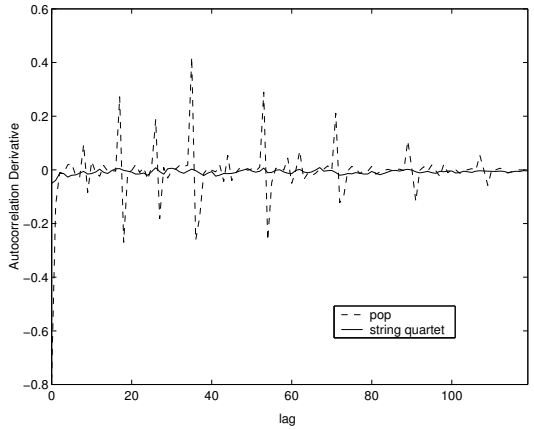


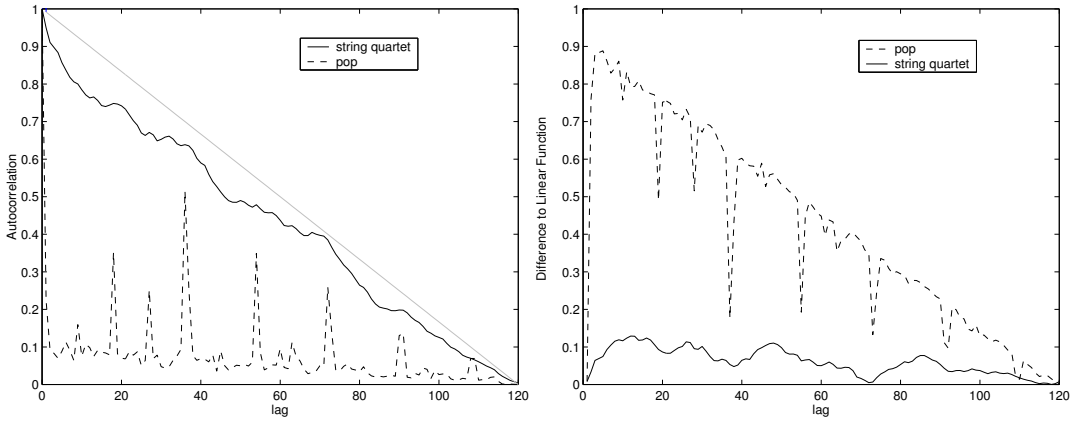
Figure 5.12: Illustration of the *RR1** feature. The figure shows the derivatives of the autocorrelations of a pop and a classical example.

- *mean of the difference between the autocorrelation and a linear function ranging from 0 to the the peak of the autocorrelation:* This can be regarded as a measure of the linearity of the autocorrelation, and will yield high values for ACs with strong peaks:

$$RR = \frac{1}{N_b} \sum_{\tau=0}^{N_b} (1 - \frac{\tau}{N_b} - AC(\mathcal{B}, \tau)) \tag{5.21}$$

Figure 5.13 illustrates this feature. Figure 5.13(a) again shows the two above autocorrelations, this time superimposed and compared to the linear function to which the difference is computed. Figure 5.13(b) shows the resulting difference functions. It is clear that their mean value will be higher for signals with higher rhythmic regularity.

We will refer to these two definitions as *Rhythmic Regularity 1* and *Rhythmic Regularity 2*, respectively.



(a) Histogram autocorrelations and linear function

(b) Difference functions

Figure 5.13: Illustration of the *RR2** feature (see text for details).

5.5 Other Features

The features grouped in this last section describe the signal regarding its dynamic properties (Root Mean Square, Envelope, Low Energy Rate, Loudness), its statistical behavior (Central Moments) and its predictivity (Predictivity Ratio).

It can be noted that, in this work, no features describing pitch in an explicit manner have been considered. As mentioned, single pitch extraction is unreliable for multi-voiced signals. On the other hand, it is questionable if pitch contents can perform well in distinguishing music types, since virtually any pitch range can be found in any genre.

5.5.1 Root Mean Square

It has been mentioned that the amplitude scaling of the audio signals should be considered irrelevant for the classification. However, the middle or long-term

variation in time of the instantaneous amplitude values can provide useful information in distinguishing audio types. For example, classical music is more likely to have a greater dynamic range than pop or rock music. In the same way, silences between words or phrases result also in rapid amplitude changes in the case of speech.

For these reasons, a feature describing the smoothed trajectory of the amplitude is desired. Smoothing is required to avoid the high-frequent oscillations of the signal around the horizontal axis (that is, the zero crossings) and to obtain a description of the dynamic tendency, rather than of the instantaneous amplitude of the samples. One possible way to perform this smoothing is to compute the RMS energy of the signal in each frame, as given by Eq. 2.3, where, in this case, N is the number of audio samples in each analysis window.

5.5.2 Envelope

Another possibility for the above mentioned smoothing is to take the maximum of the absolute amplitude values in each frame. The result is an envelope trajectory that lies on the peaks of the time-domain signal:

$$ENV_r = \max_{n=1,N} \{|x_r[n]|\} \tag{5.22}$$

To gain insight into the subtle differences between this definition of envelope and the RMS value of the last subsection, both curves have been superimposed upon a short excerpt from the above pop example in Fig. 5.14. It can be seen that the envelope curve lies on the peaks, while the RMS curve represents more averaged values of the amplitude.

5.5.3 Low Energy Rate

Another, much more reduced form for describing energy variation is given by the *low energy rate* feature [35, 44]. It is defined as the percentage of frames within

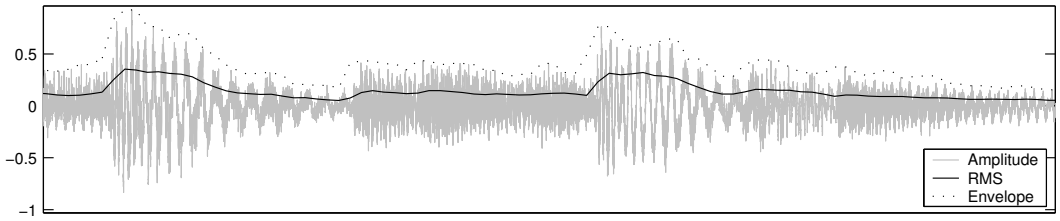


Figure 5.14: Illustration of RMS and envelope curves.

a texture window that have an RMS energy lower than the mean RMS energy across that texture window.

It should be noted that, apart from the beat-histogram-based features, this is the only feature that is not computed on a frame-by-frame basis, but on a texture window basis.

5.5.4 Loudness

The previous dynamic-related features are based on physical measures like amplitude or energy. A better adaptation to the human perception of sound dynamics is provided by the simple approximation of loudness introduced in Sect. 2.2.1. Here, the energy-based loudness approximation for each frame will be evaluated as a feature:

$$L_r = E_r^{0.23} \tag{5.23}$$

where E_r is the energy of the current frame as given by Eq. 2.1. Since we are interested in the temporal variation rather than in the absoulute values, the k constant in Eq. 2.14 has been ignored.

Figure 5.15 shows plots of the frame-based amplitude-related features presented in this section.

5.5.5 Central Moments

The estimated third and fourth order central moments of the time-domain audio signal, that is, its *sample skewness* and its *sample kurtosis* are evaluated here as possible audio features. They are given by equations 2.24 and 2.25, where in this case the samples are the amplitudes of the audio samples of the audio signal, and N is the total number of audio samples within one analysis window.

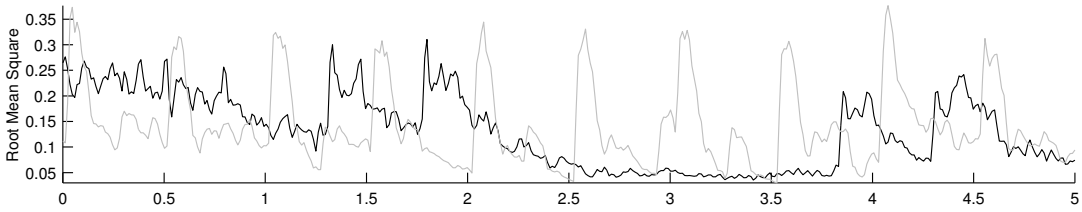
Skewness and kurtosis are mathematical descriptions of the statistical behavior of the signal, and have no straightforward interpretation in a timbral or perceptual context.

Curves of the central moments for the same music examples are plotted in Figs. 5.16(a) and 5.16(b), respectively. The peaks on both curves correspond to the strongest onsets of the amplitudes.

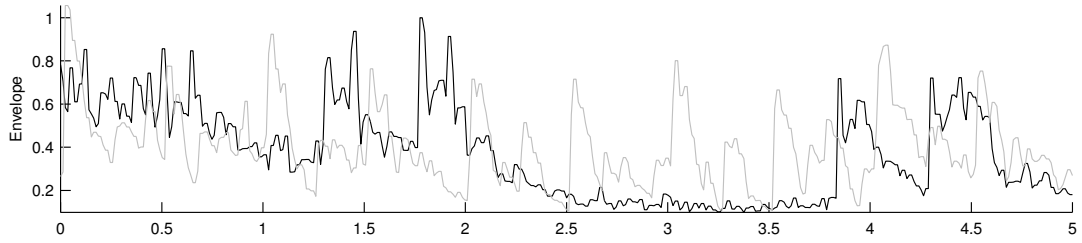
5.5.6 Predictivity Ratio

A *p-order linear prediction* $\hat{x}[n]$ of a sample $x[n]$ is a prediction of its amplitude value as a linear combination of its past p samples:

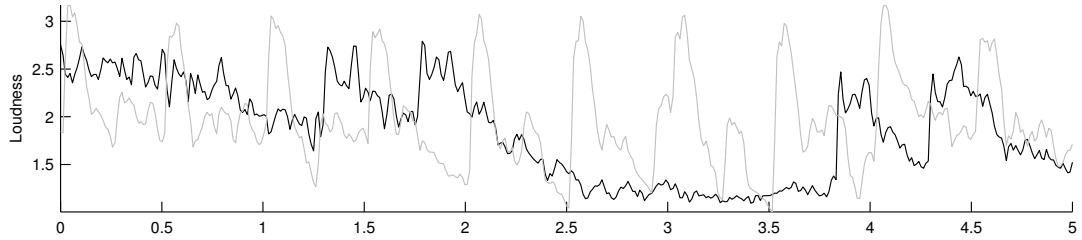
$$\hat{x}[n] = a_1 x[n-1] + a_2 x[n-2] + \ldots + a_p x[n-p] \tag{5.24}$$



(a) Root Mean Square



(b) Envelope

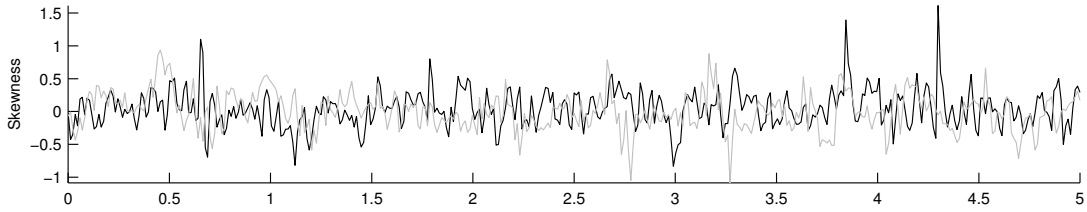


(c) Loudness

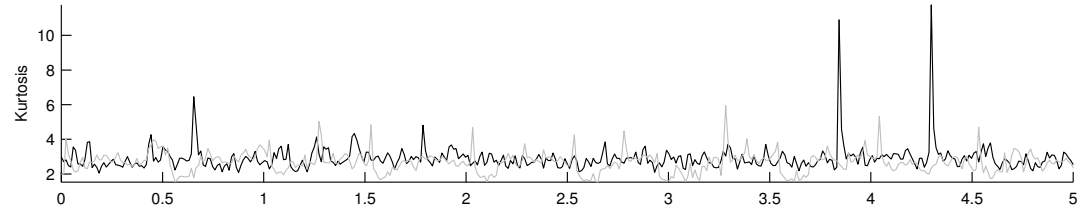
Figure 5.15: Amplitude-related features from the examples plotted in Figs. 5.1(a) and 5.1(b).

The a_i coefficients are called the *Linear Prediction Coefficients (LPC)*, and can be obtained by one of several proposed algorithms, which aim at obtaining a prediction error as lowest as possible. In this work, the MATLAB built-in implementation was used, which is based on the so-called autocorrelation method of autoregressive modeling [18].

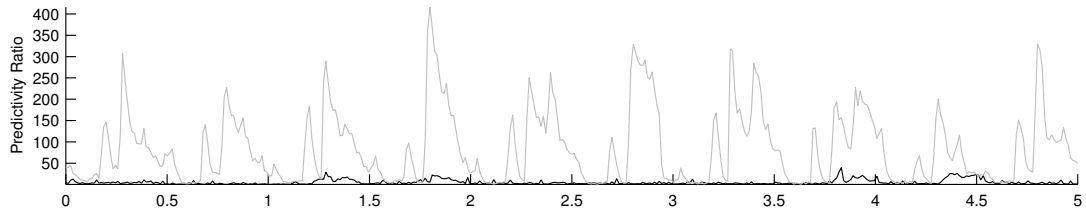
One possibility to measure the prediction error is to compute the ratio of the energy of the predicted signal to the energy of the original signal. Of course, this computation will be performed here on a frame-by-frame basis:



(a) Skewness



(b) Kurtosis



(c) Predictivity Ratio

Figure 5.16: Statistical and predictivity features from the examples plotted in Figs. 5.1(a) and 5.1(b).

$$PR_r = \frac{\sum_{n=0}^{N} |\hat{x}[n]|^2}{\sum_{n=0}^{N} |x[n]|^2} \tag{5.25}$$

An order of $p = 12$ was used here for the linear prediction. Signals with sudden amplitude changes and high noise components are more likely to yield Predictivity Ratio values far from unity. The Predictivity Ratio feature is plotted in Fig. 5.16(c).

5.6 Feature Overview and Naming Convention

In this Chapter, a total number of 20 frame-based features, plus one texture-based feature and 9 beat-histogram-based features have been reviewed. Furthermore, the 4 subfeatures introduced in Sect. 5.1.1, which are applicable to all of the 20 frame-based features, make a total number of 90 available different features, from which a subset will be selected to implement the classification system.

During the design phase described in the next sections, all these features are subjected to systematical analysis, and will be mentioned often. To facilitate an overview and improve readability in the rest of the work, all of them are summed up in the next tables, assigning to each one a short naming convention.

A feature will be denoted by the following notation:

$$FEATURE/SF$$

where SF is one of the four subfeatures given in table 5.1, and $FEATURE$ is the short name of the corresponding frame-based feature. For example, $LOUD/DS$ is the standard deviation of the derivative of the loudness. Single-valued features, to which subfeatures are not applicable, will be denoted by *. In table 5.2 all the features are listed in alphabetical order.

Short name	Full name
M	Mean
S	Standard deviation
DM	Mean of the derivative
DS	Standard deviation of the derivative

Table 5.1: Subfeatures naming convention.

Short name	Full name	Section
BH	Beat Histogram	5.4.2
*BHE**	Beat Histogram Entropy	5.4.2
*BHS**	Beat Histogram Skewness	5.4.2
*BHK**	Beat Histogram Kurtosis	5.4.2
CENTR	Centroid	5.3.2
ENV	Envelope	5.5.2
FLUX	Flux	5.3.4
KURT	Kurtosis	5.5.5
*LE**	Low Energy Rate	5.5.3
LOUD	Loudness	5.5.4
MFCC1	First MFC coefficient	5.3.5
MFCC2	Second MFC coefficient	5.3.5
MFCC3	Third MFC coefficient	5.3.5
MFCC4	Fourth MFC coefficient	5.3.5
MFCC5	Fifth MFC coefficient	5.3.5
M7CEN	MPEG-7 Centroid	5.3.6.1
M7SPR	MPEG-7 Spread	5.3.6.2
M7FLAT	Mean MPEG-7 Flatness	5.3.6.3
M7HR	MPEG-7 Harmonic Ratio	5.3.6.4
MODHR	Modified MPEG-7 Harmonic Ratio	5.3.6.5
PRED	Predictivity Ratio	5.5.6
*RR1**	Rhythmic Regularity (first definition)	5.4.3
*RR2**	Rhythmic Regularity (second definition)	5.4.3
RMS	Root Mean Square	5.5.1
ROLL	Rolloff	5.3.3
SKEW	Skewness	5.5.5
ZC	Zero Crossings	5.3.1

Table 5.2: Alphabetical listing of features and their short names. Features marked by * are single-valued and thus not compatible with the subfeatures of table 5.1.

Chapter 6

Feature Selection

The *curse of dimensionality*, which was introduced in Sect. 2.4.5, implies that it is advantageous to reduce the number of features (that is, the number of dimensions in the feature space) in order to reduce computational costs while keeping similar levels of performance, and in some cases even to improve the classification rate.

In a pattern classification application, a well-designed feature should have the two following general properties:

Invariancy to irrelevancies. A good feature should be invariant to irrelevant transformations of the input signal. In machine vision applications, irrelevancies can include translations, rotations and changes in scale of the objects. In the audio context, they can include signal quality in respect of noise and bandwidth, the number of channels, distortions or the overall amplitude scaling.

Good discriminative power. A feature should take similar values within a given class, but very different values across classes.

Uncorrelation to other features. Redundant information should be avoided in the feature space. Each new feature should add as new information about the object as possible. In a geometrical context, uncorrelation corresponds to orthogonality.

In this work, features are selected out of the initial list of 90 available features in a completely systematical way. We proceed in two steps: the first step corresponds to the above criterium of invariancy, the second to the criteria of discriminative power and uncorrelation.

