Design and Realisation of an Efficient Content Based Music Playlist Generation System

J.W. Balkema

Design and Realisation of an Efficient Content Based Music Playlist Generation System - J.W. Balkema

ISBN: 978-90-365-2886-3
DOI: 10.3990/1.9789036528863

Design and Realisation of an Efficient Content Based Music Playlist Generation System

Wietse Balkema

De promotiecommissie:

voorzitter en secretaris:	
prof.dr.ir. J. van Amerongen	Universiteit Twente
promotoren:	
prof.dr.ir. C.H. Slump	Universiteit Twente
prof.dr.-ing K. Brandenburg	Fraunhofer Institut für Digitale Medientechnik
ass.promotor:	
dr.ir. F. van der Heijden	Universiteit Twente
referent:	
dipl.-ing G. Spreitz	Bosch, Hildesheim
leden:	
prof.dr.ir. P.L. Regtien	Universiteit Twente
prof.dr.ing. W.B. Verwey	Universiteit Twente
dr. A. Hanjalic	TU Delft/Universiteit Twente

Signals & Systems group P.O. Box 217, 7500 AE Enschede, the Netherlands

Print: Wöhrmann Print Service
Typesetting: LaTeX2e

ISBN 978-90-365-2886-3
DOI 10.3990/1.9789036528863

DESIGN AND REALISATION
OF AN EFFICIENT CONTENT BASED
MUSIC PLAYLIST GENERATION SYSTEM

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. E. Brinksma,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op 27 augustus 2009 om 13.15 uur

door

Jan Wietse Balkema
geboren op 9 november 1978
te Arnhem

Dit proefschrift is goedgekeurd door de promotoren:

Prof.dr.ir. C.H. Slump
Prof.dr.-ing K. Brandenburg

ISBN 978-90-365-2886-3

There's a basic rule which runs through all kinds of music, kind of an unwritten rule. I don't know what it is. But I've got it.
- Ron Wood, guitarist of the Rolling Stones

Abstract

This thesis is on the subject of content based music playlist generation systems. The primary aim is to develop algorithms for content based music playlist generation that are faster than the current state of technology while keeping the quality of the playlists at a level that is at least comparable with that of the current state of technology. Not only need the algorithms be fast, they shall also allow flexibility for the end user to be able to tune the algorithms to match his personal requirements. For evaluation of the algorithms, a framework for automatic content based music playlist generation is developed and presented.

In order to be able to evaluate the quality of music playlist generation systems, criteria for quality judgment of playlists have to be known. To gain insight in these quality criteria, a questionnaire is developed. The responses on this questionnaire are analysed. It shows that the number of parameters that influence the perceived quality of a personal playlist is huge, and the individual variation of desired values is large. Because of the large variance in preferred values, it is impossible to find one single set of parameters that suits for all people. Using the results of the questionnaire, it is argued that playlist genre consistency is a suitable criterion for assessing playlist quality. Songs within a playlist should have approximately the same genre. The key to good music playlist generation systems therefore is a good music similarity measure, that allows finding 'similar' music. To speed up music playlist generation systems, the music similarity measures used by these systems should be fast. This thesis presents two steps towards faster music similarity measures.

The first step is early in the process of determining music similarity. The properties of each song are statistically described by a Gaussian mixture model. Similarity between songs is determined by applying the earth mover's distance on these statistical models of the songs to measure the similarity of the songs. Where the current state of technology uses a constant number of Gaussians for all songs, this thesis presents a method to estimate an optimal number of Gaussians for each individual song. This prevents the Gaussian mixture

models from overfitting the data of one song and allows the average number of Gaussians per song to be reduced without sacrificing quality of the similarity measure. The computation time decreases quadratically with the average reduction of the number of Gaussians.

The second step towards faster playlist generation algorithms is an algorithm that allows Gaussian mixture models to be embedded into self organizing maps. Self organizing maps are neural networks that traditionally allow features of high dimensionality to be mapped into a low dimensional space. In this research, self organizing maps with embedded Gaussian mixture models have been pioneered. Using Gaussian mixture models as neurons in a self organizing map allows robustness of the song models to be obtained even in the low dimensional space of the map. After training a self organizing map and mapping the song models to the map, song similarity calculations are performed in the two-dimensional space of the self organizing map, instead of the high dimensional gaussian mixture models. The two-dimensional space allows using simple distance measures like the euclidean distance, to determine the similarity between two songs. This is orders of magnitude faster than using the earth mover's distance on the statistical song models directly.

It shows that the average number of clusters per song can be reduced by using Gaussian mixture models with an individually estimated optimal number of clusters per song. This allows faster music similarity computation without sacrificing quality of the similarity measure. Projecting the Gaussian mixture models in self organizing maps allows song similarity to be calculated extremely fast. By combining multiple self organizing maps, flexible, high-quality music playlist generation systems can be realised.

Samenvatting

Dit proefschrift bespreekt systemen die automatisch (muziek)playlisten samenstellen en daarbij uitsluitend gebruik maken van de inhoud van de muziek. Het doel van het onderzoek is het ontwikkelen van algoritmes die sneller zijn dan de algoritmes die in de huidige stand der techniek toegepast worden, zonder daarbij aan kwaliteit in te boeten. Deze algoritmes moeten niet alleen snel, maar ook flexibel zijn, zodat een gebruiker de algoritmes kan aanpassen aan zijn persoonlijke wensen. Ter evaluatie van de algoritmes wordt een zelf ontwikkeld framework voor het automatische genereren van playlisten gepresenteerd.

Voor het beoordelen van de kwaliteit van automatisch gegenereerde playlisten moeten beoordelingscriteria vastgelegd worden. Om vast te stellen wat goede criteria zijn, is een enquete ontwikkeld en uitgevoerd. De resultaten van deze enquete worden besproken. Het blijkt dat het aantal parameters die de kwaliteitswaarneming van playlisten beïnvloeden groot is, en dat de voorkeurswaarden van deze parameters per persoon erg verschillend zijn. Door deze grote verschillen in voorkeur is het niet mogelijk om een eenvoudige parameterset te vinden die de voorkeuren van alle gebruikers goed beschrijven. Aan de hand van de resultaten van de enquete, wordt afgeleid dat genre-consistentie in een playlist een criterium is dat voor de kwaliteitsbeoordeling van playlisten geschikt is. Muziekstukken in een playlist moeten ongeveer hetzelfde genre hebben. Voor systemen die automatisch playlisten genereren is het dus van essentieel belang om een goede methode te hebben om te berekenen hoeveel twee muziekstukken op elkaar lijken. Om deze systemen sneller te maken, moeten de algoritmes die twee muziekstukken met elkaar vergelijken snel zijn. In dit proefschrift worden hiervoor twee methoden beschreven.

De eerste methode betreft het modelleren van muziekstukken. De eigenschappen van elk muziekstuk worden statistisch beschreven met een ‘Gaussian Mixture Model’ (GMM). Hoeveel twee muziekstukken op elkaar lijken wordt berekend door de ‘Earth Mover’s Distance’, een algoritme dat de overeenkomsten tussen twee GMMs berekent. In de huidige generatie van systemen

wordt voor elk lied hetzelfde aantal clusters per GMM gebruikt. In dit proefschrift wordt een methode gepresenteerd waarmee voor elk muziekstuk individueel een optimaal aantal clusters kan worden geschat. Het voordeel van een optimaal aantal clusters per muziekstuk is dat elk muziekstuk beschreven wordt door een model dat niet onnodig veel parameters heeft, terwijl toch de kwaliteit van het model goed genoeg is om de inhoud van het muziekstuk te beschrijven. De rekentijd die nodig is om de gelijkheid tussen twee GMMs te berekenen hangt kwadratisch samen met het gemiddelde aantal clusters in de GMMs zodat hierdoor de rekentijd aanzienlijk kan worden verkort.

De tweede methode betreft een algoritme waarmee het voor het eerst mogelijk wordt, GMMs in zelf-organiserende neurale netwerken (SOMs) op te nemen. SOMs zijn neurale netwerken die veel-dimensionale vectoren in een weinig-dimensionale ruimte kunnen afbeelden. Hierbij blijft de onderlinge afstandsverhouding ongeveer gelijk aan die in de veel-dimensionale ruimte. Door GMMs in SOMs op te nemen, kunnen de voordelen van gedetailleerde statistische beschrijvingen van muziekstukken gecombineerd worden met de voordelen van rekenen in een weinig-dimensionale ruimte in een SOM. Nadat een SOM getraind is, kunnen de veel-dimensionale muziekstukmodellen (de GMMs) in de SOM afgebeeld worden. Het voordeel hiervan is dat verdere berekeningen in de twee-dimensionale ruimte in de SOM plaatsvinden. Hierdoor is het mogelijk om in plaats van het vergelijken van GMMs met de Earth Mover's Distance, eenvoudigere afstandsberekeningen te gebruiken. In dit proefschrift wordt de simpele Euclidische afstandsberekening gebruikt welke veel sneller is dan de Earth Mover's Distance.

Het blijkt dat door het individuele schatten van een optimaal aantal clusters per GMM, het gemiddelde aantal clusters per muziekstuk verlaagd kan worden zonder dat de kwaliteit van het systeem hierdoor vermindert. Door het geringere aantal clusters per lied worden de berekeningen eenvoudiger en dus sneller. Daarnaast wordt door het projecteren van de GMMs in een SOM de snelheid van de afstandsberekeningen tussen twee muziekstukken nogmaals verhoogd. Als meerdere SOMs gecombineerd worden, kunnen systemen gerealiseerd worden die eenvoudig aan de wensen van de gebruiker aangepast kunnen worden en daarbij toch de gewenste kwaliteit leveren.

Zusammenfassung

Diese Dissertation befasst sich mit dem Thema der inhaltsbasierten und automatischen Erstellung von Playlisten. Primäres Ziel der Arbeit ist die Entwicklung neuer und schnellerer Algorithmen zum Generieren von Playlisten, wobei trotz höherer Geschwindigkeit eine mindestens vergleichbare Qualität zu bisherigen Verfahren erreicht werden soll. Neben diesen Hauptkriterien, Geschwindigkeit und Qualität, sollen die Algorithmen leicht anpassbar sein. Dies bietet den Vorteil, dass der Benutzer die Algorithmen so verändern kann, dass die Playlisten seinen Vorlieben entsprechen. Zur Evaluation der Algorithmen wird ein selbstentwickelter Framework für das schnellere automatische Generieren von Playlisten vorgestellt.

Zur Beurteilung der Qualität automatisch generierter Playlisten, müssen Qualitätskriterien definiert werden. Hierzu werden die Ergebnisse einer Nutzerbefragung, die im Rahmen dieser Arbeit entwickelt und durchgeführt wurde, analysiert, um einen Einblick in die Qualitätskriterien für personalisierte Playlisten zu erhalten. Die Befragung zeigt auf, dass die Anzahl von Parametern, welche die Qualitätswahrnehmung von Playlisten beeinflussen, sowie die Varianz dieser Parameter sehr groß ist. Diese große Varianz führt dazu, dass es nicht möglich ist, die Vorlieben aller Nutzer mit einem einfachen Parametersatz zu beschreiben. Mit den Ergebnissen der Nutzerbefragung als Grundlage wird daher argumentiert, dass Genrekonsistenz in automatisch generierten Playlisten das geeignete Qualitätsmerkmal ist, um die Qualität von Playlisten zu beurteilen: Die Lieder in einer Playliste sollten aus möglichst ähnlichen Genres stammen. Der Schlüssel zu guten automatisch generierten Playlisten ist also ein Musikähnlichkeitsmaß, dass es erlaubt 'ähnliche' Lieder zu finden. Um Systeme für das automatische Generieren von Playlisten zu beschleunigen, müssen die Algorithmen zur Musikähnlichkeitsbestimmung, die von diesen Systemen verwendet werden, schnell sein. Innerhalb dieser Dissertation werden zwei Methoden aufgezeigt, die eine schnellere Berechnung von Musikähnlichkeit ermöglichen.

Die erste Methode setzt beim Modellieren der einzelnen Lieder an. Die Liedeigenschaften werden statistisch mittels des 'Gaussian Mixture Models' (GMM) beschrieben. Die Ähnlichkeit zwischen einzelnen Liedern wird durch das Anwenden eines Distanzmaß auf diese statistischen Datenmodelle (der 'Earth Mover's Distance') berechnet. Anders als beim jetzigen Stand der Technik üblich, wird in der vorliegenden Arbeit für jedes Lied individuell die optimale Anzahl von Clustern pro GMM bestimmt. Durch die individuelle Bestimmung der optimalen Clusteranzahl kann deren durchschnittliche Anzahl verringert werden, ohne dass die Qualität der Ähnlichkeitsbestimmung darunter leidet. Dies hat den großen Vorteil, dass die benötigte Rechenzeit zur Ähnlichkeitsbestimmung erheblich reduziert wird.

Die zweite Methode stellt einen weiteren Schritt zum schnelleren automatischen Generieren von Playlisten dar. Es wird ein Algorithmus vorgestellt, der es erstmalig ermöglicht GMMs in selbstorganisierende neuronale Netze (SOMs) aufzunehmen. SOMs sind neuronale Netze, die angewendet werden können, um hochdimensionale Vektoren in einem niedrigdimensionalen Raum abzubilden. Hierbei bleiben die Distanzverhältnisse zwischen den hochdimensionalen Vektoren im niedrigdimensionalen Raum in etwa erhalten. Durch das Verwenden von GMMs in SOMs können die Vorteile von der detaillierten Liedbeschreibung mittels GMMs mit den Vorteilen von SOMs kombiniert werden: das Rechnen im niedrigdimensionalen Raum. Nachdem ein SOM trainiert worden ist, können die einzelnen Liedmodelle in die niedrig-dimensionalen SOM's hineinprojiziert werden. Nun können die Liedähnlichkeitsberechnungen im zweidimensionalen SOM-Raum stattfinden. Hierdurch wird es möglich, einfache Distanzmaße wie zum Beispiel die Euklidische Distanz zu verwenden, um die Ähnlichkeit zweier Lieder zu bestimmen. Hierdurch kann eine schnellere Ähnlichkeitsberechnungen zwischen den Liedmodellen erfolgen als mit dem Earth Mover's Distance Algorithmus.

Es stellt sich in der Evaluation heraus, dass die durchschnittliche Anzahl von Clustern pro Lied verringert werden kann, wenn von jedem Lied individuell die optimale Anzahl von Clustern bestimmt wird. Dies ermöglicht schnellere Musikähnlichkeitsberechnungen bei reduziertem Rechenaufwand und ohne Qualitätseinbußen. Des Weiteren ermöglicht das Projizieren von GMMs in SOMs eine extrem schnelle Liedähnlichkeitsberechnung. Durch das Kombinieren von mehreren SOMs können leicht anpassbare Systeme für die automatische Generierung von Playlisten realisiert werden und gleichzeitig den Qualitätsanforderungen genügen.

Contents

List of Figures

List of Tables

Abbreviations and symbols

Abbreviations

BIC	Bayesian information criterium
DRM	Digital rights management
EMD	Earth movers distance
DFT	Discrete Fourier transform
FFT	Fast Fourier transform
GMM	Gaussian mixture model
HMM	Hidden Markov model
ISMIR	International conference on music information retrieval
KMM	K-means model
MDL	Minimum description length
MDS	Multidimensional scaling
MFCC	Mel-scale Frequency Cepstral Coefficients
MIR	Music information retrieval
SOM	Self organizing map
STFT	Short time Fourier transform
TFIDF	Term frequency inverce document frequency

Symbols and Units

$\boldsymbol{m}$	Codebook vector
$\boldsymbol{\Sigma}$	Covariance
dB	Decibel (sound pressure level)
$\mathcal{G}$	Gaussian
Hz	Herz (frequency)
$\boldsymbol{\mu}$	Mean
$\mathcal{M}$	Model
$\boldsymbol{\Theta}$	Model parameters
$\mathcal{S}$	Dataset
α	Weight (cluster)
$\mathcal{X}$	Feature space

Chapter 1

Introduction

This chapter presents a short description of this thesis and indicates the relevance in the context of Music Information Retrieval (MIR). Section 1.1 justifies the work in this thesis by reviewing the current possibilities of music distribution and access. Section 1.2 gives a broad overview of the MIR research areas. The problem definition and thesis structure are given in Section 1.3.

1.1 Justification

Since the introduction and standardisation of MP3 in 1991, digital music distribution drastically changed the possibilities for accessing music. Together with ever increasing internet connection speeds, a vast amount of online music providers, and affordable portable digital music players, one can easily find all music one likes and listen to it, in theory everywhere.

In practice there are some problems with access to digital music. Digital Rights Management (DRM) prevents you from listening to the music you bought for your IPod on your Microsoft Zune player. Tools for organizing large music collections are of limited use, and you may never find that piece that is stored somewhere on your IPod again if you have forgotten its name and artist.

Finding the music one likes without having to search for hours is one of the biggest challenges the music industry of today has to deal with. Web applications like Last.FM[1], MyStrands[2], and iLike[3] offer interfaces for finding music one might like, based on observing your listening behaviour. These

[1] http://www.last.fm
[2] http://www.mystrands.com
[3] http://www.ilike.com

applications compare your 'profile' with that of other users and recommend music other similar listeners have liked.

At this moment, the web applications that fall into the category of 'social recommendation systems' dominate the market. The drawback of social recommendation systems is twofold: Before a clear profile of a song or artist is available, there must have been many listeners that listened to this song or artist and have shared their data. There will always be a delay between the release of a song and obtaining robust recommendations of this song. This issue is called the 'cold start' problem [38]. The second issue is closely related with the first issue: a well-established group of listeners of a single song or artist is needed to obtain good recommendations. This implies that 'niche' music, music that is not mainstream, is hard to capture well in social recommendation systems.

The counterpart of social music recommendation is content based music recommendation. Content based music recommender systems extract features from the audio signal and compare songs based on these extracted features. 'Similar' music will have a similar feature profile; thus recommendations can be made. Content based music recommender systems also have many drawbacks. Music perception is not purely physics, there are many factors influencing musical taste and similarity perception. The second issue that content based recommender systems have to deal with is computational complexity. An accurate description of a song requires a detailed analysis of the contents of the song. This involves a data extraction process in which the amount of extracted data can easily be more than a few megabytes per song. These extracted data can be modelled, but operations on detailed song models will be computationally complex.

This thesis aims at speeding up methods for *content-based* music recommendations for playlist generation systems. Recommendation algorithms are required to tackle the problems of navigating through large music collections. Speed is an issue especially on devices with limited resources and slim user interfaces.

1.2 Music Information Retrieval

This section provides a short description of the area of research this thesis contributes to: Music Information Retrieval. MIR is the research field of retrieving information from music. Since the mid 1990s the interest in this field has boomed [115], resulting in the first major conference in this field in

October 2000: ISMIR [116]. The following two sections present two different categoratisations of the MIR research area.

1.2.1 Seven facets of Music Information

In 2003, Downie [26] defines seven facets (classes) of music information: pitch, temporal, harmonic, timbral, editorial, textual, and bibliographic facets. MIR researchers use information from these classes to find new ways for interacting with or analyzing music. The facets provide a framework for categorizing the domain a certain music information feature relates to. In the following paragraphs, a short description of these facets is presented.

Pitch is defined as the fundamental frequency of a sound. It can be represented in *Hertz* (oscillations per second) or as one of 12 *semitones* (music) in an octave. A modern piano has 88 keys: seven octaves plus a minor third. Its lowest tone is A0 at 27.50 Hz, the highest is C8 at 4186.01 Hz. A sequence of tones is conceived as a melody.

The temporal facet describes musical events with respect to their timing. From a musical point of view, this facet can be dissected into tempo, meter, pitch duration, harmonic duration and accents. Within MIR also other temporal facets are taken into account. These include note onset time, duration of a complete musical piece or duration of logical elements within a piece.

Harmony is the combination of pitches or intervals. We distinguish between polyphony, the event that two or more pitches occur at the same instance, and monophonic intervals, single pitches played sequentially. The latter can be interpreted as implied harmony.

The fourth facet, timbre, describes tone colour. It enables us to distinguish between a note of certain pitch, played by different instruments. Instrumentation and playing style (bowing or staccato on a violin) thus have major impact on the timbre of a musical piece.

Written performance instructions, either in symbolic or in textual form, are accounted to the editorial facet. These instructions vary from notitions a composer makes on the dynamics of a piece, to a written-down solo of a musician in an orchestra.

The textual facet, song lyrics, completes the list of facets that can be retrieved from the audio signal.

Metadata describing music, such as artist, producer, song title, label, etcetera are all counted to the bibliographic facet. Traditional music search only uses these bibliographic data. Exploring other means for finding music, by incorporating the other six facets into a search, is an important issue in

MIR.

This thesis focusses on content-based playlist generation; playlist generation using only information one can directly extract from the audio signal. In the categorization of Downie, this involves the pitch, temporal, harmonic and timbral categories. In literature on content-based playlist generation, combinations of features from the different categories have been used (see chapter 2.4 for details). In this thesis, only timbral features are used.

1.2.2 Alternative MIR categorization

An alternative categorization of MIR research is possible using the music representation level. Futrelle [32] defines four categories and gives some example research topics for each category. This table is reproduced in Table 1.1. Note that the 'Visual' representation can also be represented as a subcategory of 'Symbolic', however, the field of research is very different.

Representation	**Description**	**Research**
Audio	Recordings, Streaming audio, Instrument libraries	Sound/Song spotting, Transcription, Timbre classification, Musical analysis
Symbolic	Notation (scores, charts), Event-based recordings (MIDI), Hybrid representations	Matching, Theme/Melody Extraction, Voice Separation, Musical Analysis
Metadata	Cataloging, Bibliography, Descriptions	Library testbeds, Traditional information retrieval, Interoperability
Visual	Scores	Score reading (optical music recognition)

Table 1.1: Music Representations and Research in MIR

The first three levels of representation can be perceived as semi-orthogonal data. The metadata level provides us with the descriptions of music one would use in a record store or one uses when talking about music in general terms. The symbolic layer adds the information that is required by a musician for

reproducing music without having to know anything else about the song. The audio representation level finally adds the personal interpretation of the performing musician. The word 'semi' was used, since allthough the information is from different domains, the three are tightly interconnected. Using two of the three, a trained person can deduce a lot of information of the third dimension.

The three dimensions of music; audio, symbolic and metadata, all describe the *content* of a song. Following the classification of Futrelle, the *content based playlist generation* this thesis deals with, is positioned in the audio representation domain. Within the audio domain, only timbral information will be used.

1.3 Problem definition and thesis structure

As described in Section 1.1, content based music recommendation systems have major advantages over social recommender systems when it comes to exploiting and exploring large personal or shared music databases in case not all music is mainstream. Much research has been done on improving the quality of content based music recommender algorithms. This research included many facets of these algorithms. The use of new features for providing extra information about the audio domain of the music was explored. Different data modelling techniques and feature transformations were analyzed. A multitude of classification and evaluation methods was tried on music. Still, the major drawback of content based music recommendation is the lack of robustness and the computational complexity that is inherent to performing operations on high-dimensional, complex song models.

A factor that has major influence on the acceptance of almost any electronic device is the speed of operation and response. This theses focusses at finding methods to speed up personal music recommendations based on content based music analysis. The main research question is:

> Is it possible to reduce computational complexity of playlist generation systems at playlist generation time, without sacrificing quality and flexibility of the playlist generation method?

This question is answered by dissecting this question in three questions that address different aspects of the main research question. The first aspect is measuring the quality of a playlist. Currently, there is no known universal playlist quality criterium. Without a playlist quality criterium, playlist generation systems can only subjectively be evaluated. In order to objectively

evaluate playlist generation systems, the following question has to be answered:

> Is it possible to formulate a general playlist quality criterium?

The way a person interprets the quality of a playlist depends on many factors. In order to get insight in these factors and be able to answer this question, a user questionnaire is designed and performed. The results of the questionnaire are evaluated with respect to the question if there is a universal playlist quality criterium.

Once a playlist quality criterium is found, optimisations on the signal processing chain of playlist generation systems can be performed and the effect of these optimisations on the quality of the systems can be evaluated. In particular, two questions are asked:

> Is the current method for modelling songs using Gaussian mixture models appropriate?

and

> Is it possible to find compact song representations that allow song similarity calculations at playlist generation time?

Answering these three research questions involves a detailed analysis of computing music similarity, music similarity perception and music playlist generation. Chapter 2 presents an introduction to 'music similarity'. Several concepts of music similarity are described and methods for measuring audio similarity are presented. The chapter ends with an overview of the state of technology of music similarity research.

In chapter 3 the focus is on playlist generation. First, the main concepts and state of technology of this field of research are presented. This is followed by a presentation of the results of a user survey on user requirements on playlist generation systems that was performed in the scope of this thesis. Using the results of the survey, goodness measures for playlists are discussed. The last section of chapter 3 presents the playlist generation concept that was developed. This includes the two major contributions to the field of MIR:

1. Fast song model complexity estimation for individual songs allows each song to be represented with an appropriate sized statistical model.

2. Increasing robustness of song mappings in self organizing maps by using Gaussian mixture models instead of regular Euclidean vectors as codebook entries. The process of using Gaussian mixture models in self

organizing maps is new and has been applied for a patent.

The fourth chapter presents the architectural design of the developed framework for playlist generation. It describes the algorithms and heuristics for implementing the concepts presented in chapter 3.

In chapter 5, the developed system is evaluated. Three papers that were written present detailed evaluation of the developed concepts. It is shown that by using individual song complexity estimation, music similarity tasks can be speeded up without losing accuracy. Furthermore, it is shown that self organizing maps using Gaussian mixture models show good and robust self organization properties.

The last chapter provides conclusions and directions for further research.

Chapter 2

Music Similarity

2.1 Introduction

For one person, the difference between 'doom metal' and 'gothic rock' may be completely obvious, while a piano piece by Chopin and by Satie are all the same. For another person this may be the other way around. Perception of music similarity cannot be generalized into detail, for a group of people with a diverse musical taste. But what is musical taste or musical preference? Gembris [34] names four factors that influence our 'liking of a song': demographics, audio, media and associated feelings (see Figure 2.1). These factors can all serve as a basis to determine perceived musical similarity.

This chapter describes music similarity only in a conceptual and technical context. Section 2.2 presents a few aspects of music similarity based on music theory, music perception psychology, and musical timbre. This section is followed by an overview of how music similarity can be measured based on audio signal analysis. In Section 2.4, several state-of-the-art algorithms that provide different kinds of music similarity measures are presented.

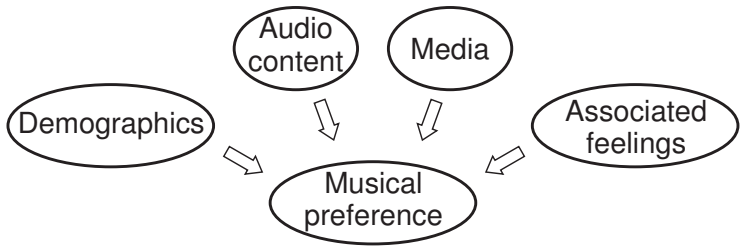


Figure 2.1: Factors influencing musical preferences (Gembris [34])

2.2 Concepts of Music Similarity

The following subsections provide a summary of music similarity concepts that are relevant in the context of this thesis. It does not provide a complete overview of all aspects of music similarity. Detailed information can be found e.g. in [43] [47] and [93].

2.2.1 Music Theory

Music theory is a field of study that investigates the principal elements of music. The elements commonly agreed upon are *harmony*, *melody*, *rhythm*, *texture* and *structure*. In the following paragraphs, the most important features of these elements and some basic similarity measures on each individual element are presented.

Harmony

In this thesis, the word *harmony* is used to describe the event of two or more tones of different *pitch* sounding simultaneously. According to this definition, harmony can only occur in polyphonic music: music with multiple voices sounding at the same time. Monophonic music, music with only one single voice at a time, can have an *implied harmony*. The human brain builds relationships between successively sounding notes and interprets these as harmony.

Three or more notes form a *chord*. The most elementary chords consist of three notes: one base tone, its major or minor third and the perfect fifth. *Chord progressions*, a sequence of chords, provide the core of harmony, and a structure for the melody of each song. The chord sequence can be normalized to the central key (tonality) of a song, to encompass for transpositions. Once the sequence of chords of a song is known, this could be used for querying a (song) database for songs with similar chord sequences. This process can be used for e.g. music cover detection.

Melody

Where harmony is defined as the simultaneous occurrence of tones, *melody* is defined as a series of musical events, of which both pitch and duration of the single notes may vary. Throughout the centuries, different tonal scales have been used. In western classical music until late 19^{th}century, a seven note scale (the *diatonic* scale) was most common. Later western music, including popular music, uses almost only the 12 tone *chromatic* scale.

A song consists of one or more melodies or melodic phrases that are repeated throughout the entire song. In this structure of repetitions we can find patterns that are characteristic for different musical styles. Pop and rock music usually consist of multiple *verses*, separated by a *chorus* whereas classical music often has one central theme followed by variations around this theme.

Rhythm

Along with melody, *rhythm* is one of the fundamental components of music. Rhythm is defined as the variation of the length and the *accentuation* of musical events. Early forms of music consisted only of rhythmical patterns, performed by clapping, shouting or drumming.

Western music rhythms are often *divisive*, each measure is divided in logical groups and a natural accentuation of the pattern occurs. This pattern repeats itself over several measures. A good example of this is the traditional waltz, where the first note of the measure is heavily emphasized.

The opposite of divisive rhythms are *additive* rhythms. Additive rhythms consist of series of patterns that are not repetitive. No regular natural accentuation of notes occurs.

Texture

The quality of a sound or a piece called *texture* is influenced by the relationship between and the number of voices, the instrumentation of these voices and the harmony of a musical piece. Texture is commonly expressed in terms as 'light', 'open' or 'raw'. Composers or artists choose to use a certain instrument to create the texture that contributes best to the expression of the feelings or moods of a certain song.

Structure

The preceding elementary music elements described the ingredients (the alphabet) a composer has to form music. The *structure* of a piece can be interpreted as the grammar, needed to formulate a story from the individual elements of the alphabet.

Typical western classical musical forms, such as the fugue, sonata, or invention consist of an exposition, development, and a recapitulation. These elements may optionally be preceded by an introduction and succeeded by a coda.

Modern western music structure is usually structured around its lyrics. A song consists of multiple verses with very similar melody, but different lyrics, connected by chorusses, with a contrasting melody and the same lyrics. An optional third element is an interlude, or bridge. It is used to connect two parts of a song, which are for instance in a different tonality.

2.2.2 Timbre Similarity

The term *timbre* is used to describe 'the quality or set of qualities that allows a listener to identify the physical source of a sound' [105]. This identification process can be divided in a preprocessing step, where the actual audio signal is converted to a series of nerve impulses in the inner ear, and a postprocessing step of matching these impulses with already known sounds. Timbral similarity measures developed by MIR researchers determine similarity largely on this first preprocessing stage. This section elaborates on the preprocessing that occurs in the human auditory system and on research that has been done on the 'elementary dimensions' of timbre.

Auditory Preprocessing

The human ear, the sensory organ for detecting sounds, consists of three parts [125]. The *outer ear* guides the sound waves to the tympanic membrane, the outer part of the ear drum. The *middle ear* contains the ear drum, performing an impedance matching for transmitting the air-guided acoustic waves to the fluid-filled *inner ear*. The inner ear consists of two organs, the vestibular system for sensations of balance and motion and the cochlea for hearing. The cochlea is a snail-formed organ consisting of two chambers, pictured in Figure 2.2. These chambers are separated by the *basilar membrane*, on which the *organ of Corti* is located. This organ contains between 15.000 and 20.000 auditory nerve receptors, each having its own *hair cell*. The hair cells are arranged in four rows, the outer three functioning as a mechanical pre-amplifier, and the inner row for the actual reception. Depending on the position on the basilar membrane, the hair cells recept frequencies between 16 Hz and 20 kHz, dependent on their position on the basilar membrane.

Perception of Pitch The cochlea acts as a spectrum analyzer, every receptive hair cell neuron transmits information on its current activation. Like the receptors in the human eye, this activation is a continuous, and not a discrete value.

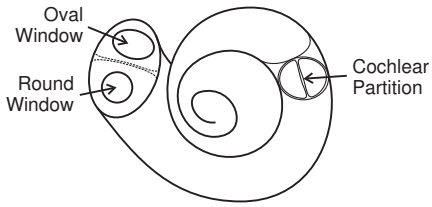


Figure 2.2: Schematic drawing of human cochlea, after [125]

There are two main model types for pitch perception, *place models* and *temporal models*. Place theory (e.g. [35]) assumes that the excitation pattern on the basilar membrane is related to the pitch. Based on statistical models of the structure of harmonics of an individual tone, the pitch or pitches are determined. Temporal theory (e.g. [63]) models the basilar membrane as a bank of linear filters. These filters resonate on a frequency which is dependent on their position and the waveform periodicity that occurs at the membrane. These excitation patterns are again analyzed by pattern recognition processes.