In the present Chapter, dimensionality reduction techniques are presented first. Advantages and drawbacks of different approaches are overviewed (Sect. 6.1 to 6.3), and the algorithm chosen is explained more in detail (Sect. 6.4). Finally, the results of the selection are given in Sect. 6.5.

6.1 The Curse of Dimensionality

It is logical to think that adding a new dimension to the feature space will improve the performance of the classifier. In other words, it seems desirable to examine as many characteristics of the input object as possible in order to make a decision about its class. In the worst case, the added feature will not provide much extra information, but, intuitively, this is not likely to harm the performance; at least, it will be kept the same.

In fact, adding a feature *does* provide new information about the object to be classified, but that is only one part of the matter. In realistic pattern recognition tasks, two main limitations exist. On the one hand, the size of the training set is finite. On the other hand, decisions are derived from this limited set, either by direct comparison (nonparametric methods) or by density estimation.

The accuracy of these decision rules depends on the number of available samples. In particular, to keep the same classifying accuracy, the density of the samples in the feature space must remain the same after adding new dimensions. When adding a feature, we can either choose to

- maintain the same density by increasing the number of samples or to
- maintain the number of samples.

The key point of the problem is that, when adding a new dimension, the number of training samples must be increased exponentially in order to keep the same density. Therefore, if samples are costly to obtain, the first above approach must be discarded. The second approach will result in a feature space in which the samples are distributed very sparsely, which in turn can result in worse density estimations or worse classification accuracies.

As a result, for a given set of training samples, there will be a threshold number of features upon which the performance will stop growing, or even begin to decrease. This *curse of dimensionality* has been experimentally observed in the tests detailed in the next Chapter.

6.2 Dimensionality Reduction Methods

Dimensionality reduction is a general problem that arises not only in pattern recognition, but also in other signal processing fields such as signal representation and source separation. As it will be noted, not all of the available methods for dimensionality reduction are suitable for the specific problem of pattern recognition. In general, dimensionality reduction algorithms [7, 16, 42, 26] can be broadly classified into two groups:

Feature Space Transformations (FST)

Given a feature space $\mathbf{x}_i \in \mathbb{R}^N$, the problem consists in this case of finding a transformation $\mathbf{y} = f(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}^M$ with $M < N$ such that the transformed vectors $\mathbf{y}_i \in \mathbb{R}^M$ preserve most of the information or structure in $\mathbb{R}^N$. This mapping will be optimal if the probability of classification error remains the same after the transformation. Although linear transformations are suboptimal in this sense, they are often used because of their theoretical and computational tractability.

One of the most widely used linear FST techniques is *Principal Component Analysis (PCA)*, which seeks to preserve as much of the variance of the features as possible in the transformed space. Intuitively, it aligns the axes of the feature space with the directions of maximum variance. Thus, it is appropriate to obtain a low-dimensional accurate representation of a signal. In the context of signal compression, PCA is called the Karhunen-Loève Transform (KLT).

In contrast to PCA, *Linear Discriminant Analysis (LDA)* takes into account the class membership of the feature vectors in order to obtain a low-dimensional space in which most of the class discriminatory information is preserved. LDA is conceptually very similar to PCA, the difference being that, instead of maximizing variance, it maximizes class separability. In the case where only two classes are considered, LDA is also called *Fisher's Linear Discriminant*.

Feature Subset Selection (FSS)

The other approach to reduce dimensions is to select an optimal subset of features from the set of initially available features without using any transformation. More formally, given a feature set $\mathcal{X} = \{f_i | i = 1..N\}$, the problem consists of finding a subset $\mathcal{Y} = \{f_{i_1}, f_{i_2}, .., f_{i_M}\}$ with $M < N$ that optimizes a given *objective function* $J(\mathcal{Y})$. This objective function will be chosen depending on the context of the problem; in the case of pattern classification, usually a measure of class separability is used.

Another possibility is to take the final performance of the classifier as the objective function. In this case, the feature selection depends on the type of classifier chosen, and implies the training and validation of the whole system each time the objective function is to be evaluated. Since different classifiers are going to be evaluated in this work, we chose an objective function that relies on the intrinsic properties of the data, such as class separability, rather than on the final performance, in order to allow computational tractability and to gain in generality.

From the above methods, only LDA and FSS based on class separability are appropriate for a classification problem. PCA is not suitable because there is no guarantee that the directions of maximum variance will contain the best features for the discrimination into classes.

On the other hand, LDA has following limitations:

- In a C-class problem, LDA can produce at most $C-1$ feature projections. If more features are needed to prevent high error rates, some other method must be used.
- The transformed features are linear combinations of the original features. Thus, the correspondence to the physical meaning of the features is lost.
- LDA implies the computation of all the original features before applying the transformation.

These three major drawbacks can be however overcome with FSS techniques. In this case, the maximum number of selected features does not depend on the number of classes, allowing $M > C$. As it will be addressed in detail in the next Chapter, we adopt a hierarchical approach in which the taxonomy tree is implemented as a series of successive classification problems, each of which consisting of a number of classes between 2 and 4. In this case, LDA would yield only 1 to 3 feature projections in each of the individual classification steps. FSS also reduces computational costs in the feature extraction process, since only the final M selected features must be computed.

For the above reasons, an FSS algorithm, in particular a Sequential Forward Selection algorithm based on class separability, has been chosen in this work to perform feature selection, and will be detailed in the next two sections.

6.3 Class Separability Measures

Intuitively, two classes will be easily separable if each of the variances of their sample feature vectors are small with respect to the difference of their means. In other words, a feature will perform well in separating classes when:

- it takes similar values within each class
- it takes very different values across classes

In the two-class problem, one possibility to express this mathematically is to define class separability as the magnitude of a vector $\boldsymbol{\delta}$ whose elements are the ratio between the difference of the feature means and some combination of their variances or standard deviations, for example:

$$\delta_i = \frac{|\mu_{1i} - \mu_{2i}|}{\max\{\sigma_{1i}, \sigma_{2i}\}} \tag{6.1}$$

In the C-class problem, a measure of how the features are spread within each class ω_k is given by the *within-class scatter matrix*:

$$\mathbf{S}_W = \sum_{k=1}^{C} \sum_{\mathbf{x}\in\omega_k} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^T \tag{6.2}$$

where $\boldsymbol{\mu}_k$ is the sample mean vector of the N_k samples in class ω_k (the hat notation is omitted here for simplicity):

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{\mathbf{x}\in\omega_k} \mathbf{x}_j \tag{6.3}$$

The within-class scatter matrix can be regarded as an overall covariance matrix that takes into account class membership.

In contrast, the *between-class scatter matrix* measures how distinct the features across the classes are:

$$\mathbf{S}_B = \sum_{k=1}^{C} N_k(\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T \tag{6.4}$$

where $\boldsymbol{\mu}$ is the sample mean of all available N samples, as given by Eq. 2.27.

The first two intuitive criteria given at the beginning of this section correspond to finding a high *between-scatter* to *within-scatter* ratio. This can be obtained by reducing the above matrices to scalar values by means of their determinants or their traces[1], thus obtaining several possible variations on a generalized definition of *class separability*[2]:

$$J_1 = \frac{|\mathbf{S}_B|}{|\mathbf{S}_W|} = |\mathbf{S}_W^{-1}\mathbf{S}_B| \tag{6.5}$$

$$J_2 = \frac{\mathrm{tr}(\mathbf{S}_B)}{\mathrm{tr}(\mathbf{S}_W)} \tag{6.6}$$

$$J_3 = \mathrm{tr}(\mathbf{S}_W^{-1}\mathbf{S}_B) \tag{6.7}$$

The J_2 and J_3 criteria are more computationally efficient, since the computation of a trace requires less operations than the computation of a determinant. In this work, the J_3 criterion has been used as the objective function. Since a matrix inversion of $\mathbf{S}_W$ is implied in this definition, singular $\mathbf{S}_W$ matrices must

[1]The trace of a matrix $\mathbf{A}$, denoted by $\mathrm{tr}(\mathbf{A})$, is the sum of the elements on its diagonal.

[2]These class separability measures constitute also the base of Linear Discriminant Analysis. In particular, LDA consists of finding the linear transformation that maximizes the J_1 criterion of Eq. 6.5.

be avoided[3]. To achieve this, a previous *regularization* step is taken, by which a small constant is added to the diagonal of all within-scatter matrices:

$$\mathbf{S}_{W,reg} = \mathbf{S}_W + \alpha I \tag{6.8}$$

where I is an identity matrix of the same size as $\mathbf{S}_W$ and, in our case, $\alpha = 0.1$.

6.4 Feature Subset Selection

A naïve approach to feature selection for pattern classification would be to choose one of the above definitions of class separability as the objective function J and compute its value for each of the features (that is, compute the separability by considering each time only a single feature in the feature space). Features could be ranked according to their separability score, and the best M would be selected from them.

However, if we proceed in this way, correlation between the features is not taken into account. This is likely to produce very poor results, since, as mentioned above, the features in an optimal subspace should not only contain as much information about the signal as possible, but they also should be as independent to each other as possible to prevent from redundant information.

For this reason, not single features (*scalar FSS*) but subsets of features should be used to evaluate the objective function (*vectorial FSS*). However, a number of 2^N subsets can be formed out of an initial set of N features, making an exhaustive testing of all possible subsets unfeasible. As an example, in this work a total number of 90 features are considered, which results in $2^{90} \approx 1.2 \cdot 10^{27}$ possible subsets. A *search strategy* is needed to direct the selection process across the feature space, to avoid evaluating all possible combinations of features.

In this work, a *Sequential Forward Selection* algorithm [16] is used as the search strategy. It consists of the following steps (the subindices of the feature set $\mathcal{Y}$ denote the steps in the algorithm):

1. Start with the empty feature set $\mathcal{Y}_0 = \{\emptyset\}$.
2. Out of the features that have not yet been chosen, select the one feature f^+ that maximizes the objective function in combination with the previously selected features: $f^+ = \underset{f \in \mathcal{X} - \mathcal{Y}_s}{\operatorname{argmax}} \{J(\mathcal{Y}_s \cup f)\}$.
3. Update: $\mathcal{Y}_{s+1} = \mathcal{Y}_s \cup f^+$, $s \rightarrow s + 1$.
4. Go to 2.

[3]A singular matrix is a square matrix that does not have an inverse matrix. A matrix is singular if and only if its determinant is zero.

6.5 Results of the Feature Selection

As mentioned at the beginning of this Chapter, feature selection has been performed in two steps. In the first step, all initially available features were tested for robustness to irrelevant transformations of the signals. The remaining features were subjected to a Sequential Forward Selection algorithm based on class separability, resulting in a set of lists in which the features are ranked according to their overall quality. The actual number of features finally chosen for classification was determined by observing the behavior of the classification rate when increasing dimension, as it will be detailed in the next Chapter.

6.5.1 Tests on Robustness to Irrelevancies

To ensure similar classification rates across a wide range of audio qualities, the following signal properties have been regarded as irrelevant in the context of this work:

Amplitude scaling. The average power of an audio signal should have no influence in making the decision about genre. This irrelevancy has already been eliminated by normalization in the pre-processing step outlined in Sect. 5.2.

Noise content. Audio should be correctly classified even if it has a moderately noisy background.

Bandwidth. [4] The bandwidth at which the audio has been recorded or transmitted should have as little influence in the classification as possible.

As the first step in the feature selection, those features that were most susceptible to noise and to moderate changes in the signal bandwidth were discarded. This has been achieved by the following two tests:

6.5.1.1 Noise Test

In order to test the features for robustness against the addition of noise, we chose four representative training samples belonging to the speech, classical music, popular music, and background noise classes. Each example was normalized and mixed with white gaussian noise of -25 dB RMS power and subjected to file-based feature extraction. The resulting features of the four noisy signals were compared with the ones extracted from the original signals by obtaining the ratio between them. The variations were averaged across the four samples. A variation threshold for discarding features has not been used, since the variation values depend on the noise power used for the test. Instead, we set a ranking

[4] Not to be confused with *instantaneous bandwidth* or *spread* (see Sect. 5.3.6.2)

threshold of 20 features, which seems a reasonable percentage of the initial 90 dimensions. Thus, the 20 worst features in the ranking were discarded.

The results are shown in tables 6.1 and 6.2, which list the 20 features that varied least and most, respectively, after the addition of noise. Numerical values in the table give the factor of relative variation of each noisy feature with respect to each original feature. For example, 0 corresponds to no variation while 0.1 corresponds to a 10% variation above or under the original value. For now on, the features will be mentioned by their short names, as listed in table 5.2.

The following general conclusions can be drawn from the results of the noise test:

- The *DS* subfeatures (standard deviations of the derivatives of the underlying features) are especially robust to noisy changes in the signal.
- The *M* and *DM* subfeatures (means and means of the derivatives of the underlying features) are in most cases highly sensitive to noise.

6.5.1.2 Filtering Test

To test the feature invariance to changes in bandwidth, the same four samples used for the noise test were lowpass-filtered with a cut-off frequency of 11025Hz. Filtered features are compared to original features in the same way, obtaining the averaged results in tables 6.3 and 6.4. Again, the 20 worst features were discarded.

Observing the results of the filtering test, we can conclude the following:

- MFCCs are highly robust to lowpass filtering (except for their *DM* variants).
- The *M* and *DM* subfeatures are in most cases highly sensitive to lowpass filtering.
- The Predictivity Ratio (*PRED*) feature is extremely sensitive to lowpass filtering in all of its four subfeature variants . The four *PRED*-related subfeatures belong to the bottom-five of the ranking.

As a result of both noise and filtering tests, a total number of 32 features (the ones that appear at least once in both bottom-ranking lists) were discarded. The following features are present in the lists in all of their subfeature variants, and therefore they were not implemented at all in the prototype application:

Ranking	Feature	Variation
1.	*LE**	0.0009
2.	*BHE**	0.0031
3.	*RMS/DS*	0.0042
4.	*MFCC4/DS*	0.0055
5.	*MFCC5/DS*	0.0085
6.	*RMS/S*	0.0089
7.	*MFCC3/DS*	0.011
8.	*BH/DS*	0.011
9.	*M7SPR/DS*	0.012
10.	*BH/S*	0.013
11.	*LOUD/DS*	0.015
12.	*BH/M*	0.019
13.	*BH/DM*	0.019
14.	*LOUD/M*	0.021
15.	*M7CEN/DS*	0.026
16.	*MFCC3/S*	0.030
17.	*BHK**	0.030
18.	*RR1**	0.033
19.	*M7SPR/S*	0.036
20.	*RR2**	0.040

Table 6.1: Noise test: 20 best features.

Ranking	Feature	Variation
90.	*MFCC1/DM*	11.89
89.	*M7CEN/M*	10.63
88.	*KURT/DM*	6.94
87.	*MFCC5/M*	4.61
86.	*ROLL/DM*	4.24
85.	*M7FLAT/DM*	2.07
84.	*MODHR/DM*	1.90
83.	*KURT/DS*	1.43
82.	*MFCC3/DM*	1.42
81.	*KURT/S*	1.27
80.	*MFCC4/M*	1.20
79.	*LOUD/DM*	1.06
78.	*BHS**	1.04
77.	*RMS/DM*	0.88
76.	*PRED/M*	0.86
75.	*PRED/S*	0.80
74.	*MFCC4/DM*	0.74
73.	*M7HR/DM*	0.73
72.	*PRED/DS*	0.72
71.	*SKEW/DS*	0.68

Table 6.2: Noise test: 20 worst features.

Ranking	Feature	Variation
1.	*MFCC2/DS*	0
2.	*MFCC1/DS*	0.0001
3.	*MFCC5/DS*	0.0001
4.	*MFCC3/DS*	0.0001
5.	*MFCC4/S*	0.0001
6.	*MFCC2/M*	0.0001
7.	*MFCC4/M*	0.0002
8.	*MFCC3/M*	0.0002
9.	*MFCC5/S*	0.0002
10.	*LE**	0.0003
11.	*MFCC2/S*	0.0003
12.	*MFCC3/S*	0.0004
13.	*MFCC5/M*	0.0007
14.	*MFCC4/DS*	0.0008
15.	*MFCC1/S*	0.0009
16.	*MFCC1/M*	0.0013
17.	*MODHR/DM*	0.0014
18.	*BHE**	0.0037
19.	*ENV/S*	0.0037
20.	*MODHR/DS*	0.0047

Table 6.3: Filter test: 20 best features.

Ranking	Feature	Variation
90.	*PRED/S*	2.66
89.	*PRED/M*	2.64
88.	*PRED/DM*	2.23
87.	*MFCC5/DM*	1.46
86.	*PRED/DS*	1.27
85.	*KURT/DM*	1.14
84.	*LOUD/DM*	0.98
83.	*ROLL/S*	0.62
82.	*M7FLAT/DM*	0.56
81.	*ROLL/DS*	0.46
80.	*CENTR/S*	0.41
79.	*M7HR/DS*	0.41
78.	*MFCC4/DM*	0.37
77.	*CENTR/DS*	0.34
76.	*CENTR/DM*	0.31
75.	*ROLL/M*	0.31
74.	*CENTR/M*	0.30
73.	*M7HR/S*	0.30
72.	*M7HR/DM*	0.29
71.	*SKEW/DM*	0.27

Table 6.4: Filter test: 20 worst features.

- Skewness of the Beat Histogram (*BHS**)
- Centroid (*CENTR*, non-MPEG-7 definition)
- Predictivity Ratio (*PRED*)
- Rolloff (*ROLL*)

6.5.2 Results of the Feature Subset Selection

The remaining 58 features were subjected to further selection according to their class separability capabilities. In the next Chapter, a direct and a hierarchical approach to classification will be evaluated and compared (Sect. 7.2). These approaches also have their counterparts in the feature selection process, and will be treated here separately.