Frequency dependency The perceived loudness of a pure tone is not only dependent on the audio level, but also on the frequency of the tone. The unit for expressing perceived loudness is the *phon*, it relates the sound pressure level (dB) to perceived loudness. Fletcher and Munson [29] have determined the equal loudness curves as presented in Figure 2.3. At 1 kHz the phon scale is equal to the decibel scale. For other frequencies, subjects were asked to adjust the loudness of the signal until it was perceived to have equal loudness as the 1 kHz signal. It can be seen that the human ear is most sensitive in the range from 1000 to 5000 Hz, approximately corresponding to the frequencies that are most important for human speech. The zero-phon level indicates the average human hearing threshold in quiet; tones at lower intensity are inaudible.

Auditory masking The general effect of influencing, or even suppressing tones by other tones is called auditory masking. Tones above the hearing threshold in quiet, can become inaudible when the intensity of a nearby tone is large enough. In Figure 2.4, masking threshold levels are plotted for various masker intensities. It can be seen that for larger intensities, the upward spread of the masking effect becomes greater than the downward spread.

Two kinds of masking are distinguished: simultaneous masking and temporal masking. Simultaneous masking is when the masker and the masked signal occur simultaneously. Temporal masking occurs just after the mask-

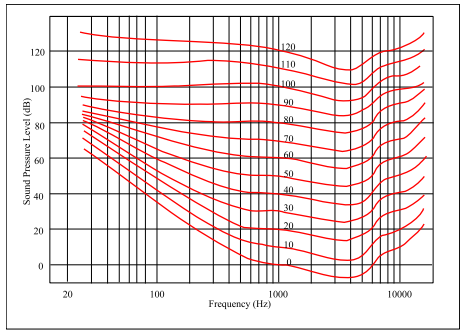


Figure 2.3: Phon scale: perceived loudness as function of frequency and intensity level, after [29]

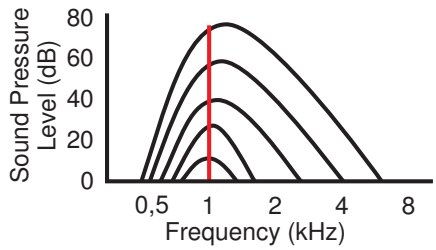


Figure 2.4: Frequency masking thresholds for various masker intensities, after [33]

ing signal is switched off (*forward masking*), or right before it is switched on (*backward masking*). Forward masking effects endure approximately 50 milliseconds, backward masking approximately 10 milliseconds. These effects may originate from the temporal integration of sound in the human brain.

Critical bands Although the frequency resolution of the human ear is very high (the difference between a pure tone of 440 and 441 Hz can be heard), this performance cannot be reached when perceiving simultaneous tones. When two simultaneous tones have approximately the same frequency, the two tones are perceived as one single tone with an amplitude modulation with a frequency equal to the frequency difference of the two tones (beating). The critical bandwidth can be determined by increasing the frequency difference of the two tones, until two separate tones are perceived. This effect can be explained by local vibrations on the basilar membrane. The length of the vibrating area on the membrane influences the bandwidth of the critical band. Again, the bandwidth is frequency dependent.

Timbre Dimensionality

The preceding paragraphs present the elementary functional blocks for reception of sound and two phenomena influencing the perception of multiple sounds. All timbral features used in MIR applications mimic (parts of) the human auditory system, using approximations of these functional elements as their building blocks.

As recognition of emotions in voice, or affection with certain sounds is presumed to rely largely on timbre perception processes, much research has been done on what the basic dimensions of timbre are (e.g. [69, 110]). The most common approach used in studies on timbre dimensions relies on multidimensional scaling (MDS) of similarity ratings. A number of listeners is asked to judge similarity of sound fragment duplets or triples. The sound fragments are either recordings of 'natural' sounds, or synthesized sounds. The similarity data are used to construct a distance matrix that is analyzed by a MDS algorithm. The MDS algorithm tries to find a space that can best be used to map the sounds and their distances to. Timbral features should at least correlate with the basic dimensions of timbre found in these studies.

Caclin et al. [18] present a confirmatory study on research that has been done thus far. Experiments with synthesized sounds of approximately 500 ms were performed. The parameters that were varied were attack time (from onset to the moment of maximal amplitude), spectral center of gravity (brightness),

spectral flux and spectral irregularity (attenuation of even harmonics), which were found to be the most important dimensions in other literature. Caclin et al. [18] confirm attack time, spectral center of gravity and spectral irregularity to be the main dimensions of timbre. Spectral flux did not influence similarity judgments significantly.

2.2.3 Perceived Similarity

Perception is the process of acquiring, interpreting, selecting and organizing sensory information. Music similarity perception is studied in the field of cognitive psychology. In this thesis, the results of a user survey on musical preferences is presented (Section 3.4). A main theme of this user survey is perception of music similarity. The following paragraphs provide a framework of concepts for perceived similarity. Two main concepts of similarity as defined in cognitive psychology are presented; the principle of *prototype theory* and *concept-based classification*. Furthermore, the concept of *integrative listening* is presented.

Prototype theory

The *prototype theory* [99] is considered to be one of the most influential theories for perceptual equivalence. Within the theory of perceptual equivalence, similarity between two objects is defined as the weighted sum of (sub-)similarities on distinctive features describing the objects. Similarity judgments are personal and are not necessarily symmetrical; the Rolling Stones may be perceived more similar to the Beatles than the other way around. This is because the weighting of features for the object similarity depends on the relative salience of referent and subject.

Prototype theory is a theory for categorizing objects to *exemplars*, prototypical examples, of a certain category. These exemplars are not necessarily real members of the category, they can also be abstract objects that have the feature properties that are most representative for all objects in that category.

In [100], the prototype theory is extended to hierarchical categorizations. A typical example for a prototype taxonomy is given in Figure 2.5. The basic level (chair, table) has superordinate and subordinate levels. The basic level is most accessible, superordinate levels are more abstract and subordinate levels show only minor variation on the basic level. Defining what taxonomy level is the basic level, is a somewhat arbitrary process. From a information theoretic point of view, one would describe the basic level as the categorization that minimizes mutual information between the categories.

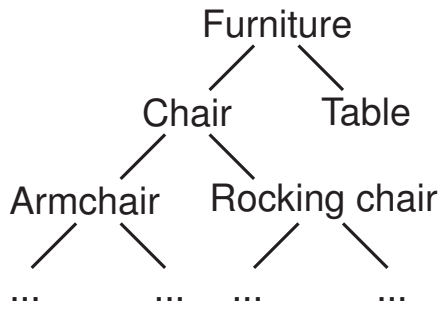


Figure 2.5: Typical taxonomy for prototype theory

A prototype taxonomy differs from *definition based* taxonomies in the fact that the distinctive features have non-binary values.

Concept-based Classification

Categorizations as found within prototype theory are feature-based. While this is a very natural approach, not all objects can be classified by just observing feature measurements.

The *concept*, or *theory-based classification* approach [71] suggests using abstract concepts for classification (e.g. musical genre), not feature based similarities. In contrast to the prototype approach, concept classification requires background knowledge, and priorizes the role of context in similarity relations. It should be noted, that often, relationships between feature- and theory-based similarities exist. These relationships can be learned while gaining experience with the particular domain (perceptual learning).

Integrative Listening

Rauhe et al. [96] presents the concept of *integrative listening*, the integration of *unconscious* and *conscious* music reception. Music similarity can be expressed as the weighted sum of similarities within conscious and unconscious listening processes. Rauhe defines six categories for music reception. Unconscious listening is subdivided in *diffuse reception*, *motoric-reflective reception* and *associative-emotional reception*. The conscious processes are *empathic reception*, *structural reception* and *subject-oriented reception*. These categories are briefly described in the following paragraphs.

Diffuse Reception The diffusive listening mode applies to listening to music as background music. The music is non-obtrusive, reception takes place at

the elementary level of *secondary* and *tertiary* sound components. The category of secondary components is formed by timbral facets. Typical tertiary components are special singing effects such as shouting or scatting. Also general sound effects such as reverb, compression or electronic sound effects are counted to this category.

A typical example of music designed for diffuse reception is the so-called Muzak or 'elevator music'. Lanza [46] presents a history of this musical genre.

Motoric-reflective Reception Triggered by rhythmic elements in music, the recipient's motoric sensory system is activated. This results in spontaneous body movements, such as tapping along with the music, or wobbling on a chair.

Associative-emotional Reception Since music has become an omnipresent component of our daily life, music gets associated with special experiences, memories and emotions. Rauhe et al. [96] distinguishes five types of associations:

- Thematic associations, with specific words or lyrics.
- Situational associations, where the music listened to during significant, meaningful situations, impressed the listener.
- Secondary and tertiary musical component associations; components also present in ones favorite music triggers associations.
- Association with the social image of the artist.
- Location and time-bound associations, for instance dance music reception in a disco.

Empathic Reception The conscious counterpart of associative-emotional reception is empathic reception. The empathic listening mode is characterized by the active process of feeling one's way into the music. Where the associative-emotional process was triggered by individual facets of the music, the empathic reception is oriented on the musical piece as a whole.

Structural Reception The process of dissecting music in its structural components and capturing the musical structure as a whole is described as structural reception. This form of reception distinguishes between structural-analytic and structural-synthetic listening. The structural-analytic listening

mode focusses on the perception of structural details, single structural elements and structural levels, where the structural-synthetic mode describes the process of placing relationships between the observed elements.

Subject-oriented Reception Subject-oriented reception is the process of projecting one's own emotions, opinions or memories on that music in which the listener recognizes his own personality. Subject-oriented reception is also called mirroring-, or projective listening.

2.2.4 Cultural Similarity

Within the MIR community, cultural similarity is understood as similarity, defined by a listener or a collection of listeners. Cultural similarity thus cannot be captured by the audio signal itself, but only by ratings (descriptions, comparisons to other music) of listeners.

The following paragraphs will briefly describe methods for investigating *musical preference*, procedures for determining *web-based similarity* and *lyrics similarity*. The last paragraph contains some critical remarks on generalizing personal similarity judgments.

Musical Preference

The ultimate goal of music recommender systems is to recommend music that matches the user's musical preference. It is assumed that a user's musical preference consists of one or more clusters of 'similar music'. In the 1920s, the early years of radio, the first marketing-driven research on musical preferences was performed. Over the years, a broad range of research methods have been explored. Gembris [34] summarizes three major aspects of research, which are briefly described in the following paragraphs.

Behavioral and Verbal Preferences Behavioral music preferences are those preferences people indicate, when actually listening to a piece of music. The common procedure for studying behavioral preference is a two step procedure: first, a 15-60 second excerpt of the music is played, after which the person is asked to judge the music with respect to a certain dimension.

The counterpart of behavioral preference studies are studies on verbal preference. The test person is not directly judging audio, but asked for preferences on certain aspects of music for instance by completing a questionnaire or in a interview. Verbal preferences concern preference for the notion of musical style or genre. Gembris [34] points out, that behavioral and verbal preference often

do not correspond. The given explanation is that the verbal preferences are coined by social influences; these 'open' preferences often represent a socially desired musical taste. The behavior preferences do not suffer from these social demands and thus represent more 'private' preferences.

Pen and Paper Methods When questioned for behavioral or verbal preferences, judgments are usually retrieved using lists with questions, scaled-ratings or with semantic differentials (e.g. good - bad, exciting - boring). Earlier research has made use of longer lists of adjectives where the test person could select one or more items from the list, describing the music best. This approach was later replaced by using pictorial scales, using icons to express how the music is judged with respect to a certain aspect. The pictorial scales were preferred over the adjective lists by the test persons, and lead to higher test-retest reliability [34].

Technical Methods The pen and paper methods just described, do not allow for instantaneous feedback on the music. Various devices have been realized to allow immediate reactions. Early devices worked like strip chart recorders; the state of push or shift buttons was recorded on a paper roll, running synchronized with the music. Newer devices read sensor values and process them in a computer. These approaches allow parallel processing of group-feedbacks.

Web-based similarity

One important source of independent, subjective opinions about music is the internet. There are many sites with music journalists writing about music, weblog entries about visited concerts, and online shops with music reviews and recommendations. These descriptions are easily available and time-aware [122]. There have been different approaches for inferring similarity relations using web-data. The main principles of these approaches will be described in the following paragraphs.

One of the early works on web-based similarity is by Pachet et al. [75], who performed a co-occurrence analysis of individual tracks on compilation CDs and in radio playlists. The main assumption behind this kind of similarity metric is that either the producer of the CD, or the DJ of the radio station make a conscious decision of which music to put in what order. The co-occurrence data were used in a hierarchical classificator to cluster 'similar' tracks. The perceived correctness of these clusters was assessed by 'five persons with a good

knowledge of music'. The clusters found by analyzing compilation CDs were found to be more meaningful than the approach using radio playlists.

Later work on web-based similarity focusses on artist similarity. Baumann and Hummel [12] use a similarity measure based on algorithms often used in text mining applications; the *term frequency - inverse document frequency* (TFIDF) measure applied to a list of unigram terms, bigram terms, noun phrases and adjectives found on web documents with the artists' name. The TFIDF term is calculated for each term in the list:

$$\text{TFIDF} = \text{TF} \cdot \log_{10}(n/\text{DF}) \tag{2.1}$$

with TF the term frequency in a set of documents known to be related to the specific artist, DF the document frequency, counting the number of documents with occurrences of a term in the entire document collection used for the evaluation and n the number of documents taken into account for the evaluation. The TFIDF scores for each term are aggregated in a single vector per artist. Artist similarity can then be calculated by, for instance, the cosine similarity measure on the artist TFIDF vectors. The cosine similarity measure between two vectors is expressed as:

$$\text{similarity}(\boldsymbol{A}, \boldsymbol{B}) = \cos(\Theta) = \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{\|\boldsymbol{A}\|\|\boldsymbol{B}\|} \tag{2.2}$$

with $\boldsymbol{A}$ and $\boldsymbol{B}$ two vectors and Θ the angle between them.

Ellis et al. [27] investigated the suitability of different web-based similarity measures as ground truth for artist similarity. Among the evaluated measures was an Erdös distance (named after the distance measure for in how many hops mathematicians have co-authored with the Hungarian mathematician Paul Erdös), based on the 'similar artist' recommendations from the All Music Guide[1]. Other measures included a co-occurrence analysis of artists in personal collections available through peer-to-peer networks and the TFIDF measure as presented above. These measures were compared with a similarity ground truth obtained from a large-scale online artist similarity experiment. The Erdös measure proved to be the best match to the ground truth data, however, the concept of a single, general ground truth is put in doubt by Ellis et al.; the subjective author similarity ratings in the online experiment shows great variance per user.

[1] www.allmusic.com

2.3 Measuring Music Audio Similarity

Perrott and Gjerdingen [88] have found that college students where capable to classify a piece of music quite accurately in a 10-class genre taxonomy, while only listening to an excerpt of 250 ms. This is "fundamentally inexplicable with present models of music perception" [105] and justifies the statement that, even in very short timeframes, the spectral content (or 'audio surface') of the audio signal contains enough information for genre classification. In these short timeframes, there can be no complex rhythmic information.

The number of features that are currently being used for measuring music similarity is increasing. These features can be divided in three classes: *low-*, *mid-* and *high-level* features.

Low-level features are extracted directly from the audio signal. The audio is divided in short timeframes, in which the audio signal can be approximated as being stationary. The framelength is usually chosen between 10 and 50 ms. Another approach is feature extraction on a segmented audio stream [121], where segmentation is performed on for instance note onsets. Because of the stationarity of the frames, only information on the audio surface or timbre can be extracted. Low-level features do not capture temporal information.

Mid-level features capture information by observing a longer sequence of short timeframes, usually low-level features. By performing statistical operations or mathematical transformations on the sequence, information on the dynamics or structure of the audio signal is extracted. Common observation lengths range up to 5-10 seconds.

In contrast to low- and mid-level features, high-level features have a *semantical* meaning. High-level features can not be obtained from the audio signal without using 'intelligence', either by using machine learning algorithms or by human interaction. Low- or mid-level features can be mapped to semantic meaningful 'anchors' Berenzweig et al. [13].

In this section, a selection of frequently used low- and mid-level features are presented. This overview gives an impression of what kinds of information can be extracted from the audio signal in the timbral domain. Techniques for generating statistical models describing the feature data and algorithms to measure distance between model instances are briefly introduced. Furthermore, the most common classification schemes used for genre classification tasks are presented.

2.3.1 Timbral and Textural Features

Most low-level features are timbral or textural features. They describe the frequency distribution characteristics in a short timeframe. In the following paragraphs, a small set of commonly used timbral features is presented. The features using a FFT typically use timeframes of 20 miliseconds and 256 (equally spaced) frequency bins.

Spectral Centroid

The spectral centroid (SC) feature is defined as the center of gravity of the magnitude spectrum of the short-time Fourier transform. It is a measure of the *brightness* of an audio signal and is defined as:

$$SC = \frac{\sum_i iS(i)}{\sum_i S(i)} \tag{2.3}$$

with S the modulus of the discrete FFT spectrum and i the index of the FFT frequency bins.

Spectral Roll-Off

The Roll-Off point is the frequency SRF, below which a certain percentage of the power of the spectrum resides. Common values for this percentage range from 80 to 95%.

$$\sum_{i=1}^{SRF} S(i) = 0.85 \times \sum_{i=1}^{N} S(i) \tag{2.4}$$

with S the modulus of the discrete FFT spectrum, i the index of the FFT frequency bins and N the number of frequency bins.

Spectral Flux

The spectral flux (SF) is a measure of the rate of change in the spectral shape on a frame-by-frame basis.

$$SF = \sum_{i=1}^{N} \left(S_t(i) - S_{t-1}(i)\right)^2 \tag{2.5}$$

with S the modulus of the discrete FFT spectrum, i the index of the FFT frequency bins, N the number of frequency bins and t the index of the current timeframe.

Zero Crossings Rate

The zero crossings rate (ZCR) is a measure of the noisiness of a signal. It is defined as the number of sign-changes per time unit [114]. The ZCR feature is correlated with the pitch of a signal as follows from applying the ZCR on a single sinewave signal.

Mel-scale Frequency Cepstral Coefficients

The Mel-scale is a nonlinear mapping between frequency and perceived pitch, based on the human auditory system. For frequencies up to around 1 kHz, pitch and frequency are perceived in a approximately linear manner, and above approximately logarithmic. At 1 kHz, the Mel-value is 1000. The mapping between Mel-scale and frequency is given by:

$$m = 1127.01048 \ln(1 + f/700) \tag{2.6}$$

with m in Mel and f the freqency in Hertz.

Experiments in voice-recognition research showed that using this Mel-scale for speech recognition systems yields better results than using linear scale (e.g. [127]). The process of calculating the MFCCs consists of the following steps:

1. Convert the signal to timeframes.

2. Convert each timeframe to the frequency domain using an STFT.

3. Take the Log of the amplitude spectrum.

4. Apply Mel-spaced triangular filters to determine the power in Mel-spaced frequency intervals.

5. Decorrelate the filter power values using a discrete cosine transform.

This process is repeaded for each timeframe. The discrete cosine transform is an approximation of the Karhunen-Loève-transform for independent component analysis. Logan [52] has studied the effects of the DCT and Mel-scaling for modeling music and concluded that the DCT is an appropriate approximation of the Karhunen-Loève-transform. Terasawa et al. [110] have investigated perceptual similarity and conclude that the MFCC vectors account for 66% of the perceptual variance of similarity judgments.

Spectral Contrast Feature

The MFCC coefficients only contain information on the average signal energy in each Mel-frequency band. An alternative measure is the spectral contrast feature [23, 120], where instead of average band power, the power difference between power peak and power valley are part of the feature. Instead of a Mel-frequency scale, an octave based scale over six octaves is used. For each band, the spectral contrast and the power value are returned. The SCF is calculated as follows: First, the energy of each of the six sub-bands of the audio signal is determined in the form of a vector $\{x_{j,1}, x_{j,2}, \ldots, x_{j,N}\}$, with j the sub-band index. The elements in this vector are then sorted *into descending order of magnitude*. Then

$$\mathrm{P}_j = \log\left(\frac{1}{\alpha N}\sum_{i=1}^{\alpha N} x_{j,i}\right) \tag{2.7}$$

$$\mathrm{V}_j = \log\left(\frac{1}{\alpha N}\sum_{i=1}^{\alpha N} x_{j,N-i+1}\right) \tag{2.8}$$

$$\mathrm{SCF}_j = \mathrm{P}_j - \mathrm{V}_j \tag{2.9}$$

where i is the total number of FFT bins in a single frequency band and α is a 'neighboring factor' with value between 0.02 and 0.2. A signal with a high spectral contrast is likely to represent a signal with a high degree of localized harmonic content. Signals with a low spectral contrast will thus have a lower degree of harmonic content and a higher degree of noise components.

Pitch Class Profile

The Pitch class profile (PCP) [31] is the most widespread feature for harmony and chord detection algorithms. The PCP defines a mapping from frequency spectrum S to a chromagram. A chromagram (or PCP) represents the likelihood of chroma occurences in an audio signal.

$$\mathrm{PCP}(b) = \sum_{m=0}^{M-1} |S(b + mB)| \tag{2.10}$$

with $b = 1, 2, \ldots, B$ the chromagram bin index and M the number of octaves. B is chosen to be equal to the number of pitches in a tonal system, for western music $B = 12$.

2.3.2 Data Modeling

Extracting one low-level feature vector of reasonable dimension (e.g. a 15-dimensional MFCC vector) every 10 milliseconds results for a song of average length in 15000 samples. Manipulating this amount of data, or comparing two songs by determining the distance between all raw song features, is computationally very expensive. Instead of working on the raw features, it is also possible to first create a statistical model that represents the original data, and then work with the statistical model for further processing.

The following presents the two models that are most commonly used in MIR systems: the *K-Means Model* and the *Gaussian Mixture Model*. After a short description of what these models are, two modeling approaches that can be used for fitting a model in a dataset are described.

k-Means Model

A dataset of samples with low variance, generated around a single mean value $\boldsymbol{\mu}$ can be efficiently described with just the mean value of all samples. A k-means model describes a dataset $\mathcal{S}$ in terms of k weighted mean values. Each mean $\boldsymbol{\mu}_i$ is a prototype for the data it represents. There is a wide variety of approaches available for estimating a good value for k, an overview of these methods is presented in Section 4.3.2.

k-Means modelling

A k-means model (kMM) can be fitted to a dataset $\mathcal{S}$ using either a batch k-means or an online k-means algorithm. A set of k random prototype vectors are selected from the dataset. For the batch algorithm, the prototype vectors are then updated as follows:

$$\Delta\boldsymbol{\mu}_i = \sum_j \epsilon(t)(\boldsymbol{x}_j - \boldsymbol{\mu}_i) \qquad \text{if } \boldsymbol{\mu}_i = \boldsymbol{s}_j \tag{2.11}$$

with $\boldsymbol{s}_j$ the closest prototype vector for sample $\boldsymbol{x}_j$. $\epsilon(t)$ is a monotonic decreasing function of t determining the learing rate of the model. t increases with each complete iteration. This procedure is repeated until a stopping criterium is fulfilled.

The online algorithm has the same principle, but adapts the position of the prototypes after each sample:

$$\Delta\boldsymbol{\mu}_i = \epsilon(t)(\boldsymbol{x}_j - \boldsymbol{\mu}_i) \qquad \text{if } \boldsymbol{\mu}_i = \boldsymbol{s}_j \tag{2.12}$$

It was shown in Bottou and Bengio [15] that the online algorithm converges faster at the first iteration steps, while the batch algorithm has better convergence properties. The fast convergence of the online algorithm can be attributed to redundancies in a dataset; only a small part of the data is enough to yield accurate first model estimates. The worse convergence properties result from stochastic noise resulting from randomly drawing samples.

Gaussian Mixture Model

A Gaussian Mixture Model (GMM) describes the density of a dataset $\mathcal{S}$ in feature space $\mathcal{X}$ as the weighted sum of k Gaussian probability density functions. The probability of sample $\boldsymbol{x}$, given a Gaussian mixture model can then be expressed as:

$$p(\boldsymbol{x}) = \sum_{i=1}^{k} \alpha_i \mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{2.13}$$

$$\mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}_i|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i)\right) \tag{2.14}$$

where $\boldsymbol{\mu}_i$ is the mean vector and $\boldsymbol{\Sigma}_i$ is the (positive definite) covariance matrix of the i^{th} Gaussian in the mixture model.

Estimation Maximization Algorithm

When the number of components in a mixture is known in advance, the Expectation Maximization (EM) algorithm [25] provides an efficient method to estimate the parameters of the distribution of $\boldsymbol{n}$ data samples $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. The EM algorithm is an iterative procedure and is guaranteed to converge [126] to a local maximum of the maximum (log-)likelihood estimate of the mixture model parameters $\boldsymbol{\Theta}$:

$$\hat{\boldsymbol{\Theta}}_{\text{ML}} = \arg\max_{\boldsymbol{\Theta}} \left(\log p\left(\mathcal{X}|\boldsymbol{\Theta}\right)\right) \tag{2.15}$$

Each iteration consists of two steps:

- **E-step:** Assign each sample to the mixture component that is most likely to have generated the sample, based on the current estimate of the model parameters.
- **M-step:** Recompute the model parameters based on the current sample membership estimation.

These steps are repeated until convergence of the likelihood estimate or until a maximal number of iterations has been performed. The set of model parameters that is found using the EM algorithm describes the distribution of the samples $\mathcal{X}$. When $\mathcal{X}$ is a set of extracted features of a song, a set of $\hat{\boldsymbol{\Theta}}_{\mathrm{ML}}$ for different songs can be used for determining similarity between songs as described in section 2.3.4.

2.3.3 Distance Measures

Distance between a sample and a model instance

During the process of fitting a model on a dataset $\mathcal{S}$ in feature space $\mathcal{X}$, the distance between each sample and the model clusters have to be calculated. For this process (also called training), there is a variety of distance measures available. A limited subset of distance measures suitable for this training process is presented here.

Euclidean Distance The Euclidean distance is the simplest distance measure and can be used k-means models. It can also be used for Gaussian mixture models when discarding the covariances and treating the GMM like a kMM. It calculates the metric distance between $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$D_{\mathrm{E}}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.16}$$

with n the dimensionality of $\boldsymbol{x}$ and $\boldsymbol{y}$. The Euclidean distance is easy to compute but does not take the distribution of the data into account.

Mahalanobis Distance The Mahalanobis distance [57] is a distance measure that takes the *global* distribution of the data into account. The distance between datapoint $\boldsymbol{x}$ and a cluster in the dataset with mean $\boldsymbol{\mu}$ is given by:

$$D_{\mathrm{M}}(\boldsymbol{x}, \boldsymbol{\mu}) = \|\boldsymbol{x} - \boldsymbol{\mu}\|_{\boldsymbol{\Sigma}_{\mathcal{X}}} = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}_{\mathcal{X}}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})} \tag{2.17}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of the entire dataset. Note that if $\boldsymbol{\Sigma}$ equals the identity matrix, the Mahalanobis distance simplifies to the Euclidean distance.

Gaussian Mixture Distance It is proven in Li and King [48] that the Mahalanobis distance is the optimal distance measure for estimating $\boldsymbol{\Sigma}$ for distributions where the covariance in all clusters is equal to the covariance of

the entire dataset. For most data this is not the case. Li and King show that an optimal covariance $\mathbf{\Sigma}_{\mathcal{C}}$ can be chosen for each sample $\boldsymbol{x}$, for distributions with non-uniform covariance matrices is given by:

$$\hat{\mathbf{\Sigma}}_{\mathcal{C}} = \left(\sum_{i=1}^{k} P(i|\boldsymbol{x}) \mathbf{\Sigma}_i^{-1} \right)^{-1} \tag{2.18}$$

where

$$P(i|\boldsymbol{x}) = \frac{\alpha_i \mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}_i, \mathbf{\Sigma}_i)}{\sum_{i=1}^{k} \alpha_i \mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}_i, \mathbf{\Sigma}_i)} \tag{2.19}$$

with k the number of clusters. The distance measure then becomes:

$$D_{\mathrm{GM}}(\boldsymbol{x}, \boldsymbol{\mu}) = \sqrt{(\boldsymbol{x} - \boldsymbol{\mu})^T \hat{\mathbf{\Sigma}}_{\mathcal{C}}^{-1} (\boldsymbol{x} - \boldsymbol{\mu})} \tag{2.20}$$

The quality of the estimated $\hat{\mathbf{\Sigma}}_{\mathcal{C}}$ strongly depends on the correct choice of the number of Gaussians k in the model.

2.3.4 Distance between model instances

Gaussian Mixture Models

A common distance measure between two Gaussian distributions is given by the Kullback-Leibler (KL) divergence [45]. For two discrete variables with probabilities P and Q, the KL-divergence is given by:

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{2.21}$$

A closed form expression of the KL-divergence for normal (Gaussian) distributions [87] is:

$$D_{\mathrm{KL}}(P\|Q) = \frac{1}{2} \log \frac{|\Sigma_q|}{|\Sigma_p|} + \frac{1}{2} \mathrm{Tr}\left(\Sigma_q^{-1} \Sigma_p\right) + \frac{1}{2} (\mu_p - \mu_q)^T \Sigma_q^{-1} (\mu_p - \mu_q) - \frac{d}{2} \tag{2.22}$$

The KL-divergence is often used as a distance measure between probability distributions. For distance measures, symmetry is a desired property. The KL-divergence is asymmetric by definition, but can easily be symmetrized:

$$D_{\mathrm{KL_S}}(P\|Q) = D_{\mathrm{KL}}(P\|Q) + D_{\mathrm{KL}}(Q\|P) \tag{2.23}$$

The closed form expression of this symmetrized KL divergence thus can be expressed as:

$$\begin{aligned} D_{\mathrm{KL_S}}(P\|Q) =& \frac{1}{2}\mathrm{Tr}\left(\Sigma_q^{-1}\Sigma_p + \Sigma_p^{-1}\Sigma_q\right) \\ &+ \frac{1}{2}(\mu_p - \mu_q)^T\left(\Sigma_q^{-1} + \Sigma_p^{-1}\right)(\mu_p - \mu_q) - d \end{aligned} \tag{2.24}$$

Unfortunately, there is no closed form expression of the KL-divergence for GMMs. However, there do exists methods to determine similarity between GMM instances. A selection of frequently used GMM distance measures is described in the following paragraphs.

Centroid Distance The centroid distance is the simplest distance measure for mixture models. It reduces the Gaussian mixture model to one single point, the centroid. This centroid is the weighted mean of all clusters:

$$\boldsymbol{\mu}_{\mathrm{centroid}} = \sum_{n=1}^{N} w_n \cdot \boldsymbol{\mu}_n \tag{2.25}$$

The centroid distance is then calculated as the Euclidean distance between the centroids of two Gaussian mixture models. Berenzweig et al. [14] show that this distance measure performs surprisingly good as a distance measure for a music similarity task.

Monte Carlo method The Monte Carlo method for determining GMM similarity is to draw random samples from one model instance and determine the likelihood of these samples in the other GMM. This method has been used successfully in Aucouturier and Pachet [4]. The authors use a symmetrized and normalized form:

$$\begin{aligned} D_{\mathrm{MC}}(P\|Q) =& \sum_{i=1}^{n} \log P(S_i^P|P) + \sum_{i=1}^{n} \log P(S_i^Q|Q) \\ &- \sum_{i=1}^{n} \log P(S_i^P|Q) - \sum_{i=1}^{n} \log P(S_i^Q|P) \end{aligned} \tag{2.26}$$

with n the number of samples. The major drawback of this method is that it is computationally expensive. The precision of the measure depends of n. For a reliable distance estimation, a minimum of 2000 samples is recommended.

Earth Mover's Distance The Earth Mover's Distance (Rubner et al. [101]) is a distance measure based on the minimal cost (amount of work), needed to transform one 'signature' into another. Signatures can be histograms (used by Rubner et al.) or any other data model where a distance measure between the individual components is defined. The principle of the EMD on histograms is illustrated in Figure 2.6. The cost of transforming the top histogram (the supplier) to the bottom histogram (the customer) can be expressed as the amount (flow) of 'earth' to be moved between supplier and customer times the distance of the transport. Let P and Q be the two histograms (or signatures) from Figure 2.6 with m the number of bins in P and n the number of bins in Q. The 'weight' of bin p_i is w_{p_i} and the sum of weights over all bins for each histogram equals one. The distance between p_i and q_j is $d_{p_i q_j}$, the flow is $f_{p_i q_j}$. The distance between P and Q can then be found by solving a minimization problem:

1. Minimize the cost function W as a function of the 'flow' f:

$$W = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{p_i q_j} f_{p_i q_j} \tag{2.27}$$

under the constraints:

$$\begin{aligned} f_{p_i q_j} &\geq 0 && 1 \leq i \leq m, 1 \leq j \leq n \\ \sum_{j=1}^{n} f_{p_i q_j} &\leq w_{p_i} && 1 \leq i \leq m \\ \sum_{i=1}^{m} f_{p_i q_j} &\leq w_{q_j} && 1 \leq j \leq n \end{aligned}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} f_{p_i q_j} = \min\left(\sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{q_j}\right)$$

2. Determine the distance between P and Q by normalizing W to the sum of all flows as:

$$D_{\mathrm{EMD}}(P, Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{p_i q_j} f_{p_i q_j}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{p_i q_j}} \tag{2.28}$$

Equation 2.27 is a linear programming problem, for which efficient solutions exist, i.e. the transportation problem.