6.5.2.1 Direct Feature Subset Selection

The direct approach to classification is a single-stage 17-class problem (see Fig. 7.2). In this case, all training samples were used for the feature selection using the overall separability of the 17 classes as the criterion function. Applying the Sequential Forward search based on the J_3 criterion defined in Eq. 6.7 yields a list in which the 58 features are ordered according to their quality in separating classes. The best 20 entries of that list are given in table 6.5.

It should be noted that it is not possible to give an absolute numerical measure of the quality of each feature, since its position in the ranking depends on the previously selected features.

6.5.2.2 Genre-dependent Feature Subset Selection

The hierarchical approach consists of successive classification problems with a number of classes ranging from 2 to 4, with the hierarchy corresponding to the audio taxonomy tree depicted in Fig. 4.1. Rather than using the whole training database to obtain a single list of selected features, a genre-dependent feature selection is proposed in this work, in which only the training samples belonging to the current branch in the classification tree are used to evaluate the separability of the current 2, 3 or 4 classes.

As a result, a set of 9 feature lists was obtained, one for each split in the tree. These lists provide useful information about which features best distinguish between a given set of music or audio subgenres. They are not only useful for the implementation of the system, but also for gaining insight into the physical, perceptual, or musical measures that are appropriate in differentiating a given subgenre. In this way, they can be taken as a guideline to future implementations of more sophisticated genre-dependent features. The best 20 features for each

Ranking	Feature
1.	*BHE**
2.	*MFCC2/S*
3.	*LOUD/DS*
4.	*FLUX/S*
5.	*M7FLAT/M*
6.	*MFCC2/M*
7.	*RR2**
8.	*MFCC4/S*
9.	*KURT/M*
10.	*LOUD/S*
11.	*MFCC1/S*
12.	*MFCC3/M*
13.	*M7FLAT/DS*
14.	*LOUD/M*
15.	*M7SPR/S*
16.	*MFCC5/DS*
17.	*M7SPR/M*
18.	*ZC/M*
19.	*ZC/DS*
20.	*M7CEN/S*

Table 6.5: Direct feature selection: best 20 features.

split are given in tables 6.7 to 6.15. The captions in the tables refer to the split naming convention presented in table 6.6. We will often use this convention from now on.

The following conclusions can be drawn from the results of the genre-dependent feature selection:

- While the *M*, *S*, and *DS* subfeatures appear often throughout the best-20 tables, the *DM* subfeature appears only 4 times (from a total of $9*20 = 180$ features). That is, the mean of the derivative is not an effective texture-based measure for classification. This can be explained by the fact that the mean of the derivative is very likely to take values close to zero.

- The modified version of the Harmonic Ratio, *MODHR*, shows a better overall performance in classification purposes than the original *MPEG-7* definition. It appears 8 times among the tables, while the *M7HR* does not appear at all. It performs particularly well in separating chamber music and orchestral subgenres, as can be seen on tables 6.12 and 6.13.

- The second proposed definition for rhythmic regularity (*RR2**), based on a linearity measure of the autocorrelation of the beat histogram (Sect. 5.4.3), is a better separator than the other proposed definition (*RR1**), based on the derivative of the autocorrelation. *RR2** has excellent separating

Short name	Full name
1	speech/music/background
2a	male/female/speech+background
2b	classical/non-classical
3a	chamber music/orchestral music
3b	rock/pop/jazz
4a	chamber subgenres
4b	orchestral subgenres
4c	hard rock/soft rock
4d	pop subgenres

Table 6.6: Split naming conventions.

performance in all splits except for the speech (2a) and the classical splits (3a, 4a and 4b). In all other splits, it belongs to the top-3 of the ranking. In contrast, *RR1** does not appear on any top-20.

- The *ZC/DS* feature is an excellent separator. It tops the list in 4 occasions, which are: separation between classical and non-classical music (2b) and classical subseparations (3a, 4a and 4b).

- Despite its simplicity, the *LOUD* approximation of loudness works fairly well as separator, most of all at the highest levels in the hierarchy (1, 2a, 2b). This is a motivation for the future usage of more sophisticated perceptual loudness definitions as a feature.

- It is a surprising result that beat strength features (*BHE** and *BHK**)

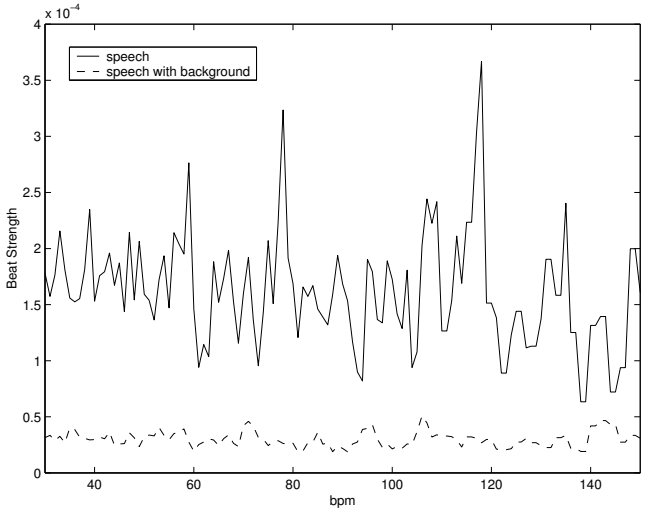


Figure 6.1: Beat Histograms of a speech and a speech with background example.

play a key role in separating male speech, female speech and speech with background, as can be seen on table 6.8. However, this phenomenon can be explained in that word and phrase onsets in speech signals produce strong "rhythmic" peaks on the histogram, while the noisy or musical background cause these amplitude jumps to be smaller. Figure 6.1 shows the Beat Histograms from a male speech sample (dark line) and a TV-commercial sample (dotted line). A measure of Rhythmic Regularity would yield very low values in both cases.

Ranking	Feature
1.	*MFCC2/S*
2.	*MFCC4/DS*
3.	*RR2**
4.	*LOUD/DS*
5.	*FLUX/S*
6.	*LOUD/S*
7.	*M7FLAT/M*
8.	*MFCC1/S*
9.	*SKEW/S*
10.	*MFCC4/S*
11.	*LOUD/M*
12.	*MFCC2/M*
13.	*M7CEN/S*
14.	*MODHR/S*
15.	*M7SPR/S*
16.	*MFCC1/M*
17.	*BHK**
18.	*M7FLAT/DS*
19.	*MFCC5/DS*
20.	*ZC/S*

Table 6.7: Split 1 best 20 features.

Ranking	Feature
1.	*BHE**
2.	*MFCC4/S*
3.	*LOUD/S*
4.	*FLUX/S*
5.	*M7FLAT/M*
6.	*BHK**
7.	*M7SPR/S*
8.	*LOUD/M*
9.	*MFCC2/S*
10.	*MFCC2/M*
11.	*MFCC1/M*
12.	*FLUX/DM*
13.	*MFCC5/DS*
14.	*MFCC3/M*
15.	*MFCC5/S*
16.	*MFCC4/DS*
17.	*FLUX/DS*
18.	*LOUD/DS*
19.	*M7CEN/S*
20.	*M7FLAT/S*

Table 6.8: Split 2a best 20 features.

Ranking	Feature
1.	*ZC/DS*
2.	*LOUD/M*
3.	*RR2**
4.	*SKEW/S*
5.	*MFCC2/M*
6.	*MFCC3/M*
7.	*M7FLAT/M*
8.	*BHE**
9.	*MFCC3/S*
10.	*MFCC1/S*
11.	*FLUX/S*
12.	*LOUD/DS*
13.	*MFCC2/S*
14.	*LOUD/S*
15.	*MFCC4/DS*
16.	*KURT/M*
17.	*MFCC1/M*
18.	*RMS/M*
19.	*RMS/S*
20.	*ZC/DM*

Table 6.9: Split 2b best 20 features.

Ranking	Feature
1.	*ZC/DS*
2.	*FLUX/DS*
3.	*ZC/M*
4.	*SKEW/S*
5.	*MFCC5/S*
6.	*ZC/S*
7.	*LOUD/S*
8.	*BHK**
9.	*MODHR/M*
10.	*MFCC3/DS*
11.	*FLUX/M*
12.	*MODHR/DS*
13.	*MFCC2/M*
14.	*M7SPR/M*
15.	*MODHR/S*
16.	*ZC/DM*
17.	*MFCC1/S*
18.	*MFCC1/M*
19.	*M7FLAT/S*
20.	*ENV/M*

Table 6.10: Split 3a best 20 features.

Ranking	Feature
1.	*RR2**
2.	*MFCC1/M*
3.	*MFCC3/M*
4.	*MFCC4/S*
5.	*LOUD/S*
6.	*FLUX/S*
7.	*ZC/M*
8.	*MFCC2/M*
9.	*ENV/M*
10.	*LOUD/M*
11.	*M7SPR/S*
12.	*MFCC3/S*
13.	*MFCC3/DS*
14.	*MODHR/M*
15.	*RMS/M*
16.	*BHE**
17.	*M7SPR/DS*
18.	*M7CEN/S*
19.	*ZC/S*
20.	*M7SPR/M*

Table 6.11: Split 3b best 20 features.

Ranking	Feature
1.	*ZC/DS*
2.	*FLUX/M*
3.	*M7FLAT/M*
4.	*MODHR/M*
5.	*SKEW/S*
6.	*BHE**
7.	*RR2**
8.	*MFCC2/M*
9.	*MFCC2/DS*
10.	*FLUX/DS*
11.	*M7SPR/M*
12.	*M7SPR/DS*
13.	*ZC/M*
14.	*ZC/S*
15.	*MFCC5/S*
16.	*MFCC2/S*
17.	*KURT/M*
18.	*FLUX/DM*
19.	*FLUX/S*
20.	*BHK**

Table 6.12: Split 4a best 20 features.

Ranking	Feature
1.	*ZC/DS*
2.	*FLUX/DS*
3.	*MFCC2/M*
4.	*MFCC5/S*
5.	*MODHR/M*
6.	*KURT/M*
7.	*MFCC3/M*
8.	*MFCC1/DS*
9.	*MFCC1/S*
10.	*LE**
11.	*MFCC1/M*
12.	*ZC/M*
13.	*MFCC2/S*
14.	*MFFC3/S*
15.	*ENV/M*
16.	*MODHR/DS*
17.	*MFCC5/DS*
18.	*FLUX/M*
19.	*LOUD/M*
20.	*M7CEN/DS*

Table 6.13: Split 4b best 20 features.

Ranking	Feature
1.	*MFCC1/M*
2.	*RR2**
3.	*MFCC2/M*
4.	*ZC/M*
5.	*ENV/M*
6.	*LOUD/M*
7.	*FLUX/M*
8.	*MFCC3/S*
9.	*M7SPR/DS*
10.	*RMS/M*
11.	*MFCC3/M*
12.	*M7SPR/M*
13.	*MFCC1/S*
14.	*LE**
15.	*SKEW/M*
16.	*BHK**
17.	*M7FLAT/S*
18.	*RMS/S*
19.	*MFCC2/DS*
20.	*KURT/M*

Table 6.14: Split 4c best 20 features.

Ranking	Feature
1.	*RR2**
2.	*MFCC1/M*
3.	*MFCC4/S*
4.	*MFCC3/M*
5.	*FLUX/DS*
6.	*FLUX/M*
7.	*LOUD/M*
8.	*MFCC1/DS*
9.	*ENV/M*
10.	*BHE**
11.	*MFCC1/S*
12.	*ZC/M*
13.	*MFCC3/S*
14.	*SKEW/S*
15.	*MFCC2/M*
16.	*ZC/S*
17.	*M7CEN/S*
18.	*BHK**
19.	*RMS/M*
20.	*KURT/M*

Table 6.15: Split 4d best 20 features.

Chapter 7

Classification

Once the audio has been reduced to a small set of numbers by the feature extraction process, it is now turn for the *audio-unaware* classification algorithms to examine them and decide a class label. From the large set of proposed classifiers, the *k-Nearest Neighbor (kNN)* and the *Gaussian Mixture Model (GMM)* classifiers have been extensively used and proven to be adequate in audio classification tasks [35, 21, 33, 9, 44] (see Chapter 3). In this work, the two are thoroughly evaluated and compared.

Nearest Neighbor classifiers are theoretically simple and have already been introduced in Sect. 2.4.4. In contrast, GMMs are considerably more complex. Although it has already been mentioned that they generalize simple gaussian models by modeling each class as a linear combination of gaussian densities, they will be considered in more detail in the first section of this Chapter.

7.1 Gaussian Mixture Models

In some cases, normal densities cannot accurately model a distribution of feature vectors in the feature space. If the nature of the data is of a certain complexity, the distribution can take a *multimodal* form, in which several clusters of samples are present, in contrast to a *unimodal* distribution with a single gaussian cluster. *Gaussian Mixture Models (GMMs)* account for this situation by modeling a distribution as a weighted sum of gaussian densities. In the context of pattern recognition, each class ω_k can be modeled as a mixture model, obtaining class likelihoods of the following form (see Sect. 2.4.2):

$$p(\mathbf{x}|\omega_k) = \sum_{m=1}^{M} w_{km}\, p_{km}(\mathbf{x}) \tag{7.1}$$

where w_{km} are the weights of each density and p_{km} is a normal density of the form of Eq. 2.30. The m index denotes the number of density within each class, the k is used as the class index like in the previous chapters. The individual

densities are called the *components* of the mixture. Using this notation, we can write $p(\mathbf{x})_{km} \sim N(\boldsymbol{\mu}_{km}, \boldsymbol{\Sigma}_{km})$ to emphasize that each component within each class has its own mean vectors and covariance matrices. It can clearly be seen that the simple gaussian (GS) model is a particular case of a GMM in which $M = 1$. To denote a M-component GMM, we will use the notation M-GMM[1].

Figure 7.1 illustrates a set of data samples that has been modeled as a 3-component GMM. Ellipses denote levels of equal probability density for each of the components. Their centers are given by the mean vector of each component and their form is determined by their covariance matrices (see Sect. 2.3.3).

As with the case of the normal density, training a GMM is done by ML parameter estimation, in which the parameter vector $\hat{\boldsymbol{\theta}}_k$ that maximizes Eq. 2.35 is sought (Sect. 2.4.3). But in this case, the parameter vector for each class has to contain not only M different mean vectors and M different covariance matrices, but also the mixture weights w_{km}:

$$\boldsymbol{\theta}_k = (w_{km}, \boldsymbol{\mu}_{km}, \boldsymbol{\Sigma}_{km}), \quad m = 1, \ldots, M.$$

This expanded parameter vector makes the ML estimation far more complex than in the simple gaussian case. Usually, it is accomplished by making use

[1]In benefit of generality, we will denote a GS model as 1-GMM, the same way as we denote a NN classifier as 1-NN.

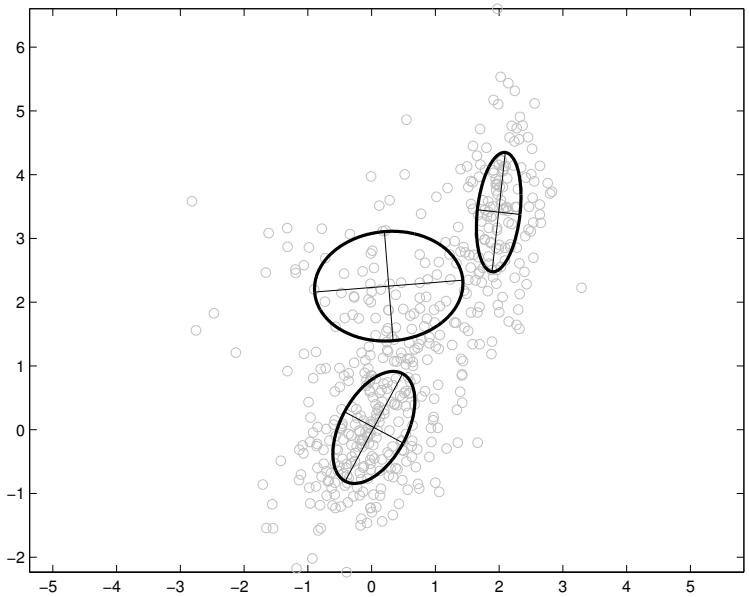


Figure 7.1: Illustration of a 3-centre Gaussian Mixture Model. Ellipses denote levels of equal probability density for each mixture component. Their principal axes are determined by their covariance matrices. This figure was obtained from the demonstrations provided with the Netlab toolbox for MATLAB [28].

of a so-called *expectation-maximization (EM)* algorithm [7, 50]. This consists of a set of iterations in which the parameters are updated in such a way that the likelihood $p(\mathcal{X}_k|\boldsymbol{\theta}_k)$ of Eq. 2.35 increases monotonically until it has reached a certain threshold value. A detailed explanation of the EM algorithm can be found in the cited literature, but is out of the scope of this work.

For pattern recognition purposes, it is common use to reduce the covariance matrices of the components to diagonal covariance matrices, ignoring the off-diagonal elements, in order to reduce computational costs [48, 33]. The off-diagonal elements are the covariances between each pair of features (see Sect. 2.3.2), and will have low values if the features are reasonably uncorrelated. This is likely to be the case here, because the feature selection method outlined in the preceding section also avoided correlation between successively selected features. In fact, we observed in our experiments that this simplification had virtually no influence in the final classification accuracy, and therefore, it was also adopted in this work.

7.2 Direct and Hierarchical Classification

For each of the classifier models tested here (*kNN* and *GMM*), there are furthermore two possible approaches to implement the classification process: a *direct* approach and a *hierarchical* approach.

Direct approach

In the direct classification scheme, only one decision step is taken to classify the input audio into one of the 17 classes of the taxonomy. This is the common approach in virtually all previous works, not only speech/music discriminators in which there are at most 3 classes, but also in music genre detectors with up to 8 or 10 classes (see Chapter 3). This approach is illustrated in Fig. 7.2. To improve readability, we will often make use of short names for the audio classes, as seen in the figure. The short names are listed in Table 7.1.