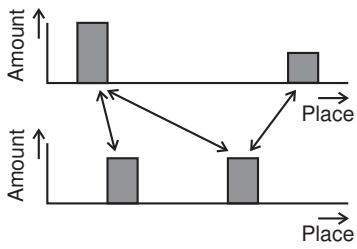


Figure 2.6: Minimal cost transformation between two histograms

Asymptotic Likelihood Approximation The Asymptotic Likelihood Approximation (ALA) as presented by Vasconcelos [118] is an analytic solution of the KL-divergence for the special case that the clusters of a GMM in feature space $\mathcal{X}$ do not overlap. When the clusters do overlap, it still holds as a good approximation. Since higher-dimensional feature spaces are more sparsely populated than low-dimensional spaces, the overlap will is expected to be small and the approximation will be of reasonable quality.

The ALA is defined as the distance between Gaussians P and Q with n and m clusters respectively:

$$P(\boldsymbol{x}) = \sum_{j=1}^{n} w_j \mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu_{p_j}}, \boldsymbol{\Sigma}_{p_j})$$
$$Q(\boldsymbol{x}) = \sum_{k=1}^{m} w_k \mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu_{q_k}}, \boldsymbol{\Sigma}_{q_k})$$

$$\mathrm{ALA}(P\|Q) = \sum_j w_j \Bigg(\log w_{q_{\beta(j)}} + \left[\log \mathcal{G}(\boldsymbol{\mu}_{p_j}, \boldsymbol{\mu}_{q_{\beta(j)}}, \boldsymbol{\Sigma}_{q_{\beta(j)}}) - \frac{1}{2} \operatorname{trace}[\boldsymbol{\Sigma}_{q_{\beta(j)}}^{-1} \boldsymbol{\Sigma}_{p_j}] \right] \Bigg) \quad (2.29)$$

where $\beta(j)$ the index of the cluster in Q that is closest to cluster j in P according to:

$$\beta(j) = k \Leftrightarrow \|\boldsymbol{\mu}_{p_j} - \boldsymbol{\mu}_{q_k}\|^2_{\boldsymbol{\Sigma}_{q_k}} - \log w_k < \|\boldsymbol{\mu}_{p_j} - \boldsymbol{\mu}_{q_l}\|^2_{\boldsymbol{\Sigma}_{q_l}} - \log w_l, \qquad \forall l \neq k.$$

2.3.5 Common Classifiers

The goal of classification algorithms is to assign a sample $\boldsymbol{x}$ from sample space $\mathcal{X}$ to the class it most likely belongs to. The set of classes $\mathcal{Y}$ contains a finite number of classes $\mathcal{C}$. We distinguish between *eager* and *lazy* classifiers. An eager classifier determines a relationship between $\mathcal{X}$ and the set of labels $\mathcal{Y}$ at the training stage, while lazy learners search for a relation between $\boldsymbol{x}$ and $\mathcal{Y}$ at classification stage. In this section, a selection of classification algorithms being used in the MIR community is described.

k-Nearest Neighbors

Among the simplest classification procedures is the k-nearest neighbor algorithm (kNN). For a testvector $\boldsymbol{x}$, the k-nearest neighbours are determined. The sample is assigned to the majority class of its neighbors. Selecting an appropriate value for k is a tedious problem, the optimal value depends on the dataset. Large values will reduce classification noise but result in less distinct class borders. At small k, the algorithm is sensitive to the local structure of data.

Two problems of the kNN algorithm are that classes with more frequent examples dominate less prominent classes and that the algorithm does not 'scale' well for large datasets. In this context, 'scaling' refers to the increase of computational complexity with the increase of the dataset. Scaling efficiency can be improved by segmenting feature space (e.g. by using k-dimensional trees [67]). Instead of searching through the entire dataset, only the neighboring leaf nodes of the tree need to be traversed.

Naive Bayes

A naive Bayes classifier uses Bayes' rule to determine the likelihood of a set of samples in a set of classes described by a probability model. In the training phase of a Bayes classifier, a probabilistic model (e.g. a Gaussian mixture model) is created for each class $\mathcal{C}$. Once the class models are trained, samples can be classified using Bayes' rule:

$$p(\mathcal{C}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \frac{p(\mathcal{C})p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n|\mathcal{C})}{p(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)} \tag{2.30}$$

Since the denominator is equal for all classes, only the numerator is of interest. When assuming the samples $\boldsymbol{x}_j$ are independent, the expression can be

simplified to:

$$p(\mathcal{C}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = p(\mathcal{C}) \prod_{j=1}^{n} p(\boldsymbol{x}_j|\mathcal{C}) \tag{2.31}$$

The most frequent desicion rule is based on the maximum a posteriori criterium:

$$\text{class} = \arg\max_c p(\mathcal{C} = c|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \tag{2.32}$$

Decision Tree Classifiers

Decision tree classifiers build a decision tree, splitting feature space at each decision node in two or more partitions. Each splitting decision is based on one single attribute. The choise on what attribute to split on is made by analyzing all possible splits with respect to an optimization criterion. Typical examples of these optimization or goodness criteria are the information gain or information gain ratio, based on the increase of entropy of the selected subsets relative to the entropy before the split, or the gini index, based on the 'purity' of the classes.

The basic procedure for training a decision tree is:

1. Start with all training examples in the root.
2. Choose the attribute to split the training examples in the root in two child nodes on, using a goodness criterium.
3. Assign training examples to child nodes according to splitting function.
4. Proceed recursively until a stopping criterium is reached.

The stopping criterium for perfect classification of the trainingset is when all leaf nodes contain only one single class. Since this often results in overfitted classifiers, full-grown trees are often *pruned*; brances are cut before the original stopping criterium is reached. There are two methods for pruning: prepruning and postpruning. The prepruning procedure introduces an extra stopping criterium: stop splitting when the goodness criterium does not improve above a certain threshold. It is difficult to choose an appropriate threshold. Postpruning removes branches from a full-grown tree progressively. Classification accuracy can be assessed on a training dataset at different stages of pruning.

Support Vector Machines

A support vector machine is a linear classifier, separating two classes by fitting a hyperplane between the classes. The optimal classification is obtained with

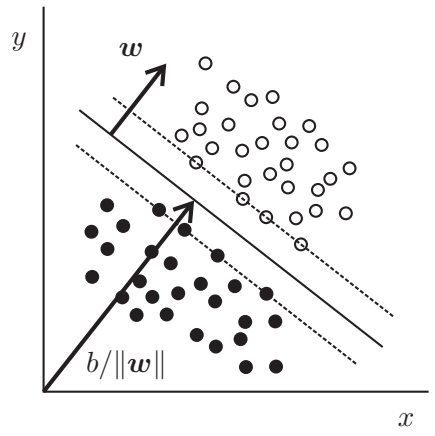


Figure 2.7: Linear separation of two classes using SVM. The solid line illustrates a decision hyperplane, the dashed lines represent the upper and lower margins (Equations 2.33 and 2.34).

a maximum margin hyperplane (Figure 2.7), which can be fitted in the feature space using a quadratic programming approach [17]. The optimal position of the hyperplane solely depends on its *support vectors*: those samples along the hyperplane. The points $\boldsymbol{x}$ on a hyperplane satisfy $\boldsymbol{x} \cdot \boldsymbol{w} - b = 0$, with $\boldsymbol{w}$ the normal vector perpendicular to the hyperplane, and $b/\|\boldsymbol{w}\|$ the perpendicular distance from the hyperplane to the origin. For the upper and lower margin we can be described by:

$$\boldsymbol{w} \cdot \boldsymbol{x} - \boldsymbol{b} = 1 \tag{2.33}$$

$$\boldsymbol{w} \cdot \boldsymbol{x} - \boldsymbol{b} = -1 \tag{2.34}$$

Once $\boldsymbol{w}$ and b are determined, samples can be classified using $\boldsymbol{w} \cdot \boldsymbol{x} - b \leq -1$ or ≥ 1.

Many classification problems are not linear seperable in the feature input space. SVMs can still be used for these problems by transforming the non-linear seperable input space to a higher dimensional space in which the classification problem can be solved in a linear fashion. These transformations are carried out using *kernel functions*, functions providing a 1 to 1 mapping from one space to another.

Since SVMs can only differentiate between two single classes, various strategies for multiclass classification are developed. The standard method for an N class classification problem is training N SVMs, each separating one class ver-

sus all remaining classes. The final classification result is the class with highest classification output. Other methods include 1 vs. 1 classifiers in which SVMs are trained for each class pair. Classification output of this 'max wins' strategy is determined by majority vote: the class receiving most votes wins. This method has been refined by Platt et al. [89] using directed acyclic graphs, to reduce the number of 1 vs. 1 comparisons.

2.3.6 Self Organizing Maps

Song similarity based on content based music similarity measures is hard to visualize in a comprehensible manner; the single features or feature sets have high dimensionality, and so have the statistical models of songs. Data of more than three dimensions cannot easily be displayed on a two-dimensional surface. Methods for projecting high dimensional data to a low dimensional space are available. One method doing this, while keeping distance relations approximately intact, is mapping the data using a self organizing map (SOM).

A SOM is a neural network that accomplishes a mapping of a high dimensional space to a lower dimensional space. The term 'self organizing' refers to the property that the neural net can be trained by a training set such that the mapping approximately preserves the topology of the training set: neighbors in the training set are mapped to neighbors in the lower dimensional space. For training it uses the principles of competitive learning. Given a variable $\boldsymbol{x}(t) \in \mathbb{R}^n$ and a set of reference or *codebook vectors* $\boldsymbol{m}_i(t) \in \mathbb{R}^n$, $i = 1 \dots k$. Here, k is the number of neurons. The variable t is a counter that indexes the iteration count during the learning phase of the SOM. We refer to t as 'time'. $\boldsymbol{x}(t)$ is drawn from a learning set. $\boldsymbol{m}_i(0)$ is initialized with random vectors from the learning set. Each codebook vector is stored in a neuron that has an *activation function* monotonically decreasing with the distance of its codebook vector to the input sample $\boldsymbol{x}(t)$.

At time t, the sample $\boldsymbol{x}(t)$ is presented to all neurons, the winning neuron is that one having the highest activation, i.e. the neuron with the codebook vector most similar to input $\boldsymbol{x}(t)$:

$$\boldsymbol{m}_c(t) \in \boldsymbol{m}_{1\dots k} : \boldsymbol{m}_c(t) = \min_{\forall i} \|\boldsymbol{x}(t) - \boldsymbol{m}_i(t)\| \tag{2.35}$$

The winning codebook vector $\boldsymbol{m}_c(t)$ is then adapted to decrease the distance to $\boldsymbol{x}(t)$:

$$\boldsymbol{m}_c(t+1) = \boldsymbol{m}_c(t) + \alpha(t)\left[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)\right] \tag{2.36}$$

with $\alpha(t)$ a monotonic decreasing function, determining the learning rate of

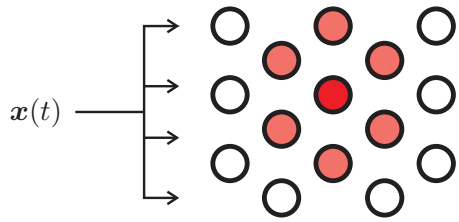


Figure 2.8: Competitive learning with neighborhood functions. Each circle represents a neuron. The dark-red neuron is $\boldsymbol{m}_c$, the light-red neurons are the neurons that are adapted due to the neighborhood function.

the system. For sufficiently large t, the codebook vectors $\boldsymbol{m}_i$ will describe $\boldsymbol{x}(t)$ with a minimal residual error. This result is similar to that of a vector quantization algorithm.

The described 'network' does not have any notion of topology, no neighborhood relationships between the neurons were defined. Kohonen [44] aligned the neurons on a grid and introduced a neighborhood function ϕ giving the neurons topological relationships. The update function now not only updates the winning neuron $\boldsymbol{m}_c$, but also its neighbors N_c (see Figure 2.8):

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) + \alpha(t)\phi(t)\left[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)\right] \qquad \forall i \in N_c(t) \qquad (2.37)$$

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) \qquad \forall i \notin N_c(t) \qquad (2.38)$$

The neighborhood function is a monotonic decreasing function over distance to the winning neuron. Its radius decreases over time. Due to the sequential overlaps of the neighborhood functions at each iteration of the algorithm, the values of the codebook vectors tend to be smoothed and become ordered. Similar input samples $\boldsymbol{x}$ will map to neurons that are topographically close to each other.

A special form of SOM is the continous or circular SOM. This SOM has no 'borders'. In the case of a SOM that is aligned in a rectangular grid of 10×10 positions, the neuron at e.g. $(10, 4)$ is a direct neighbor of the neuron at $(1, 4)$. This kind of SOM has been used e.g. in [70].

Self organizing maps can work with arbitrarily complex codebooks, as long as two conditions are fulfilled. A distance measure with monotonic decreasing distance function between training data and codebook entry is available and there should be an algorithm for adapting the codebook entry to better represent the training data.

2.4 State of Technology

In this section, the state of technology of music genre classification systems, of social and of content-based music recommender systems is described.

2.4.1 Genre Classification

The most common evaluation procedure for music similarity measures is genre classification. The assumption one has to make, is that songs of the same genre are (more or less) similar. Although the validity of using genre classification as test for music similarity is put in doubt by some authors (e.g. [4]), other authors argue to keep pursuing genre classification, but in a more cross-disciplinary way as has been the case up till now [61]. Musical genre is a feature that is not primarily defined by the audio contents of a song. Definitions of genre change over time and humans often disagree on what is the correct genre of a song [65].

There are many sources of ground truth data available (e.g. [16, 36, 62]). These sources often have different genre taxonomies which makes comparison between data from different sources even more difficult [73]. Due to copyright restrictions, no commonly available test dataset has been agreed upon [54]. This lack of a commonly used test dataset makes it hard to compare genre classification results of different authors.

At the yearly International Conference on Music Information Retrieval (ISMIR), researchers around the world are invited to participate in the Music Information Retrieval Evaluation eXchange (MIREX) tasks. MIREX gives the opportunity to compare performance of genre classification systems on one single (closed) dataset.

Feature selection

The elementary component of a classification systems is the featureset describing the objects to be classified. There have been various approaches for choosing features to describe music similarity. In the following paragraphs, the basic methods that have been investigated are described.

Single domain Dan-Ning et al. [23] compare MFCC features with the spectral contrast feature (SCF) in a five genre database. The genres are modeled with 16 component Gaussian mixture models and 10 seconds audio clips are classified into the most probable class according to the Bayesian criterion. The MFCC features are outperformed by the SCF by a few percent. Aucouturier

and Pachet [6] also compared MFCC with SCF and only have found a marginal improvement of 1% in favor of the SCF.

The usage of rhythmic features for classification of ballroom music is investigated by Gouyon et al. [37]. From a wide range of features, a MFCC-like decomposition of inter-onset interval histograms provides the largest contribution to classification accuracy. It has to be noted that for ballroom music, tempo and rhythm are essential characteristics so that using rhythmic features for this kind of music is a natural choice.

Multiple domains Tzanetakis and Cook [114] have combined three feature sets, describing timbral, rhythmic and pitch content of the audio. Classification on three datasets was performed using k nearest neighbor, single Gaussian and Gaussian mixture model based classification. The Gaussian mixture model approach using three Gaussians outperformed the other classifiers. The individual contributions of the timbral, rhythmic and pitch content featuresets was assessed, the timbral features proved to give the highest classification accuracy, but the combination of all features performed best. Pampalk et al. [80] also combines timbral features with rhythmic features and find that weighted similarity with 70% timbral and 30% rhythmic features gives optimal performance.

Whitman and Smaragdis [122] successfully combine a simple spectrum-based feature with web-based artist similarity for a genre classification task using a k nearest neighbour classificator. Genres that are largely culturally defined, are found to be hard to distinguish based on the audio properties only. The genres hardcore rap and R&B were difficult to classify using web-based similarity, but could be classified using the audio based approach.

Feature transformations Lippens et al. [50] use a feature that attempts to represent the physiology of the human ear. The structure of the feature is similar to that of standard MFCC vectors, but takes effects of the propagation in the outer ear and compression in the hair cells into account. The more complex feature did not improve genre classification accuracy when comparing with the MFCC feature.

Lidy and Rauber [49] explored the use of psycho-acoustic transformations on a series of rhythmic features. By transforming the input data to a dB loudness scale, applying an equal loudness contour correction and converting to phon scale, dB loudness, equal loudness contour and phon scale transforms had positive influence on a genre classification task.

Automatic feature selection There have been multiple studies on automatic feature selection. Mörchen et al. [69] use a set of 519 low level features, on which a set of 164 static and temporal statistic operations are performed. The performance of each individual resulting feature was assessed by analyzing class seperability using the amount of overlap on the individual class Pareto density estimations. Three new featuresets were defined, each consisting of the 20 top features of either overall class seperability, specialist class seperability or a combination of overall and specialist class seperability score. The sets were selected using a greedy selection algorithm with a correlation filter.

Mierswa and Morik [64] and Zils and Pachet [128] use a genetic programming approach for finding new features. A set of operators for temporal, spectral and statistical operations is defined. These operators are combined using feature design patterns and heuristics. After each iteration of the system, the 'fittest' feature is selected and transformed to form a new set of similar features. This process is repeated untill a stopping criterium is fulfilled. Although this method is very succesful, the found feature set is tailored to the training set which can cause serious overfitting problems.

Model based features The features described in the preceding paragraphs did not take notion of the semantical structure of the audio. Reed and Lee [97] present a classification method based on *universal acoustic models* and *acoustic segment models* (ASM), a modelling approach developed for speech recognition applications. These models are based that each word consist of a limited set of phonemes, whose relationships can be described using a set of grammatical rules. Reed and Lee extend these models to form a set of phonemes for music: the audio of a training set is segmented using a minimal distortion metric on successive frames and characteristic segments are determined using a vector quantization algorithm. After re-estimization of the segments, a HMM is trained on each 'phoneme'. After all phoneme models are trained, a song is represented as a string of phoneme symbols, which can be classified using text classification algorithms.

Pohle et al. [91] use a different approach. Inspired by image classification, a set of 'patches', consisting of consecutive Mel-sone feature vectors is analysed with an independent componant analysis algorithm. Songs can then be represented as activation histograms of the available independent components. Similarity between songs can now be determined by comparing the activation histograms of the independent components. Performance of the similarity measure was evaluated with a leave-one-out 1-Nearest-Neighbour genre classification task, which showed reasonable classification accuracy.

Models and classifiers

Various data modelling and classifier techniques have been used for genre classification tasks. Tzanetakis and Cook [114] have compared single Gaussian, Gaussian mixture models with 2-5 clusters, and k-nearest neighbor classification on three datasets. Each song was represented by one single feature vector, representing timbral, temporal and pitch content. On all three datasets, the GMM classificator with three clusters delivered the best classification accuracy. Lippens et al. [50] compared the single Gaussian classifier with results from a human classification task on the same dataset of 160 songs from 10 genres. The human classification results were on average about 20% better than the results obtained with the single Gaussian classifier (88 % vs 65 % accuracy [50]).

Aucouturier and Pachet [6] performed extensive experiments on finding an optimal parameterset for modelling MFCC based timbre similarity with Gaussian mixture models. Each song is modelled with a 50 component Gaussian mixture model, in a 20 dimensional MFCC feature space. Using monte carlo sampling, a song similarity matrix is build. The similarity matrix is input to a k-nearest neighbor algorithm used for genre classification. Although 50 Gaussians performed optimal, it was noted that this number can be reduced without much loss of classification accuracy.

Later work by Aucouturier and Pachet [7] analyzed *hub occurrences* as a function of GMM *homogenization*. Hubs are songs that are returned in nearest neighbour queries way more often than statistically plausible. A homogenization operator was used to subsequently remove the smallest clusters in a 50-component GMM. Discarding the smallest clusters that account for five percent of the data, caused a dramatic increase in the number of hubs.

The influcence of using a hidden Markov model to also model the temporal relationships of the MFCC vectors was also investigated. The best performing HMM has about the same number of free parameters as the 50 component GMM, but performs not better than the GMM. Flexer et al. [30] find similar results: HMMs are found to better model a song's datastructure, but this has no positive influence on genre classification.

Berenzweig et al. [14] compared different distance measures for Gaussian mixture models: the asymptotic likelihood approximation (ALA), earth movers distance (EMD) and the centroid distance. Nearest neighbor lists were retrieved and compared with lists obtained from a subjective similarity measure based on a user survey. The EMD slightly outperformed the ALA. The simple centroid distance performed remarkably well.

Mitri et al. [65] compared a ZeroR, OneR and C4.5 decision tree classifier on a 10-genre classification task. A 58 dimensional vector consisting of both temporal as timbral features was extracted of 10 second excerpts of audio. The OneR classifier selected one single feature providing optimal separation of the classes. This single feature discrimination surprisingly showed better classification results as the C4.5 tree classifier using all available features.

Mandel and Ellis [58] compare support vector machine classifiers with a k-nearest neighbor classifier on both a genre- as on an artist classification task. Both a frame based feature approach as song level features were evaluated on a genre and on an artist classification task. The support vector machine classifier consequently outperforms the k-nearest neighbor classifier and song level features outperformed the frame based approach. In order to reduce the 'album' effect, songs from the training and test set were selected from different albums. Without this restriction, classification accuracies are much higher, but do not generalize well. A similar proposal was made by Pampalk et al. [80] for genre classification tasks: The training set should contain different artists as the test set.

2.4.2 Social Music Recommender Systems

Aucouturier and Pachet [6] state that there exists a 'glass ceiling' for classifiers only relying on features extracted from the audio content of music. This suggests the use of other data, that cannot be retrieved from the audio itself, is needed to overcome this barrier.

Currently, the music recommendation services market is dominated by social recommender systems using the principles of *collaborative filtering* [38]. These systems (e.g. lastFM (www.last.fm), iLike (www.ilike.com)) base their recommendations on data obtained by analyzing user preferences, either by observing user behavior, or by analyzing user ratings of songs.

The goal of collaborative filtering systems is to learn relationships between items by observing user interaction with and user ratings of these items. Using collaborative filtering for recommending music to users, two approaches are available: the user-based and the item-based approach.

The user-based approach collects statistical data on what items a user is interested in by either observing the user's browsing behavior or by analyzing the user's ratings on a set of items. These statistical data, also called 'user profile', are then compared to other user profiles. Recommendations can now be done by selecting user profiles that show high similarity with that of the current user. Highly rated items from these other user's profiles that are not

yet rated or seen by the current user, are selected as potentially interesting items and are recommended to the user.

The item-based approach does not build models of users, but models the items: When a user is interested in item A and item B, a relation between A and B is set up. When an other user now searches for item A, item B can be recommended, since it is related to A.

2.5 Summary

In this chapter, an introduction to music similarity is given. In the first section, the concept of music similarity was described with respect to music theory, timbre similarity, perceived similarity and cultural similarity. Music can be described using the music theoretical elements of *harmony*, *melody*, *rhythm*, *texture* and *structure*. Each of these elements can be used for expressing similarity between two musical pieces. The concept of *timbral similarity* is explained by analyzing the human auditory system and a short overview of studies on the elementary dimensions of timbre. All music theoretical and timbral elements that were described are put into context in the part on *perceived similarity*: the process of recognizing these elements and the different modes of reception were described. The first section ends with an overview of methods for measuring *cultural similarity*: similarity defined by a listener or a collection of listeners.

Section 2.3 presented an overview of commonly used low- and mid-level features for describing audio signals and methods for modelling them. The section ends with an introduction of several classifiers. In section 2.4 these features and classifiers return as part of the state of technology for music genre classification algorithms.

Chapter 3

Playlist Generation

3.1 Introduction

The concept of playlists originated from radio stations publishing a list of songs they were to play or had played. This list not necessarily listed the songs in the order of play, it could also just present the pool of songs of which only a small part was played. As personal media devices have gained access to large amounts of music, the current meaning of 'playlists' emerged: an ordered list of songs, to be played in given sequence.

This thesis is about the automatic generation of personal playlists. In this context, a personal playlist is defined as follows:

> A personal playlist is a list of a number of songs in a certain order. It has a certain length and matches the musical preference of a user.

In this chapter, the main concepts of playlist generation and the state of technology of current playlist generation systems is described. Furthermore, a detailed report on a questionnaire that was designed to analyze user's listening behavior and requirements on personal playlists is presented. This section is followed by a discussion on possible playlist quality criteria. The chapter ends with the presentation of the new approach for content based playlist generation systems that was developed in the course of this research.

3.2 Concepts

The number of ways to use playlists is very large. It ranges from a personal playlist, only listened to one single time, by a single person, to playlists for nation-wide radio stations. This wide range of scenarios results in different user

demands for software offering support for playlist generation. In this section, common use-cases and user demands and some of the psychological aspects that should be dealt with are presented.

3.2.1 Use-cases and User Demands

Radio playlists

In the keynote speech of ISMIR 2006, Huron [42] posed that music consumption will soon be 'AMAWAT': Any Music, AnyWhere, AnyTime. The entertainment industry is among the largest industries in the world. The main business of radio stations will not be to entertain, but to trade consumers to advertisers. In order to have a homogenous audience, each radio station has a certain *programming format*: the overall content of the radio station. Example formats include classic rock, classical or easy listening. Within a song collection of a certain format, *playlist rotation rules* are applied. These rules ensure a mix without undesired frequent repetitions and enough variations within a playlist. As an example, the format and most important rotation rules for the German local classic rock radio station Radio21[1] are presented here.

Radio21 targets at a male audience of 30 years and older, liking music of the 60s, 70s and 80s with a 'guitar sound'. Music is categorized by year, energy (1 = slow, 4 = fast) and whether the music sounds 'hard' or 'soft'. A small subset of the collection is labeled as 'power song', songs that are frequently named to be listener's favorite songs. These user preference data are obtained through telephonic enquiries among the audience. The typical listening length of a Radio21 listener is 15-20 minutes. The most important property of a Radio21 playlist is variation: the year of production of subsequent songs follows the simple pattern 70s-60s-70s-80s-..., both the energy and hard/soft curve make no large steps at once. Songs of female artists may never be played without a male intermission. The mean energy over the entire day lies at 2.6, in the morning this tends to 3.0. Radio21 playlists are created by a scheduling program. This program selects the songs based on given constraints and outputs playlist for as long as one week. The music editorial staff checks the playlists and makes small changes.

Personal playlists

Contrasting to the marketing driven radio playlists, a personal playlist can contain any music. A classic example of a user playlist consists of all tracks

[1] www.radio21.de

of a single CD, or tracks from the *active items* (Cunningham et al. [22]) in a person's CD collection. In large digital personal music collections other playlist creation methods are used, ranging from random play through an entire collection, playing all music of a single artist, album or genre, to careful manual selection of single tracks.

Cunningham et al. [21] analyzed the way playlists and mixes are created by users of the Art of the Mix website[2]. These playlists consist of carefully selected songs '*where one song responds to another song*' for a special event, of a certain mood or for a certain activity. The website offers the possibility for users to exchange opinions on each other's playlists and ask other users for help or suggestions when creating a playlist. Cunningham et al. categorizes such help requests in 10 organizing principles. The categorization of 115 of such help requests is listed in Table 3.1. The percentages do not sum up to 100% since some requests overlapped different categories. The list gives an interesting mix of content and contextual aspects important for these personal playlists. Not all persons making playlists make such detailed thoughts about playlists, but the categories provide a frame for aspects to be taken into account for personal playlists.

Category	Percentage
Artist/Genre/Style	25.2%
Event or Activity	25.2%
Romance	19.1%
Message or Story	16.5%
Mood	16.5%
Challenge or Puzzle	10.4%
Orchestration	7.0%
Characteristics of Mix Recipient	6.1%
Cultural References	6.1%
Other	2.6%

Table 3.1: Categorization of Organizing Principles for Mix Help Requests. Reproduced from [21].

The wide variety of playlist concepts and categories poses requirements on playlist generation systems that are still out of reach of the current state of technology. A large part of the mix help requests in Table 3.1 do not directly concern the music genre or instrumentation, but are about using music in a certain situation or context. These kind of metadata have been explored by

[2] www.artofthemix.org

Hu et al. [41]: *Usage metadata* showed to be partly correlated with genre, but only to a limited extend.

Apart from the wide variety in usage scenarios of playlists, playlist generation systems have to tackle the problem of exploiting large datasets in mobile media devices having very limited processing power and a limited user interface. Current portable media players suffer from the *long tail problem*: only about 20% of the music available on a medium size collection, is actually listened to. This long tail is revealed when plotting the song play counts, sorted on play count: a exponential-decay like plot will show up. As a consequence of failing music access strategies, the majority of the music remains unheard.

3.2.2 Psychological Aspects

Music can have a soothing effect under stressful circumstances. Wiesenthal et al. [123] presented a study where car drivers were either listening to their favourite music, or not listening to music. The group listening to music showed to be less aroused under heavy traffic conditions. In [83], distraction by music was investigated: a group of listeners listening to their preferred music showed to be less distracted than a group listening to their least preferred music. A good playlist should thus contain music that is preferred by a listener.

A list of songs matching one's personal music taste is not necessarily a good playlist. A person having a wide musical taste may listen to very different kinds of music at different times or different occasions. In a study performed by Pauws and Eggen [84], users express two main wishes: the wish for *coherence* and the wish for *variation*. The wish for coherence shows that there should be some kind of relation between the songs in a playlist, for instance a shared conceptual description of the songs in the list. Contrasting to the wish for coherence is the wish for diversity. Introducing new, unpredictable musical content provide the user with a satisfying 'surprise effect'. It is also this effect that is adressed by the 'aha-slider' of Aucouturier and Pachet [4]. The balance between coherence and variation is further studied in section 3.4.

3.3 State of the art in playlist generation

Various methods for music playlist generation have been proposed. These methods use either purely content based features [20, 53, 82, 92], high level metadata [1, 94] or a combination of both [3, 39, 74, 84]. These approaches all have in common, that 'similar' songs are sought, which are played after each other. Purely content-based methods apply nearest neighbor-like strategies,

while the use of high level metadata allows posing a set of constraints a playlist has to satisfy. In this section, nearest neighbor based strategies, constraint satisfaction techniques and alternative ways to generate playlists are described.

3.3.1 Nearest Neighbor and variations

Nearest neighbor approaches to playlist generation are all based on a distance measure expressing similarity between each song as a distance. 'Similar' songs have a small distance to each other. Using a heuristic, those songs that are similar to each other are enqueued in a playlist.

Early work by Logan and Salomon [55] explore the use of an MFCC based timbral similarity measure for simple playlist generation. The performance of the similarity measure was analyzed by retrieving playlists consisting of the N nearest neighbors of a seed song. The quality (or 'goodness') of the playlists was assessed by analyzing the average number of songs in the same genre or by the same artist, within the closest 5, 10 or 20 nearest neighbors.

Logan extends the nearest neighbor approach to *song trajectories* in [53]. A fully connected graph is formed from all songs in the database. Each song is a node and the link strengths are given by the distances between the songs found by the MFCC based distance measure. The first trajectory strategy is selecting the shortest path of length N starting at a *seed song*. In case of a loop in the path, either the next closest song of the current song, that is not part of the playlist is chosen, or the path is restarted from the next closest song from the original seed song. The second trajectory strategy includes automatic relevance feedback: of each M nearest neighbors of the seed song, N-closest playlists are generated. The final playlist has the songs ranked by their cumulative song position in all M playlists. Logan evaluated both trajectory strategies with the same measure in [55]. The second strategy showed a slightly worse performance as the simple N-closest playlist baseline.

Herrera et al. [39] combine timbre, tempo, genre, mood and year similarity in their E-Mu jukebox. Users can set the weights of the individual features by dragging feature weight adapters in a feature weighting space. The returned playlist consists of the nearest neighbors of a seed song according to a weighted similarity distance measure over all feature.