In this case, feature selection is performed considering all classes, as explained in Sect. 6.5.2.1.

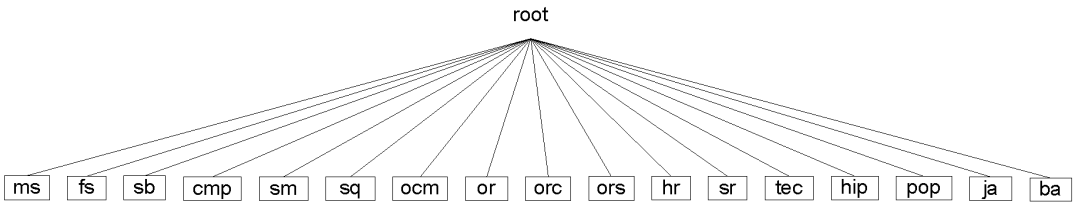


Figure 7.2: Direct classification approach. For the class names, see Table 7.1.

Short name	Full name
ba	Background
c	Classical music
ch	Chamber music
cmp	Chamber music with piano
ep	Electronic/pop
fs	Female speech
hip	Rap/Hip-Hop
hr	Hard Rock
ja	Jazz
m	Music
ms	Male speech
nc	Non-classical music
o	Orchestral music
ocm	Other chamber ensembles
or	Symphonic music
orc	Orchestra with choir
ors	Orchestra with soloist
pop	Pop
r	Rock
s	Speech
sb	Speech with background
sm	Solo music
sq	String quartet
sr	Soft Rock
tec	Techno/Dance

Table 7.1: Class naming conventions.

Hierarchical approach

However, in this work a special effort has been made in exploring a hierarchical alternative, which consists on a tree-based succession of class decisions corresponding with the class taxonomy depicted in Fig. 4.1. In this way, level one of the taxonomy corresponds to a three-class classification problem. Following the decision of this first problem, we then switch to one of the three or two-class problems at the second level (splits 2a or 2b), and so on[2].

Figure 7.3 shows the taxonomy again, this time in a more schematic manner. Roman numbers indicate the different levels of classification. The combination of numbers and letters found beside each split on the tree indicate the naming convention for the splits, which was already presented in Table 6.6.

[2]This hierarchical approach can be regarded as a *classification tree*, since it divides a single-stage problem into a set of reduced-class stages. However, this term usually refers in pattern recognition literature to a classifier in which the decision at each split is based on the comparison *of a single feature* with a given threshold value, different single features being evaluated at different stages [4]. For this reason, the term *classification tree* will be avoided here.

Not only classification, but also feature selection has been made hierarchically. That is, features were selected depending on the current split they must separate, by using only the training samples belonging in each case to its child classes (Sect. 6.5.2.2).

The special focus of this work on this hierarchical method was motivated by the advantages it offers in comparison with the direct approach [3]:

- A hierarchical approach allows to account for the *class dependency of the features*. Clearly, some features will be more suitable than others when classifying into a given set of subgenres. For example, features describing rhythmic regularity are more likely to perform better in separating classical from pop music than in classifying into chamber music subgenres. This is exactly the motivation for the genre-dependent feature selection addressed in the previous Chapter.

- It allows the errors to be more *graceful* than in the case of a direct classification. For example, if a symphonic music sample is wrongly classified as orchestral music with soloist, it is not so bad as if it were classified as Hip-Hop. Dividing the decision in subdecisions makes the errors concentrate within the given subgenre.

- It closely reflects the underlying audio taxonomy, thus allowing to evaluate the separability of broad common-used genres, such as pop, rock and jazz, and their suitability for automatic classification purposes.

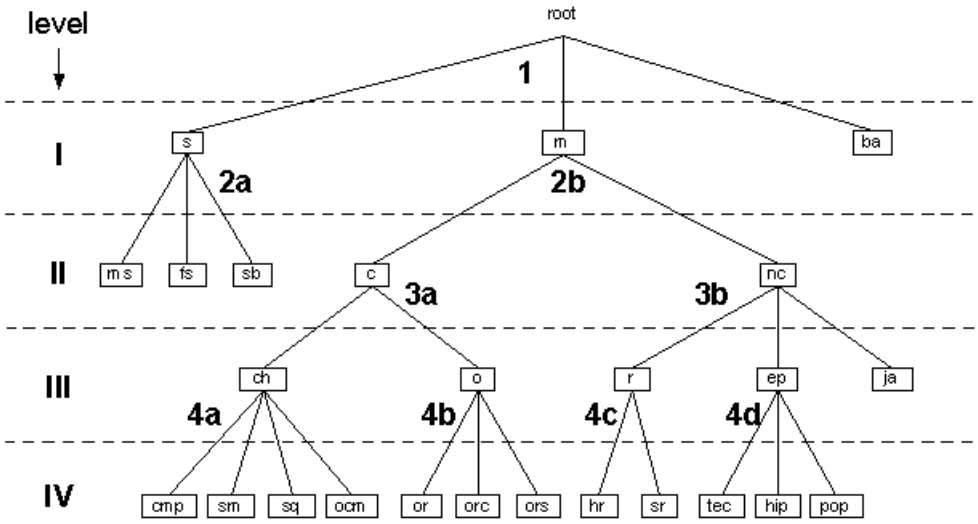


Figure 7.3: Hierarchical classification approach. Roman number indicate classification levels. Numbers with letters denote the splits. For the class names, see Table 7.1.

- It provides the framework for the future design of more sophisticated genre-dependent features. Possibilities include, for example, a measure of overall distortion for detecting Hard Rock, singer detection to distinguish opera, a measurement of reverberation to separate chamber music from symphonic music or more sophisticated loudness models.

- It makes future expansions of the taxonomy easier. If a new class were added to the direct scheme, the feature selection algorithm would have to be run again with all training samples. Furthermore, each class would have to be re-trained according to the new result of the feature selection. In contrast, in a hierarchical approach, only the genre branch to which a new class is added should be modified with respect to feature selection and training, the rest of the models remaining unchanged. In this way, an end-user would be able to create its own subsplits on the taxonomy without needing the whole training database for the rest of genres.

However, it has the following three major drawbacks:

- The classification rates are multiplicative across levels. For example, a signal correctly classified as string quartet must have been correctly classified as chamber music at the previous level, as classical music at the previous level and as music at the highest level. As it has been found by the experiments explained in the present Chapter, this leads to a slightly poorer performance than the direct case. To compensate the higher possibility of error, the genre dependent features must be very well designed to fit their particular classes.

- It presents much more complexity in the implementation.

- It is more computationally expensive, not only because more classification decisions must be met for a given input signal, but also because it is likely that a higher number of features than in the direct case must be computed.

As a result of the above considerations, both direct and hierarchical variants of the kNN and GMM classification were implemented in MATLAB and compared in the following experiments. For the kNN and GMM algorithms, the publicly available Netlab toolbox was used [28].

7.3 Feature Preprocessing

As can be derived from the feature curves across Chapter 5, each feature has a very different value range. Classification must not be influenced by the numerical ranges of the features, but by their statistical behavior. To ensure this, all test and training feature vectors must be normalized in such a way that their values

lie within comparable ranges. This can be accomplished by normalizing each element of a feature vector according to:

$$x_{i,norm} = \frac{x_i - \mu_i}{\sigma_i} \tag{7.2}$$

where μ_i and σ_i are the mean and standard deviation of the feature x_i *across all N available samples of all classes*:

$$\mu_i = \frac{1}{N}\sum_{j=1}^{N} x_{ij} \tag{7.3}$$

$$\sigma_i = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(x_{ij} - \mu_i)^2} \tag{7.4}$$

Each new unknown input feature vector is normalized using the previously stored μ_i and σ_i parameters.

The normalization is especially important in the case of the kNN classifier, since its class decision is based on the computation of a distance metric (see Sect. 2.4.4), and is thus extremely dependent on the scaling of the feature space axes.

7.4 Design of the Classifier

Goal of this work is not only to examine several methods to perform classification, but also to select one of them to be implemented in a final prototype application programmed in the *C/C++* language. This section details the experiments to which the different algorithms were subjected, and provides the line of argument in choosing the final model.

The parameters that have to be chosen to obtain a final working system are the following:

- Classification model: kNN or GMM
- Classification approach: direct or hierarchical
- Number of training samples
- Classifier parameter: number of neighbors k for the kNN or number of centres M for the GMM
- Number of features

The evaluations performed in this Chapter were done using the *holdout* method [16]: 10% of the audio database samples were used for testing and the other 90% for training, resulting in a test set of 85 samples and a training set of 765 samples. Particularly, we use a *stratified holdout* method, which means that each class is represented by the same number of samples in the test and training sets. As a result, 5 samples from each class were randomly selected to form the test set.

The main drawback of the holdout method is that the performance measures depend on the particular test set chosen. To obtain a better estimate of the final performance of a classification system, a *cross-validation* method is needed for the evaluation. Cross-validation will be used in Chapter 9 to measure the performance of the final application. However, in the present Chapter, the focus is to compare a wide range of classifier and parameter combinations and to observe tendencies in the behavior of their performances rather than to obtain an accurate prediction. For this reason, the much less computationally costly holdout method was used here.

It should also be noted that all the classification rates given in this Chapter correspond to the *overall* classification rate of the whole system, considering the 17 classes. Also, the file-based approach (one feature vector per database sample) was used. When testing the final application, more detailed indications about performance in particular levels and sublevels, as well as with both texture-based and file-based approaches, will be given.

The curves on the next two pages present the results of the experiments. They give the classification performance as a function of the number of features selected per classification stage for the two types of classifiers and for k and M parameters ranging from 1 to 6. The particular case $k = 1$ corresponds to a nearest neighbor classifier, and the case $M = 1$ corresponds to a simple gaussian classifier.

Of course, the features are selected in the order displayed in the lists of Sect. 6.5.2. It can be seen from the figures that increasing the number of features does not increase the performance substantially for high dimensions. In most cases, the curves tend to stop increasing significantly at a certain number of features, as a result of the *curse of dimensionality*.

Choice of the Model

Table 7.2 shows the performance of the different classification approaches averaged across the number of features for each different k or M parameter. The last column in turn averages the values across the k/M parameters.

It can be seen that the different approaches have very similar averaged performances, ranging from 59.5% to 63.3%. This fact makes averaged performance a weak argument in favouring one model over the other. Therefore, we rather based our decision in regarding the behavior of the performance curves, as follows:

Curves for the direct kNN approach tend to reach the highest performance

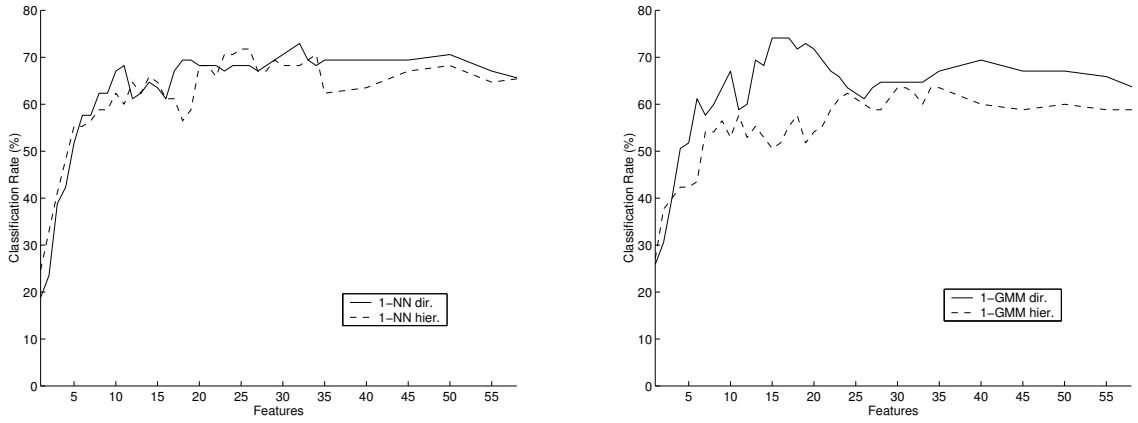


Figure 7.4: Performance of the 1-NN and 1-GMM classifiers.

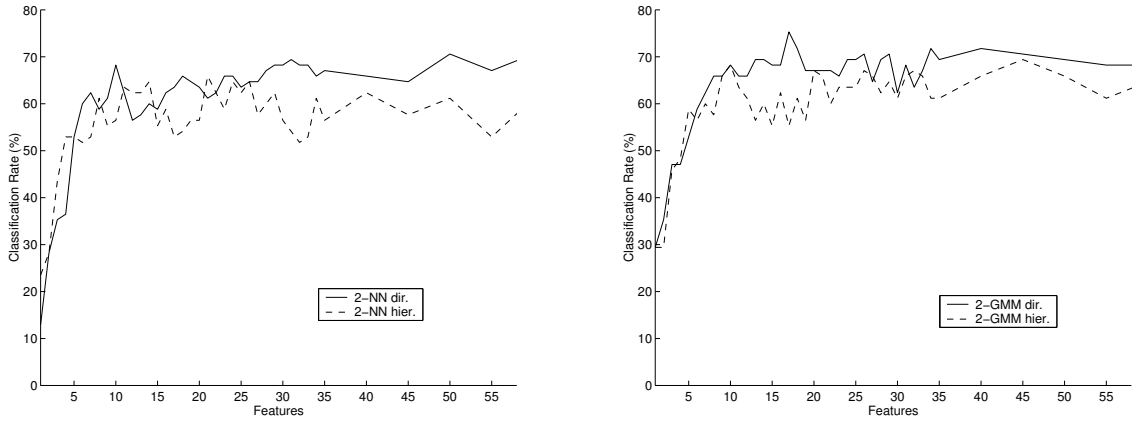


Figure 7.5: Performance of the 2-NN and 2-GMM classifiers.

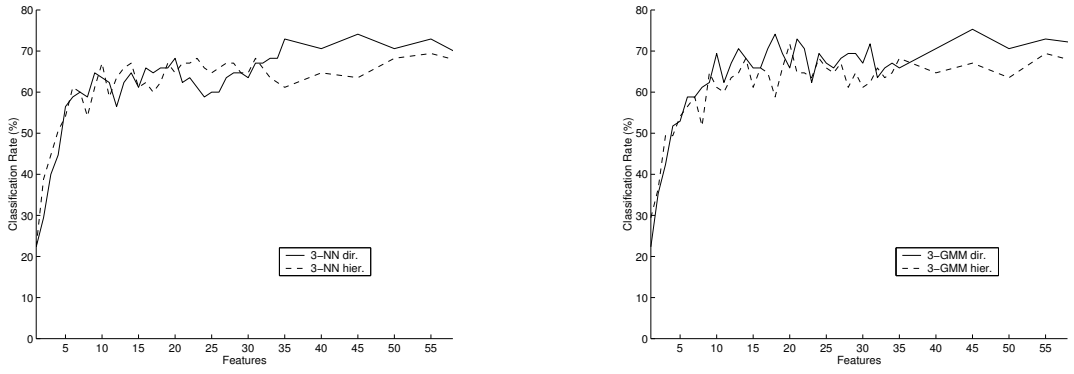


Figure 7.6: Performance of the 3-NN and 3-GMM classifiers.

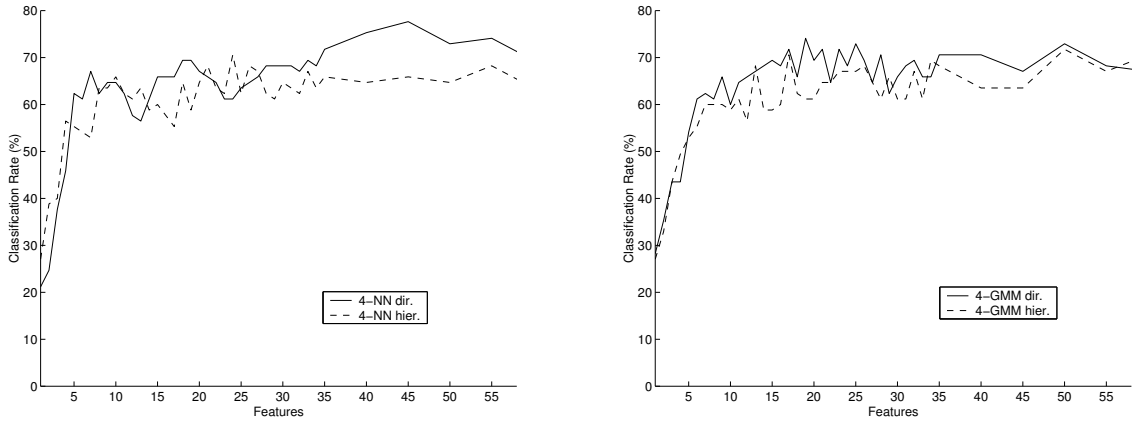


Figure 7.7: Performance of the 4-NN and 4-GMM classifiers.

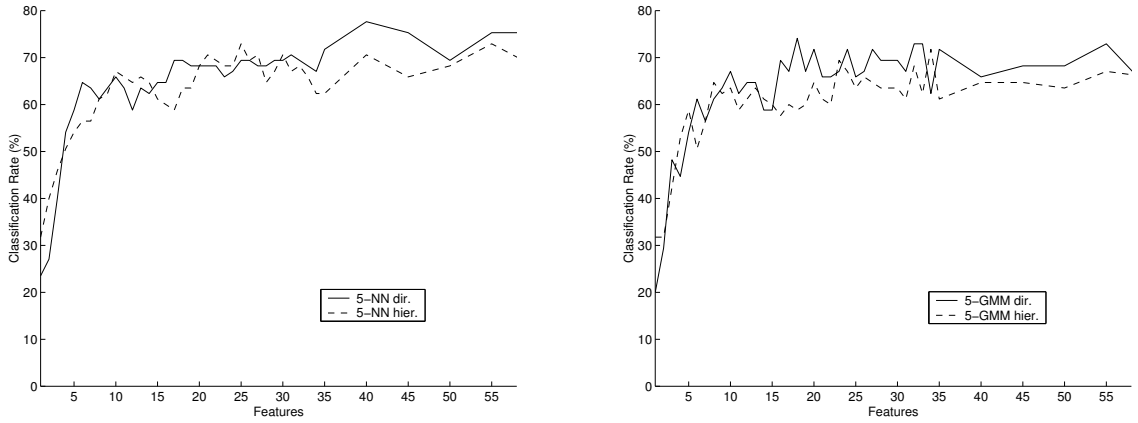


Figure 7.8: Performance of the 5-NN and 5-GMM classifiers.

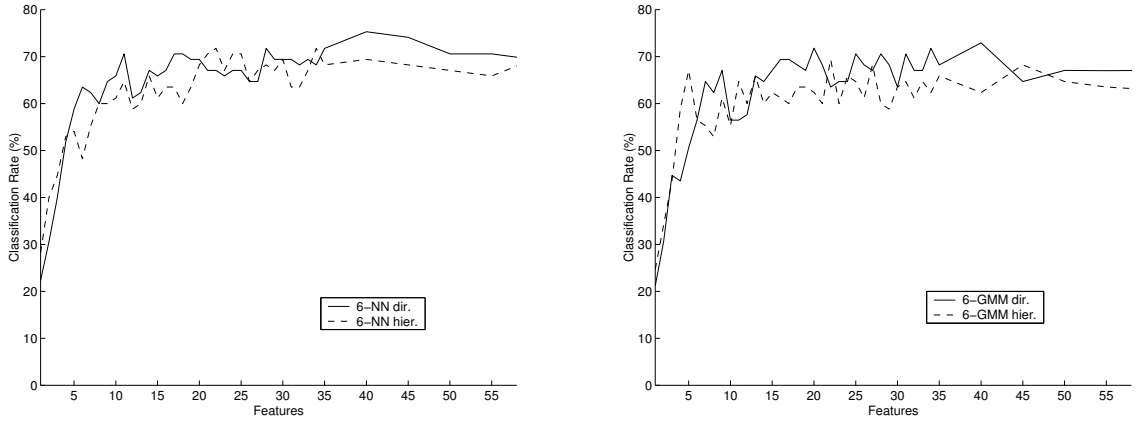


Figure 7.9: Performance of the 6-NN and 6-GMM classifiers.

k/M	1	2	3	4	5	6	Average
kNN dir.	62.8	60.3	61.3	62.8	64.2	64.3	62.6
kNN hier.	61.8	56.2	61.4	60.5	63	62.2	60.9
GMM dir.	62.7	64.6	64.1	64.1	63.3	61	63.3
GMM hier.	55	60	61.3	60.6	60.2	60	59.5

Table 7.2: Performance of the direct and hierarchical approaches averaged across the number of features.

at around 35 to 50 features, and for the hierarchical approach around 20 to 35 features. For the GMM, curves tend to peak at a lower number of features, around 10 to 25, both in the direct and hierarchical cases. These peak performances lie in both kNN and GMM cases on the same range (65% to 75%). This means that with the GMM model, the highest performance is reached with less features than with the kNN model, allowing to reduce computational costs.

Another major drawback of the kNN classifier is that it tends to fit the decision boundaries too closely on the training data (see for example Fig. 2.6), resulting in a lack of generality due to overfitting, as mentioned in Sect. 2.4.5. GMM models each class statistically, and is therefore more likely to reflect the true nature of data.

Yet another disadvantage of kNN, as mentioned in Sect. 2.4.4, is that it requires the storage of the feature vectors of the whole training database in order to search for the nearest neighbors, as well as the computation of the distance between the unknown vector and each of the N stored vectors. In contrast, the GMM solution needs to store only one set of model parameters (means and covariances) for each of the classes. When classified, an input vector only needs to be compared with each of these C models.

Consequently, the GMM algorithm allows similar performance as the kNN while highly reducing computational costs and ensuring generality, and for these reasons, it was chosen as the model for the final implementation.

It can also be seen on the table that the hierarchical approach leads to slightly lower performance in both kNN and GMM cases. As mentioned in Sect. 7.2, this is explained by the higher probability of error that results from adopting a tree-like stepwise classification. Considering that the difference in performance is small, it has been nevertheless opted for the hierarchical approach, since it features the additional advantages of allowing genre-dependency of the features, flexibility and upgradability, as outlined in the mentioned section.

Choice of the Number of Training Samples

Some authors in pattern recognition literature have noted that it can be disadvantageous to train a classifier with substantially different number of samples per class [19]. However, it is well known that in some situations, data complexity

is also substantially different among classes, requiring that more complex data should be represented in the database by more training samples than less complex data. For example, very few training samples would be needed if we would like to train a *silence* class, in contrast to the number of samples that would be required to train a broad *music* class.

In this work, the audio database consists of 50 samples for each of the 17 classes. In the direct approach, the GMM is trained with an equal number of samples per class (45 training samples per class). However, in the hierarchical case, this will strongly depend on the classification level. For example, at the highest level, it is differentiated between speech, music and background. The speech class contains 3 subclasses, the music class 13 leaf-classes and the background class is itself an end class. This results in an unbalanced set of 135, 585 and 50 samples for each of the classes, respectively. The same applies to other splits on the tree.

To proof if unequal training sets can really make performance worse in our particular audio situation, we reevaluated the hierarchical classification methods described above, but this time ensuring that at each stage, all classes were trained using the same number of samples. Considering the same example, the background leaf-class on stage 1 constrains to select 45 training samples out of all the samples belonging to the speech and music classes. Table 7.3 shows the results.

k/n	1	2	3	4	5	6	Average
kNN hier.	57.6	53.7	57.9	57.7	59.6	59.4	57.7
GMM hier.	54	57.4	56.8	56.8	56.3	54.4	56

Table 7.3: Performance of the direct and hierarchical approaches averaged across the number of features using equal number of training samples per class.

When compared with table 7.2, this results show clearly that in this case, it is better to keep as much training samples as possible, rather than to select balanced training sets.

Choice of the Model Parameter

As can be seen on table 7.2 the GMM hierarchical approach yields very similar performances for values of M from 2 to 6, peaking at $M = 3$ (61.3%) and staying around 60% in the other cases. The simple gaussian case ($M = 1$) does perform somewhat poorer (55%). Of all the considered classifier/parameter combinations, the GS case is the one for which the direct and the hierarchical approach differ most in performance, as can be also seen on Fig. 7.4. From the rest of parameters, a value of $M = 3$ was chosen, which shows a slightly better performance.

Choice of the Number of Features

Finally, the last parameter that remains to be chosen is the actual number of features to be selected at each of the hierarchy steps. It has been stated in repeated occasions that the optimal number of features is closely related to the number of training samples per class. Unfortunately, there exist no specific design equations that relate number of samples and number of dimensions to classification error for the particular case of a GMM classifier, as for most of all other possible algorithms.

Consequently, this value has to be obtained empirically. It has been mentioned that the performance curves of the GMMs tend to reach the maximum at around 10 to 25 features before staying at comparable levels for higher number of features. In the particular case of the 3-GMM (Fig. 7.6), these peaking occurs at a number of 20 features. With this number of features, an optimal relationship between classification accuracy and computational performance is achieved. Hence, the 20 best features at each stage will be selected and computed (see tables 6.7 to 6.15).

It is worth emphasizing that, in this particular system, 20 is the optimal number of features *given a number of 50 training samples per class*. Extending the audio database would result in a higher optimal number of features, and vice versa. Although increasing the optimal number of features is a possible strategy to improve performance, it is rather a costly one, since it would require to augment the number of training samples exponentially, as a result of the *curse of dimensionality*.

Another important consideration is that the choice of the number of features was done observing the overall performance. Observing the individual stage performances is likely to result in a different optimal number of features for each split in the tree, since at each stage, the number of training samples per class is different (and equal to 50 times the number of leaf classes each father class comprises). This is a possible issue to be investigated in the future, since it can probably benefit performance.

The final specifications of the classifier to be implemented in the final application are outlined as follows:

- Classification model: 3-GMM
- Classification approach: Hierarchical
- Number of features: 20

Chapter 8

Implementation

All the implementations and computations that have been detailed so far in the present work have been programmed in MATLAB, which because it is an interpreted language is useful for simulation and testing, but far too slow for a practical application. Besides the exploration of possible solutions to the audio classification problem, constructing a prototype application in a compiled language which could demonstrate the selected algorithm and which could constitute a base for future improvements was a goal of this work. In this case, the *C/C++* language was chosen for the implementation.

The result is the `classify.exe` command-line application that runs on Windows operating systems. In contrast with the MATLAB implementation, in which the feature extraction process of a single 30-second audio file lasted several minutes, faster than real-time feature extraction has been achieved with the application.

The `classify.exe` application is included on the enclosed CD-ROM, together with a set of WAV test examples (see Appendix C). Of course, the provided WAV files were not a part of the training database used to train the specific class models included in this version.

8.1 Program Functionality

The program takes an audio file in WAV format as input and displays a class label on the screen after the processing. As argued in the preceding Chapter, classification is performed using a hierarchical 3-GMM classifier that follows the taxonomy of Fig. 4.1. At each level, 20 features are extracted from the audio, according to the lists given in Sect. 6.5.2.2. The classification is file-based, and assumes that only one type of audio is present within the file.

The application is intended to demonstrate classification, and does not offer the possibility of training new classes or re-training old ones. Each class was trained using the MATLAB implementation used earlier in the work. The result-

ing parameter vectors $\boldsymbol{\theta}_k$ are passed to the C program as binary data files. Of course, the implementation of the training process is one of the possible future upgrades of the initial version.

Besides these basic functionalities, the application also has the following characteristics that were not present in the MATLAB implementation:

- Possibility to work in *single vector* mode or in *texture window* mode (see Sect. 5.1.2). Both modes have been evaluated and compared in the next Chapter.
- Possibility to stop classification when a certain level in the tree is reached. This allows the flexible usage of the tool as just a music/speech/background discriminator, as a classical/non-classical separator, as a rock/pop/jazz separator, and so on, if it is not desired to reach the most detailed subgenres at the bottom of the tree.
- *Efficient feature extraction:* only the features that have not been extracted at the previous levels are computed, the rest are loaded from a set of feature buffers. If the feature is frame-based, its whole time-trajectory across the given texture window is saved, so that all of its four *M*, *S*, *DM* or *DS* subfeatures can be extracted at lower levels without having to recompute the feature.
- *Flexible buffer processing:* As has been seen in Chapter 5, different STFT parameters (FFT length N, window length L, step length R) are used to compute the different features. The processing function of the program takes these parameters as input arguments and adapts its buffering operations according to them. This allows the easy addition of features with any combination of these parameters (with or without zero padding, with different grades of overlapping, etc.).
- *Detailed probability output:* Rather than only a single final genre decision, the different decision probabilities with which the analyzed signal has been classified into the candidate genres are displayed at the output. In this way, we gain insight into the reliability of the given classification decision.
- *Tree implementation optimized for future upgrades:* Although the initial demo application does not allow to create or train new classes, the audio taxonomy has been implemented as a *dynamic tree*, which allows easy future expansions.

8.2 Program Structure

The program is structured following the canonical *main program/library* architecture. The feature extraction and the classification routines are encapsu-

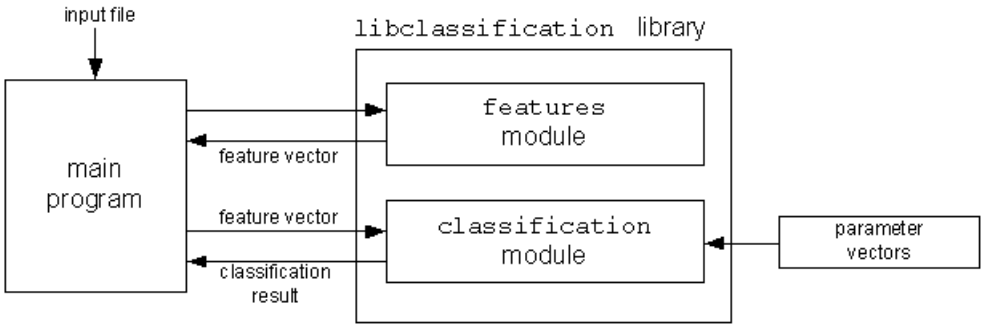


Figure 8.1: Program structure.

lated in the corresponding `features` and `classification` modules within the `libclassification` library, as shown in Fig. 8.1.

The main program loads the current feature list and model parameters from the taxonomy tree, and then runs a processing loop during which, at each iteration, a new frame is read from the audio file and the feature extraction module is called to compute the desired features for that frame, storing each value in the trajectory buffers. All the features are implemented within the `features` module, except for the computation of the beat histogram, for which the `BeatTracker` library by zplane.development is called.

When the end of the file or of the texture window is reached, the resulting feature vector is passed as input argument to the `classification` module. According to its class prediction, the new feature lists and model parameters are loaded and the whole process restarts.

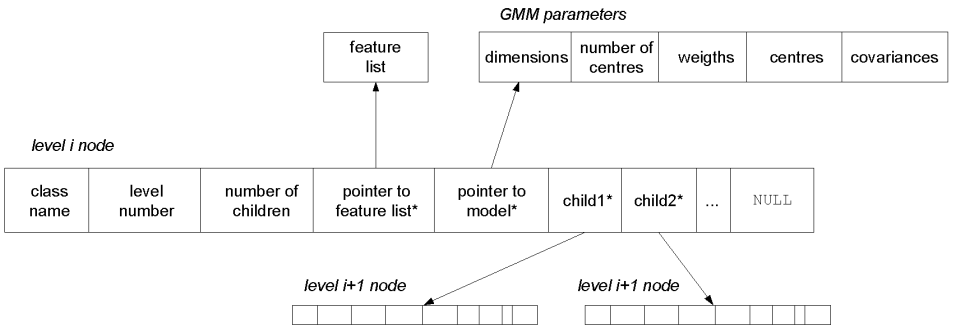


Figure 8.2: Implementation of a tree node. Asterisks denote pointers.

Implementation of the taxonomy tree

As has been mentioned in the preceding section, the class taxonomy has been implemented as a *dynamic tree*. These are tree-like data structures in which each node contains pointers to each of his child nodes. In our case, each node corresponds to an audio class and, apart from the children pointers, contains all the information regarding its corresponding list of features and its GMM model. The structure of each node is detailed in Fig. 8.2.

The classification process is done recursively: first, the root node is taken as the starting node and its list of features and the GMM models of its children are loaded. When a classification decision is met, the corresponding child is taken as the starting node and the whole process is repeated until the user-defined maximum level or a leaf node has been reached (leaf nodes are those whose lists of children pointer contain only `NULL`s).

8.3 User Interface

The program is started from the shell prompt by typing

```
classify audiofile [level] [step]
```

The optional `level` argument indicates the maximum classification level desired. Its range is 1 to 4, where 1 corresponds to a speech/music/background discriminator and 4 to the full 17-class audio taxonomy. The default value is 3.

The optional `step` argument indicates the desired classification step (that is, the desired texture-window length) in seconds. The default value is 30 seconds. A value of 0 means single-vector classification. The minimum is set to 0.2, since lower values would produce conflicts with the analysis windows. If the analyzed file is shorter than the number of seconds typed, single-vector classification is performed.

When invoked, the program displays the classification tree up to the desired level. If the single-vector mode is selected, the whole file is processed, and afterwards, the classification result is displayed. The different classification probabilities (which can be interpreted as the *certainty* of the classification) are displayed beside the corresponding leaf nodes of the tree. These are the *a posteriori* probabilities from the classes, computed from their GMM likelihoods $p(\mathbf{x}|\omega_k)$ (Eq. 7.1) using Bayes' law (Eq. 2.32). Since we assume that all classes are equally probable, we can simplify the factor $P(\omega_k)$ from Bayes' law, reducing the equation to:

$$P(\omega_k|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_k)}{\sum_{k=1}^{C} p(\mathbf{x}|\omega_k)} \tag{8.1}$$

where, in this case, C is the number of candidate classes *within the current split*. Each probability is multiplied with that of its father node.

The probabilities are tabulated to match the tree structure and to improve readability. All the probabilities within each of the tabulated levels must add up to the probability of their father node. This is shown in Fig. 8.3 for the particular case of a level 4 classification.

If texture-window mode is selected, classification is updated at the configured intervals. Each time a decision is made, a green square is added to the horizontal bar beside the corresponding tree branch, and the probability indications are updated. The bars at the right side of the tree form the *classification histogram* of the current signal. Figure 8.4 shows the program in texture-window mode, for a level 2 classification.

8.4 Operation Modes

The MATLAB implementation described in the previous chapters based the classification of each test audio sample (around 30s long) on a single feature vector describing the whole file. That is, means, standard deviations, and the other subfeatures where computed over a texture window corresponding to the length

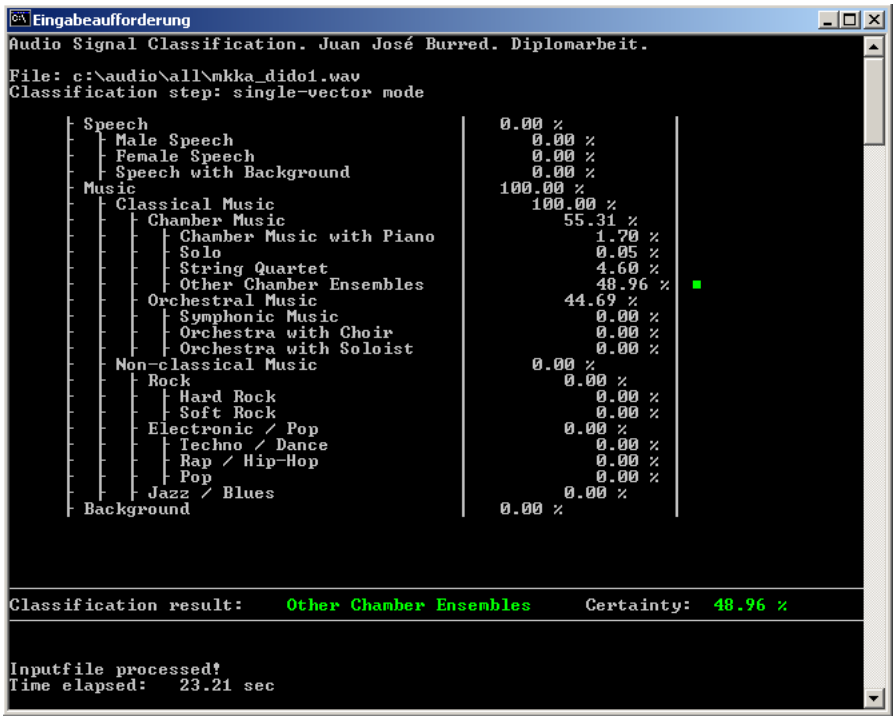


Figure 8.3: Screenshot of the `classify` application in single-vector mode and level 4 classification.