Pohle et al. [92] explore circular playlists covering an entire music database with more than 3000 songs using a timbre-based distance measure and the traveling salesman algorithm to determine the shortest path through all songs. The traveling salesman algorithm is originally used in operations research. The name is a metaphor for the problem of determining the sequence of visiting all

cities in a country exactly once. The optimal sequence has the shortest possible path length. Four algorithms approximating the optimal traveling salesman problem (TSP) solution on a music dataset were evaluated with respect to overall path length and the short time and long time genre consistency. The short time and long time genre consistency measure assesses the number of different genres occurring within a short- (e.g. 12 consecutive songs, the average number of songs on a CD) or long-time (e.g. 75 consecutive songs, music for one day) interval in a playlist. The algorithms providing the shortest overall path length show high short-time genre consistency, but as soon as an artist filter is applied, disallowing tracks of one artist to occur twice within six consecutive songs, both route length and consistency severely degrade. Two algorithms providing better long-time genre consistency showed to be less sensitive to the artist filter; after the artist filtering, short-time consistency is comparable to the other algorithms.

Pampalk et al. [82] present a dynamic playlist generation system based on a audio based similarity measure [80] and user skipping behaviour. The performance of four playlist generation heuristics are evaluated for three use-cases: 1) the user wants to listen to songs similar to a seed song. 2) like 1, but not songs of a particular artist. 3) the user's preference changes over time from genre A to genre B. Songs that do not fit within the current user preference are skipped. The analyzed heuristics are:

a) Play the N nearest neighbors of a seed song. Once a song is skipped, the next item in the list is played. This non-dynamic algorithm serves as baseline.

b) Play the nearest neighbor of the latest accepted song.

c) Play the closest to any of the accepted songs.

d) Let d_a and d_s be the distances to the nearest accepted and nearest skipped song respectively. If $d_a < d_s$ add the candidate song to the set S. Play the song in S with smallest d_a. If S is empty, add the song with the smallest d_a/d_s ratio.

These heuristics are depicted in Figure 3.1. Evaluation took place on a 2522 song dataset containing 22 genres. The median and mean number of skips indicate how well each heuristic performs. For all three use-cases, heuristic **d** significantly outperforms the other heuristics. A playlist of 20 correct songs for the first use-case requires a median of 11 skips, vs 37 for the (static) heuristic **a**.

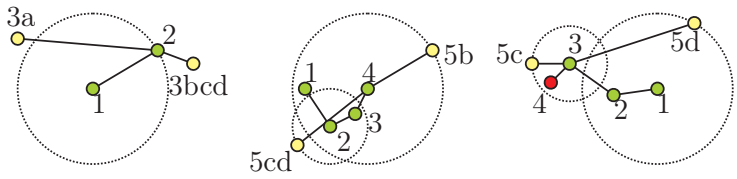


Figure 3.1: Playlist generation heuristics **a-d**. The dots represent songs mapped in a two-dimensional surface. The three subfigures (from left to right) show the differences (yellow) between the four heuristics. Red indicates that a song was actively skipped. The dashed circles show decision distances for the different heuristics. Reproduced from [82]

3.3.2 Constraint Satisfaction

The constraint satisfaction approach to playlist generation allows specification of a set of constraints a playlist has to fulfill. These constraints may either be *coherence constraints* imposing restrictions on relations between songs, or *absolute constraints* defining overall properties of a playlist (e.g. which percentage of all songs in a playlist must have a certain property).

A good example of using constraint satisfaction for playlist generation is provided by Pachet et al. [74]. The used features are placed in a taxonomy and similarity relations between the feature values are defined. For a feature like tempo, this is trivial, but for genre relations this is a tedious process. A typical playlist request then may look like:

- The playlist must contain 12 songs.
- Only medium and fast tempos.
- At least 30 percent of the songs should be of a female artist.
- Genre should be close to the neighboring genres.
- Each artist may only occur once.

This constraint satisfaction problem is not rewritten as a minimization problem but is implemented in a two step process. In the *backtracking procedure*, variables are progressively initialized with values that are available in their respective domains. If an inconsistency is detected that precludes finding a valid solution, the procedure goes back and tries again. The backtracking phase is followed by a *domain reduction phase*. In this phase, those values of the variable domains that cannot pertain to a valid solution are removed from the search space. This reduced domain allows faster search for valid solutions.

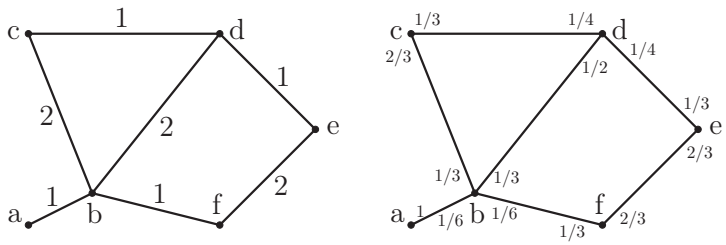


Figure 3.2: Connected graph (left) and Markov random field with exit probabilities (right) for playlist generation. Each node (a ... f) represents a song. The exit probabilities in the Markov random field (indicated as fractions at the outgoing links) are determined by the sum of link weights of the node and the link weight of the corresponding outgoing link.

Aucouturier and Pachet [5] introduced adaptive search as an alternative to constraint satisfaction for playlist generation. Adaptive search is a local search technique, iteratively optimizing the *costs* of a search solution. The advantage of this adaptive search strategy is that the search can be terminated at any time, providing a feasible, but non-optimal solution. Non-optimal solutions are acceptable for a large number of applications and it requires only about 25% of the time to find a nearly-optimal playlist.

3.3.3 Other

Ragno et al. [94] use online playlists shared by radio stations to infer similarity between songs. Each song is represented as a node in a graph. For each time two songs are played after each other, the link weight between the two songs is increased. When the amount of playlist data used to create the graph is large, this process results in a tightly connected graph, with songs that are likely to appear together with a high link weight. In order to create playlists, the graph is mapped to a Markov random field. To this purpose, the weights of all arcs w_a terminating on a node are summed up to w_N. Each arc is then replaced by two directed arcs, each with an exit probability of w_a/w_N (see Figure 3.2). Once the Markov random field is known, playlists can be generated by random traversal through the now directed graph.

Song distance matrixes can be generated from the Markov random field by converting the node exit probabilities to distances. This can for instance be done by taking the log likelihood of the exit probability to determine the distance between two songs in one direction. Using a shortest path finding algorithm between two nodes, the distance between any pair of songs can be

expressed as the sum of the link distances.

Pauws and Eggen [84] use a decentralized clustering approach in order to allow for the required coherence within the clusters together with the demand of enough variation in a playlist. The used clustering procedure is dynamic, and uses a learning song distance measure that adapts itself to positive and negative feedback of the user. The distance measure uses a large set of high level features (e.g. year of release, rhythm, instrumetation, etcetera). Song similarity is expressed as the sum of weighted feature similarity. The feature weights are adapted by training a decision tree on the user feedback labels 'preferred' and 'rejected'. The depth of a feature in the tree is used to adapt the feature weighting.

Song clusters are formed in a two-dimensional Euclidean space of finite size. All songs move around through this space with a certain speed and direction. The initial state is initialized with random positions, speeds and directions for each song. At each time step, a random song 'senses' whether there is a song in its vicinity. If this is the case, and the song is very similar, it adapt its speed and direction to match that of the similar song. Playlists are generated by selecting songs that are close to each other in the euclidean clustering space.

3.4 User Survey

There is extensive literature on playlist generation algorithms, music similarity functions and perceived music or timbre similarity (e.g. [1, 5, 14, 74]). Some studies present a human evaluation of how well a music similarity measure performs (e.g. [27, 84]) or whether automatically generated playlists make sense (e.g. [92]), but no literature was found on the actual user demands on playlists.

3.4.1 Goals

In scope of this research, an online questionnaire was designed to gain insights in user requirements on automatic playlist generation systems. The two main questions to be answered were: 'On what criteria does a user decide if he/she likes a song?' and 'What criteria are of relevance for generating a playlist?'.

A difference between conscious, *active listening* and unconscious, *passive listening* is assumed for playlist listening behavior. The validity of this assumption is assessed by comparing criteria importance ratings for both listening modes.

A wide variety in genre taxonomies is available, with hierarchical genre taxonomies expressing relationships between main and sub-genres. Part of the questionnaire aimed at analyzing listening behavior, musical preferences and the relations between favorized genres.

3.4.2 Questionnaire

Visitors of six online music forums and members of MIR-related mailing lists have been invited to take part in the online questionnaire, both available in english and german. The target audience is between 14 and 55 years of age and uses new media devices for music consumption.

The questionnaire consists of six parts. The first part is about musical preferences and listening habits and consists of 11 questions. These questions ask for preferences and dislikes of musical genres and the reasons thereof. Furthermore, this part contains questions on the influence music has on personal mood, on what situations music is listened to and for how long.

The second part aims at getting a detailed description of what aspects are named when describing music or musical genre. Part three is dedicated to in-car music listening behavior. Influences of the traffic situation on music preference are analyzed. What media devices are used to listen to music by the participant and whether he/she is an active musician is asked for in part four.

The questions relating to playlists are grouped in part five. The participants are asked to judge the importance of 12 playlist characteristics for a playlist. This question differentiates between listening to music as 'background music' and listening to music as primary occupation if the participant perceives these to be different.

Part six concludes the questionnaire with some questions on demographic data. The entire questionnaire contains 33 questions.

The questionnaire was available online from May 23rd until June 22nd 2007.

3.4.3 Evaluation

The intended target audience for the questionnaire was the population between 14 and 55 years of age, using new media such as mp3-player, pc and the internet. By choosing the internet as primary medium, only those people that are online are reached. This introduces a bias toward a higher educated, male audience. As a consequence of inviting for participation on music-related internet fora and mailinglists, the results of this research are also biased towards people having strong interest in music. Participation was voluntary, which means

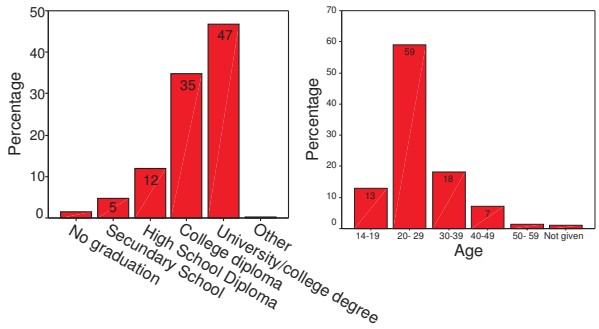


Figure 3.3: Education and age of questionnaire participants

that the self-selection process already filtered out people with low interest in the topic. These biases prevent the results from being easily generizable to a broader audience.

Sample

While the questionnaire was available online, 275 participants have answered all the questions. During the pre-test and a street interview phase, an additional 43 complete questionnaires were filled out, resulting in a total of 318 responses. 62% of the respondents were male, with an average age of 27,5 years. The female participants were only slightly younger, 26,9 years on average. The participants were divided in five age categories: 14-19, 20-29, 30-39, 40-49 and 50-59 years.

The educational level of the participants is high, almost 50% has a college or university degree.

Musical style and song preferences

In the first part of the questionnaire, the participants are asked for their favorite genres and an indication of on basis of what elements these genres are favorized. The participants can choose from a list of thirteen elements: rhythm, topicality, structure, artist, song mood, associated feelings, instrumentation, lyrics, melody, tempo, vocals, popularity and sound. The three elements that were named most often are song mood (67.4%), melody (58.6%) and rhythm (56.9%). Topicality and popularity are the least important with 3.9% and 0.7% respectively.

A very similar question is on what elements are important for liking a single song. The participants can choose from a list of fifteen elements, the thirteen

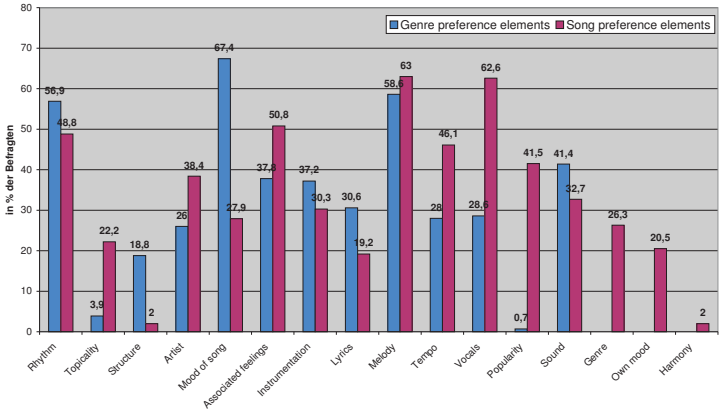


Figure 3.4: Elementary elements for judging preference for a genre and for a song

from the previous question and genre, own mood and harmony. It was expected that the same elements are important for judging genre preference and song preference. This expectation turned out not to be true; the most important elements for single song preference are melody (63.0%), vocals (62.6%) and associated feelings (50.8%).

In Figure 3.4 the answers of both questions are shown. Comparing the aspects for genre and song preferences, there are a few remarkable differences. For judging song preference, topicality (22.2% vs 3.9%), artist (38.4% vs 26%) and popularity (41.5% vs 0.7%) gain major importance. Song mood turns out to be less important for the liking of a single song. A further look reveals that for genre preferences, the analytical elements song structure, lyrics and instrumentation are more important.

Using the results of question one, an alternative genre similarity graph can be created. The most frequently named genre preference combinations that were named as favorite musical style are shown in Figure 3.5. Graphs like these preferred genre combinations graph can be used by constraint satisfaction approaches for playlist generation; genre continuity constraints can restrict allowed genre combinations within a distance to the genre of the first song in a playlist.

Listening behavior

The first part of the questionnaire contained 20 questions on personal listening behavior. The main topics of these questions were the influences of music on personal well-being, factors influencing the choice what music to listen to and

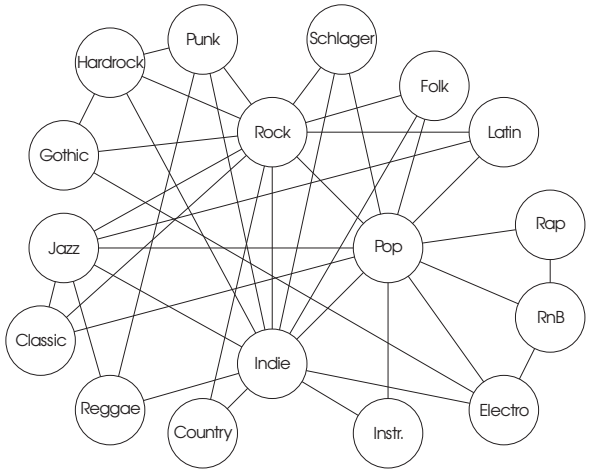


Figure 3.5: Frequently named preferred genre combinations

the main properties of the active music collection.

Music is important for 91.3% of the people. 95.0% indicate that music can have a positive influence on personal mood. The opinion on whether music can have a negative influence on mood is less uniform; 52.7% of the participants indicate that music can worsen personal mood. The statement 'Music helps me to relax' was said to apply for 88.3% of the people. 43.5% can relax best with calm music. Just over 40 percent of the people indicate that they can concentrate better while listening to music.

Preferences for what music to listen to are influenced by many factors. Most people (81.5%) indicate that music should match the mood one is in. For only 20.1% of the people, musical preferences of the people in their close environment is preferred to other music. 25.8% indicate favoring listening to recent hits. 31.3% of the people listen to other music in the evening than during daytime. The weather is said to influence the choice of music for 27.1% of the participants.

The majority of the people regularly get back to 'similar' music; 77.0% often listens to the same music, 67.6% often listens to music of once specific genre and 50.8% often listens to music of one specific artist. 45.6% of the participants often return to music one specific mood. 75.3% indicated to appreciate listening to different styles of music, which may also be music that is new to the listener (65.5%).

Active listening		**Passive listening**		**Universal listening**	
1	Variation	1	Volume	1	Volume
2	Same mood	2	Same mood	2	Variation
	Known songs	3	Harmony	3	Known songs
3	Volume	4	Known songs	4	Same mood
4	Harmony	5	Variation	5	Similarity

Table 3.2: Playlist feature preferences for three listening modes.

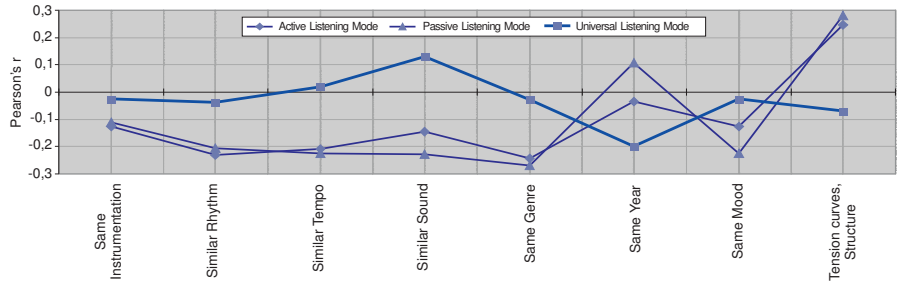
Playlist preferences

In the part of the questionnaire dealing with playlists, the participants are split in two groups. The first group consists of the people that indicate to have different criteria for judging playlists in an active listening mode and in a passive listening mode. The second group indicates not to have different criteria for both listening modes. This will be referred to as the universal listening mode. Both groups were asked which criteria out of a list of 12 possible criteria are important for a good playlist. For the first group (69% of the participants), this question was asked both for the active and the passive listening mode.

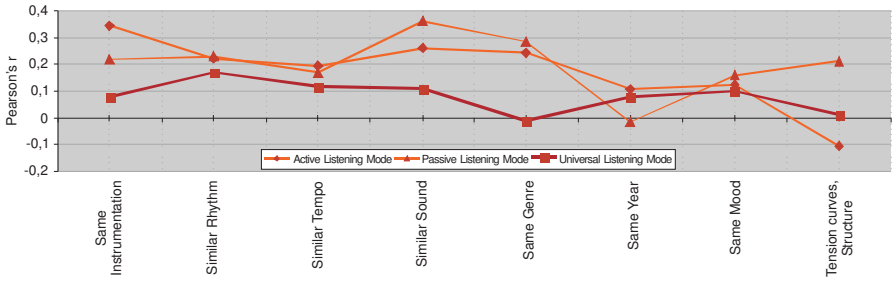
The most frequently named properties for all listening modes are listed in Table 3.2. The top five of the active and passive listening group contains the same properties, the universal listening group trades 'harmony' for 'similarity'. For the active listening case, variation stands out of the other criteria: it is named by 47.5% of the participants, with the next properties following at 36.1%. For the other listening modes the difference between the most and second-most frequently named feature preference is not larger than 3.4%. Except for the variation criterium for the active listening mode, the same tendencies for all three listening modes can be observed.

In the next question, the participants are asked to judge the importance of eight properties of playlists; songs in a playlist should have: same instrumentation, similar rhythm, similar tempo, similar sound, same genre, same year of production, same mood and clear tension-curves or structure. The question was answered only for the most characteristic listening mode of the participant. 41% answered the question for the passive listening mode, 59% for the active listening mode. Each criterium could be scored as 'very important', 'important', 'less important' and 'not important'.

The answering patterns for both active as passive listening again show the same tendencies. For the passive listening mode, all criteria are rated more



(a) Correlation of users indicating 'variation' as being important property vs other playlist properties



(b) Correlation of users indicating 'similarity' as being important property vs other playlist properties

Figure 3.6: Pearson's r correlation of playlist properties for the different listening modes.

important as for the active listening mode. This is in line with the expectations, since variation scored very high for the active listening mode in the previous question. The only feature that was rated as 'not important' by the majority of the participants was the year of production. The tension curves and structure of a playlist was rated more important for the active listening mode.

The feature preference listed in Table 3.2 are assumed to be related with the feature importance judgments of the individual participants. This relation has been analyzed using the Pearson's r correlation measure. In Figure 3.6, the individual correlations for the variation and similarity properties for the three listening modes are shown. The correlations are weak, although there is a relation, there is wide variance between the individual importance judgements. The group of participants judging 'variation' as important, shows the expected negative correlation with almost all similarity criteria for the active and passive listening mode. This group generally finds a playlist structure more important. Participants indicating 'similarity' as important property of playlists generally find the similarity criteria for playlists more important. The active listening

mode for 'similarity listeners' shows weak negative correlation with the playlist structure.

3.4.4 Summary

We were interested in the criteria used by people for judging song or genre preferences. Contrary to the expectations, these criteria are not the same: song mood, melody and rhythm are the most frequently named criteria for genre preference, while song preference is judged mainly by melody, vocals and feelings associated with the song.

The criteria for judging playlist quality are more or less the same for the three distincted listening modes. The users differentiating between active and passive listening, rated 'variation' to be the most important property for active listening, for the passive listening mode, this property is found at fifth' rank. Surprisingly, 'volume' was named very frequently, it shows up within the top three for all listening modes.

Using the participants genre preference data, a graph with genre similarity relations was created. This can be used as a source for genre distances in constraint satisfaction algorithms for playlist generation.

3.5 Playlist quality criteria

In the first section of this chapter, the principles of current playlist generation algorithms have been presented. Nearest neighbor, collaborative filtering and other approaches were explored. The second section presented the results of a user survey on music listening behavior, music and playlist preferences. In this section, the evaluation methods for the current generation of playlist generation systems are analyzed. The playlist quality criteria found in the user survey are analyzed with respect to application in automatic playlist generation systems.

3.5.1 Current methods

Playlist generation algorithms have been evaluated using both quantitative and qualitative methods. The quantitative approaches measure performance in terms of computational complexity or the number of user actions needed to generate a playlists. Qualitative playlist quality studies aim at measuring the perceived quality of playlists and thus involve user evaluations.

Both Logan and Salomon [55] and Pohle et al. [92] use high level metadata for evaluating the relevance and consistency of the generated playlists. Logan

and Salomon use genre and artist labels and count the average number of returned items with the same genre or artist within the closest 5, 10 or 20 nearest neighbors. Within the list of nearest neighbors, there is no quality criterium on subsequent songs. An 'optimal playlist' according to this measure thus consists of songs only of the same genre or artist. Pohle et al. use short time and long time genre consistency for comparing different playlist generation algorithms. Again, the criterium for quality of a playlist is optimized when only songs of one single genre are returned. A first step away from just returning the most similar songs is the 'artist filter': within n tracks, an artist is only allowed to occur once. Both measures do actually measure the quality of how well the similarity measure captures the similarity defined by the music metadata. No quality measure for the playlists as such is used.

Pampalk et al. [82] measure performance of various dynamic playlist generation algorithms on the amount of user input needed to fulfill the demands of three use-cases. The use-cases are based on genre and artist similarity. Suggested songs that do not fit in a use-case profile are skipped. The number of skipping actions needed to create a playlist of 20 songs is a measure for the quality of the playlist generation algorithm. The E-Mu jukebox by Herrera et al. [39] is evaluated by 22 test persons who were invited to create a playlist of ten songs. The only criterion given was the quality of the playlists. The performance of the system was evaluated with both objective and subjective measures: The time spent and the number of actions needed for creating a playlist are measured. The perceived quality and ease of use of the system are analyzed. The quality of the systems is assessed by measuring the amount of user input, or the time needed to adapt the settings in order to create a satisfactory playlist. This again is no objective quality measure of the playlists as such. Instead, the adaptivity of the algorithm and the quality of the similarity measure are rated.

Playlist generation algorithms based on constraint satisfaction allow posing constraints on the individual relationships between songs in a playlist. These methods have been analyzed with respect to their computational complexity. Aucouturier and Pachet [5] pose that 'the quality of the playlists generated with regards to some user preferences [...] depends on the quality of the musical metadata [...] involved in these preferences'. Since constraint satisfaction based systems can work with arbitrary sets of constraints, indeed any playlist can be created when the metadata are of high quality and detail. Evaluating qualitative playlist quality thus merely describes how well the metadata catches the preferences of the user and if the available constraints on song sequence in the playlist can capture the user preferences.

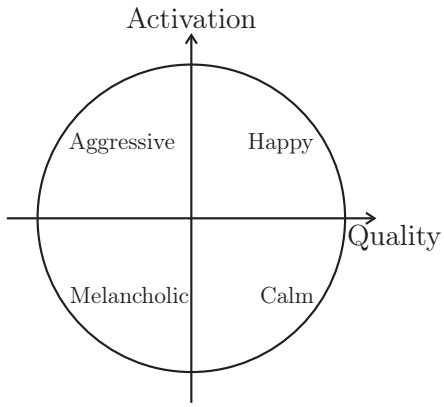


Figure 3.7: Thayer's model of mood, after [111]

3.5.2 User Survey

The playlist feature priorities listed in Table 3.2 show that both similarity (same mood, similarity) and variation are important properties of playlists for the different listening scenarios. These two properties are in conflict with each other when restricted to one single aspect of music similarity. Similar to the correlation analysis for variation and similarity in Figure 3.6, a correlation analysis on the other feature preferences with the feature importance judgments was performed. It was expected that preference for similarity on a certain feature is correlated with the importance judgments of the other features. When these correlations exist, a set of similarity and variation criteria can be created.

The strongest cross-feature Pearson's r that was found, occurs between 'same genre' and 'similar sound' at $r = 0.3$ for the universal listening mode. We therefore conclude that, for the group of user survey participants, there is no significant correlation between the similarity or variance preference and importance ratings. The participants' demands on playlists thus show too large variation to be captured by a small set of universal playlist quality criteria.

Some constraints on the songs in a playlist can still be posed, using the results of the questions on common listening behavior. Since the large majority of the participants (81.5%) indicated that music should match the mood one is in, mood continuity is a possible general playlist criterium. Liu et al. [51] and Tolos et al. [113] successfully performed mood classification using the model of mood defined by Thayer [111]. This model (see Figure 3.7) spans an *emotional space* by two orthogonal dimensions: *activation* and *quality*. Tolos

et al. performed a human mood classification experiment and found that there was low agreement between the ratings in the lower half of Thayer's model. It is suggested to only use three mood categories: 'Happy', 'Agressive' and the combination of 'Melancholic' and 'Calm'. They compare the user's ratings with automatic classification using a small set of timbral features and find that the automatic classification error is about as large as the level of disagreement in the human classification results. Liu et al. use a combination of timbral, rhythmic and 'intensity' features, where the intensity feature represents the volume of the music. A two-stage classification procedure, first classifying the activation dimension using the intensity feature and then classifying the quality dimension using the timbral and rhythmic features is proposed.

3.5.3 Summary

In this section we described that current content based playlist generation systems have been evaluated using artist and genre consistency as main criteria. These consistency measures express how well an audio similarity measure performs on genre or artist clustering tasks but are no criteria for the quality of a playlist. Constraint satisfaction approaches have not been evaluated on playlist quality, but on computational complexity. Playlist quality depends on the quality of the metadata and the available constraints.

The user survey provided no clear playlist quality profiles, but one constraint can be posed: songs in a playlist should match the current personal mood. Mood classification experiments using timbral, rhythmic and intensity features showed to be succesful [51, 113]. Succesful music genre classification systems make use of feature sets that exactly cover the domains that are also used by mood classification algorithms. Since both genre and mood classification make use of comparable feature sets, mood continuity and genre continuity may largely have similar effects on playlist quality.

The discrepancy between similarity criteria and the request for variation remains. In order to deliberately apply variation in a playlist, it is of vital importance to know what exactly is similarity in the first place. When 'variation' can be expressed as 'music with a certain distance to each other', content based playlist generation systems are very well able to generate playlists balancing on the borders between similarity and variations.

3.6 A new approach to playlist generation

The methods for playlist generation that are presented in this chapter, show that there are many possible strategies for selecting songs out of a large music database. In this section, the demands on the playlist generation system that was developed in the course of this research are presented. An architecture for the new procedure is proposed and the main functional blocks are discussed.

3.6.1 Demands

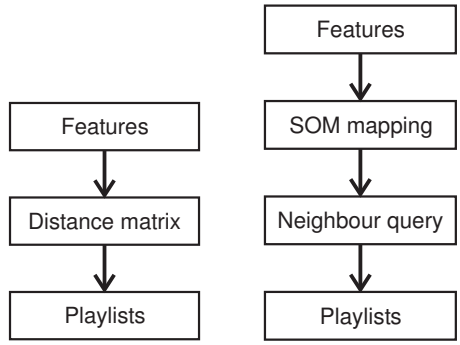
The questionnaire shows that it is impossible to define a universal set of playlist quality constraints. A playlist should contain similar songs, but at the same time have enough variation not to be boring. In order to consciously apply variation, one must first know what music is perceived as being similar. The first demand on the playlist generation system thus can be formulated as follows:

> A content-based playlist generation system should be able to find **similar** music.

The six music listening modes defined by Rauhe et al. [96] show the wide range of possible similarity criteria. Depending on the mode, the user may pay attention to the acoustic surface of the sound (diffuse reception), or focus on feelings associated with the music (associative-emotional reception). To encompass for individual similarity perception, the music similarity measure should be adaptable to the user's notion of similarity. User-controlled similarity measures showed to be appreciated over fixed similarity measures by Vignoli [119]. The second demand on the system can be formulated as:

> The music similarity measure should be **adaptable** by the user at playlist generation time.

Since the storage capacity of media devices is ever increasing, scalability of automatic playlist genertion systems is an important issue. Personal music collections with over 5000 songs are not uncommon anymore. Even on these large collections, the playlist generation should be fast. Processing times of up to a maximum of one second are not perceived as annoying. The user's flow of thought is not interrupted (Nielsen [72]). Processing time is not only relevant at playlist retrieval time. When a feature extraction or similarity training phase is needed, these should be of reasonable speed as well. Since these issues become more important at large systems, we can formulate the third demand as:



(a) Traditional architecture (b) Proposed architecture

Figure 3.8: Playlist generation architectures

The playlist generation system should **scale** well on large music collections

These three criteria will be the main design criteria for the new playlist generation system.

3.6.2 Proposal

In the previous sections, the current generation of playlist generation systems is presented, the user demands are analyzed and the requirements on a new playlist generation system are defined. The common architecture of the current generation of playlist generation systems is shown in Figure 3.8(a). A (set of) features is extracted from each song. The feature extraction phase is followed by a data reduction phase in which a probability density model of the features is generated (e.g. [3, 56, 80]). Using a suitable distance measure, a full song distance matrix is calculated. This distance matrix expresses the feature based similarity between all songs. The distance matrix serves as input for the playlist generation heuristic (e.g. [55, 80, 92]).

The disadvantage of the approach based on full distance matrices is the scalability. The distance between all song pairs has to be calculated. The size of the distance matrix is proportional with the square of the number of songs. With an asymmetric distance measure, a 5000 song distance matrix thus has 25 million entries. If new song gets added to the music collection, the distances to all songs already in the collection have to be (re)calculated.

In the proposed architecture (see Figure 3.8(b)), the distance matrix is

replaced by a self organizing map. Self organizing maps have been used in MIR systems for visualizing music databases and show good song clustering results. Mitri et al. [65] and Mörchen et al. [70] each use a fixed feature set to project the songs onto the map. Pampalk [77] trained several aligned SOM layers with different feature weighting, which allows visual browsing of a music collection for different similarity criteria.

The traditional playlist generation architecture uses the song distance matrix to find similar songs. In this thesis, the possibilities of using a SOM for playlist generation are explored. The assumed advantages of this approach are:

Finding similar music: Self organizing maps are capable of mapping high dimensional data to a low dimensional space, keeping distance relationships approximately intact. Given an appropriate featureset, 'similar' songs are thus mapped close to each other in the map. Finding similar songs thus only requires searching the neighbourhood of a song in the map, and not the entire database. To better represent the songs in the SOMs, the neurons contain Gaussian mixture models trained on the features.

Adaptable similarity measure: By training multiple SOMs on different features (equivalent to the work of Pampalk [77]), different similarity concepts are easily available at playlist retrieval time. By weighted linear combination of song distances on the SOMs, the similarity measure can be adapted to user needs.

Scalability: The proposed approach does not need a full song distance matrix, but requires only the mapped position of each song in the SOM. The amount of datapoints thus scales linearly with song database size, and not quadratic as in the distance matrix case. Since the mapped song coordinates are discrete values, neighbouring songs can be easily retrieved using efficient database queries. When adding a new song to the database, only the best matching neuron in the SOM has to be found. Typically, the number of neurons in a SOM will be much smaller than the amount of songs in the database. Pampalk [76] recommends the number of neurons to be in the range of $\sqrt{n}$, with n the number of data items.

3.6.3 Justification

Self organizing map based mappings of songs provide an intuitive method for exploring music databases (Rauber and Frühwirth [95]). The use of appealing

visualisations (e.g. *Islands of Music*, Pampalk [76]) further reduces the barrier of exploring new regions in unknown music databases. Pohle et al. [92] used circular SOMs for playlist generation, and found that after applying an artist filter, the generated playlists using the SOMs performed very good on a short- and long-term concistency quality measure.

Still, the performance of self organizing maps for music organization is limited. Each neuron in a SOM can only contain a single vector, which limits the song representation accuracy. Rauber and Frühwirth [95] uses a two-stage SOM mapping: the first stage SOM is trained on the raw feature vectors of the entire database. At the second stage, the feature vectors of one song are mapped to the trained SOM. Of each neuron in the SOM, the number of times it was selected as best match to an input feature vector is counted. These best match counts for the entire SOM form the 'neuron activation pattern'. The neuron activation patterns for each song in this first SOM are used to train the second SOM. Pampalk [76] has evaluated different strategies, using indexes of characteristic feature models and simple mean feature values over entire songs. Pohle et al. [92] use the first 30 components of a principal component mapping of each column of a full song distance matrix.

To extend the song representation accuracy in a SOM, it would be desirable to allow for more complex song models in the neurons of a SOM. This is possible as long as the two codebook requirements that are presented in Section 2.3.6 are met: There is a distance function to express similarity between training data and codebook entries and there is an algorithm that allows to adapt the codebook entry to better represent the training data.

Single Gaussians and Gaussian mixture models perform very well on music similarity tasks [59, 78, 79, 90]. To the best of my knowledge, both single Gaussians and Gaussian mixture models have not been used as codebook entries in self organizing maps thus far. For single Gaussian models, both codebook entry requirements are met: similarity can be measured by the Kullback-Leibler divergence [87] or by just taking the euclidean distance over the means of the Gaussians. Two Gaussians can be made more similar to each other by linear combination of the Gaussian mean and covariance matrixes. For Gaussian mixture models, multiple distance measures or approximations thereof are available. Berenzweig et al. [14] use EMD, ALA and centroid distances, Aucouturier and Pachet [4] use Monte-Carlo sampling. For the second requirement, however, no adequate method is known.

Background estimation processes in video surveillance applications use adaptive mixture models [108, 129]. These methods are designed to slowly adapt a mixture model to sequential single input samples. To realize signif-

icant changes in a mixture model, a high number of samples is required and in order to accurately represent a song, a high number of samples have to be drawn from a song model. For determining model similarity using Monte-Carlo sampling, Aucouturier and Pachet [6] recommend using at least 1000 samples for a mixture model consisting of 3 Gaussians in 8-dimensional space. For accurate sequential adaption, a even higher number of samples can be expected to be necessary. In Section 4.4 a heuristic is presented that allows updating a Gaussian mixture model to better match a second Gaussian mixture model without having to draw random samples from one of the mixtures.

Once all songs are mapped in the SOM, the playlist generation heuristics as presented by Pampalk et al. [82] can be used. Instead of navigating through the origininal feature space, the heuristics now use the low-dimensional SOM space.

The entire process from signal analysis to playlist generation now consists of the following steps:

1. Extract the desired features from the audio data and build statistical model per song.
2. Train SOM using a set of representative training data.
3. Map all songs to the trained SOM.
4. Select playlist seed song and find position in SOM.
5. Query for neighbouring songs in SOM.
6. Apply playlist generation heuristics.

To allow individual similarity measures for different users, the neighbouring song query can be combined over multiple SOMs, trained on different features as shown in Figure 3.9. The distances of each song pair in the individual SOMs are weighted with a set of personalized weight factors and summed up. The resulting list of neighbouring songs thus depends on the individual feature weightings. Since these distance computtations are in the low-dimensional SOM space, they can be performed real-time.

3.7 Summary

This chapter provided insight in playlist generation systems. In the first section, different use-cases for playlists and the accompanying user demands on

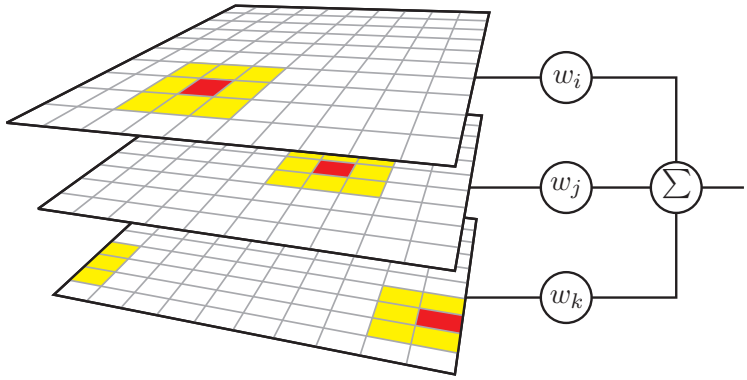


Figure 3.9: Combining multiple SOM mappings for playlist generation. The fields shaded red contain a playlist seed song and all songs within the yellow-shaded fields belong to the neighbourhood of the seed song. By applying weights w_i, w_j and w_k for each user, a personal notion of similarity can be realised for each user.

these playlists is analyzed. A short tour in the psychological aspects of listening to music shows the benefits of selecting music matching the personal taste and preferences of a listener.

Section 3.3 presented a review of current playlist generation algorithms. This section is followed by a report of a questionnaire on user's listening behavior and requirements on personal playlists that was developed and performed during this research. It shows that the participants of the questionnaire use a wide range of playlist quality criteria. These criteria are largely the same for three identified listening modes.

In section 3.5 the quality criteria currently in use for evaluating the quality of playlists generated by automatic playlist generation systems are discussed. The results of the user survey are analyzed with respect to quality criteria. It is reasoned that playlist consistency with respect to genre labels is a reasonable method to evaluate playlist quality.

The last section presents the new approach for playlist generation that was developed in the course of this research. Using a SOM for mapping high dimensional song models to a low dimensional space allows efficient neighborhood queries. Combining several SOMs by using a weighted sum of SOM feature distances can provide each user with an individual similarity measure. This similarity measure can be adapted real-time.

Chapter 4

Design

4.1 Introduction

In the previous chapters, the basics of content based music genre classification and playlist generation systems were described. Section 3.6 describes the new approach to playlist generation that was developed in the course of this research. In this chapter, the realization of the system is described.

Section 4.2 describes the general architecture of the framework that has been developed. The requirements on each component of the framework are analyzed and the structure of the main elements is described. The next section describes the principles of model complexity estimation: a first step towards reducing model complexity for each individual song. Section 4.4 presents the design issues for using self organizing maps with Gaussian mixture models and section 4.5 shows how to use these self organizing maps for efficient playlist generation. The chapter ends with a short summary.

4.2 Framework Architecture

4.2.1 Introduction

Apart from the perceived quality of a generated playlist, other research and design criteria also play a major role when designing a complete content-based playlist generator framework. The (subjective) playlist quality criterium becomes far less critical when generating a short playlist takes several hours while the user only wants 15 minutes of music but starting right away.

In the previous chapters, the individual components of content based playlist generation systems are introduced, in this section they are placed

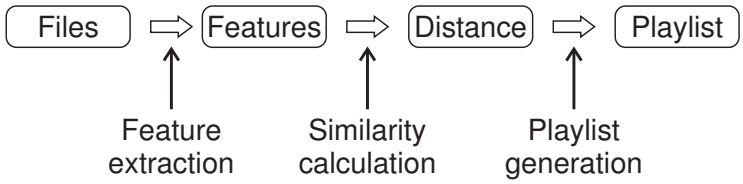


Figure 4.1: Identification of playlist generation process steps

in the context of the entire framework. The presented architecture aims at realizing a system that realizes the following design criteria:

Quality: The system shall use state of the art methods for determining music similarity, and aims at obtaining robust, personal music playlists.

Flexibility: The system shall be easily extensible with new content-based features, without having to recompute previous music similarity results. Adding new music to the database does not require re-analyzing the entire database again.

Portability: The system aims at reducing the amount of data to be stored to a minimum, to allow operation on portable devices.

Speed: The system implements efficient music similarity measures at playlist generation time.

In this section, the general framework architecture is described. The major functional blocks are placed in their context and the program flow is explained.

4.2.2 Architecture design

Before designing a playlist generation framework, all functional blocks have to be identified. The simplified architecture of Figure 3.8(a) shows the elements of a traditional playlist generation system: features, distance and playlists. The operations that have to be performed in these elements can be designed to fulfill the four design criteria. These operations are shown in Figure 4.1. It are these processing steps that form the three main functional blocks of each content based playlist generation system.

The four design criteria (quality, flexibility, portability and speed) are not of equal importance to all of the main three functional blocks of the playlist generation system. In Figure 4.2 the critical criteria for the three functional blocks of a playlist generator framework are identified. In the following paragraphs, these criteria are explained in more detail.

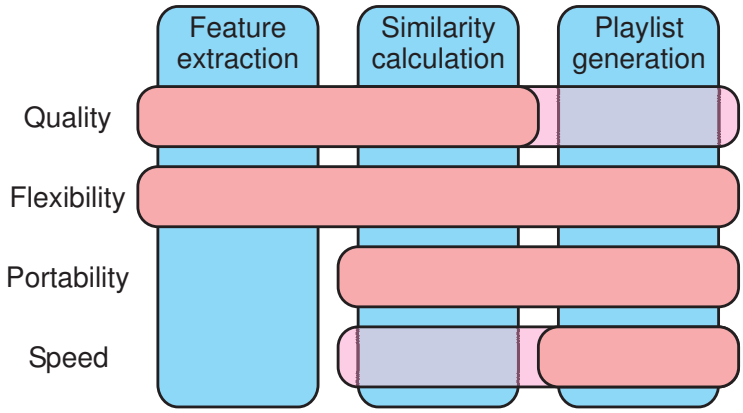


Figure 4.2: Identification of critical criteria for the three functional blocks of a playlist generator framework. The semitransparent selections indicate that corresponding criterium is important for the functional block, but that it is not a main criterium.

Quality criterium

The quality criterium states that the system shall use state of the art methods to obtain robust, personal music playlists. This criterium is most important to the feature extraction and similarity calculation blocks. In order to be able to generate a 'good' playlist, the system has to be able to reliably identify 'similar' music or music that is preferred by a user. Of the factors influencing music similarity or preference perception that are described in Section 2.2, only the audio content is available for content based similarity analysis.

Even if the rules for creating a 'perfect' personal playlist based on a detailed representation of the audio signal would be known, systems need an accurate description of the audio signal and measures that are able to express the personal similarity perception based on these descriptions. The feature extractor thus has to be able to extract at least the most relevant features that are in use for music similarity tasks. These features have to be accessible to similarity measures, to express the similarity between two individual songs as a function of the personalized weighted sum of similarities measured on the individual features.

The playlist generation step itself is also identified as being a relevant parameter for overall quality of the playlist. Since any playlist that is automatically generated based on a content based music similarity analysis ultimately depends on the quality of the expressed music similarity, the quality criterium is more relevant for the first two functional blocks.

Flexibility criterium

The flexibility criterium has a twofold goal: it aims at keeping the music similarity measures used for generating the playlist up-to-date with the newest available features and the system shall allow single songs to be added to a music collection without having to recompute an entire similarity matrix over all songs in the database. This criterium thus poses constraints on all three functionality blocks: the feature extractor must be easily extensible with new features, the similarity calculation must be able to use new features, and the playlist generation part must not require a full song distance matrix expressing the individual song distances for all songs for the current set of features.

These constraints thus require strong modularity of the feature extraction and similarity calculation parts of the framework: adding a new feature to the feature extraction functional block should not require an entire rewrite of the block, but should allow reuse of functions and intermediate results of other, similar features. The features should be available to the similarity calculations in an abstract container format that allows expression of similarity based on each individual feature, independent of the feature format and value range.

Portability criterium

The mass market for media players is dominated by portable devices. Unlike home or desktop systems, the computational power and the available resources on these devices is quite limited. The amount of data to be stored that is necessary for similarity calculations thus should be kept to a minimum, while keeping the quality of the description as high as possible. Compact data descriptions both save storage space and reduce the complexiness of operations on these data: many distance measures between data models have a complexity that is linear or even quadratic with the level of detail of the data description.

To some extent, the music similarity relations required for playlist generation algorithms can be calculated off-line: Similarity relationships based on individual features can be pre-calculated on a desktop system, the result of these pre-calculations can be used by a playlist generation algorithm at playlist generation time on the portable device.

Playlist generation algorithms based on a full song similarity matrix do not scale well: the size of the similarity matrix increases quadratically with the number of songs in the song database. Algorithms not requiring a full song similarity matrix thus are preferred.

Speed criterium

One of the major annoyances of using portable devices with limited computational power is having to wait for a result of user input. Response time is one of the most important criteria for user acceptance [103]. Of the entire process of playlist generation, only the actual playlist generation is time-critical. A personal music collection does not change every minute, feature extraction and pre-calculation of partial song similarities have only to be performed on adding a song to the collection. For the actual playlist generation algorithm to be fast enough, it should operate on an absolute minimum amount of data: song representations should be as compact as possible and wherever applicable, song similarities should be pre-calculated without using flexibility for applying user-adaptable similarity measures.

Proposed framework architecture

Not all of the four design criteria pose restrictions on the framework architecture. The quality criterium only states that the features and similarity measures to be used should be state of the art. The flexibility criterium requires a modular music similarity measure and no full distance matrices at playlist generation time. The portability and the speed criterium both aim at data reduction, where the speed criterium is more relevant at playlist generation time. Since speed is not an issue at feature extraction time, an architectural separation between the feature extraction and playlist generation process lies at hand.

The architecture that was designed to meet the design requirements, consists of three major functional blocks, depicted in Figure 4.3. The feature extraction and modeling module provides a set of modules for extracting a range of low level features and modeling those features as a first data reduction phase. The central song(model) management block stores the song feature models and metadata in a database and provides access to the data for the playlist generation module. This third module implements a self organizing map for mapping the songs to a low dimensional space, and provides playlist generation heuristics. These three functional blocks will be described in detail in the following sections.

4.2.3 Feature Extraction and Modelling

The task of the feature extraction and modelling module is to access and decode the audio files and then extract the selected features from the audio

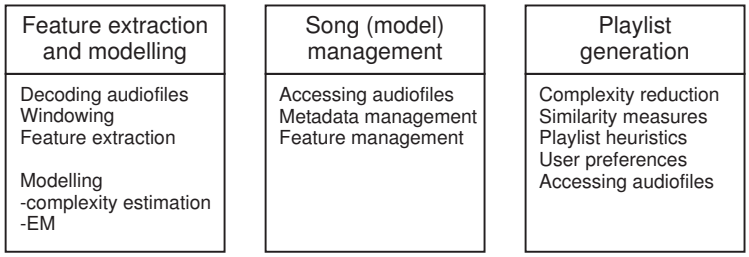


Figure 4.3: Principal components of the playlist generation framework architecture

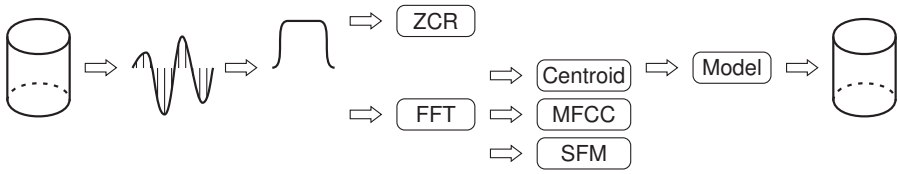


Figure 4.4: Program flow of feature extractor

data. The feature extraction phase is followed by a feature modelling stage. After modelling, the feature models have to be made available to the rest of the framework. The program flow within the entire feature extraction and modelling module is shown in Figure 4.4.

Structure

The module was implemented as a separate executable, consisting of the components as shown in Figure 4.5. The first module reads the extraction and modelling parameters from the command-line. These parameters are passed to the network management module. This module has the task to initialize the signal processing components in the module and allocate and manage the required buffers to store intermediate and final feature values and models thereof. After processing and modeling the features of a single song, the results are written to either an XML or a Matlab file.

Feature Extraction Components

The feature extraction process is mapped to a small number of modules performing basic operations on the signal flow. The audiofile is first converted to a series of samples. Features based in the time domain (e.g. ZCR) directly access these sample series. For frequency-based features the time series are

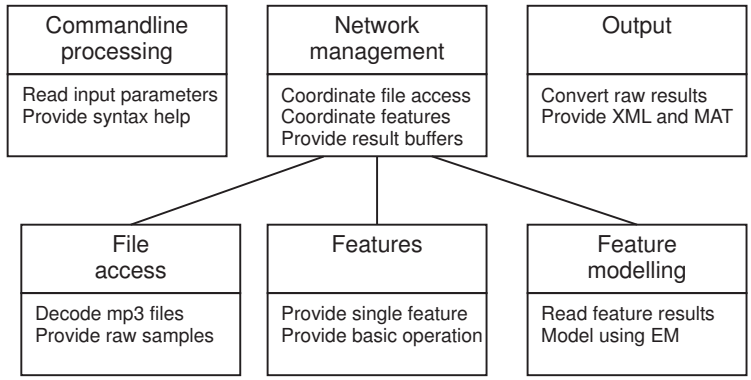


Figure 4.5: Feature extractor components

windowed using a Hann or raised cosine window before performing an FFT. During the experiments on playlist generation, a windowsize of 20 milliseconds was used. After processing one window, a next window is formed by shifting the time offset by 10 milliseconds.

Feature Modeling Modules

The feature modeling modules allow modeling the extracted features with a K-means model or with a Gaussian mixture model. Selecting the initial clusters of a K-means model has large impact on the quality of the model (see Peña et al. [86] for a comparison of several methods). The framework implements a variation of the Kaufman initialization algorithm: K samples are selected that have a maximum distance to each other. These samples are selected from a random subset of the data. The Gaussian mixture modeling module uses the K-means modeling module for initializing the initial cluster positions.

The major disadvantage of traditional KMM and GMM approaches, is that the number of clusters for modeling a dataset has to be known in advance. When not specifying a number of clusters, the framework uses the basic sequential algorithmic scheme (BSAS, see page 84) algorithm to determine the number of clusters to use and initialize the initial means.

Output

The extracted features and feature models have to be made available to the rest of the framework. To this goal, the MPEG-7 standard has been used. This standard provides an XML scheme for multimedia content description (see Martinez et al. [60]). The raw features can also be exported to a Matlab

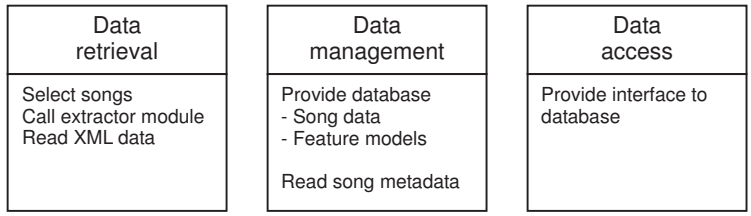


Figure 4.6: Song (model) manager components

data file for easy visualizations.

4.2.4 Song (model) management

The task of the song (model) management module is to manage a database with song data, to call the feature extractor for extracting the desired feature data and import them into the database, and to make the feature models available to the playlist generation module.

Structure

The song (model) management module consists of three blocks, as is shown in Figure 4.6. The data retrieval block allows the user to select the features, parameters thereof and modeling options using a graphical interface. Once the data retrieval component is started, the path and names of the files to be analyzed are retrieved from the data management module and the settings are converted to the commandline syntax of the feature extractor. After extracting and saving the desired features of one file to disk using the MPEG-7 XML format, the data are parsed by the XML parser of the data retrieval block and then presented to the data management module.

The data management module keeps the database structure up-to-date and extracts and stores additional song metadata from each song. These metadata include artist and album name, track numbers and song genres. These data are read from the ID3 tags available in most mp3 files.

Song models and metadata can be retrieved from the database by the functions of the data access module. The data are available by ID or by path and filename.

Database structure

The database is implemented as a relational database, using linking tables to link each song to its author, album and other data. The entire database

Figure 4.7: Structure of the relational database. All information that can be assigned to multiple songs (artist name and album name) is stored in separate tables

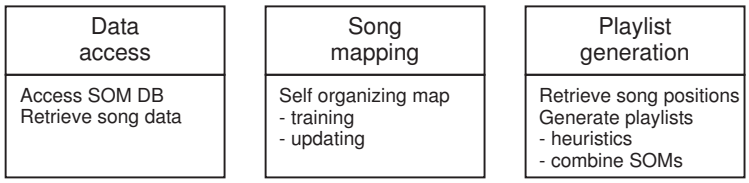


Figure 4.8: Architecture of the playlist generation module

structure is shown in Figure 4.7. Each song's feature model is stored in the corresponding feature table. The SongID/SOMID table stores the mapping of a song in a self organizing map used for playlist generation, which will be described in the next section.

4.2.5 Playlist generation

The ultimate task of the playlist generation module is to generate the music playlists based on the song feature model based data. Since the speed criterium is important mostly at playlist generation time, an additional data reduction phase is introduced in this module. This data reduction is performed by mapping the song feature models to a low dimensional space in a self organizing map, see Section 3.6.2 for an outline of this procedure.

Structure

The playlist generation module consists of three components, shown in Figure 4.8. The two central components are the song mapping and the playlist generation blocks. In the SOM mapping module, the high dimensional song feature modules are mapped to a two-dimensional self organizing map.

Song mapping

Since the standard SOM implementations only work on vector data, and not on Gaussian mixture models, an algorithm enabling the use of GMMs in SOMs was designed. This algorithm is presented in Section 4.4. After training a SOM, the mapped song positions are stored in the database. Each entry in the database consists of four integer values: the song ID, the SOM ID, and two integer coordinates in the SOM.

Playlist generation

The playlist generation block uses the mapped song positions to determine song similarities, songs lying close to each other in a SOM have a high similarity. Multiple SOMs, trained on different features, can be combined to obtain a (personal) weighted song similarity measure. The process of playlist generation is described in more detail in Section 4.5.

4.2.6 Operational architecture

The architectural elements described in the sections above all constitute part of the necessary signal processing chain. As the software framework is designed for experimenting with playlist generation, it allows experimenting with the individual components and reusing previously computed results. In order to generate a playlist the following steps have to be performed:

The first step is to select the features and feature parameters that are to be extracted from a set of songs. Multiple features can be extracted in one single run, for later processing using multiple SOMs. After selecting the features, the modelling parameters for the EM algorithm for calculating GMMs of each song's features have to be set. The last step of the feature and modelling phase is to select the location where the songs are located and start the feature extraction phase. The extracted features are saved in a database as shown in Figure 4.7.

After the features have been extracted, the size and training parameters of a new SOM can be set. These parameters include the number of training iterations to be performed and the distance measure that is to be used for determining song model similarity. Instead of using randomly initialized neurons, a set of songs can be selected for initializing the neurons with. Once the SOM is initialized with existing song models, a larger set of songs can be selected for training the SOM. After successfully training a SOM, the SOM can be saved to a database for later use. The size of the SOM in the database only depends

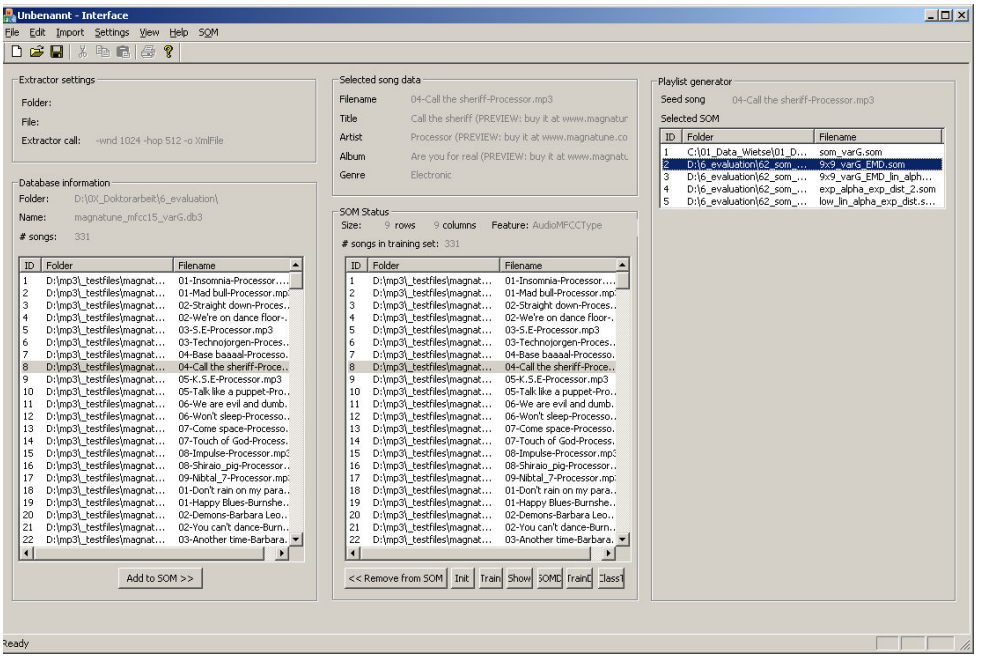


Figure 4.9: Screenshot of the framework interface

on the number of songs mapped to the SOM: each song requires four integer values. The actual size in bytes depends on the database implementation.

When one or multiple SOMs have been trained and saved in a database, playlists can be created. As described in Section 4.2.5, one or multiple SOMs that are stored in the database can be selected. After selecting the database(s), a seed song can be picked from the list of songs in a SOM by doubleclick.

4.3 Model complexity estimation

4.3.1 Introduction

In the study on music similarity measures performed by Aucouturier and Pachet [6], the number of clusters of the statistical models of the song features, was identified as being a parameter to be optimized. Based on the *Occam's razor* theorem, which states that the explanation of any phenomenon should make as few assumptions as possible, while attaining a certain accuracy about the observed phenomenon, information criteria have been developed that yield an optimal tradeoff between model complexity and accuracy of the model. In

this section, several optimization criteria are described, algorithms for optimizing these criteria are discussed and the implemented method is presented.

4.3.2 Optimization criteria

Let $\mathcal{M}_k$ be a class of models having k free parameters ($\boldsymbol{\Theta}$). The quality of fit of a model of class $\mathcal{M}_k$ on a dataset $\mathcal{S}$ can be expressed by the ML criterion. The maximized likelihood estimate on the data is a non-decreasing function of k since $\mathcal{M}_k \subseteq \mathcal{M}_{k+1}$. Finding a good model for representing the data $\mathcal{S}$ thus seems to be a case of using a model with many free parameters (complex models): the ML estimate is ever increasing.

The problem with models of high complexity is their lack of generalizing properties. The data the model is trained upon may be fitted very well, but other data generated by the same source is not represented well. This problem is called *overfitting*. There are various criteria available for expressing the 'optimality' of a model, where an 'optimal' model has a certain tradeoff between goodness of fit of the data, and the number of parameters of the model: The number of parameters is selected according to criterium:

$$\hat{k} = \arg\min_k \{\mathcal{C}(\hat{\boldsymbol{\Theta}}(k), k)\}, \quad k = k_{\min}, \ldots, k_{\max} \tag{4.1}$$

with $\mathcal{C}$ the criterium function, consisting of a part rewarding the goodness of fit of the model and a part penalizing a high number of parameters.

Bayesian Information Criterium

The Bayesian information criterium (BIC) [106] consists of the maximum likelihood estimate of the model on the data and a penalty function dependent on the number free parameters and the number of samples in the dataset:

$$\text{BIC} = -2\log(p(\mathcal{S}|\hat{\boldsymbol{\Theta}}_{ML}) + k\log(n) \tag{4.2}$$

with $\mathcal{S}$ the dataset, $\hat{\boldsymbol{\Theta}}_{ML}$ the set of model parameters, k the number of parameters in the model and n the size of the dataset. Minimizing the BIC yields a maximally informative model.

Minimum Description Length Criterion

The Minimum Description Length (MDL) criterion [98] is based on the concept of *Kolmogorov complexity*. This concept states that data compression is a form of learning regularities of the data. The shortest representation of data $\mathcal{S}$ is

given by a program $\boldsymbol{\Theta}$ written in a general purpose programming language, that outputs the datasequence $\mathcal{S}$. According to Shannon, the shortest length for representing $p(\mathcal{S}|\boldsymbol{\Theta})$ is given by $-\lceil {}^2\log(\mathcal{S}|\boldsymbol{\Theta})\rceil$ (bits). For decoding these data, the recipient must also know the program $\boldsymbol{\Theta}$. The entire length required for encoding and transmitting the data is thus given by:

$$\text{Length}(\boldsymbol{\Theta}, \mathcal{S}) = \text{Length}(\mathcal{S}|\boldsymbol{\Theta}) + \text{Length}(\boldsymbol{\Theta}) \tag{4.3}$$

which is the criterium that MDL is trying to minimize.

The optimal number of model components c according to the MDL criterion can be found by:

$$\hat{c}_{\text{MDL}} = \arg\min \left\{ -\log p(\mathcal{S}|\hat{\boldsymbol{\Theta}}(c)) + \frac{c}{2}\log n \right\} \tag{4.4}$$

which is equivalent to the minimization problem of the BIC.

4.3.3 Algorithms for complexity estimation

There is a variety of algorithms for complexity estimation available. In this section, several different concepts are described.

X-Means

X-Means (Pelleg and Moore [85]) is an extension of the traditional K-means algorithm that makes use of the BIC to estimate the optimal number of clusters given a certain dataset. The algorithm consists of a parameter optimization phase and a structure optimization phase, that are repeated iteratively. The parameter optimization phase works like the traditional K-means algorithm: given the current cluster means, the average squared distance from data points to their cluster means is optimized. After this optimization phase, the model structure is adapted: Each cluster is split in two clusters, and a K-means algorithm is ran on these two clusters to divide the data points originally belonging to the 'parent' cluster between the two new clusters. Determining whether the split was successful is done by assessing the BIC for both the splitted and original configuration. After each parameter/structure iteration, the BIC score is saved for that configuration.

This entire process is repeated for K starting at a lower bound until a maximum K is reached. The optimal model is the model that has the lowest overall BIC score.

Gaussian Merging Splitting

The Gaussian Merging Splitting (GMS) algorithm (Sankar [104]) is an iterative algorithm that can both split and merge Gaussians. The number of Gaussians to split can be limited to a fixed number, only the Gaussians with the highest variances are split.

The Gaussian merging is performed with an agglomerative clustering procedure. The clustering distance is a weighted function of increase in entropy and the number of samples belonging to the two clusters. Clusters are merged until all clusters have at least a threshold amount of data.

Figueiredo

Figueiredo and Jain [28] present a method for unsupervised learning of finite mixture models, using the minimum message length criterion. The method uses EM for fitting a GMM in a dataset $\mathcal{S}$ and starts with a high number of mixture components. By choosing this number of components much higher than the expected optimal number of components, the algorithm is robust with respect to initialization: the EM is guaranteed to find a local optimum estimate and is not able get out of a local optimum by itself.

The EM method of Figueiredo has a M-step that can annihilate components with little support of the data. By iteratively removing mixture components with low support from the data, it is likely that the eventual solution has clusters there, where they are maximally supported. Since when starting with a large number of components, there might be many clusters having little support, a component-wise EM algorithm (CEM, Celeux et al. [19]) is used. When the CEM algorithm with the annihilating M-step has converged, and the number of components still is above a threshold minimum, the least probable component is annihilated. During the entire annihilation process, the value of a model optimization criterium is tracked for each number of components. After reaching the minimum threshold number of clusters, the number of components that has the best value for the optimization criterium is picked as the optimal number of components.

Basic Sequential Algorithmic Scheme

Sequential modeling algorithms can be used for finding clusters in a dataset, when the number of clusters is not known in advance. The most basic algorithm is the 'Basic Sequential Algorithmic Scheme' (BSAS [112]). As can be seen from the following pseudocode, BSAS only has two parameters: The threshold Θ for

Figure 4.10: Complexity estimation flow

determining whether a new cluster has to be formed, and N_{maxClust}, the maximum number of clusters to be formed. With N_{clust} the number of clusters during the estimation and C_i the set of samples $\boldsymbol{x}$ assigned to the cluster with index i, the BSAS algorithm has the following steps:

1: $N_{\text{clust}} = 1$
2: $C_1 = \{\boldsymbol{x}_1\}$
3: **for** $j = 2$ to n **do**
4: find C_i: $d(\boldsymbol{x}_j, C_i) = \min_{\forall k} d(\boldsymbol{x}_j, C_k)$
5: **if** $(d(\boldsymbol{x}_j, C_k) > \theta)$ and $(N_{\text{clust}} < N_{\text{maxClust}})$ **then**
6: $N_{\text{clust}} = N_{\text{clust}} + 1$
7: $C_{N_{\text{clust}}} = \{\boldsymbol{x}_j\}$
8: **else**
9: $C_k = C_k \cup \{\boldsymbol{x}_j\}$
10: **end if**
11: **end for**

By choosing an appropriate value for threshold Θ, the number of model clusters found by more complex algorithms can be approximated reasonably well. The use of the BSAS for music similarity measures is assessed in Section 5.2.

4.3.4 Implementation of model complexity estimation

The number of clusters in a song model has direct influence on song model similarity calculations, the quality of the song representations and on the size of the extracted data. For this reasons, a model complexity estimation is implemented in the framework. In [11], we describe a method using the simple BSAS algorithm for approximating the number of clusters found by Figueiredo's algorithm (Figueiredo and Jain [28]) is described. This paper is reproduced in Section 5.2. The signal flow in the complexity estimation block consists of the three functional elements as depicted in Figure 4.10.

In the first block, the threshold parameter Θ for the BSAS algorithm is determined. It shows that choosing Θ according to the following equation results in an approximately equal number of components as provided by Figueiredo's

algorithm:

$$\Theta = \sqrt{\text{trace}(\text{cov}(\mathcal{S}))} \tag{4.5}$$

with the trace being the sum of the diagonal elements of the covariance matrix.