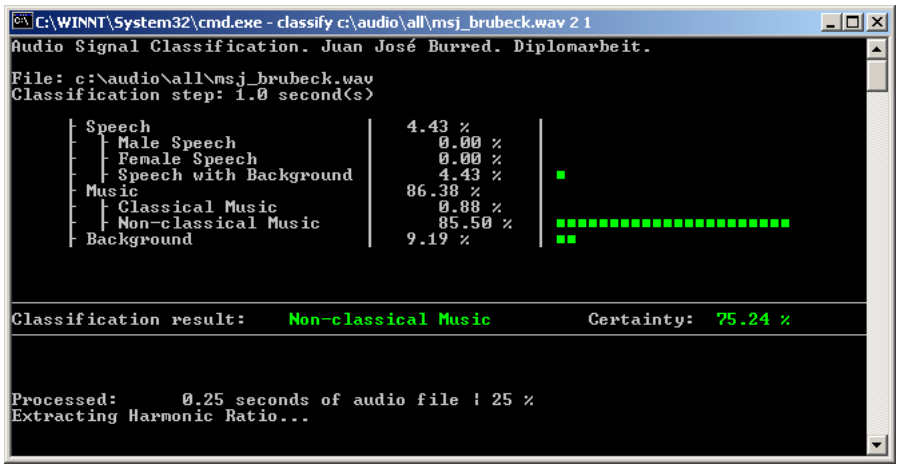


Figure 8.4: Screenshot of the `classify` application in texture-window mode and level 2 classification.

of the file. In contrast, as mentioned above, the `classify` application outlined here is able to operate using either of the approaches for file-based classification described in Sect. 5.1.2.

Single vector mode

In this case, only one class decision is made for each file. The selected class is the one with the highest certainty within the current set of leaf subclasses. Only a single square appears at the corresponding leaf, and the certainty is displayed beside the final class name (Fig. 8.3).

Texture window mode

A simple approach in the texture-window case would be to count the number of times each class has been detected, and declaring the one with the highest count (that is, the one with the longer bar in the histogram) as the final class. However, this can lead to poor results because of the following two reasons:

- The classification probabilities are not taken into account.
- If two classes happen to have the same histogram count, which can often be the case with large texture windows, the class decision would be indeterminate.

Therefore, both the decision counts *and* the probabilities have to be regarded in order to make the final decision. This in turn requires to average the probabilities in time, which cannot be done in the straightforward way of adding the

successive probabilities of each class and dividing by the total number of classification steps taken so far. This would only work if all leaf classes belonged to the same level, and if all final class probabilities would be computed at each classification step. The complex tree structure handled here does not fulfill these conditions and forces, first, to average each probability only with the number of times it was computed, rather than with the total number of steps, and then, to multiply with the father certainties.

When the end of the file has been reached, the averaged probabilities $\langle P(\omega_k|\mathbf{x})\rangle$ are weighted with the histogram counts:

$$\langle P(\omega_k|\mathbf{x})\rangle_w = \frac{W_k}{W}\langle P(\omega_k|\mathbf{x})\rangle \tag{8.2}$$

where W_k is the number of times class ω_k was selected, and W is the total number of texture-windows. The final class is the one that has the largest $\langle P(\omega_k|\mathbf{x})\rangle_w$ value. This value is displayed beside the name of the final class.

Chapter 9

Evaluation

In chapter 7, many possible combinations of classifiers and their parameters were tested for their all-class overall performance using the *holdout method*. In the present chapter, the implemented application will be subjected to a more detailed performance evaluation in which not only the all-class performance, but the performances at each level will be analyzed. The hierarchical approach allows to give genre-dependent performance values, thus gaining insight into how easy or difficult it is to separate a given set of subgenres.

In the last section of the Chapter, the computational performance of the program will also be analyzed.

9.1 Cross-validation

The *holdout* method used in Chapter 7 is highly dependent on the used test set. For example, if the test set happens to be an especially "lucky" one, the results will be positively biased and it will give a too optimistic performance estimation.

One approach to avoid this and to obtain a realistic estimation of performance is to repeat the evaluation K times, using each time a different test set. In the *K-fold cross-validation* method, the evaluation is iterated K times, each time using a randomly selected test set consisting of $K\%$ of the samples of the whole database. Then, the performances across the iterations are averaged to obtain the final estimation of the classification rate R:

$$R = \frac{1}{K} \sum_{i=1}^{K} R_i \tag{9.1}$$

The test samples that are already part of a test set are ignored for the rest of the random selection. In this way, it is assured that all the examples in the data set are eventually used for both training and testing. If we additionally impose

that each class must be represented by the same number of samples in the test set, the cross-validation is said to be *stratified*.

A value of $K = 10$ is a common choice in the evaluation of pattern recognizers, and has been also adopted in this work. Therefore, the algorithm used here will be a *stratified 10-fold cross-validation*.

9.2 Classification Performance in Single-Vector Mode

When evaluating a classification tree such as the one used here, some subtle issues must be taken into consideration. For example, if it is said that jazz samples were classified with an accuracy of 70%, this can actually have two meanings:

- 70% of the jazz samples that have been correctly classified as non-classical music at the previous level have been correctly recognized as jazz.
- 70% of all the jazz samples contained in the test set have been correctly recognized as jazz.

According to these two interpretations, for each split on the tree two different performance indications will be given:

- *Accumulative performance:* Percentage of samples of the test set correctly classified.
- *Independent performance:* Percentage of samples correctly classified at level i that have been correctly classified at level $i + 1$.

The *accumulative performance* is a more demanding measure, since it considers the samples that have been incorrectly classified elsewhere in the tree, accounting for the multiplicative nature of the performance of tree-based classifiers. On the other hand, the *independent performance* is useful if we are wishing to estimate how good the system is in separating two given subclasses. For the particular case of split 1 (speech/music/background), accumulative and independent performances will be identical. The all-class performance is implicitly accumulative.

Table 9.1 shows the results of the 10-fold cross-validation for each of the splits in the tree (the percentages are given as mean values plus their standard deviations across the iterations) when using the single-vector approach (one feature vector per training or test sample). In order to improve readability, only the final results are shown here. The complete results of each of the iterations of the cross-validation experiment are given in detail in Appendix A.

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	94.59 ± 1.77	94.59 ± 1.77
2a	male/female/speech+background	76.67 ± 8.46	82.31 ± 8.63
2b	classical/non-classical	91.08 ± 3.68	96.08 ± 2.02
3a	chamber music/orchestral music	74.29 ± 7.25	81.52 ± 7.88
3b	rock/pop/jazz	63.67 ± 6.17	70.33 ± 8.65
4a	chamber subgenres	42.50 ± 12.08	54.67 ± 13.92
4b	orchestral subgenres	52.67 ± 10.63	75.21 ± 11.83
4c	hard rock/soft rock	55.00 ± 16.50	79.52 ± 20.18
4d	pop subgenres	62.00 ± 9.96	76.15 ± 9.55

Table 9.1: Single vector split classification performance using the hierarchical approach.

Level	Accumulative performance	Independent Performance
I	94.59 ± 1.77	94.59 ± 1.77
II	83.88 ± 6.07	89.20 ± 5.33
III	68.98 ± 6.71	75.92 ± 8.27
IV	53.04 ± 12.29	71.39 ± 13.87

Table 9.2: Single vector level classification performance using the hierarchical approach.

Table 9.2 averages the obtained values across each level, to obtain a measure of the level-related accuracy.

Finally, the all-class performance value is given in table 9.3 for each of the evaluation iterations, and in average.

Fold	All-class performance
1	56.47
2	61.18
3	63.53
4	56.47
5	61.18
6	56.47
7	58.82
8	61.18
9	55.29
10	56.47
Average	58.71 ± 2.85

Table 9.3: All-class performance.

A more detailed information about the classification performance is given by the so-called *confusion matrix* (Fig. 9.1). Its rows correspond to the actual classes and its columns to the predicted classes. For example, an entry of 8 in element (1,2) of the matrix denotes that 8% of the male speech samples were wrongly classified as female speech. Correct classifications correspond to the diagonal. The square margins and the different grades of shading denote the splits on the hierarchy. The class short names were introduced in Table 7.1.

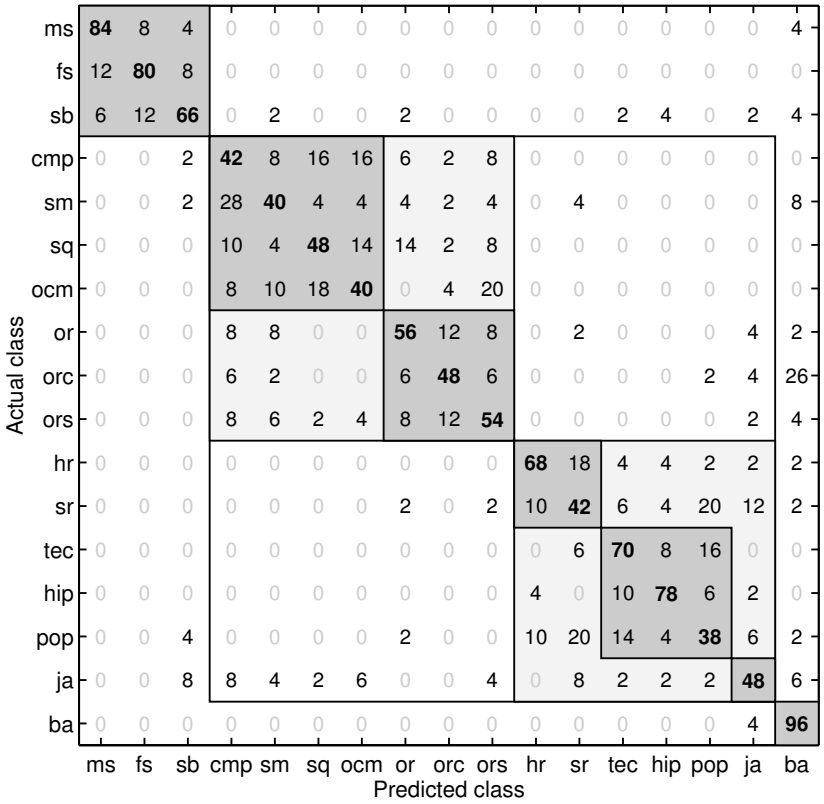


Figure 9.1: Confusion matrix. An entry of R in element (i,j) of the matrix means that R% of the test samples of class i were classified as belonging to class j. The square margins and the different grades of shading denote the successive decision splits.

The following conclusions can be drawn from the observation of the confusion matrix:

- It is clear to see that the classification errors tend to appear within the same class groups or subgroups, which indicates the ability of the hierarchical approach to produce *graceful errors* (see Sect. 7.2).
- Many music samples were wrongly classified as background. Perhaps the most surprising result of the confusion matrix is the high number of choral

music samples (26%) that have been wrongly clasified as background noise. On the other hand, only 4% of the background samples were classified as music.

- All the non-classical music samples misclassified as chamber music are jazz examples. This is musicologically consistent, since, from a purely instrumental point of view, it can be said that jazz is chamber-music-like.
- The matrix also confirms the difficulty of separating the rock and pop classes. Particularly, the most difficult genres in the hierarchy are Soft Rock (42% of correct classifications) and Pop (38% correct). It can also be seen that many rock examples have been classified as pop, and vice versa.
- The musical genres with the best classification rates are Rap / Hip-Hop (78%) and Techno (70%). This indicates that they are very separable classes, in which their belonging examples have similar characteristics within a class (low within scatter) but differentiated characteristics when compared with the rest of classes (high between-scatter).

9.2.1 Comparison with the Direct Approach

To obtain a more reliable comparison between the hierarchical and the direct approach, we repeated the above 10-fold cross-correlation test with the direct 3-GMM approach implemented in MATLAB. Tables 9.4 and 9.5 show the results. It can be seen that, although the all-class performance is slightly better than with the hierarchical approach (59.76% against 58.71%), as predicted in Chapter 7, there are some splits at which the classification is worse in the direct approach.

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	96.35 ± 1.70	96.35 ± 1.70
2a	male/female/speech+background	76.67 ± 9.03	81.00 ± 9.19
2b	classical/non-classical	94.31 ± 3.48	96.67 ± 2.45
3a	chamber music/orchestral music	75.43 ± 7.15	78.06 ± 6.81
3b	rock/pop/jazz	65.33 ± 5.49	71.83 ± 9.38
4a	chamber subgenres	50.50 ± 9.26	63.05 ± 13.24
4b	orchestral subgenres	52.00 ± 15.65	75.86 ± 18.26
4c	hard rock/soft rock	59.00 ± 19.69	78.91 ± 21.09
4d	pop subgenres	58.67 ± 16.87	71.57 ± 16.04
All classes		59.76 ± 5.23	

Table 9.4: Single-vector split classification performance using the direct approach.

Level	Accumulative performance	Independent Performance
I	96.35 ± 1.70	96.35 ± 1.70
II	85.49 ± 6.26	88.83 ± 5.82
III	70.38 ± 6.32	74.95 ± 8.10
IV	55.04 ± 15.36	72.35 ± 17.16

Table 9.5: Single-vector level classification performance using the direct approach.

9.2.2 Performance Using Only MPEG-7 Features

As another experiment, it has been tested how the system would work if it was fully "MPEG-7-compliant", that is, if only the reviewed *MPEG-7* features were used for the single vector, hierarchical classification. These make a total of four descriptors (*M7CEN*, *M7SPR*, *M7FLAT*, *M7HR*) plus their associated subfeatures, resulting in a total of 16 features. Since these number is lower than the optimal number of features per class (20), all of the 16 features were used for these evaluations, without prior subset selection.

Table 9.6 shows the results of this experiment, for which also a 10-fold cross validation was used (the test sets are the same as the used in the preceding sections).

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	85.41 ± 2.61	85.41 ± 2.61
2a	male/female/speech+background	59.33 ± 11.95	69.66 ± 14.07
2b	classical/non-classical	79.38 ± 3.18	92.98 ± 2.61
3a	chamber music/orchestral music	66.86 ± 4.51	75.91 ± 5.77
3b	rock/pop/jazz	41.67 ± 5.50	60.44 ± 7.86
4a	chamber subgenres	32.50 ± 11.84	47.17 ± 15.00
4b	orchestral subgenres	44.67 ± 9.45	69.32 ± 8.37
4c	hard rock/soft rock	38.00 ± 14.76	67.75 ± 18.00
4d	pop subgenres	26.00 ± 16.47	53.96 ± 26.70
All classes		41.41 ± 5.20	

Table 9.6: Single vector split classification performance using only *MPEG-7* features.

The best independent performance was obtained in the distinction between classical and non-classical music (92.98%), which is comparable to the one obtained with the other features (96.08%). All the other splits show significantly poorer results, especially in the case of the accumulative performances. This results suggest that the evaluated MPEG-7 features do not provide good results by themselves for a classification purpose using the approach proposed in this work. However, the DDL language defined within the standard (see Sect. 3.2) provides the infrastructure to allow the definition of new features if it were necessary.

As far as the classification algorithm is concerned, all the tools needed for its full-MPEG-7-compliant implementation are already defined in the standard. For the implementation of the class tree and its relationships, a Description Scheme called *ClassificationScheme* could be used. For the statistical computations (means, variances, etc.), the standardized *Probability Models* could be used. A *GaussianMixtureModelType* is also normalized in the document. All these tools belong to Part 5 of the standard: *Multimedia Description Schemes* [2].

9.3 Classification Performance in Texture Window Mode

In Sect. 8.4, the texture window operation mode of the application was introduced. One of the goals of the evaluation tests described in this work was to find which of both file-based classification approaches worked better.

For that purpose, a 10-fold cross validation experiment was carried out for each of the following 7 texture window sizes: 0.5, 1, 2, 5, 10, 15 and 20 seconds. It should always be kept in mind that all the training and test samples are approximately 30 seconds long.

Table 9.7 summarizes the results for the independent performances at each level (the complete evaluation tables are given in Appendix A), comparing them with the results of the single-vector mode (last column). It can be seen that, with the largest texture window sizes, the levels of performance reached are similar to those of the single-vector approach.

Level	0.5s	1s	2s	5s	10s	15s	20s	single-vector
I	66.24	68.82	73.18	82.82	88.47	92.71	92.12	94.59
II	56.11	67.90	79.82	84.57	83.94	87.08	88.08	89.20
III	63.42	61.00	62.12	65.8	72.11	74.64	74.79	75.92
IV	50.35	50.19	52.50	58.77	61.11	66.89	68.44	71.39
All classes	24.12	27.88	34.00	41.65	46.69	53.76	53.76	58.71

Table 9.7: Summarized results of the texture window mode evaluation. The first row of the table indicates the lengths of the texture-windows.