Given parameter Θ, the BSAS algorithm is used to find an initial clustering result, which serves as input for a traditional EM algorithm. This process is further described in Section 5.2.

4.3.5 Summary

In this section, an introduction to model complexity estimation was given. The tradeoff between the data description accuracy and the generalizing properties of a model was described and several optimization criteria were presented. These criteria were followed by a selection of algorithms using these criteria for determining the number of model parameters (the model complexity). In the last section, the architecture of the model complexity estimation block was shortly described.

4.4 Self Organizing Map

4.4.1 Introduction

As is described in Section 2.3.6, Self Organizing Maps can be used for mapping high dimensional data to a low dimensional space as long as two conditions are fulfilled:

- A distance measure with monotonic decreasing distance function between training data and a codebook entry is available.
- There is an algorithm for adapting a codebook entry to better represent the training data.

When using GMMs as codebook entries, only the first condition is fulfilled. For the second condition, no applicable algorithm has been found.

In this section, an algorithm for adapting a GMM codebook entry is presented and the algorithmic details of training a SOM having GMMs as codebook entries are explained.

4.4.2 Updating Codebook Vectors

The training phase of a SOM is a two-step iterative process: For each song GMM in the trainingset, first the 'winning' codebook entry is determined, then

this codebook entry and its neighbors are adapted towards the training song GMM. This process is repeated for several iterations with decreasing adaption strength and decreasing neighborhood size. For determining the winning codebook entry with a GMM codebook, any distance function between GMMs that is monotonic decreasing can be used. Given such a distance measure, the second condition can be rephrased:

> There is an algorithm for adapting a codebook entry to decrease the distance between an instance of the training data and a codebook entry, given a certain monotonic decreasing distance measure.

On-line adaption of GMMs

There is a multitude of algorithms available that allow on-line adaption of GMMs (e.g. [40, 109]). These algorithms adapt a GMM as soon as new samples are presented to the algorithm: when a new sample lies within a certain distance of one of the clusters (usually expressed in terms of a multitude of the standard deviation of the cluster) of the already existing GMM, this sample is added to that cluster and its parameters are updated. When the new sample lies outside of this threshold, new clusters can be initiated.

Song and Wang [107] introduce an intermediate step, by first modeling a series of newly observed data values with a separate GMM. All clusters in this temporary GMM are compared with the clusters in the existing GMM. For each cluster in the temporary GMM, statistically equivalent clusters are searched in the existing GMM. When these are found, the temporary cluster is merged with the cluster in the existing GMM. Temporary clusters not having an equivalent cluster are transferred to the new GMM. The algorithm can create new clusters until a maximum number of clusters has been reached.

New approach

When training a SOM with GMMs as codebook entries, and having song GMMs as training data, requiring samples for on-line updating of the codebook entries is undesirable. Drawing representative samples from a GMM is a computational expensive operation, in order to well represent a statistical model, the number of samples should be large.

The approach presented by Song and Wang [107] allows merging of two GMMs, but will eventually cause each codebook entry having the maximum number of allowed clusters. Since the computational cost of distance mea-

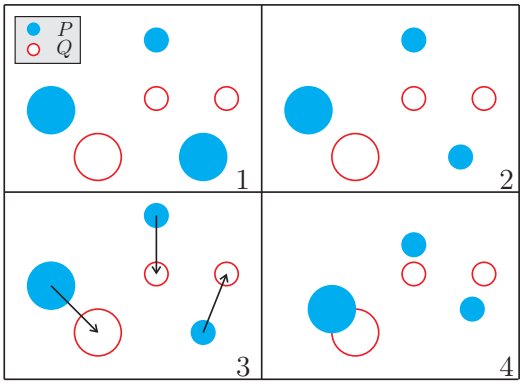


Figure 4.11: Four steps for adapting GMMs using earth mover's flow. P an Q are two GMMs in a two-dimensional space. The arrows in the third step show the flow between P and Q.

sures between GMMs are linearly or quadratically dependent on the number of clusters in a GMM, this is not desirable.

The main requirement that the algorithms described in the previous section do not fulfill is the lack of an *adaption strength* parameter, as is required for training a SOM (see Equation 2.36). For this reason, a new algorithm that allows to adapt a GMM to another GMM with an adaption strength factor α has been designed. The algorithm is based on the earth mover's distance and uses the 'probability mass flow' for determining which clusters from training and codebook GMM relate to each other.

The algorithm consists of four steps, which are illustrated in Figure 4.11. Let P be a codebook GMM and Q a training GMM.

1. Determine the distances between all clusters of P and Q using the same distance measure as used in the EMD. Store the distances in matrix D.

2. For each cluster in P, adapt the weight to the nearest cluster's weight in Q with factor α.

3. Normalize the weights in P and deterine the EMD flow between P and Q. Store the flow components in matrix F. Calculate the cost matrix $C_{(i,j)} = D_{(i,j)}F_{(i,j)}$.

4. For each cluster in P, adapt the mean and covariance to the cluster in Q having the highest cost with factor α.

The details and evaluation of this method are described in Section 5.4.

4.4.3 Training Self Organizing Maps

Once a song database, distance measure and codebook adaption algorithm are available, several other parameters have to be set. These parameters are the number of codebook entries and the size of the SOM, the neighborhood function $\phi(t, d_{\text{SOM}})$ determining the shape and size of the neighborhood, and the form of the monotonic decreasing function $\alpha(t)$.

The size of the SOM is directly related to the 'resolution' of the map: the more codebook entries, the smoother the transitions between neighboring codebook entries will be. The quality of the mapping in the SOM can be measured with the average quantization error (the mean of $\|x - m_c\|$ with x the training data and m_c the codebook entries) or the average distortion measure (the mean of $\phi\|x - m_c\|^2$, with ϕ the neighborhood function). Note that a SOM having the same number of codebook entries as it has training data achieves zero quantization error when each training data item is just mapped to one single codebook entry. There exists no general rule for estimating an optimal number of codebook entries given the number of training data: data of low variance requires less codebook entries to obtain the same quantization error as data of high variance with a larger SOM.

In Section 3.4 it was shown that a playlist quality criterium is very personal, requirements on subsequent song similarity show great variation. As the size of the SOM determines the resolution of the similarity captured in the SOM, no strict rule for the size of a SOM can be given. The test database that was used consisted of 331 songs of 6 distinct genres. A SOM of 9×9 neurons showed good clustering results on this dataset. The SOM was setup in a circular fashion: the neuron at position $(4, 0)$ is a direct neighbor of the neuron at position $(4, 9)$. The distance between two neurons is determined using the normalized euclidean distance on the coordinates in the SOM. Since the SOM is circular, the maximal normalized distance is $1/\sqrt{2}$.

A KNN genre classification task was used for evaluating the quality of the song mapping in the SOM for different neigborhood functions $\phi(t, d_{\text{SOM}})$. On the test set, the optimal function for $\phi(t, d_{\text{SOM}})$ showed to be:

$$\phi(t, d_{\text{SOM}}) = \alpha(t) e^{-500 \cdot d_{\text{SOM}}^2 \cdot t} \tag{4.6}$$

Using this function, different settings for $\alpha(t)$, the learning rate, are explored. Figure 4.12 shows the obtained genre clasification accuracies for different functions for $\alpha(t)$. Functions with exponential and linear decay are evaluated. The exponential decay variations use the base function $\exp(-4 \cdot \frac{t}{t_{\max}})$, the lin-

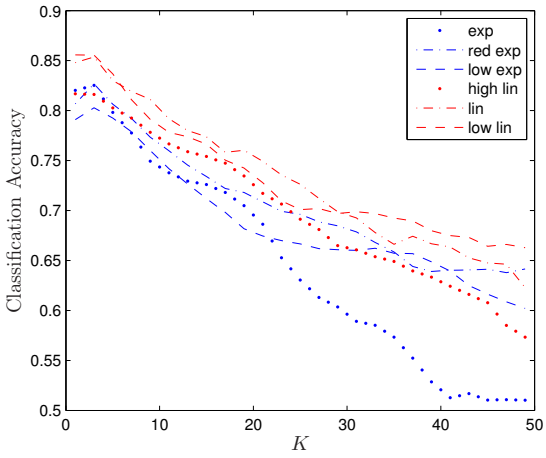


Figure 4.12: KNN SOM genre classification results

ear decay variations use the base function $\frac{t_{\max}-t}{t_{\max}}$. These base functions were multiplied with a fixed value between $[1 \ldots 0.3]$.

The best combination showed to be the 'low lin' setting: the linear basis function for $\alpha(t)$ multiplied with a factor of $(1/3)$. For low values of K, classification accuracy is almost as good as for 'lin', but for high K, it outperforms all other combinations. $\phi(t, d_{\mathrm{SOM}})$ with this optimal value for $\alpha(t)$ is plotted in Figure 4.13 for ten iterations.

4.4.4 Summary

In this section, a heuristic for adapting two GMMs using the earth mover's flow was presented. This heuristic allows adapting two GMMs with a factor α, which allows GMMs to be used in SOMs. The functions used for determining α for each iteration and neighboring distance were presented.

4.5 Playlist Generation

4.5.1 Introduction

The previous sections describe the process of optimal model generation and mapping these song models in a SOM. In this SOM, those songs that are similar with respect to the extracted feature are mapped close together. This section describes the architecture that enables fast and efficient playlist generation, while satisfying the requirements of finding similar music according to

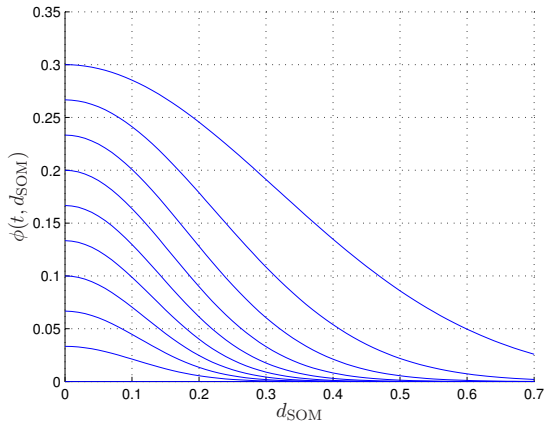


Figure 4.13: Neighborhood function for 10 iterations. $t = 0.1 \ldots 1.0$

an adaptable similarity measure and being scalable to larger databases.

4.5.2 Playlist Retrieval

As was described in Section 4.2.4, the mapped song positions in each SOM are stored in a relational database (Figure 4.7). The SOMID - SongID table has the following structure (the shown table entries are arbitrary example values):

SOMID	**SongID**	**Pos$_X$**	**Pos$_Y$**
1	23	3	4
1	24	4	4
...	...	...	...

Table 4.1: Structure of the SOMID - SongID table

The playlist generation process is based on nearest neigbor queries of a seed song. For one single SOM, the positions of all songs in the SOM are retrieved. Once a seed song of the playlist has been selected, the distances of all other songs in the list to this seed song are calculated using the euclidean distance measure on the mapped song positions. The resulting distances to the seed song are then used to sort the list on increasing distance. This list of songs is the basic playlist generation heuristic (heuristic **a** in Figure 3.1) that was presented by Pampalk et al. [82]. Since during the playlist generation only the mapped positions in the SOM are used, the playlist generation heuristic is computationally inexpensive.

4.5.3 Adaptable similarity measure

In order to be able to provide a personalized similarity measure, even for large music collections, the personalized distance between two songs can be expressed as a weighted sum of distances on individual features:

$$D(A\|B) = \sum_{i=1}^{N} w_i D_i(A\|B) \tag{4.7}$$

$$\sum_{i=1}^{N} w_i = 1$$

where $D_i(A\|B)$ is the distance between songs A and B based on the feature with index i, w_i the feature weights and N the number of features that are available.

Instead of computing one single SOM based on a weighted combination of features, multiple SOMs can be trained on independent features. After selecting a seed song, the individual song distances in the set of SOMs can be combined using Equation 4.7. By allowing the user to individually adapt the w_i, a personalized distance can be realized at playlist generation time.

4.5.4 Segmenting similarity space

In a well-trained SOM, similar songs are mapped close to each other. This fact can be used to further reduce the computational expense for retrieving a nearest neighbor list, since primarily similar songs are of interest for a playlist. Depending on the size of the SOM and the number of songs in the database, only songs that are mapped in the rows and columns around the position of the seed song can be retrieved from the database. This extra filtering can provide an extra decrease of computational cost.

4.5.5 Summary

This section describes the playlist generation module that was implemented in the course of this research. By using mapped song positions an euclidean distance measure can be used for determining song similarities. Since this is computationally inexpensive, nearest neighbour lists from a chosen seed song can be generated at playlist generation time. By choosing different weights for single feature similarities, a personal similarity measure can be created. The computational expense for generating playlists can be further reduced by only retrieving songs that are mapped close to the seed song.

4.6 Summary

In this chapter, the architecture of the framework that was developed during this research was described. The importance of the four design criteria (quality, flexibility, portability and speed) was differentiated for the three modules of the framework: feature extraction and modelling, song (model) management and playlist generation. After outlining the general structure of the framework in Section 4.2, the most important components of each module are described in Section 4.3 - 4.5.

Chapter 5

Evaluation

5.1 Introduction

In this chapter, the concepts that are developed to improve the quality of music playlist generation systems are evaluated. The chapter consists of three papers that were published in the course of the research.

In the first paper, 'On Mixture Model Complexity Estimation for Music Recommender Systems' [11], the validity for using the BSAS algorithm (Section 4.3.3) for approximating more complex model complexity estimation algorithms is assessed. The paper 'Variable-size Gaussian Mixture Models for music similarity measures' [9] uses the model complexity estimation algorithm that was presented in [11] for a music genre classification task. Individually assessing song model complexity for each song yields better genre classification results at an average lower model complexity. In the third paper, 'Variable-size Gaussian Mixture Models for music similarity measures' [10], the variable-sized song models are mapped in a SOM using the heuristic that was described in Section 4.4.

5.2 On Mixture Model Complexity Estimation for Music Recommender Systems

5.2.1 Abstract

Content-based music navigation systems are in need of robust music similarity measures. Current similarity measures model each song with the same model parameters. We propose methods to efficiently estimate the required number of model parameters of each individual song. First results of a study on

relationships between a small set of basic audio features are presented. We conclude that there are only very small correlations between models on low- and on high-dimensional features.

When we compare a very simple clustering algorithm with an algorithm that estimates model parameters using the MDL criterium, we find a surprisingly strong correlation between the estimated number of mixture components.

5.2.2 Introduction

With the ever increasing availability of digital music, new music access strategies are needed. Perrot [88] found that college students where capable to classify a piece of music quite accurately in a 10-class genre taxonomy, while only listening to an excerpt of 250 ms. This is "fundamentally inexplicable with present models of music perception" [105] and justifies the statement that the audio surface contains a lot of information that can be used for music genre classification.

Content-based music recommendation seems to be the most promising solution for finding music in large music collections. These systems usually extract a set of high dimensional features from the audio signal and model these with a statistical model, where a Gaussian Mixture Model with a fixed number of gaussians (between 10 and 100) is most common.

The main problem of content based music recommendation engines is the lack of robustness of the recommendations. Aucouturier [2] finds that certain songs are always ranked very high in a nearest neighbor search. He suggests that these songs (named *hubs*) contain outlier frames, that have great impact on the song model. It is interesting to note that the 'hubness' of a song is found to be not an intrinsic property of the song, but rather a property of a given algorithm. For this reason, we present an alternative approach for modelling music.

Each song has its own characteristics: Instrumentation, vocals, rhythm, etc. These characteristics influence the spectrum and structure of the song and thus the data distribution of the extracted features. We hypothesize that the appearance of hubs can be reduced by analyzing a song's feature complexity and adapting the number of components of a song's model accordingly. To the best of our knowledge, this has not yet been applied in the field of Music Information Retrieval.

In section 5.2.3, we will give a short overview of Music Information Retrieval (MIR) and some commonly used music descriptor features. Section 5.2.4 is on modelling these features and section 5.2.5 deals with complexity estimation

of music data. We present two methods to estimate the model complexity of high-dimensional features. In section 5.2.6 we present some results of both methods.

5.2.3 Musical features

Music Information Retrieval is the science of extracting information from music for various purposes. In large music collections we want to minimize the number of required actions of a user to find the music he likes. Some ten years ago, Wold [124] identifies four methods how to access sounds:

- *Simile*: Find sounds that belong to a certain class.
- *Acoustical/perceptual features*: Find sounds that fulfill certain feature criteria.
- *Subjective features*: Find sounds using a personalized description scheme.
- *Onomatopoeia*: Query by humming.

These methods are still major areas of research in MIR.

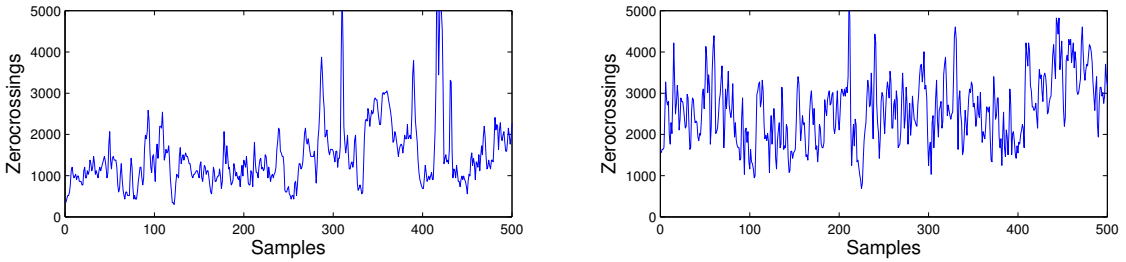
Current music retrieval systems mostly rely on *timbre*-based features. Timbre is the collection of properties that distinguish the sound of a musical note, when this note is generated from different sources or instruments. Three timbre-based features are the *Zero Crossings Rate*, *Spectral Centroid* and *Mel-scale Frequency Cepstrum Coefficients*. These features are calculated over time-frames in which the audio signal is quasi-stationary. A common framelength is 20ms [6].

Zero Crossings Rate

The zero crossings rate is a time-domain based feature. It is a measure of the *noisiness* of an audio signal. The ZCR is defined as:

$$ZCR = \frac{1}{T}\sum_{n=1}^{N} |\text{sign}(x[n]) - \text{sign}(x[n-1])| \tag{5.1}$$

where T is the time in seconds, $x[n]$ the time domain signal of the audio signal and $N = T * Samplerate$. Figure 5.1 depicts two 5 second intervals of songs of two different genres. It is clear to see that the Beatles (Figure 5.1(a)) have a 'cleaner' sound than Greenday (Figure 5.1(b)).



(a) Beatles (b) Greenday

Figure 5.1: Five second interval of Zero Crossings Rate for two songs

Spectral Centroid

The spectral centroid is defined as the center of gravity of the magnitude spectrum of the short-time Fourier transform. It is a measure of the *brightness* of the musical piece and is defined as:

$$SC = \frac{\sum_k kS(k)}{\sum_k S(k)} \tag{5.2}$$

where $S(k)$ is the power of the spectrum in the k^{th} frequency bin.

Mel-scale Frequency Cepstrum Coefficients

MFCC have first been used in speech recognition research and have proven to give a compact representation of the perceptually relevant frequency components in an audio signal. The MFCC is calculated as follows:

1. Convert the signal to frames
2. Take the discrete Fourier transform
3. Take the log of the amplitude spectrum
4. Apply the Mel-scaling and smoothing
5. Take the discrete cosine transform

The mel-scale is a nonlinear scale modelling perceived pitch. It can be approximated by:

$$\text{mel}(f) = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{5.3}$$

Aucouturier [6] systematically explored MFCC feature space and found the optimal number of components to use is 20.

5.2.4 Mixture Models

The timbre features mentioned above are calculated over $20ms$ windows, with a hopsize of $10ms$. For a three-minute song we thus have 18000 samples. For practical applications, this amount of data is too high. We therefore model the data with a mixture model.

A mixture model for a d-dimensional random variable $\boldsymbol{x}$ is given by:

$$p(\boldsymbol{x}, \boldsymbol{\Theta}) = \sum_{m=1}^{k} \alpha_m p_m(\boldsymbol{x}, \boldsymbol{\Theta}_m) \tag{5.4}$$

where k is the number of components in the mixture and $\boldsymbol{\Theta} = \{\alpha_1, \ldots, \alpha_k, \boldsymbol{\Theta}_1, \ldots, \boldsymbol{\Theta}_k\}$ are the model parameters. The mixture weights α_m are nonnegative and add up to one.

Gaussian Mixture Model

The most widespread mixture model type is the Gaussian Mixture Model. Each mixture component is a gaussian probability distribution. The parameters $\boldsymbol{\Theta}$ of a gaussian are its mean $\boldsymbol{\mu}$ and its covariance matrix $\boldsymbol{\Sigma}$:

$$\begin{aligned} \mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}_{\mathcal{X}}) = & \frac{1}{(2\pi)^{N/2} |\boldsymbol{\Sigma}_{\mathcal{X}}|^{1/2}} \\ & \cdot \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}_{\mathcal{X}}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right) \end{aligned} \tag{5.5}$$

The covariance matrix has to be positive definite.

Parameter Estimation

When the number of components in a mixture is known, the Expectation Maximization (EM) algorithm [25] provides an efficient method to estimate the parameters of the distribution of n datasamples $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. The EM algorithm is an iterative procedure and is guaranteed to converge to a local maximum of the maximum (log-)likelihood estimate of the mixture parameters:

$$\hat{\boldsymbol{\Theta}}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\Theta}} \left(\log p(\mathcal{X} | \boldsymbol{\Theta}) \right) \tag{5.6}$$

Each iteration consists of two steps:

- **E-step:** Assign each sample to the mixture component that is most likely to have generated the sample, based on the current estimate of the model parameters.

- **M-step:** Recompute the model parameters based on the current sample membership estimation.

These steps are repeated until convergence of the likelihood estimate.

5.2.5 Complexity Estimation

One major problem of the EM algorithm is that the number of mixture components in a mixture should be known in advance. When listening to various kinds of music, it is clear that there are broad variations in musical structure and sound. Even if this is recognized for music genre classification, where different feature sets are used for determining class likelihood (eg. [68]), no song-level feature model optimization is performed. We think that by modelling each song with an optimal number of components significantly improves classification accuracy. For 'complex' songs, the number of components will be higher as for 'simple' songs.

We perform a study on feature complexity correlations between high and low dimensional features in order to be able to predict the optimal number of gaussians for a high dimensional feature by analyzing the complexness of a low dimensional feature. For this, we compare our predicted number of mixture components $\hat{k}$ with a conventional ground truth k_{opt}. We use the algorithm presented by Figueiredo [28] to find k_{opt} for a small set of features.

Optimal Model Selection

Model selection algorithms try to find the number of components k, that minimize the cost function $\mathcal{C}(\hat{\boldsymbol{\Theta}}(k), k)$:

$$\hat{k} = \arg\min_{k} \{\mathcal{C}(\hat{\boldsymbol{\Theta}}(k), k)\}, \quad k = k_{\min}, \ldots, k_{\max} \tag{5.7}$$

The cost function $\mathcal{C}(\hat{\boldsymbol{\Theta}}(k), k)$ consists of two parts:

- A part expressing the goodness of fit of a model with k components. This function is a monotonically increasing function of k.
- A part penalizing models with higher k.

The method presented by Figueiredo uses a cost function that is based on the Minimum Description Length (MDL) criterion. This criterion is based on the idea that if you can describe some observed data with a short code, you have a good model of the source generating the data.

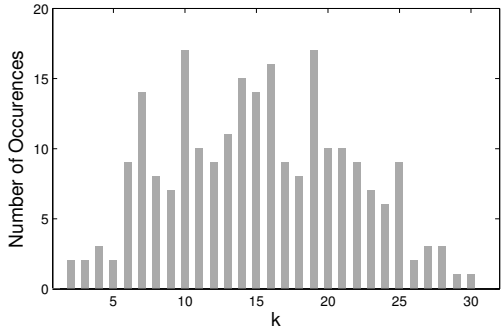


Figure 5.2: Distribution of k_{opt} on a dataset of 234 songs

In Figure 5.3 we see the optimal number of mixture component as found by Figueiredo's algorithm on a dataset of 234 songs. We see that k_{opt} varies between 3 and 30 around a center value of 15 components. Note that this value for k_{opt} is significantly less than the fixed value of $k = 50$ found by Aucouturier [6].

There are numerous algorithms that are also based on the MML criterion. Zivkovic [130] presents an algorithm that uses a coarse approximation of MML. It is much faster than Figueiredo, but also less robust.

Model Estimation

Although model selection algorithms like Figueiredo or Zivkovic presented provide reasonable speed, they are not suitable to find the optimal number of model components for a dataset consisting of 20-dimensional features of 5000 songs because of computation time considerations. Therefore, we search for methods to efficiently estimate the minimal required model complexity of individual songs that are computationally less expensive than the algorithms mentioned above.

Correlation between features We assume that the required model complexity is an intrinsic property of a song. Based on this assumption, we expect a relationship between the required number of components for modelling simple features such as the ZCR or SC and the required number for complex features such as MFCC.

Correlation between algorithms When required model complexity is an intrinsic property of a song, different component estimation algorithms must

also correlate. We have investigated correlations between the number of components k, found by very simple clustering algorithms and k_{opt} as found by Figueiredo. The most basic algorithm we use, is the "Basic Sequential Algorithmic Scheme" (BSAS [112]). As can be seen from the following pseudocode, BSAS only has two parameters: The threshold Θ for determining whether a new cluster has to be formed, and N_{maxClust}, the maximum number of clusters to be formed.

1: $N_{\text{clust}} = 1$
2: $C_1 = \{x_1\}$
3: **for** $i = 2$ to N **do**
4: find C_k: $d(x_i, C_k) = \min_{\forall j} d(x_i, C_j)$
5: **if** $(d(x_i, C_k) > \Theta)$ and $(N_{\text{clust}} < N_{\text{maxClust}})$ **then**
6: $N_{\text{clust}} = N_{\text{clust}} + 1$
7: $C_{N_{\text{clust}}} = \{x_i\}$
8: **else**
9: $C_k = C_k \cup \{x_i\}$
10: **end if**
11: **end for**

5.2.6 Results

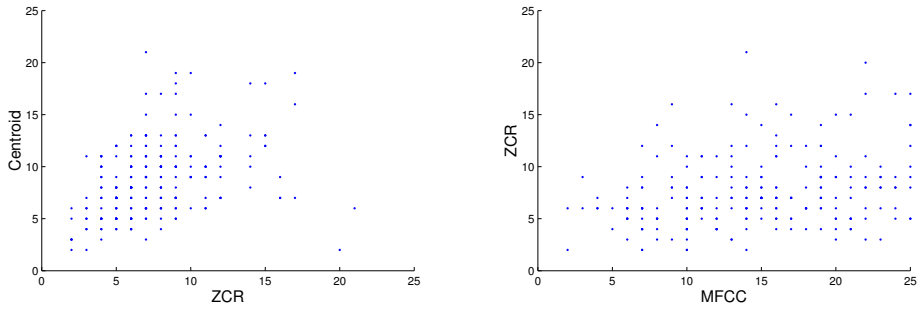
We have selected 234 songs from 26 different genres and analyzed k_{opt}, the optimal number of components found by Figueiredo's algorithm. Then, we compare k_{opt} with estimations of the number of components, obtained via the two methods that have been described in section 5.2.5: Correlation between features and correlation between algorithms.

Correlation between features

In Figure 5.3(a) we have plotted the relation between the number of mixture components as found by Figueiredo's algorithm on the ZCR and SC, k_{ZCR} and k_{SC}. We find the Pearson's correlation coefficient to be 0.39.

In Figure 5.3(b) we see the relation between k_{ZCR} and k_{MFCC}, the optimal number of mixture components of the complete 20-dimensional MFCC, as found by Figueiredo. The Pearson's correlation coefficient is only 0.27.

The correlation coefficients we found are too low to meaningfully predict the number of mixture components of the complete 20-dimensional MFCC vector.



(a) k_{opt} ZCR vs SC (b) k_{opt} ZCR vs MFCC

Figure 5.3: Correlation of k_{opt} between features

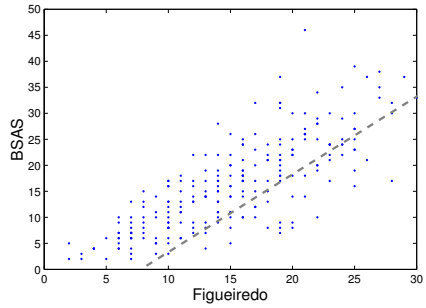


Figure 5.4: k_{opt} on MFCC data, found by Figueiredo vs BSAS

Correlation between algorithms

We used BSAS on our dataset with an Euclidean distance measure, $\Theta = 4$ and $N_{\text{maxClust}} = 400$. BSAS tends to find a huge amount of very small clusters on outlier frames. When we discard clusters of less than 80 samples (out of 9000), we get the following relationship between k_{opt} as found by Figueiredo and BSAS: The Pearson's correlation coefficient is 0.78, which is significantly higher than for the feature-based correlation approach. We can approximate the relationship between the number of components found by BSAS and found by Figueiredo as:

$$k_{\text{Figueiredo}} \cong (k_{\text{BSAS}} - 8) \cdot 1.5 \tag{5.8}$$

This expression overestimates the number of components for most cases. This is deliberate since the loss of information when modelling with less mixture components than required, gives a bigger loss of information than models with

a too high number of components.

5.2.7 Conclusion

We presented two methods to estimate the required model complexity of individual songs. We found only very weak correlations between different audio features. Especially between low- and high-dimensional features, correlation is neglectable. Probably, the single dimension as used by the ZCR or SC contains far too little information to accurately predict the number of clusters in a higher dimensional feature space.

Further research on correlations between two- or three-dimensional features and high-dimensional features is needed to explore the possibilities of complexity estimation using simpler features.

The use of simple clustering algorithms, such as BSAS shows much better results. We can approximate the relationship between the number of mixture components as found by BSAS and the conventional ground truth as found by the algorithm of Figueiredo with a linear expression.

5.3 Variable-size Gaussian Mixture Models for music similarity measures

5.3.1 Abstract

An algorithm to efficiently determine an appropriate number of components for a Gaussian mixture model is presented. For determining the optimal model complexity we do not use a classical iterative procedure, but use the strong correlation between a simple clustering method (BSAS [112]) and an MDL-based method [28]. This approach is computationally efficient and prevents the model from representing statistically irrelevant data.

The performance of these variable size mixture models is evaluated with respect to hub occurrences, genre classification and computational complexity. Our variable size modelling approach marginally reduces the number of hubs, yields 3-4% better genre classification precision and is approximately 40% less computationally expensive.

5.3.2 Introduction

In the current boom of web 2.0, the market for music recommender systems seems to have taken off. Last.fm, iLike, myStrands and others analyze a user's

listening behaviour and compare it with other user's profiles. Music can be tagged with personal tags which allows new ways to explore music collections.

One of the major problems of these community based recommender systems is their robustness in new databases and dealing with underrepresented data. Once a song is chosen as favourite, a loop mechanism can keep this song as favourite for a long time. Community based recommenders can be sensitive to attacks that try to influence a specific song's rating. Two kinds of attack strategies are *shilling* (promoting an item) and *nuking* (demoting an item) [66].

Another approach for recommender systems that do not suffer from loops, shilling or nuking is content-based recommendation. The acoustical content of a song is analyzed, and songs found to be 'similar' to songs a user likes are recommended. Audio similarity is multifaceted, so a common approach to evaluate audio similarity measures is to perform a genre classification task. Pampalk et al. [80] found that a combination of 70% timbral and 30% temporal features provide a good audio similarity measure.

Hubs, songs that are found to be very similar to a very large number of other songs, are a major problem for audio-based music recommender systems. Aucouturier and Pachet [7] showed that in a purely timbre-based nearest neighbor retrieval system, the number of hubs significantly increases when discarding the 5% least significant clusters from a Gaussian mixture model.

The computational complexity for calculating the distance between Gaussian mixture models scales linearly with the number of clusters in a mixture model for most distance measures. Reducing the number of clusters in a model thus has great impact in computational complexity, but influences performance.

We present a method to reduce the number of mixture components without sacrificing retrieval performance. The required number of Gaussians in a Gaussian mixture model is estimated for each song individually. The number of clusters is then used by the Estimation Maximization algorithm to model the song data. Using individual song complexity estimation prevents overfitted models on 'simple' songs with too complex models, while still offering 'complex' songs an adequate model complexity.

The remainder of this paper is organized as follows: In section 5.3.3 we present a short overview of related work. Section 5.3.4 describes our feature modelling approach in detail. This section is followed by a performance analysis with respect to the effect of our algorithm on hubs and on a genre classification task. In our last section we summarize our results and give some recommendations for further research.

5.3.3 Related Work

Berenzweig et al. [14] compare anchor space based and GMM based similarity measures with a similarity matrix retrieved from a user survey. The anchor space method performs very similarly to the GMM method.

Aucouturier and Pachet [6] systematically explore feature parameter space for timbre similarity experiments and evaluate performance with a nearest neighbour retrieval task. Optimal R-precision was found with 20 dimensional MFCCs and a Gaussian mixture model with 50 components. The number of model components however is of less influence than the number of feature dimensions. Their conclusion is that 'Everything performs the same' and that there seems to be a glass ceiling in R-precision.

Flexer et al. [30] compare Hidden Markov Models (HMMs) with Gaussian Mixture Models (GMM) describing spectral similarity of songs. It is shown that HMMs are capable of representing the underlying data better than GMMs, even if the GMM has more degrees of freedom. In a genre classification task, both methods show very similar results.

5.3.4 Feature Modelling

We calculate song similarity on 15 dimensional MFCC vectors (without the 0^{th} coefficient), modelled with a Gaussian Mixture Model:

$$p\left(\boldsymbol{x}, \boldsymbol{\Theta}\right) = \sum_{i=1}^{k} \alpha_i \mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}_{\mathcal{X}}) \tag{5.9}$$

with $\boldsymbol{x}$ a single feature vector and $\boldsymbol{\Theta}$ the model parameters: cluster mean $\boldsymbol{\mu}$ and cluster covariance $\boldsymbol{\Sigma}$. The mixture weights α_i are nonnegative and add up to one.

Parameter Estimation

When the number of components in a mixture is known in advance, the Expectation Maximization (EM) algorithm [25] provides an efficient method to estimate the parameters of the distribution of $\boldsymbol{n}$ data samples $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$. The EM algorithm is an iterative procedure and is guaranteed to converge to a local maximum of the maximum (log-)likelihood estimate of the mixture parameters:

$$\hat{\boldsymbol{\Theta}}_{\text{ML}} = \arg\max_{\boldsymbol{\Theta}} \left(\log p\left(\mathcal{X}|\boldsymbol{\Theta}\right)\right) \tag{5.10}$$

Each iteration consists of two steps:

- **E-step:** Assign each sample to the mixture component that is most likely to have generated the sample, based on the current estimate of the model parameters.
- **M-step:** Recompute the model parameters based on the current sample membership estimation.

These steps are repeated until the likelihood estimate converges.

Complexity Estimation

During the training phase of the EM algorithm, the number of mixture components remains constant, even if the model over- or underfits the data. When listening to various kinds of music, it is clear that there are broad variations in musical structure and sound. Mörchen et al. [68] recognized this issue on a genre level and used different feature sets for each genre for determining genre likelihood.

Pampalk [78] allows variable-size models for each individual song. A k-means model is fitted to the song features and a minimal distance between clusters is defined. When two cluster centers are within this minimal distance, they are merged.

We introduce a similar approach to Pampalk, and use it to generate gaussian mixture models.

Optimal Models Model selection algorithms try to find the number of components k, that minimize the cost function $\mathcal{C}(\hat{\mathbf{\Theta}}(k), k)$:

$$\hat{k} = \arg\min_{k}\{\mathcal{C}(\hat{\mathbf{\Theta}}(k), k)\}, \quad k = k_{\min}, \ldots, k_{\max} \tag{5.11}$$

The cost function $\mathcal{C}(\hat{\mathbf{\Theta}}(k), k)$ consists of two parts:

- A part expressing the goodness of fit of a model with k components. This function is a monotonically increasing function of k.
- A part penalizing models with higher k.

Figueiredo and Jain [28] presented an algorithm that optimizes a cost function based on the Minimum Description Length (MDL) criterion. This criterion is based on the assumption that if one can describe some observed data with a short code, one has a good model of the source generating the data. Other algorithms optimizing a cost function like in Equation 5.11 exist (eg. [85], based on the Bayesian Information Criterion), but have not been investigated.

Figueiredo uses a modified version of the EM algorithm for fitting a GMM in the dataset. The algorithm starts with a high number of components and eliminates components of the mixture in the M-step.

$$\text{for} \quad i = 1, \ldots, k: \\ \hat{\alpha}_i(t+1) = \frac{\max\left\{0, \left(\sum_{j=1}^{n} w_i^{(j)}\right) - \frac{N}{2}\right\}}{\sum_{i=1}^{k} \max\left\{0, \left(\sum_{j=1}^{n} w_i^{(j)}\right) - \frac{N}{2}\right\}} \tag{5.12}$$

where $w_i^{(j)}$ is the conditional expectation that sample j belongs to mixture component i. When the EM algorithm is converged and the number of components is still larger than $k_{\min}$, the component with the smallest support is forced to zero. This procedure is repeated until $k = k_{\min}$.

Optimal Model Approximation As a consequence of using EM to iterate through the various model sizes, Figueiredo's algorithm is very slow. We found that the number of clusters found by the much simpler 'Basic Sequential Algorithmic Scheme' (BSAS, [112]) shows high correlation with the number of clusters as found by Figueiredo. This algorithm only takes two parameters: the threshold θ for determining whether a new cluster has to be formed, and N_{maxClust}, the maximum number of clusters to be formed.

The basic algorithm in pseudocode consists of the following steps:

1: $N_{\text{clust}} = 1$
2: $C_1 = \{\boldsymbol{x}_1\}$
3: **for** $j = 2$ to n **do**
4: find C_i: $d(\boldsymbol{x}_j, C_i) = \min_{\forall k} d(\boldsymbol{x}_j, C_k)$
5: **if** $(d(\boldsymbol{x}_j, C_k) > \theta)$ and $(N_{\text{clust}} < N_{\text{maxClust}})$ **then**
6: $N_{\text{clust}} = N_{\text{clust}} + 1$
7: $C_{N_{\text{clust}}} = \{\boldsymbol{x}_j\}$
8: **else**
9: $C_k = C_k \cup \{\boldsymbol{x}_j\}$
10: **end if**
11: **end for**

This algorithm was modified to accept new clusters even if the maximum number of clusters has already been reached, but only if there is a cluster that was assigned less than 1% of the data. This smallest cluster is then discarded and replaced by the new cluster. After the algorithm has finished, all clusters

containing less than 1% of the data are discarded. The cluster centers found by BSAS are used as input for an EM algorithm to fit a GMM in the data. The EM algorithm uses all samples, including those in the clusters that were discarded in the BSAS algorithm.

Initializing the EM process with the clustering result of BSAS significantly decreases the number of iterations the EM algorithm needs to converge when we compare it with methods that discard insignificant clusters in the training phase of the algorithm.

We compared the number of components found by Figueiredo with that of BSAS, on a subset of 234 songs from the Magnatune dataset. This subset covers all music genres available in the Magnatune data. The Pearson's correlation coefficient between the number of clusters found by both algorithms is 0.78.

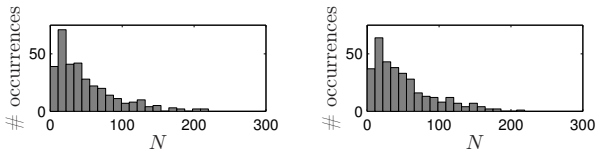
5.3.5 Evaluation

We selected a subset of 331 songs from the Magnatune dataset, covering six genres. This dataset is modelled both with fixed size GMMs with 20 clusters and with variable size GMMs with a maximum 30 clusters. The number of clusters in the variable size model case is determined by the BSAS algorithm as presented in section 5.3.4. The mean number of clusters over our dataset was 15. The EM modelling complexity scales approximately linearly with the number of clusters, we therefore obtained a 25% computing time gain for the variable size models. We use the Earth Mover's Distance to determine the distance between the GMMs [14]. Computation of the full song similarity matrix was approximately 40% faster for the variable size models.

Hub-analysis

Robustness of music similarity measures can be evaluated by means of a hub-analysis. Aucouturier [7] uses two methods to assess the hubness of various algorithms: *N-occurrences* and *Neighbour Angle*. In this subsection we evaluate the hubness of our dataset, modelled with the variable and the fixed size models, using the N-occurrence measure.

N-occurrences The N-occurrence measure counts the number of times a song occurs within the N nearest neighbours of all songs in a data set. The measure is a constant-sum measure: the mean N-occurrence is equal to N. When studying hubs, we are interested in the amount of songs that occur much more frequently in the N nearest neighbours than the expected average value.



(a) Fixed size models, 20 Gaussians (b) Variable size models

Figure 5.5: N-occurrence analysis

In Figure 5.5 we show the N-occurrence histograms for $N = 50$. The number of songs that occur more than 150 times differs only marginally between the two model types: 11 for the variable size models and 12 for the fixed size models. The use of variable size models thus seems to have small positive impact on hub occurrences.

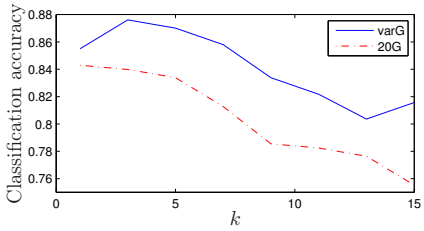
Aucouturier [7] showed that discarding statistically irrelevant clusters (*homogenization*) caused a dramatic increase in the number of hubs. With this experiment we showed that reducing the number of mixture components can be done without having negative influence on the number of hubs. Apparently, not all songs require the same number of mixture components.

Genre classification

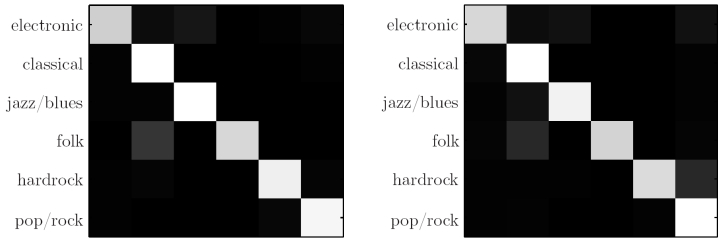
The most common evaluation procedure for music similarity measures is genre classification. Since we are only interested in the comparison between fixed- and variable size models, we do not apply an artist-filter as has been suggested by Pampalk et al. in [80].

Aucouturier and Pachet [4] dispute the use of genre classification for evaluating timbre similarity. Different artists within one single genre may have a very broad 'timbral' spectrum. Our data set only contains very few artists per genre. As a consequence of this, and under the assumption that each single artist only uses a narrow timbral spectrum, we can generalize genre distance to timbral distance.

Classification results We use a simple k-nearest neighbour classifier and classify with a leave one out cross validation procedure. In Figure 5.6(a) we depict the classification accuracy for a range of k, for variable-size models with 15 Gaussians on average and for fixed-size models with 20 Gaussians. We see that the variable-size models consequently outperform the fixed-size models, even with an average lower model complexity.



(a) k vs accuracy



(b) Variable size (c) 20 Gaussians

Figure 5.6: Genre classification performance

Inter- and intra genre distance Aucouturier and Pachet [4] use the mean distance between songs within a single genre and between different genres to express the limited use of genre classification for timbral similarity evaluations. Artists within a certain genre without a 'coherent' sound make it difficult to find a direct relationship between timbre similarity and genre similarity.

Our database consists of few artists per genre and all have a 'coherent' sound. We can thus use the measure to compare timbre discrimination performance of the fixed size Gaussian models with the variable size models to each other. Both for the 20 Gaussians and the variable size models we find a ratio of 1 : 1.32. Although the variable size models have a lower mean number of components, the timbre information seems to be captured just as well as by the more complex fixed size models.

5.3.6 Conclusions

In this paper we presented an algorithm to estimate an optimal number of cluster components for each individual song. We compared the number of hub occurrences between a 20-Gaussian model and our variable size modelling approach with 15 clusters on average. Our variable size modelling approach marginally reduces the number of hubs.

We analyzed the timbral discrimination performance of our measure with a genre classification task on a small database with homogenous genres. Variable size models outperformed fixed size models with respect to genre classification by 3-4% and shows the same mean inter- to intra genre distance ratio at average lower model complexity.

Computation of a full song distance matrix using the Earth Mover's Distance is approximately 40% faster for the variable size models.

5.4 Music Playlist Generation by assimilating GMMs into SOMs

5.4.1 Abstract

A method for music playlist generation, using assimilated Gaussian mixture models (GMMs) in self organizing maps (SOMs) is presented. Traditionally, the neurons in a SOM are represented by vectors, but in this paper we propose to use GMMs instead. To this end, we introduce a method to adapt a GMM such that its distance to a second GMM decreases at a controllable rate. Self organization is demonstrated using a small music database and a music classification task.

5.4.2 Introduction

The music industry has made a huge step towards digital distribution of their products. It is possible to buy individual tracks of high audio quality online at a small fee, even without copyright protection. Online platforms like Amazon successfully apply collaborative filtering techniques: Users profiles are made up by monitoring and analyzing buying, searching and rating behaviors. These profiles are matched to the behavior of new visitors to find music that might be of their interest. This very same principle is applied by various websites dedicated to music recommendation: Last.fm, iLike, myStrands and others are so called social music recommendation websites. But social music recommendation has several major disadvantages: a large amount of collective data has to be collected and non-mainstream listeners have problems with underrepresented data causing bad recommendations. This paper is on content-based music recommendation. A heuristic that allows fast and robust content based music exploration is presented. The following paragraphs give a brief overview of the elementary components of current content-based playlist generation systems: the process of getting music recommendations, modelling the content of

songs, and current methods for content-based music playlist generation. The section ends with an outlook on the contents of the paper.

Content based music recommendation and exploration systems are based on methods expressing similarity between every pair of songs in a collection of songs. This similarity is calculated by modeling the statistical distribution of a set of extracted features of each individual song (called song model) and determining the similarity between these distributions of all song pairs. Music recommendations can be done by asking a user for one or multiple *seed songs* that one likes, and presenting songs with small distance to these songs. The quality of content based music recommendations and music exploration applications is highly dependent on the method used for determining the similarity between songs. Selecting the features to describe each song is a tedious process and falls out of the scope of this paper. The similarity measure used in this paper is based on the timbral 'Mel-scale Frequency Cepstral Coefficients' (MFCC) feature [52], wich is known to be a suitable feature for music genre classification systems [6].

There are several methods available for modeling the statistical distribution of the extracted features of each song. Both Gaussian mixture models (GMMs) and hidden Markov models (HMMs) have been used [6]. These models both use a probability density function that is made up of a weighted sum of Gaussian distributions. A GMM uses a single set of Gaussians for modeling the content of an entire song, a HMM has multiple smaller sets that each describe a smaller part of a song and between which temporal relationships can exist. Aucouturier and Pachet [6] explore the parameter space for music similarity measures. The number of Gaussians in GMMs are systematically varied for song models of high dimensional MFCC features and GMMs were compared with HMMs. The quality of the song models is evaluated using a precision measure. This measure determines the amount of retrieved songs belonging to the same (predefined) cluster as that of a seed song in a nearest neighbor search. The best performing parameter set for a music similarity task used 20-dimensional MFCC vectors, modelled using GMMs having 50 Gaussians per GMM. HMMs showed no better performance than the conceptually easier GMMs with a similar number of degrees of freedom.

Current content based playlist generation algorithms [e.g. 81] pre-calculate a full distance matrix, expressing all pair distances between the song models of all songs in a music database. This distance matrix is used as input to a set of playlist generation heuristics, defining rules for how to select songs that are close to a given seed song. Others [e.g. 95] use Self Organizing Maps (SOMs) for music exploration. A SOM is a neural network projecting an N-dimensional

Euclidean vector space to a low dimensional (usually 2) grid while keeping the topology between the vectors approximately intact. Self organizing map based mappings of songs provide an intuitive method for exploring music databases [95]. The use of appealing visualizations [76] reduces the barrier of exploring new regions in unknown music databases.

Both the distance matrix and the SOM based approaches have limitations. Although the distance matrix based approach allows similarity measures based on dedicated song models, the computational complexity is very large: when one single song gets added to the database, the distances to all other songs in the database have to be computed and added to the distance matrix. Also the power of SOMs for music organization is limited. Each neuron in a SOM can only contain a single vector, which limits the song representation accuracy. Several strategies have been presented for selecting a maximally informative vector that captures the properties of a song. Rauber and Frühwirth [95] presents a two-stage SOM mapping: the first stage SOM is trained on the raw feature vectors of the entire database. The SOM activation patterns for each song in this first SOM are stored in a vector and are then used to train the second SOM. Pampalk [76] evaluated different strategies, using indexes of characteristic feature models and simple mean feature values over entire songs as vectors.

This paper focusses at reducing the computational complexity of playlist generation systems while keeping the method for determining song similarity robust. This is achieved by combining the modeling power of song GMM models with the self organization properties of SOMs. GMMs allow capturing the statistical properties of songs in great detail, but performing similarity calculations on GMMs is computationally expensive. SOMs allow intuitive user interfaces [76] and simple similarity calculations in the low-dimensional space of the SOM. We pose that combining the modelling accuracy of GMMs with the simple similarity calculations in SOMs provides fast and robust playlist generation systems. SOMs have almost exclusively been used to map an N-dimensional Euclidean vector space to a 2D grid. Since modeling a song with a single Euclidean vector severely limits the accuracy and level of detail of the song representation, we introduce a heuristic that allows assimilating GMMs in SOMs. This heuristic works with GMMs with a fixed number of Gaussians as well as with variably-sized GMMs [9].

The remainder of this paper is organized as follows: In Section 5.4.3 the basics of Gaussian mixture modeling are introduced. Section 5.4.4 introduces the principles of self organization in self organizing maps. This section is followed by an introduction to adaptive Gaussian mixture models and the

presentation of the heuristic that enables the assimilation of GMMs in a SOM. The heuristic is evaluated with a music genre classification task using a self organizing map in Section 5.4.6. In section 5.4.7, it is argumented that self organizing maps are well suited for personalized playlist generation. The paper concludes with Section 5.4.8, the conclusions.

5.4.3 Gaussian Mixture Models

Gaussian mixture models (GMMs) are being used in a wide range of data modeling and classification tasks. A GMM describes the distribution of an object $\mathcal{S}$ in feature space $\mathcal{X}$ as the weighted sum of k Gaussian probability density functions. The probability of a sample $\boldsymbol{x}$ given a GMM $\boldsymbol{\Theta}$ can then be expressed as:

$$p(\boldsymbol{x}|\boldsymbol{\Theta}) = \sum_{i=1}^{k} \alpha_i \mathcal{G}(\boldsymbol{x}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \tag{5.13}$$

with $\mathcal{G}$ a single Gaussian distribution where $\boldsymbol{\mu}_i$ is the mean vector and $\boldsymbol{\Sigma}_i$ is the (positive definite) covariance matrix of the i^{th} Gaussian in the mixture model. The individual mixture weights α_i sum up to one. In our case, $\mathcal{S}$ is a particular song. $\mathcal{X}$ is the space of MFCC features, and $\boldsymbol{x}_j$ is the j^{th} MFCC vector extracted from that particular song.

A GMM can be trained on a dataset using the Expectation-Maximization (EM) algorithm [25]. The EM algorithm is an iterative procedure and is guaranteed to converge to a (possible local) maximum of the log-likelihood function

$$\log p\left(\boldsymbol{x}_{1\ldots n}|\boldsymbol{\Theta}\right)$$

of the mixture parameters.

Each iteration consists of two steps:

- **E-step:** Based on the current estimate of the model parameters, assign each sample to the mixture component that most likely generated the sample.
- **M-step:** Recompute the model parameters based on the current sample membership estimation.

These steps are repeated until convergence of the likelihood estimate or until a maximum number of iterations has been performed.

When GMMs are being used for a classification task, a learning set of feature vectors of each single class are used to train a GMM. During the classification stage, the log-likelihood of each model $\boldsymbol{\Theta}$ is determined for a sequence

of feature vectors from the object $\mathcal{S}$:

$$\log p(\boldsymbol{x}_{1\ldots\mathrm{n}}|\boldsymbol{\Theta}) = \sum_{j=1}^{n} \log p(\boldsymbol{x}_j|\boldsymbol{\Theta}) \tag{5.14}$$

The log-likelihood of the feature vectors is determined for all classes. The class with the highest log-likelihood is chosen as winner.

It was shown in [9] that the average number of Gaussians per song can be reduced without losing performance in a simple music genre classification task. This was realised by estimating an optimal number of Gaussians for each single song.

5.4.4 Self Organizing Maps

Introduction to Self Organizing Maps

A self organizing map (SOM) is a neural network that accomplishes a mapping of a high dimensional space to a lower dimensional space. The term 'self organizing' refers to the property that the neural net can be trained by a training set such that the mapping approximately preserves the topology of the training set: neighbors in the training set are mapped to neighbors in the lower dimensional space. For training it uses the principles of competitive learning. Given a variable $\boldsymbol{x}(t) \in \mathbb{R}^n$ and a set of reference or *codebook vectors* $\boldsymbol{m}_i(t) \in \mathbb{R}^n$, $i = 1 \ldots k$. Here, k is the number of neurons. The variable t is a counter that indexes the iteration count during the learning phase of the SOM. We refer to t as 'time'. $\boldsymbol{x}(t)$ is drawn from a learning set. $\boldsymbol{m}_i(0)$ is initialized with random vectors from the learning set. Each codebook vector is stored in a neuron that has an *activation function* monotonically decreasing with the distance of its codebook vector to the input sample $\boldsymbol{x}(t)$.

At time t, the sample $\boldsymbol{x}(t)$ is presented to all neurons, the winning neuron is that one having the highest activation, i.e. the neuron with the codebook vector most similar to input $\boldsymbol{x}(t)$:

$$\boldsymbol{m}_c(t) \in \boldsymbol{m}_{1\ldots k} : \boldsymbol{m}_c(t) = \min_{\forall i} \|\boldsymbol{x}(t) - \boldsymbol{m}_i(t)\| \tag{5.15}$$

The winning codebook vector $\boldsymbol{m}_c(t)$ is then adapted to decrease the distance to $\boldsymbol{x}(t)$:

$$\boldsymbol{m}_c(t+1) = \boldsymbol{m}_c(t) + \alpha(t)\left[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)\right] \tag{5.16}$$

with $\alpha(t)$ a monotonic decreasing function, determining the learning rate of the system. For sufficiently large t, the codebook vectors $\boldsymbol{m}_i$ will describe

$\boldsymbol{x}(t)$ with a minimal residual error. This result is similar to that of a vector quantization algorithm.

The described 'network' does not have any notion of topology, no neighborhood relationships between the neurons were defined. Kohonen [44] aligned the neurons on a grid and introduced a neighborhood function ϕ giving the neurons topological relationships. The update function now not only updates the winning neuron $\boldsymbol{m}_c$, but also its neighbors N_c:

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) + \alpha(t)\phi(t)\left[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)\right] \quad \forall i \in N_c(t) \tag{5.17}$$

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) \qquad \forall i \notin N_c(t) \tag{5.18}$$

The neighborhood function is a monotonic decreasing function over distance to the winning neuron. Its radius decreases over time. Due to the sequential overlaps of the neighborhood functions at each iteration of the algorithm, the values of the codebook vectors tend to be smoothed and become ordered. Similar input samples $\boldsymbol{x}$ will map to neurons that are topographically close to each other.

5.4.5 GMMs in SOMs

Self organizing maps can work with any kind of codebook type, as long as two conditions are fulfilled. A distance measure with monotonic decreasing distance function between training data and codebook entry is available and there is an algorithm for adapting the codebook entry to better represent the training data.

In order to be able to use GMMs in SOMs, an algorithm capable of adapting a GMM in a SOM neuron to better represent the training GMM has to be available. In this section, current methods for adapting GMMs are reviewed and a new heuristic for this goal is presented.

On-line adaption of GMMs

Heuristics for adapting GMMs are already used in video surveillance applications. In these applications, scene background and foreground separation is one of the key functional blocks. Since, for instance, lighting conditions may vary over time, subsequent images of a scene background show slight variations. Stauffer and Grimson [109] presented an on-line algorithm for modeling the background of a scene using GMMs. The distribution of observed values is modeled using a GMM for each single pixel. Every newly observed pixel

value is checked against the GMM of that pixel: the pixel is compared with all Gaussians in the mixture. If the pixel lies within 2.5 standard deviations of the most likely Gaussian, the pixel is assigned to that cluster. The cluster parameters are then updated using a simple update rule. When the pixel does not match any clusters in the GMM, the cluster with the smallest weight is discarded and a new cluster is formed with the observed pixel value as new mean. A very similar algorithm was presented by Hu and Su [40].

Song and Wang [107] presented an alternative approach. A series of newly observed values of a pixel x are modeled with a separate GMM $a(x)$. All clusters in $a(x)$ are compared with the clusters in the existing GMM $g^N(x)$ of that pixel. Statistical cluster equivalence between the clusters in $a(x)$ and $g^N(x)$ is assessed using the W statistic and Hotelling's T^2 test. The statistical equivalent clusters in $a(x)$ and $g^N(x)$ are then merged in a new $g^N(x)$, and the non-equivalent clusters are added to the new $g^N(x)$. The algorithm can create new clusters until a maximum number of clusters has been reached.

New approach for adaptive GMMs

The on-line adaption algorithms for GMMs that were presented in the preceding subsection, allow an existing GMM to be updated to also represent newly observed data. With this, the second requirement for codebook entries in SOMs seems to be fulfilled, but this is not the case. For using GMMs in a SOMs, an algorithm is needed that allows to *gradually* adapt a codebook GMM to reduce the distance between this codebook GMM and a training GMM, as given by Equation 5.16. The method of Song and Wang [107] only allows for complete data representation, no learning rate α can be set. The methods by Stauffer and Grimson [109] and Hu and Su [40] allow a learning rate to be set by varying the number of samples presented to the algorithm, but require random samples of the observed dataset. The quality of the adaption depends on how well these random samples represent the dataset. In this section a heuristic is proposed that allows to *gradually* adapt a GMM to another GMM, where a learning rate α allows to control the rate of adaptation.

The heuristic that is proposed in this paper is based on the Earth Mover's Distance (EMD) as presented by Rubner et al. [102]. The EMD is a distance measure based on the minimal cost (amount of work), needed to transform one 'signature' into another. Signatures can be histograms (used by Rubner et al.) or any other data model where a distance measure between the individual components is defined. The principle of the EMD on histograms is illustrated in Figure 5.7. The cost of transforming the top histogram (the supplier) to

the bottom histogram (the customer) can be expressed as the amount (flow) of 'earth' to be moved between supplier and customer times the distance of the transport. Let P and Q be the two histograms (or signatures) from Figure 5.7 with m the number of bins in P and n the number of bins in Q. The 'weight' of bin p_i is w_{p_i} and the sum of weights over all bins for each histogram equals one. The distance between p_i and q_j is $d_{p_i q_j}$, the flow is $f_{p_i q_j}$. The distance between P and Q can then be found by solving a minimization problem:

1. Minimize the cost function W as a function of the 'flow' f:

$$W = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{p_i q_j} f_{p_i q_j} \tag{5.19}$$

under the constraints:

$$\begin{aligned} f_{p_i q_j} &\geq 0 && 1 \leq i \leq m, 1 \leq j \leq n \\ \sum_{j=1}^{n} f_{p_i q_j} &\leq w_{p_i} && 1 \leq i \leq m \\ \sum_{i=1}^{m} f_{p_i q_j} &\leq w_{q_j} && 1 \leq j \leq n \end{aligned}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} f_{p_i q_j} = \min\left(\sum_{i=1}^{m} w_{p_i}, \sum_{j=1}^{n} w_{q_j}\right)$$

2. Determine the distance between P and Q by normalizing W to the sum of all flows as:

$$D_{\mathrm{EMD}}(P, Q) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} d_{p_i q_j} f_{p_i q_j}}{\sum_{i=1}^{m} \sum_{j=1}^{n} f_{p_i q_j}} \tag{5.20}$$

Solving the transportation problem of Equation 5.19 is a linear programming problem, for which efficient solutions exist [e.g. 24].

When using the EMD for determining the distance between two GMMs, a distance function between two Gaussians is needed. In this paper, a symmetrized version of the Kullback-Leibler (KL) divergence is used. A closed

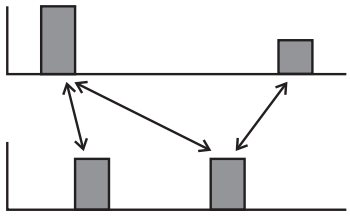


Figure 5.7: Minimal cost transformation between two histograms

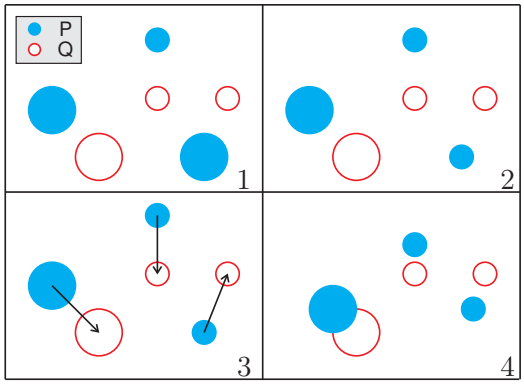


Figure 5.8: Four steps for adapting GMMs using earth mover's flow. P an Q are two GMMs in a two-dimensional space. The arrows in the third step show the flow between P and Q.

form expression for the KL divergence exists and can be expressed as:

$$\begin{aligned} D_{\mathrm{KL}}(p\|q) =& \frac{1}{2}\log\frac{|\Sigma_q|}{|\Sigma_p|} + \frac{1}{2}\mathrm{Tr}\left(\Sigma_q^{-1}\Sigma_p\right) \\ &+ \frac{1}{2}(\mu_p-\mu_q)^T\Sigma_q^{-1}(\mu_p-\mu_q) - \frac{d}{2} \end{aligned} \tag{5.21}$$

with Tr the trace operator and $|\Sigma|$ the determinant of Σ. The symmetrized version of the KL divergence is given by:

$$\begin{aligned} D_{\mathrm{KL}_{\mathrm{symm}}}(p\|q) =& \frac{1}{2}\mathrm{Tr}\left(\Sigma_q^{-1}\Sigma_p + \Sigma_p^{-1}\Sigma_q\right) \\ &+ \frac{1}{2}(\mu_p-\mu_q)^T\left(\Sigma_q^{-1}+\Sigma_p^{-1}\right)(\mu_p-\mu_q) - d \end{aligned} \tag{5.22}$$

with d the dimension of the data.

The proposed heuristic consists of four major steps, illustrated in Figure 5.8. The histogram bins of Figure 5.7 are now replaced by Gaussians in a

two dimensional space. P and Q represent two GMMs. P is going to be adapted towards Q. In the first step, the distances between all Gaussians in the mixture models are determined using Equation 5.22 and stored in matrix $\mathcal{D}$. The weight of each Gaussian in P are adapted towards the weights of the nearest Gaussian in Q in the second step. After normalizing the GMM P, the third step is using the EMD to find the flow matrix $\mathcal{F}$ between P and Q. The flow matrix is used to calculate the cost matrix: $\mathcal{C}$ with $\mathcal{C}_{ij} = f_{ij}d_{ij}$. Each entry in $\mathcal{C}$ thus represents the contribution to the amount of work (Equation 5.19) that the EMD tries to minimize. For each Gaussian in Q, the most 'expensive' supplier Gaussian in P is identified. In the fourth step, Q's Gaussians are shifted towards their most expensive supplier Gaussian in P, to obtain the highest possible cost reduction.

When formalizing the process, the heuristic can be described as follows. Let P be the codebook GMM and Q be a training GMM. The process consists of the following steps:

1: Calculate distance matrix $\mathcal{D}$ using Eq. 5.22
2: **for** $i = 1$ to m **do**
3: find $q_k \in Q$: $q_k = \min_{\forall j} D(i, j)$
4: $w_{p_i} = w_{p_i} + \alpha(w_{q_k} - w_{p_i})$
5: **end for**
6: **for** $i = 1$ to m **do**
7: $w_{p_i} = w_{p_i} / \sum_i w_{p_i}$
8: **end for**
9: Find the cost matrix $\mathcal{C}$ using EMD
10: **for** $i = 1$ to m **do**
11: find $q_k \in Q$: $q_k = \max_{\forall j} \mathcal{C}_{ij}$
12: $\boldsymbol{\mu}_{p_i} = \boldsymbol{\mu}_{p_i} + \alpha(\boldsymbol{\mu}_{q_k} - \boldsymbol{\mu}_{p_i})$
13: $\boldsymbol{\Sigma}_{p_i} = \boldsymbol{\Sigma}_{p_i} + \alpha(\boldsymbol{\Sigma}_{q_k} - \boldsymbol{\Sigma}_{p_i})$
14: **end for**

with $\alpha \in [0 \ldots 1]$ an adaption strength factor.

The Gaussian weight adaption is explicitly placed before calculating the EMD flow, and not at the later mean and covariance adaption stage. The reason behind this design choice is, that clusters that are close to each other, with the same weight, are likely to have only one single flow path. When later moving the clusters towards the most expensive supplier, a distance improvement is guaranteed. Clusters having more than one supplier can be moved towards the most expensive supplier cluster, at the risk of increasing the cost of a secondary supplier cluster.