Fig. 9.2 is a plot of Table 9.7. It can clearly be seen that the curves stop growing at a window size of 15s. This indicates that observing longer blocks from the files does not provide much additional information, assuming the audio content is reasonably homogeneous. For this reason, the system is expected to maintain similar performance also in the case in which longer texture windows were considered.

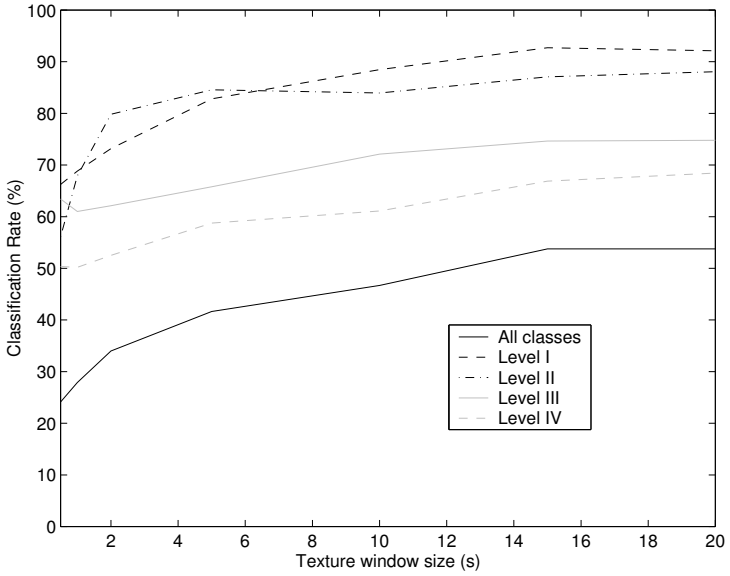


Figure 9.2: Results of the texture window mode evaluation (see Table 9.7).

9.4 Computational Performance

For each classification decision, the most time of the computation is taken by the feature extraction process. In comparison, the GMM decision is virtually instantaneous, since it consists only on the computation of 17 probabilities and on the choice of the greatest one among them. For this reason, the file-based and texture-based approach for a given file take exactly the same time, although in the latter case several classification decisions are met.

In such a hierarchical system, each final classification decision will take a different time, depending on the features that must be computed and on the maximum level reached. Therefore, to test computational performance, 7 different samples belonging to the 7 different splits in which at least one leaf node

Final class group	Relation to real-time
1 (background)	0.64
2a	0.57
4a	0.69
4b	0.71
4c	0.72
4d	0.70
3b (jazz)	0.68
Average	0.67

Table 9.8: Computational performance. The given numbers are fractions of real-time.

is present, were selected. The numbers on Table 9.8 correspond to the ratio of computation time to sample duration. A value of 1 means real-time computation, lower values mean faster than real-time computation.

This test was run on a Pentium III processor at 1000 MHz.

In all cases, the program has proven to operate around 1/3 faster than real-time. It should be noted that real-time computation of the features is meant here, not real-time classification. As mentioned in Sect. 5.1.1, real-time classification is only possible when the frame-based approach for feature extraction is used. In our case, the timely rate of classification decisions will be given by the length of the texture window (or of the file in single vector mode).

Chapter 10

Results and Conclusions

In the first chapters of this thesis, the theoretical background and some of the previous research works in the field of audio classification were presented in detail. This provided the context to situate the main part of the work, which consisted on the thorough description of the process of designing an audio classifier, as well as of its strengths and difficulties.

A total number of 90 features have been considered. They were systematically selected according to their susceptibility to noise and signal bandwidth and to their class separating ability. It was found that, for the given number of 50 training samples per class, a number of 20 features results in an optimal classification performance to computational performance ratio. These 20 features were selected in a class-dependent, hierarchical way, allowing to evaluate their quality in separating a given subset of classes. These results can constitute the starting point for the design of more sophisticated, genre-dependent features.

The line of argument in selecting the classifier was provided by the evaluation and comparison of its classification performance, computational performance, flexibility and generality. These considerations led to the choice of a hierarchical, GMM-based classifier, which in turn constituted the basis for implementing a prototype tool for classifying audio files.

10.1 Summary of Results

The classification rates given in the preceding chapter differ substantially across levels of the tree, showing the different grades of difficulty in separating each corresponding set of classes. The best independent performances were achieved at the highest levels in the tree, for example achieving 94.59% accuracy in differentiating between speech, background and music, 96.08% in separating classical from non-classical music and 81,52% in separating chamber music from orchestral music.

In contrast, the main difficulties arise in the most specific genres at the bottom

levels of the tree, especially in the case of the four chamber music subgenres, where the total classification accuracy is only of 54.67%. The bottom levels of the tree, as well as the high number of classes considered, make the all-class classification rate drop to 58.71%. To achieve higher rates at these levels, more sophisticated, genre-specific features are needed.

As mentioned in Chapter 3, it is not possible to make a reliable comparison across systems with respect to classification accuracy, most of all because of the different number of classes considered (see Tables 3.1 and 3.2). The system that is most similar to the one described in this work in respect of taxonomy complexity and classification method is the one by Tzanetakis and Cook [44]. In this case, a performance of 61% when classifying into 10 classes is reported, compared with the 59.76% (direct approach) and 58.71% (hierarchical approach) rates achieved here when classifying into 17 classes.

The 3-class highest classification level (speech/music/background), for which accuracies of 96.35% (direct approach) and 94.59% (hierarchical approach) were obtained, can be compared with the systems by Zhang and Kuo [50] and Lu *et al.* [24], which achieve performances of 90% and 96.51%, respectively, with exactly the same three classes. The differentiation between male speech, female speech and speech with background (81% direct, 82.31% hierarchical) can be compared to the equivalent classification in Tzanetakis and Cook [44] (74%).

The implemented prototype application can work faster than real-time, and can constitute the base of a practical file-based classifier operating at the highest levels of the tree. The 17-class performance is still too low for considering a practical application comprising all classes in the tree.

The two possible methods for file-based classification (single-vector and texture-window-based classification) were compared. It was obtained that, for texture windows longer than about 15 seconds, classification rates became similar to the ones obtained with the single-vector approach. This suggests that observing more than 15 seconds of homogeneous audio does not provide much additional information for the classification task.

10.2 Contributions to the State of the Art

The following novel issues have been presented in this work:

Hierarchical approach. A fully hierarchical approach is proposed, both in the classification and in the feature selection. It was a special focus of this work to explore the performance of such a hierarchical approach in comparison with the more commonly used direct approach. As described in detail in Sect. 7.2, the hierarchical approach has many advantages with respect to the direct approach, such as less costly errors, upgradability and possibility of implementing genre-dependent features, but also the risk of obtaining

worse overall classification performance than the latter. However, this work shows that this performance difference is only of about 1%, making the hierarchical approach a promising option for the future development of classification systems.

Evaluation of the problems of dimensionality. In the audio classification context, very little attention had been paid to evaluating the problems of high dimensionality in the feature space. Here, the problem is addressed in detail (see Chapter 6), and the convenience of a reduction in dimensionality has been confirmed. This has been made by means of an automatic feature selection algorithm.

New features. A set of new features to represent beat strength and rhythmic regularity is proposed. In particular, the new Rhythmic Regularity feature worked especially well as class separator. Also, a simplified model of loudness as well as the time-domain skewness and kurtosis of the signal are proposed as features.

Evaluation of robustness to noise and bandwidth changes. Features that were most susceptible to noise and moderate changes in the signal bandwidth were discarded, to ensure similar performance across a wide range of audio qualities.

Evaluation of MPEG-7 descriptors. We tested the performance of the *AudioSpectrumCentroid*, *AudioSpectrumSpread*, *AudioSpectrumFlatness* and *Harmonic Ratio* MPEG-7 descriptors as features in the general audio classification problem.

Improvement of the MPEG-7 definition of the Harmonic Ratio. Some inconsistencies were regarded in the definition of the Harmonic Ratio provided in the standard. The proposed implementation was modified obtaining a Harmonic Ratio that worked significantly better in the classification process.

General audio taxonomy. A special effort was made in defining a musicologically consistent taxonomy which tries to be both simple and as general as possible. This taxonomy has resulted in the highest number of classes (17) that has been considered so far in a general audio classification system. Also, the following new classes were introduced: chamber music, chamber music with piano, orchestra with soloist and orchestra with choir.

10.3 Outlook

Automatic Audio Classification is a discipline that is still in its beginnings, especially in the case of music classification. Although algorithms classifying very

differentiated and broad classes such as speech, classical music or popular music have already reached practical levels of performance, there is still much work to be done in order to obtain a system capable of classifying a higher number of classes with high accuracy. Possible general directions in future research include the following:

Design of new features. In this and virtually all previous works it has been observed that performance across different proposed classification algorithms are very similar. This indicates that the success of a system depends mainly not on the pattern recognition algorithm, but on the design of the audio features. In particular, to improve performance at specific subgenres in the lower levels of the taxonomy, genre-dependent features are needed. Possibilities include models of distortion or roughness, detection of singing style, measures of reverberation, etc.

Search of new models. Another direction in future research is to find other statistical models that could represent distributions of audio features more accurately than the usual current models.

Exploration of other dimensionality reduction methods. Many methods for the reduction of dimensions in the feature space have been proposed in the pattern recognition literature [7, 16]. Some of them can be probably succesfully applied in an audio context.

Expansion of the taxonomy tree. This is an obvious direction for improvement, but must be taken with care, examining how likely are the new classes to be separated with the available features.

The above considerations are applicable to the audio classification field as a whole. The following refer to this work in particular and present more specific possibilities for the development of its results:

Implementation of real-time classification. The program could be expanded to allow not only file-based classification, but also real-time, stream-based classification, for which only frame-based feature vectors could be used (see Sect. 5.1.1).

Implementation of training in the application. The application described in Chapter 8 does not have training capabilities, obtaining the model parameters from the MATLAB implementation. An obvious possibility is to upgrade it to include this possibility.

Use of a more sophisticated model of loudness. In spite of the simplicity of the loudness model used here, it worked fairly well as feature. This motivates the implementation of more sophisticated psychoacoustical models considering excitation patterns or partial masking.

Genre-dependent number of features. We based our decision of the number of features on the overall performance of the classifier, and then used the same fixed number in each classification stage. As it has already been mentioned, another possibility would be to address the choice of the number of features[1] in an independent manner, observing in each case the performance at the corresponding level, and choosing a different number of features per stage. This can probably improve performance.

Genre-dependent model parameters. Similarly, it could be explored if selecting a different number of GMM centres at each stage could benefit the classification rate.

Full MPEG-7 compatibility. The DDL language defined within the standard allows to include "non-MPEG-7" features into a full-MPEG-7-compliant application. Also, as mentioned in Sect. 9.2.2, all the other needed tools, such as the hierarchy tree, the statistical measurements, as well as the GMM distribution, are already defined in Part 5 of the standard [2].

[1] not to be confused with the choice of the features themselves, which was already addressed hierarchically

Appendix

Appendix A

Complete Evaluation Data

Single-Vector Mode

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	91.76	91.76
2a	male/female/speech+background	73.33	73.33
2b	classical/non-classical	81.54	91.38
3a	chamber music/orchestral music	80	90.32
3b	rock/pop/jazz	63.33	86.36
4a	chamber subgenres	45	47.37
4b	orchestral subgenres	40	66.67
4c	hard rock/soft rock	70	100
4d	pop subgenres	60	75
All classes		56.47	

Table A.1: Single vector split classification performance (fold 1).

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	94.12	94.12
2a	male/female/speech+background	66.67	76.92
2b	classical/non-classical	92.31	96.77
3a	chamber music/orchestral music	71.43	78.13
3b	rock/pop/jazz	66.67	71.43
4a	chamber subgenres	55	73.33
4b	orchestral subgenres	66.67	100
4c	hard rock/soft rock	50	71.43
4d	pop subgenres	53.33	61.54
All classes		61.18	

Table A.2: Single vector split classification performance (fold 2).

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	97.65	97.65
2a	male/female/speech+background	73.33	78.57
2b	classical/non-classical	93.85	95.31
3a	chamber music/orchestral music	77.14	79.41
3b	rock/pop/jazz	66.67	74.07
4a	chamber subgenres	55	68.75
4b	orchestral subgenres	60	81.82
4c	hard rock/soft rock	50	71.43
4d	pop subgenres	66.67	76.92
All classes		63.53	

Table A.3: Single vector split classification performance (fold 3).

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	95.29	95.29
2a	male/female/speech+background	93.33	93.33
2b	classical/non-classical	90.77	96.72
3a	chamber music/orchestral music	74.29	81.25
3b	rock/pop/jazz	60	66.67
4a	chamber subgenres	35	43.75
4b	orchestral subgenres	53.33	80
4c	hard rock/soft rock	40	80
4d	pop subgenres	60	69.23
All classes		56.47	

Table A.4: Single vector split classification performance (fold 4).

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	95.29	95.29
2a	male/female/speech+background	80	80
2b	classical/non-classical	90.77	95.16
3a	chamber music/orchestral music	74.29	81.25
3b	rock/pop/jazz	60	66.67
4a	chamber subgenres	45	64.29
4b	orchestral subgenres	60	75
4c	hard rock/soft rock	50	83.33
4d	pop subgenres	66.67	83.33
All classes		61.18	

Table A.5: Single vector split classification performance (fold 5).

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	96.47	96.47
2a	male/female/speech+background	73.33	73.33
2b	classical/non-classical	92.31	96.77
3a	chamber music/orchestral music	60	67.74
3b	rock/pop/jazz	66.67	68.97
4a	chamber subgenres	20	36.36
4b	orchestral subgenres	40	60
4c	hard rock/soft rock	70	100
4d	pop subgenres	80	92.31
All classes		56.47	

Table A.6: Single vector split classification performance (fold 6).

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	94.12	94.12
2a	male/female/speech+background	86.67	100
2b	classical/non-classical	95.38	98.41
3a	chamber music/orchestral music	71.43	75.76
3b	rock/pop/jazz	63.33	65.52
4a	chamber subgenres	25	38.46
4b	orchestral subgenres	66.67	83.33
4c	hard rock/soft rock	60	85.71
4d	pop subgenres	60	75
All classes		58.82	

Table A.7: Single vector split classification performance (fold 7).

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	92.94	92.94
2a	male/female/speech+background	73.33	78.57
2b	classical/non-classical	90.77	98.33
3a	chamber music/orchestral music	71.43	80.65
3b	rock/pop/jazz	76.67	82.14
4a	chamber subgenres	50	71.43
4b	orchestral subgenres	46.67	63.64
4c	hard rock/soft rock	70	70
4d	pop subgenres	73.33	84.62
All classes		61.18	

Table A.8: Single vector split classification performance (fold 8).

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	95.29	95.29
2a	male/female/speech+background	80	85.71
2b	classical/non-classical	92.31	96.77
3a	chamber music/orchestral music	88.57	96.88
3b	rock/pop/jazz	53.33	57.14
4a	chamber subgenres	45	47.37
4b	orchestral subgenres	53.33	66.67
4c	hard rock/soft rock	20	33.33
4d	pop subgenres	53.33	80
All classes		55.29	

Table A.9: Single vector split classification performance (fold 9).

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	92.94	92.94
2a	male/female/speech+background	66.67	83.33
2b	classical/non-classical	90.77	95.16
3a	chamber music/orchestral music	74.29	83.87
3b	rock/pop/jazz	60	64.29
4a	chamber subgenres	50	55.56
4b	orchestral subgenres	40	75
4c	hard rock/soft rock	70	100
4d	pop subgenres	46.67	63.64
All classes		56.47	

Table A.10: Single vector split classification performance (fold 10).

Texture-Window Mode

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	66.24 ± 8.48	66.24 ± 8.48
2a	male/female/speech+background	33.33 ± 14.05	42.99 ± 13.03
2b	classical/non-classical	42.61 ± 10.10	69.23 ± 14.23
3a	chamber music/orchestral music	20.86 ± 11.03	76.35 ± 24.50
3b	rock/pop/jazz	27.67 ± 3.16	50.48 ± 9.62
4a	chamber subgenres	9.50 ± 4.38	35.75 ± 24.04
4b	orchestral subgenres	4.57 ± 8.92	28.00 ± 45.41
4c	hard rock/soft rock	19.00 ± 12.87	72.00 ± 33.93
4d	pop subgenres	25.33 ± 10.33	65.63 ± 11.43
All classes		24.12 ± 3.77	

Table A.11: 0.5s texture-window split classification performance.

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	68.82 ± 6.78	68.82 ± 6.78
2a	male/female/speech+background	42.67 ± 8.43	52.43 ± 8.50
2b	classical/non-classical	53.08 ± 9.09	83.38 ± 10.81
3a	chamber music/orchestral music	33.14 ± 13.21	69.44 ± 16.78
3b	rock/pop/jazz	30.33 ± 4.83	52.55 ± 7.74
4a	chamber subgenres	13.5 ± 4.74	34.09 ± 15.33
4b	orchestral subgenres	9.33 ± 8.43	40.67 ± 32.50
4c	hard rock/soft rock	20.00 ± 14.91	59.67 ± 37.92
4d	pop subgenres	27.33 ± 9.66	66.31 ± 11.27
All classes		27.88 ± 2.71	

Table A.12: 1s texture-window split classification performance.

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	73.18 ± 2.92	73.18 ± 2.92
2a	male/female/speech+background	58.67 ± 12.88	67.23 ± 14.32
2b	classical/non-classical	63.08 ± 5.32	92.40 ± 6.60
3a	chamber music/orchestral music	44.29 ± 12.07	67.48 ± 12.80
3b	rock/pop/jazz	34.00 ± 5.16	56.75 ± 8.14
4a	chamber subgenres	19.50 ± 6.85	35.43 ± 10.95
4b	orchestral subgenres	16.00 ± 11.84	42.55 ± 27.66
4c	hard rock/soft rock	26.00 ± 16.47	66.14 ± 39.23
4d	pop subgenres	28.67 ± 9.96	65.90 ± 10.81
All classes		34.00 ± 3.21	

Table A.13: 2s texture-window split classification performance.

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	82.82 ± 2.37	82.82 ± 2.37
2a	male/female/speech+background	71.33 ± 8.92	74.38 ± 9.73
2b	classical/non-classical	74.77 ± 2.92	94.76 ± 2.89
3a	chamber music/orchestral music	55.71 ± 10.28	69.31 ± 11.07
3b	rock/pop/jazz	42.67 ± 6.81	62.29 ± 6.59
4a	chamber subgenres	25.50 ± 6.43	39.89 ± 4.80
4b	orchestral subgenres	30.67 ± 13.77	65.08 ± 27.25
4c	hard rock/soft rock	31.00 ± 18.53	67.64 ± 31.03
4d	pop subgenres	35.33 ± 12.59	62.45 ± 16.14
All classes		41.65 ± 4.44	

Table A.14: 5s texture-window split classification performance.

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	88.47 ± 2.87	88.47 ± 2.87
2a	male/female/speech+background	70.00 ± 12.27	72.67 ± 14.45
2b	classical/non-classical	82.00 ± 3.91	95.20 ± 2.99
3a	chamber music/orchestral music	62.86 ± 7.25	74.86 ± 6.91
3b	rock/pop/jazz	55.00 ± 5.50	69.35 ± 7.94
4a	chamber subgenres	30.00 ± 10.54	42.55 ± 12.72
4b	orchestral subgenres	36.67 ± 11.00	69.19 ± 20.82
4c	hard rock/soft rock	39.00 ± 17.29	67.35 ± 14.05
4d	pop subgenres	47.33 ± 10.16	65.35 ± 10.23
All classes		46.59 ± 4.23	

Table A.15: 10s texture-window split classification performance.

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	92.71 ± 1.98	92.71 ± 1.98
2a	male/female/speech+background	76.00 ± 10.51	79.18 ± 10.53
2b	classical/non-classical	87.54 ± 3.44	94.99 ± 2.84
3a	chamber music/orchestral music	70.29 ± 10.19	78.97 ± 10.17
3b	rock/pop/jazz	60.33 ± 5.76	70.31 ± 5.86
4a	chamber subgenres	39.00 ± 13.08	52.93 ± 15.85
4b	orchestral subgenres	46.67 ± 11.76	71.40 ± 15.20
4c	hard rock/soft rock	46.00 ± 15.78	72.75 ± 22.74
4d	pop subgenres	55.33 ± 7.73	70.49 ± 9.65
All classes		53.76 ± 4.87	

Table A.16: 15s texture-window split classification performance.

Split	Classes	Accumulative performance	Independent Performance
1	speech/music/background	92.12 ± 2.42	92.12 ± 2.42
2a	male/female/speech+background	77.33 ± 7.17	80.71 ± 8.60
2b	classical/non-classical	87.84 ± 4.38	95.45 ± 2.60
3a	chamber music/orchestral music	68.57 ± 8.73	77.05 ± 7.96
3b	rock/pop/jazz	62.67 ± 6.63	72.52 ± 7.50
4a	chamber subgenres	36.50 ± 13.34	49.46 ± 15.62
4b	orchestral subgenres	46.00 ± 16.16	72.47 ± 20.63
4c	hard rock/soft rock	51.00 ± 13.70	77.99 ± 18.26
4d	pop subgenres	60.00 ± 10.42	73.85 ± 12.43
All classes		53.76 ± 5.26	

Table A.17: 20s texture-window split classification performance.

Appendix B

Results of the Musical Questionnaire

The tables on the next two pages show the complete results of the musical questionnaire outlined in Sect. 4.2. Numerical values are the percentages of replies that assigned the corresponding broad genre (rock, pop or jazz) as a father genre for the given example. For each example, a list of the proposed specific subgenres is given. Numbers in parenthesis give the total number of proposals given for each subgenre.

The last column lists which genre assigned the implemented application (level 3 classification) to each of the examples. In this case, parenthesis indicate the certainty of each decision. The two examples wrongly classified as classical music correspond to music excerpts with little beat strength (example 8) or with no drum set at all (example 2).

The numbers listed in the first column correspond to the file names of the music excerpts, as included on the enclosed CD-ROM (see Appendix C).

Total number of replies : 75

	Artist and title	Rock	Pop	Jazz	Proposed subgenres (number of proposals)	Classified by the system as
1.	James Brown: "Papa's Got A Brand New Bag" (1965)	41.9	16.2	41.9	Soul (15), Funk (12), Blues (3), Rhythm 'n' Blues (2), Swing (2), 60's (1), 70's (1), Boogie-Woogie (1), Pop-Rock (1), Rock-Jazz (1), Classic Rock (1), American Rock (1), Big Band (1)	Pop (72.2%)
2.	Tom Waits: "Pony" (1999)	32.4	24.3	43.2	Country (14), Blues (7), Folk (7), Songwriter (3), Experimental Rock (1), Melodic Pop (1), Soul (1), Rock Ballad (1), Rhythm 'n' Blues (1), 60's (1)	Classical (71.4%)
3.	Toto: "Stop Loving You" (1988)	12	86.7	1.3	80's (13), Soft Rock (2), Pop-Rock (2), Arena (1), Melodic Pop (1), Ballad (1), Commercial Pop (1)	Rock (99.9%)
4.	Tom Waits: "Big in Japan" (1999)	93	5.6	1.4	Heavy Metal (3), Electronic (3), Independent (2), Alternative (2), Funk (2), Hard Rock (2), Punk (1), Rockabilly (1), Degenerated Rock (1), Electric Rock (1), Electronic Rock (1), Post-Modern Rock (1), Soft Rock (1), Blues Rock (1), Psychedelic Rock (1), 80's (1), Retro (1)	Pop (99.2%)
5.	World Party: "Is It Like Today?" (1993)	16	82.7	1.3	Pop-Rock (3), Alternative (2), Folk (2), Independent (2), Country (2), Acoustic Rock (1), Soft Rock (1), Guitar Pop (1), Songwriter (1), Romantic Rock (1), 90's (1), Folk-Rock (1), Classic Rock (1), Brit-Pop (1), Light Pop (1), Happy Pop (1)	Rock (94.7%)
6.	The Cranberries: "Animal Instinct" (1999)	26.7	72	1.3	Pop-Rock (8), Alternative (1), Irish Rock (1), Irish Pop-Rock (1), Folk (1), Guitar Pop (1), Folk Pop (1), Happy Pop (1), Celtic Rock (1)	Rock (99.8%)
7.	Dire Straits: "So Far Away" (1985)	30.7	64	5.3	Pop-Rock (6), Soft Rock (2), 80's (2), 60's (1), Blues Pop (1), Country (1), Guitar Pop (1), Blues (1), Classic Rock (1), Ballad (1), Melodic Pop-Rock (1)	Pop (99.9%)
8.	The Eagles: "Hollywood Waltz" (1975)	43.2	43.2	13.6	Country (27), Blues Pop (1), New Age (1), Folk (1), Country Waltz (1), Blues (1), Texan Folk (1), Ballad (1), Western (1), Catalan Rock (1)	Classical (51.2%)

Table B.1: Results of the musical questionnaire (first part).

	Artist and title	Rock	Pop	Jazz	Proposed subgenres (number of proposals)	Classified by the system as
9.	Extreme: "More Than Words" (1990)	25	72.4	2.6	Ballad (16), Pop-Rock (2), Melodic Pop (2), Folk (2), 90's (2), 70's (1), Light Pop (1), Alternative (1), Acoustic Rock (1), Soft Rock (1), Classic Rock (1), Swing (1)	Jazz (99.72%)
10.	Fermín Muguruza & Dub Manifest: "Lagun Nazakezu?" (1999)	32.4	56.8	10.8	Reggae (21), Ska (13), Pop-Rock (4), Latin (4), Samba (2), Folk (2), Funk (1), Pop-Folk (1), Merengue (1)	Rock (62%)
11.	Eddie Rabbit: "When I Think About You" (1981)	30.1	43.8	26.1	Country (28), Folk (3), Gospel (3), Blues (2), Country Blues Pop (1), Skiffle (1), Western (1), Pop-Rock (1), Oldies (1), Swing (1)	Pop (99.1%)
12.	Reel Big Fish: "Scott's A Dork" (1998)	46.6	49.3	4.1	Ska (28), Pop-Rock (4), Punk (2), 90's (2), Funk (1), Ballad (1), Folk (1), Beat (1), Latin (1), New Age Rock (1), Mazurka (1)	Rock (100%)
13.	Shakira: "Poem To A Horse" (2001)	67.6	32.4	0	Pop-Rock (11), Latin Pop (2), Latin Rock (2), Alternative (1), Hard Rock (1), 90's (1), Commercial Pop-Rock (1)	Rock (99.9%)
14.	Sting: "The Soul Cages" (1991)	29.7	67.6	2.7	Pop-Rock (3), Rock Ballad (1), Guitar Pop (1), Folk (1), 90's (1), 80's (1), Brit-Pop (1), Electric Pop (1), Symphonic Rock (1), Jazz-Pop (1), Romantic Pop (1), Jazzy Rock (1), Art Rock (1), Progressive Rock (1), Rock Fusion (1), Melodic Pop (1), Slow Classical Rock (1), Soft Rock (1)	Rock (99.8%)
15.	Travis: "Flowers In The Window" (2001)	16.9	80.3	2.8	Brit-Pop (4), Pop-Rock (3), Alternative (2), Guitar Pop (2), Ballad (2), Independent (2), Southern Rock (1), Folk Ballad (1), Slow Rock (1), Light Pop (1), U2-Style (1), Christian Rock (1), Gay (1), Depressive Pop-Rock (1), Psychedelic Pop (1), 90's (1), Oldies (1)	Pop (68%)
16.	The Beatles: "You Can't Do That" (1964)	53.4	38.4	8.2	Pop-Rock (5), Beat (3), Blues (2), 60's (2), 50's (1), Rhythm 'n' Blues (1), Soul (1), Classic Rock (1), Brit Sound (1), Old Pop (1)	Rock (97.3%)
17.	Bob Marley: "So Much Trouble In The World" (1979)	25.4	50.7	23.9	Reggae (51), Rhythm 'n' Blues (1), Rasta Style (1)	Pop (100%)

Table B.2: Results of the musical questionnaire (second part).

Appendix C

Contents of the CD-ROM

The contents of the enclosed CD-ROM are organized in the three following directories:

- `classify`: Contains the `classify.exe` application, together with the needed `*.gmm` and `*.par` files, which store the model parameters. For instructions on usage, see Chapter 8.
- `test`: Contains a test set of 85 audio `*.wav` files (10% of the whole database). The included application was trained using the remaining 90% audio files from the database.
- `questionnaire`: Contains the 17 audio examples used for the musical genre questionnaire outlined in Sect. 4.2 and in Appendix B. The file numbers correspond to the numbers listed on Tables B.1 and B.2.
- `figure_examples`: Contains the two audio examples represented in the figures throughout Chapter 5.

List of Figures

List of Tables

Bibliography

[1] ISO/IEC FDIS 15938-4. *Information Technology - Multimedia Content Description Interface - Part 4 Audio*. 2001.

[2] ISO/IEC FDIS 15938-4. *Information Technology - Multimedia Content Description Interface - Part 5 Multimedia Description Schemes*. 2001.

[3] J-J. Aucouturier and F. Pachet. Representing musical genre: A state of art. *Journal of New Music Research*, 2002.

[4] L. Breiman, J. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.

[5] M. Casey. General sound classification and similarity in mpeg-7. *Organized Sound*, 6:2, 2002.

[6] H. Deshpande, U. Nam, and R. Singh. Mugec: Automatic music genre classification. Technical report, Stanford University, 2001.

[7] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley, 2000.

[8] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. John Wiley, 1973.

[9] K. El-Maleh, M. Klein, G. Petrucci, and P. Kabal. Speech/music discrimination for multimedia applications. In *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, pages 2445–2448, June 2000.

[10] F. Fabbri. Browsing musical spaces: Categories and the musical mind. *Intern. Ass. for the Study of Popular Music (IASPM)- Conference*, 1999.

[11] B. Feiten and S. Günzel. Automatic indexing of a sound database using self-organizing neural nets. *Computer Music Journal*, 18(3), 1994.

[12] J. Foote. A similarity measure for automatic audio classification. *Proc. of the AAAI 1997 Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora. Stanford.*, 1997.

[13] J. Foote. An overview of audio information retrieval. *Multimedia Systems*, 7, 1999.

[14] J. Foote and S. Uchihashi. The beat spectrum: A new approach to rhythm analysis. *IEEE International Conference on Multimedia and Expo*, 2001.

[15] M. Frühwirth and A. Rauber. Self-organizing maps for content-based music clustering. *Proc. of the 12th Italian Workshop on Neural Nets (WIRN01)*, May 2001.

[16] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, 1990.

[17] F.J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66, No. 1, January 1978.

[18] L.B. Jackson. *Digital Filters and Signal Processing.* Kluwer Academic Publishers, 1989.

[19] A.K. Jain and B. Chandrasekaran. Dimensionality and sample size considerations in pattern recognition practice. *Handbook of Statistics, ed. Krishnaiah, P.R. and Kanal, L.N.*, Vol. 2, North-Holland, 1987.

[20] D.-N. Jiang, L. Lu, H.-J. Zhang, J.-H. Tao, and L.-H. Cai. Music type classification by spectral contrast feature. *ICME - International Conference on Multimedia and Expo*, 2002.

[21] T. Lambrou, P. Kudumakis, R. Speller, M. Sandler, and A. Linney. Classification of audio signals using statistical features on time and wavelet transform domains. *Proc. IEEE ICASSP*, 1998.

[22] Stan Z. Li. Content-based audio classification and retrieval using the nearest feature line method. *IEEE Transactions on Speech and Audio Processing*, Vol. 8, No. 5, September 2000.

[23] B. Logan. Mel frequency cepstral coefficients for music modeling. *Proc. of the 1st Annual International Symposium on Music Information Retrieval (ISMIR)*, 2000.

[24] L. Lu and H. Jiang. Content analysis for audio classification and segmentation. *IEEE Transactions on Speech and Audio Processing*, Vol. 10, No. 7, October 2002.

[25] B.S. Manjunath, P. Salembier, and T. Sikora, editors. *Introduction to MPEG-7.* Wiley, 2002.

[26] W.S. Meisel. *Computer-Oriented Approaches to Pattern Recognition.* Academic Press, 1972.

[27] B. C. J. Moore, B.R. Glasberg, and T. Baer. A model for the prediction of thresholds, loudness and partial loudness. *J. Audeo Eng. Soc.*, 45 (4), April 1997.

[28] I. Nabney and C. Bishop. Netlab neural network software. Technical report, http://www.ncrg.aston.ac.uk/netlab/.

[29] A. Oppenheim and R. Schafer. *Discrete-Time Signal Processing.* Prentice Hall, 1989.

[30] F. Pachet and D. Cazaly. A taxonomy of musical genres. In *Proc. RIAO Paris*, April 2000.

[31] E. Pampalk. Islands of music: Analysis, organization, and visualization of music archives. Master's thesis, Technische Universität Wien, December 2001.

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

[33] D. Pye. Content-based methods for the management of digital music. *Proc. 2000 IEEE International Conference on Acoustic Speech and Signal Processing (ICASSP)*, 2000.

[34] A. Rauber and M. Frühwirth. Automatically analyzing and organizing music archives. *Proc. of the 5. European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, September 2001.

[35] E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *Proc. ICASSP*, 1997.

[36] E.D. Scheirer. Tempo and beat analysis if acoustic musical signals. *J. Acoust. Soc. Am.*, 103 (1), January 1998.

[37] M. Slaney. Auditory toolbox: A matlab toolbox for auditory modeling work. Technical report, Interval Research Corporation.

[38] H. Soltau. *Erkennung von Musikstilen.* PhD thesis, Institut für Logik, Komplexität und Deduktionssysteme, Universität Karlsruhe, 1997 (in German).

[39] H. Soltau, T. Schultz, M. Westphal, and A. Waibel. Recognition of music types. *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 1998.

[40] S.S. Stevens. The measurement of loudness. *Journal of the Acoust. Soc. of America*, 27(5), September 1955.

[41] E. Terhardt. The spinc function for scaling of frequency in auditory models. *Acustica*, 77, 1992.

[42] J.T. Tou and R.C. González. *Pattern Recognition Principles*. Addison-Wesley, 1974.

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

[44] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, Vol. 10, No. 5, July 2002.

[45] G. Tzanetakis, G. Essl, and P. Cook. Automatic musical genre classification of audio signals. *Proc. of the 2nd Annual International Symposium on Music Information Retrieval (ISMIR)*, 2001.

[46] G. von Bismarck. Sharpness as an attribute of the timbre of steady sounds. *Acustica*, 30, 1974.

[47] G. von Bismarck. Timbre of steady sounds: A factorial investigation of its verbal attributes. *ACUSTICA*, 30, 1974.

[48] E. Wold, T. Blum, D. Keislar, and J. Wheaton. Content-based classification, search and retrieval of audio. *IEEE Multimedia*, Fall, 1996.

[49] T. Zhang and J. Kuo. Content-based classification and retrieval of audio. *Conference on Advanced Signal Processing Algorithms, Architectures and Implementations, VIII*, July 1998.

[50] T. Zhang and J. Kuo. Hierarchical system for content-based audio classification and retrieval. *Proc. of International Conference on Acoustic, Speech and Signal Processing.*, Vol. 6, 1998.

[51] U. Zölzer (Ed.). *DAFX: Digital Audio Effects*. Wiley, 2002.

[52] E. Zwicker and H. Fastl. *Psychoacoustics*. Springer, 1999.

Index