5.4.6 Evaluation

Evaluating automatic playlist generation systems is a hard task. First, musical preference is very personal and largely based subjective criteria. Second, the goodness criteria of a playlist is also personal and context dependent. In this paper, the quality of the GMM adapting heuristic is thus not directly judged by assessing 'playlist quality', since there is no universal goodness criterium.

In this section, the performance of the heuristic itself and the heuristic applied in a SOM with GMMs is evaluated using two methods for music genre classification, and compared to previous results obtained on the same dataset [9]. The dataset consists of 331 songs from six different musical genres. Of each song, a 15 dimensional MFCC vector [52] is extracted every 10 milliseconds, resulting in approximately 20000 samples per song. Each song is then modeled with variably-sized models with on average 15 clusters and a maximum of 30 clusters per song [see also 9].

The heuristic itself is evaluated using a music genre classification task. The song GMMs are used to train a single GMM for each genre, using the proposed heuristic. Each song in the database is then classified using the EMD between the song and the genre models.

Song organization within a self organizing map is also evaluated using a genre classification task. A SOM is trained using the same music as in the genre classification task described above. Instead of using the neuron GMMs for genre classification, the song positions in the map are used as input for a k-nearest neighbor classifier.

Evaluation of the Heuristic

One single genre GMM is iteratively adapted to the song GMMs of one single genre. For this step, Equation 5.16 is used. The success of adapting the GMM is measured by analyzing the distance reductions between the genre GMM and the song GMMs as a function of α: The distance between the song and the genre GMM is measured using the EMD before and after time the heuristic is applied. Distance improvements are given as ratio of absolute distance improvement divided by the old distance.

For each genre, the song model with the highest number of Gaussians was picked as initial genre representation. After selecting the genre models, these were iteratively adapted using the proposed heuristic, with α given by an

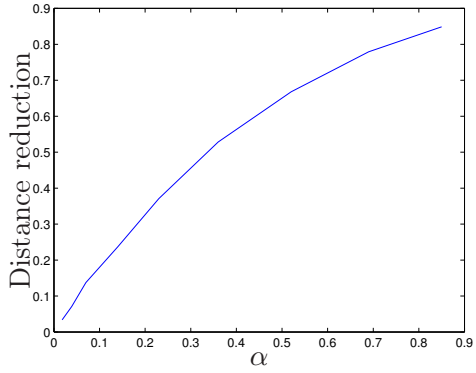


Figure 5.9: Average distance reduction as function of parameter α

exponential decay function:

$$\alpha = \exp \frac{-4 \times t}{t_{\max}} \tag{5.23}$$

Figure 5.9 shows the averaged distance reduction over all songs in the dataset, during the genre classification task, as a function of the parameter α. It can be seen that the obtained distance reductions do not depend linearly on α. For α between 0 and 0.4, a small increase results in an over-proportional increase of distance reduction. For α between 0.5 and 1, the opposite is true. This effect can be explained by analyzing the behavior of the KL divergence (Equation 5.22). Suppose the covariance matrices are identity matrices. The symmetrized KL divergence then clearly shows quadratic behavior as a function of the distance between the means of two Gaussians. For small α, it is likely that the cost reduction of moving clusters towards their most expensive supplier is larger than the cost increase of secondary supplier clusters. For larger α, the cost increase of secondary supplier clusters plays a more important role than for small α. After a large shift, the new cluster constellation is likely to result in new supplier/client relationships.

After training the genre models, the EMD is used to determine the distance of each song to all of the genre models, each song is assigned to the class having the lowest distance. The described procedure results in a genre classification accuracy of 63%, which is comparable with the results obtained in [8]. The confusion matrix is shown in Table 5.1.

The classification accuracy of the 'Electronic' genre (54%) and the 'Pop / Rock' genre (45%) clearly degrade the overall classification performance. These genres were populated with music from different sub-genres and thus show

	Electronic	Classical	Jazz / Blues	Folk	Hardrock	Pop / Rock
Electroninc	54.4	8.8	11.8	0.0	0.0	25.0
Classical	21.8	65.5	4.6	0.0	0.0	8.1
Jazz / Blues	29.4	0.0	70.6	0.0	0.0	0.0
Folk	0.0	5.3	0.0	94.7	0.0	0.0
Hardrock	4.8	2.3	0.0	0.0	88.1	4.8
Pop / Rock	12.3	12.3	0.0	0.0	29.6	45.8

Table 5.1: Confusion matrix for GMM based genre classification task

greater variation within the main genre than the other, well-classified genres. This phenomenon is inherent to using single mixture models for classification tasks with ill-defined classes. k-nearest neighbor approaches using low values for k show better accuracy for these cases.

SOM Organization

By incorporating the GMMs into a SOM, the advantages of using GMMs for representing a class of data and the dimension reduction obtained by the SOM mapping, can be combined. For evaluating the quality of the SOM organization, a k-nearest neighbor genre classification task is used. The song GMMs were used to train a 9×9 SOM. A suitable learning rate function $\alpha(t)$ and neighborhood function $\phi(t, d)$ showed to be:

$$\alpha(t) = \frac{t_{\max} - t}{3 \cdot t_{\max}} \tag{5.24}$$

$$\phi(t, d) = \alpha(t) e^{-500 \cdot d^2 \cdot t} \tag{5.25}$$

with d the normalized distance in the SOM.

After training the SOM, each song is assigned to the neuron in the SOM that has the smallest (EMD) distance. The k-nearest neighbor classifier then uses this neuron's coordinates in the 9×9 grid as song positions to determine the nearest neighbors of each song. The major advantage of this approach is that instead of calculating the EMD on high dimensional GMMs, the classifier can use a simple Euclidean distance in a two-dimensional SOM.

In previous work [9], the same dataset was used as input for a k-nearest

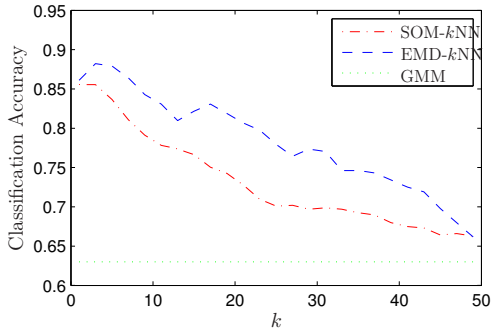


Figure 5.10: Genre classification accuracy for different classification approaches

neighbor classifier that used the high dimensional GMMs to determine the nearest neighbors. Using this approach, a classification accuracy of 87% was reached.

Figure 5.10 shows the genre classification performances for the SOM-based k-nn classification. the EMD classification as presented in [9] and, for reference, the GMM classification accuracy presented in Section 5.4.6. The SOM-based k-nn classification clearly outperforms the GMM classification strategy. This is attributed to the fact that in the SOM, one single genre may occur in more than one area of the SOM: the SOM adapts to the locality of the data, the confusion matrix shows little off-diagonal classification errors (Table 5.2). The EMD-kNN classification still outperforms the SOM based classification, but at most by 10%. The EMD classification however, used GMMs in a 15 dimensional space with on average 15 clusters to represent each song. In the SOM based approach, each song is only represented by a two-dimensional coordinate in a 9×9 SOM grid. By only using these coordinates, the computational complexity for computing song similarity is drastically reduced. It allows real-time applications for music management and playlist generation.

5.4.7 Music Management and Playlist Generation

In the previous section, it was shown that the SOM using GMMs adapts well to the song GMM models. The neighborhood of each song is likely to be dominated by songs of the same musical genre. This opens possibilities for content-based playlist generation.

	Electronic	Classical	Jazz / Blues	Folk	Hardrock	Pop / Rock
Electroninc	83.3	5.0	5.6	0.2	0.0	6.0
Classical	7.9	88.1	1.0	0.0	0.3	2.7
Jazz / Blues	11.4	4.7	81.2	2.4	0.0	0.3
Folk	4.7	4.7	0.0	90.6	0.0	0.0
Hardrock	1.9	1.7	0.0	0.0	89.8	6.6
Pop / Rock	6.9	3.8	0.0	0.0	5.3	84.0

Table 5.2: Confusion matrix for SOM based genre classification task

Argumentation

Traditional content based playlist generation systems require a full song distance matrix. Based on this distance matrix, several heuristics can be used to generate playlists [e.g. 82]. The major disadvantage of approaches using a full distance matrix for playlist generation shows up when adding new songs to a collection. In order to update the distance matrix, the distance to all songs in the database has to be calculated. To this end, each song's feature data or statistical model thereof has to be available. By building a tree of the songs in a distance matrix, the number of distance calculations may be reduced. Only the songs that are close to the new song need accurate distance information. Nevertheless, the song's feature data or statistical model still has to be available for all songs.

The SOM based approach does not require the song data or models to be available after having trained the SOM. Once an initial dataset is available, representing the musical tastes of the user(s) well, the SOM can be initialized and trained. After training, of each song the mapped position in the SOM is stored. When adding new songs to the database, only the mapped position in the SOM has to be found. This requires the song to be compared only with each neuron in the SOM, and not with all the songs already available in the database.

Retrieving a distance matrix of all songs in the database only requires the two-dimensional coordinates of each song in the SOM. Instead of the EMD between single GMMs a simple Euclidean distance measure can be used. Since playlists usually require the songs in the list to be similar with respect to certain

criteria, it is unlikely that the full distance matrix is needed; only those songs mapped around a seed song in the SOM are of relevance.

Since retrieval of a local distance matrix within a SOM is computationally inexpensive, distance matrices of multiple SOMs representing different features (e.g. features representing rhythmical or tonal information of a song) can be combined at playlist generation time. This flexibility allows for a personal definition of music similarity. The distance matrix used for playlist generation can be generated as a weighted sum of multiple distance matrices. One person can then assign a higher weight on rhythmical similarity while another finds tonal information to be more relevant.

Evaluation

For evaluating the suitability of SOM based music organization for playlist generation, a set of 775 songs from 60 genres was mapped in SOMs of sizes $10 \times 10 \ldots 14 \times 14$. Each song was modelled with 15 dimensional Gaussian mixture models using 15 clusters. After mapping, a playlist is generated for each song in the SOM, using a nearest neighbor query. Of each set of generated playlists, the percentage of songs of the same genre as the query song within the k nearest songs is determined. The results are compared with the results obtained by following the same procedure on a song distance matrix computed on the GMM song models.

Data $\diagdown$ k	1	5	10
EMD Distance matrix	28.77	20.80	16.44
SOM 14x14	19.67	15.39	12.47
SOM 13x13	20.37	17.49	14.26
SOM 12x12	17.98	16.83	13.94
SOM 11x11	19.02	17.72	14.39
SOM 10x10	19.18	17.86	14.61
Sum of SOMs	26.20	19.82	15.96

Table 5.3: Percentage of songs of the same genre within k nearest neighbours

Table 5.3 lists the percentage of songs of the same genre as the seed songs within the k nearest neighbours for the different data sets. For larger k, the results of the SOMs get closer to the results obtained from the EMD distance matrix. The differences between the different SOM sizes are small. As the data of the SOM mappings are only two-dimensional, distances between songs can easily be computed and results of multiple SOMs can be combined. The

last row of the table shows the performance of such an 'accumulated' SOM: it nearly reaches the performance of the underlying data.

5.4.8 Conclusions

This paper presented a heuristic that allows assimilating GMMs into SOMs. The heuristic allows to adapt one GMM towards a second GMM with a adaption factor α. This adaption factor is a necessary condition for a GMM adaption algorithm to allow assimilation in SOMs. The heuristic was tested using two music genre classification tasks. The first classification task used the heuristic to train GMMs for each genre and then classified the songs using the EMD. The second classification task used a mapping of GMM based high dimensional song models in a two dimensional SOM using the proposed heuristic. The SOM showed good self organizing properties and the genre classification task showed good classification rates. The SOM combines the advantages of using complex statistical models for describing music and the data reduction by mapping complex high dimensional data in a low dimensional space. By only using the mapped song positions in a SOM for determining song similarity, computationally inexpensive distance measures may be used. When combining multiple SOMs using simple distance measures on the mapped SOM positions, performance nearly equals the performance that can be obtained on the original input data.

5.5 Summary

This chapter shows the power of using variable-size Gaussian mixture models in self organizing maps for music playlist generation. In Section 5.2, algorithms for determining an optimal number of Gaussians for individual song models were compared. It was shown that the BSAS algorithm is a reasonable approximation of Figueiredo's MDL criterium [28] at much lower computational cost.

The variable-size Gaussian mixture models are used in a genre-classification task in Section 5.3. It was shown that while using variable-size GMMs the classification accuracy could be increased compared to using fixed size GMMs having on average a higher number of Gaussians. The obtained classification accuracy increase was only $3 - 4\%$, but the computational complexity was reduced by 40%.

Finally, the variable size GMMs are mapped in a SOM using the heuristic that was presented in Section 4.4. The performance of the SOM was evaluated

using a genre classification task in which the mapped song positions were used as input to a k-nearest neighbour classifier. The classification accuracy using these mapped song positions was only slightly worse than the classification accuracy obtained with a k-nearest neigbour classifier operating in the high-dimensional GMM space.

Chapter 6

Conclusions

6.1 Results

During the last four years, content-based music playlist generation systems have not been able to penetrate in the market of automatic playlist generation systems. It was found by Aucouturier and Pachet [6] that there seems to be a 'glass ceiling' for content based music similarity measures. Indeed, no system was able to prove the opposite, the perfect content based music similarity measure is not yet found.

The main research question that is adressed in this thesis is if it is possible to reduce computational complexity of playlist generation systems at playlist generation time, without loosing quality of the systems. This question is assessed by answering three questions each adressing part of the main research question.

The first question that is answered is on playlist quality criteria. The result of a user survey on playlist requirements is presented. Visitors of several online music fora were asked to participate in the survey, which resulted in a total of 318 responses. The survey aimed at gaining insight in musical preferences, listening behavior and playlist preferences from a broad target audience. From this user survey, it can be concluded that there is no general quality measure on playlists. The participants have a broad range of requirements on playlists, the individual variation on what is perceived as 'good' is very large. Most participants agree on the fact that the songs in a playlist should be of similar mood. It is argued that for evaluating playlist quality, this similar mood criterium can be simplified to assessing music genre consistency.

The second question is wether the current use of Gaussian mixture models for modelling songs is appropriate. Prior art systems model the feature distri-

bution of each song with a fixed number of Gaussians in a Gaussian mixture model, for all songs. It is shown in this thesis (Section 5.3) that by assessing the required number of Gaussians for each song individually, song similarity can be captured better at an average lower number of Gaussians per song. This increase of performance was demonstrated using a genre classification task: A higher classification accuracy was obtained. Since the computational complexity for determining song similarity depends on the number of modelling parameters, the computational complexity for classification was reduced. This result was presented at a paper at the international conference on music information retrieval in 2007 [9].

The third question is if it is possible to find a more compact song representation that allows song similarity calculation at playlist generation time. A heuristic is developed that allows Gaussian mixture models to be assimilated in self organizing maps. Traditional playlist generation systems compute song similarity on high-dimensional statistical models of the songs. Often, Gaussian mixture models are used for capturing the distriution of a feature set computed over each song. The use of self organizing map for organization of large music databases was already explored by others. The performance of these systems was limited by the fact that only regular vectors can be assimilated in a self organizing map. In this thesis, an algorithm is presented that enables assimilating Gaussian mixture models into self organizing maps. By enabling Gaussian mixture models instead of conventional high dimensional vectors, the amount of information that is available in the SOM mapping procedure is increased substantially. By mapping Gaussian mixture models in a SOM, similarity calculations can be performed in the mapped, low-dimensional euclidean space. It is shown in Section 5.4 that SOMs using GMMs can well be used for music exploration tasks. The performance of SOMs using GMMs is assessed with a genre classification and a playlist generation task. Both task show performance similar to algorithms operating in high dimensional feature spaces, but only use the two-dimensional positions in the SOM for determining song similarity. The contents of Section 5.4 is submitted for publication in [10] and a patent on the method for mapping GMMs in SOMs is applied for.

6.2 Outlook

The broadness of the music information retrieval field of research is reflected in the complexiness of designing, realizing and testing music playlist generating systems. As the understanding of how people use and enjoy digital media

will continually improve, it will be possible to agree on a more detailed set of requirements on automatic playlist generation systems than the set that was available during the course of the research that is presented in this thesis. As not only the understanding will evolve, but also computational ressources and power management in mobile devices will continue to develop, the constraints for mobile MIR applications will be less tight than they are now.

Mobile playlist generation systems will soon be able to make use of 'the wisdom of the crowds' by providing an online connection to social music recommender sites. By combining social recommender data with detailed and correct metadata providing song and artist descriptions, a big step towards the end of mobile music players as being 'music graveyards' will be made. Eventually also content-based recommendations will be available to the mobile music consumer. These methods complete the spectrum of music recommendations by closing the cold-start problem that social recommenders have and providing new ways of finding similarities between songs. The presented methods for improving speed of content based playlist generation systems while keeping the required robustness of recommendations intact, provide a step towards content based music recommendation that is realizable on low-resource mobile platforms.

The entertainment industry is one of the biggest industries in the world and will provide new impulses to media consumption. These new impulses, being new kind of media, new transportation channels or new data bearers, will requre the MIR researchers to keep being innovative and find better ways on how to handle large quantities of media data. Music will be consumed evertime, new musical instruments will be invented and new music genres will appear. This constantly changing landscape of music provides huge challenges to the MIR community and guarantees many years of challenging research on the road ahead of us.

Bibliography

[1] M. Alghoniemy and A. H. Tewfik. A network flow model for playlist generation. In *Multimedia and Expo, 2001. ICME 2001. IEEE International Conference on*, pages 329–332, 2001.

[2] J. J. Aucouturier. *Ten Experiments on the Modelling of Polyphonic Timbre*. PhD thesis, Laboratoire d'Informatique de Paris 6, 8 rue du Capitaine Scott, 75015 Paris, France, June 2006.

[3] J. J. Aucouturier and F. Pachet. Finding songs that sound the same. In *Proceedings of IEEE Benelux Workshop on Model based Processing and Coding of Audio*. University of Leuven, Belgium, November 2002. Invited Talk.

[4] J. J. Aucouturier and F. Pachet. Music similarity measures: What 's the use? In *Proceedings of the 3rd International Conference on Music Information Retrieval*, pages 157–163, October 2002.

[5] J. J. Aucouturier and F. Pachet. Scaling up music playlist generation. pages 105–108, Lausanne, August 2002. IEEE International Conference on Multimedia and Expo.

[6] J. J. Aucouturier and F. Pachet. Improving timbre similarity: How high's the sky? *Journal of Negative Results in Speech and Audio Sciences*, January 2004.

[7] J. J. Aucouturier and F. Pachet. A scale-free distribution of false positives for a large class of audio similarity measures. *Pattern Recognition*, pages 272–284, 2007.

[8] J. J. Aucouturier, F. Pachet, and P. Nanappe. From sound sampling to song sampling. In *Proceedings of the 5th International Conference on Music Information Retrieval*, October 2004.

[9] W. Balkema. Variable-size gaussian mixture models for music similarity measures. In *Proceedings of the 8th International Conference on Digital Audio Effects (DAFx)*, pages 491–494, September 2007.

[10] W. Balkema and F. van der Heijden. Music playlist generation by assimilating GMMs into SOMs. *Pattern Recognition Letters*, submitted.

[11] W. Balkema, F. van der Heijden, and B. Meijerink. On mixture model complexity estimation for music recommender systems. In *Proceedings ProRISC 2006*. STW, November 2006.

[12] S. Baumann and O. Hummel. Using cultural metadata for artist recommendations. In *Proceedings of the Third International Conference on Web Delivering of Music*, pages 138–141, September 2003.

[13] A. Berenzweig, D. P. Ellis, and S. Lawrence. Anchor space for classification and similarity measurement of music. In *ICME 2003*, pages 29–32, July 2003.

[14] A. Berenzweig, B. T. Logan, D. P. Ellis, and B. Whitman. A large-scale evaluation of acoustic and subjective music similarity measures. In *Proceedings of the 4th International Conference on Music Information Retrieval*, pages 99–105, October 2003.

[15] L. Bottou and Y. Bengio. Convergence properties of the k-means algorithms. In *Advances in Neural Information Processing Systems*, volume 7, pages 585–592. The MIT Press, 1995.

[16] J. Buckman. Magnatune, an open music experiment. *Linux Journal*, 2004(118), 2004.

[17] C. J. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[18] A. Caclin, S. McAdams, B. K. Smith, and S. Winsberg. Acoustic correlates of timbre space dimensions: A confirmatory study using synthetic tones. *The Journal of the Acoustical Society of America*, 118(1):471–482, 2005.

[19] G. Celeux, S. Chrétien, F. Forbes, and A. Mkhadri. A component-wise EM algorithm for mixtures. *Journal of Computational and Graphical Statistics*, 10(4):697–721, 2001.

[20] D. Cliff. Hang the DJ: Automatic sequencing and seamless mixing of dance-music tracks. Technical report, HP Laboratories, Bristol BS34 8QZ, June 2000.

[21] S. J. Cunningham, D. Bainbridge, and A. Falconer. 'more of an art than a science': Supporting the creation of playlists and mixes. In *Proceedings of the 7th International Conference on Music Information Retrieval*, pages 240–245, October 2006.

[22] S. J. Cunningham, M. Jones, and S. Jones. Organizing digital music for use: An examination of personal music collections. In *Proceedings of the 5th International Conference on Music Information Retrieval*, pages 82–90, October 2004.

[23] J. Dan-Ning, L. Lie, Z. Hong-Jiang, T. Jian-Hua, and C. Lian-Hong. Music type classification by spectral contrast feature. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, volume 1, pages 113–116, August 2002.

[24] G.B. Dantzig. Application of the simplex method to a transportation problem. *Activity Analysis of Production and Allocation*, pages 359–373, 1951.

[25] A. Dempster, D. Rubin, and N. Laird. Maximum likelihood from incomplete data via the EM algorithm. *J.Royal Statistical Soc., Series B (Methodological)*, 1(39):1–38, 1977.

[26] J.S. Downie. *Music Information Retrieval*, volume 37 of *Annual Review of Information Science and Technology*, chapter 7, pages 295–340. 2003.

[27] D. P. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence. The quest for ground truth in musical artist similarity. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, pages 170–177, 2002.

[28] M. A.T. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, March 2002.

[29] H. Fletcher and W.A. Munson. Loudness, its definition, measurement and calculation. *The Journal of the Acoustical Society of America*, 5(2): 82–108, 1933.

[30] A. Flexer, E. Pampalk, and G. Widmer. Hidden markov models for spectral similarity of songs. In *Proceedings of the 8th International Conference on Digital Audio Effects (DAFx)*, Madrid, Spain, September 2005.

[31] T. Fujishima. Realtime chord recognition of musical sound: a system using common lisp music. In *Proceedings of the International Computer Music Conference*, pages 464–467. International Computer Music Association, 1999.

[32] J. Futrelle and J.S. Downie. Interdisciplinary communities and research issues in music information retrieval. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, pages 215–221, 2002.

[33] S.A. Gelfand. *Hearing: An Introduction to Psychological and Physiological Acoustics*. Marcel Dekker, New York, 4th edition, October 2004.

[34] H. Gembris. Musikalische präferenzen. In *Enzyklopädie der Psychologie, Spezielle Musikpsychologie*, pages 279–342. hrsg. von R. Oerter & Th. Stoffer, Göttingen, 2005.

[35] J. L. Goldstein. An optimum processor theory for the central formation of the pitch of complex tones. *The Journal of the Acoustical Society of America*, 54(6):1496–1516, 1973.

[36] M. Goto. Aist annotation for the rwc music database. In *Proceedings of the 7th International Conference on Music Information Retrieval*, pages 359–360, October 2006.

[37] F. Gouyon, S. Dixon, E. Pampalk, and G. Widmer. Evaluating rhythmic descriptors for musical genre classification. In *AES 25th international conference*, pages 196–204, June 2004.

[38] J.L. Herlocker, J.A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250, 2000.

[39] P. Herrera, J. Bello, G. Widmer, M. Sandler, O. Celma, F. Vignoli, E. Pampalk, P. Cano, S. Pauws, and X. Serra. Simac: Semantic interaction with music audio contents. In *Proceedings of 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies*, Savoy Place, London, UK, 2005.

[40] J. Hu and T. Su. Robust background subtraction with shadow and highlight removal for indoor surveillance. *EURASIP Journal on Advances in Signal Processing*, 2007.

[41] X. Hu, J.S. Downie, and A. F. Ehmann. Exploiting recommended usage metadata: Exploratory analyses. pages 19–22, October 2006.

[42] D. B. Huron. Music distribution in a global context: Policy research challenges. Keynote Address, October 2006.

[43] G.T. Jones. *Music theory.* Barnes & Noble Books, 1974.

[44] T. K. Kohonen. Self-organized formation of topologically correct feature maps. *Biolog. Cybern.*, 43:59–69, 1982.

[45] S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[46] J. Lanza. *Elevator Music: a surreal history of Muzak, easy-listening, and other moodsong.* St. martin's Press, New York, 1994.

[47] J. LaRue. *Guidelines for Style Analysis.* Norton, 1970.

[48] X. Li and I. King. Gaussian mixture distance for information retrieval. In *Proceedings of the International Conference on Neural Networks*, pages 2544–2549, 1999.

[49] T. Lidy and A. Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proceedings of the 6th International Conference on Music Information Retrieval*, pages 34–41, September 2005.

[50] S. Lippens, J. Martens, M. Leman, B. Baets, H. Meyer, and G. Tzanetakis. A comparison of human and automatic musical genre classification. In *Proceedings of the IEEE International Conference on Audio, Speech and Signal Processing (ICASSP)*, volume 4, pages 233–236, May 2004.

[51] D. Liu, L. Lu, and H. Zhang. Automatic mood detection from acoustic music data. In *Proceedings of the 4th International Conference on Music Information Retrieval*, pages 81–87, 2003.

[52] B. T. Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the International Symposium on Music Information Retrieval 2000*, page 11p, October 2000.

[53] B. T. Logan. Content-based playlist generation: Exploratory experiments. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, pages 295–296, October 2002.

[54] B. T. Logan, D. P. Ellis, and A. Berenzweig. Towards evaluation techniques for music similarity. In *The MIR/MDL Evaluation Project White Paper Collection*, pages 81–85. GSLIS, Champaign, Illinois, 2003.

[55] B. T. Logan and A. Salomon. A content-based music similarity function. Technical report, Cambridge Research Laboratory, HP Laboratories Cambridge, June 2001.

[56] B. T. Logan and A. Salomon. A music similarity function based on signal analysis. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pages 745–748, August 2001.

[57] P.C. Mahalanobis. On the generalized distance in statistics. *Proc Natl Inst Sci India*, 2(1):49–55, 1936.

[58] M. I. Mandel and D. P. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 594–599, October 2005.

[59] M. I. Mandel and D. P. Ellis. Song level features and svms for music classification. Discussion of mirex results, October 2005.

[60] J.M. Martinez, R. Koenen, and F. Pereira. MPEG-7: the generic multimedia content description standard, part 1. *IEEE Multimedia*, 9(2): 78–87, April 2002.

[61] C. McKay and I. Fujinaga. Musical genre classification: Is it worth pursuing and how can it be improved? In *Proceedings of the 7th International Conference on Music Information Retrieval*, pages 101–106, October 2006.

[62] C. McKay, D. McEnnis, and I. Fujinaga. A large publicly accassible prototype audio database for music research. In *Proceedings of the 7th International Conference on Music Information Retrieval*, pages 160–163, October 2006.

[63] R. Meddis and M. J. Hewitt. Virtual pitch and phase sensitivity of a computer model of the auditory periphery. ii: Phase sensitivity. *The Journal of the Acoustical Society of America*, 89(6):2883–2894, 1991.

[64] I. Mierswa and K. Morik. Automatic feature extraction for classifying audio data. *Machine Learning*, 58(2-3):127–149, February 2005.

[65] G. Mitri, A. L. Uitdenbogerd, and V. Ciesielski. Automatic music classification problems. In *Proceedings of the 27th conference on Australasian computer science*, volume 26, pages 315–322, 2004.

[66] B. Mobasher, R. Burke, C. Williams, and R. Bhaumik. Analysis and detection of segment-focused attacks against collaborative recommendation. In *Advances in Web Mining and Web Usage Analysis*, volume 4198, pages 96–118, 2005.

[67] A. W. Moore. Very fast em-based mixture model clustering using multiresolution kd-trees. In M. Kearns and D. Cohn, editors, *Advances in Neural Information Processing Systems*, pages 543–549, 340 Pine Street, 6th Fl., San Francisco, CA 94104, April 1999. Morgan Kaufman.

[68] F. Mörchen, I. Mierswa, and A. Ultsch. Understandable models of music collections based on exhaustive feature generation with temporal statistics. In *Proceedings of the Twelveth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2006.

[69] F. Mörchen, A. Ultsch, M. Thies, and I. Löhken. Modeling timbre distance with temporal statistics from polyphonic music. *IEEE Transactions on Speech and Audio Processing*, 14:81–90, January 2006.

[70] F. Mörchen, A. Ultsch, M. Thies, I. Löhken, M. Nöcker, C. Stamm, N. Efthymiou, and M. Kümmerer. MusicMiner: Visualizing timbre distances of music as topographical maps. Technical Report 47, Dept. of Mathematics and Computer Science, University of Marburg, Germany, 2005.

[71] G.L. Murphy and D.L. Medin. The role of theories in conceptual coherence. *Psychological Review*, (92):289–316, 1985.

[72] J. Nielsen. *Usability Engineering.* Morgan Kaufmann, San Fancisco, 1994. ISBN 0-12-518406-9.

[73] F. Pachet and D. Cazaly. A taxonomy of musical genres. In *Content-Based Multimedia Information Access Conference (RIAO) proceedings*, 2000.

[74] F. Pachet, P. Roy, and D. Cazaly. A combinatorial approach to content-based music selection. *IEEE Multimedia*, 7(1):44–51, March 2000.

[75] F. Pachet, G. Westerman, and D. Laigre. Musical data mining for electronic music distribution. In *Proceedings of the 1st WedelMusic Conference*, 2001.

[76] E. Pampalk. Islands of Music: Analysis, Organization, and Visualization of Music Archives. Master's thesis, Vienna University of Technology, Austria, December 2001.

[77] E. Pampalk. Aligned self-organizing maps. In *Proceedings of the Workshop on Self-Organizing Maps, Kitakyushu, Japan*, pages 185–190, Kitakyushu, Japan, September 2003. Kyushu Institute of Technology.

[78] E. Pampalk. Speeding up music similarity. Technical report, Queen Mary University, London, UK, September 2005.

[79] E. Pampalk. Audio-based music similarity and retrieval: Combining a spectral similarity model with information extracted from fluctuation patterns. Discussion of mirex results, October 2006.

[80] E. Pampalk, A. Flexer, and G. Widmer. Improvements of audio-based music similarity and genre classification. In *Proceedings of the 6th International Conference on Music Information Retrieval*, pages 628–633, October 2005.

[81] E. Pampalk and M. Gasser. An implementation of a simple playlist generator based on audio similarity measures and user feedback. In *Proceedings of the 7th International Conference on Music Information Retrieval*, pages 389–390, October 2006.

[82] E. Pampalk, T. Pohle, and G. Widmer. Dynamic playlist generation based on skipping behavior. In *Proceedings of the 6th International Conference on Music Information Retrieval*, pages 634–637, October 2005.

[83] J.A. Parente. Music preference as a factor of music distraction. *Percept Mot Skills*, 43-07:337–338, August 1976.

[84] S. Pauws and B. Eggen. Pats: Realization and user evaluation of an automatic playlist generator. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, pages 222–230, October 2002.

[85] D. Pelleg and A. W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, San Francisco, 2000.

[86] J. M. Peña, J. A. Lozano, and P. Larrañaga. An emperical comparison of four initialization methods for the k-means algorithm. *Pattern recognition Letters*, 20(10):1027–1040, 1999.

[87] W. D. Penny. Kullback-leibler divergences of normal, gamma, dirichlet and wishart densities. Journal, Wellcome Department of Cognitive Neurology, March 2001.

[88] D. Perrott and R. Gjerdingen. Scanning the dial: An exploration of factors in the identification of musical style. In *Proceedings of the 1999 Society for Music Perception and Cognition*, page 88, 1999.

[89] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In S.A. Solla, T.K. Leen, and K.-R. Mueller, editors, *Advances in Neural Information Processing Systems 12*, pages 547–553, 2000.

[90] T. Pohle. Post processing music similarity computations. Discussion of mirex results, October 2006.

[91] T. Pohle, P. Knees, M. Schedl, and G. Widmer. Independent component analysis for music similarity computation. In *Proceedings of the 7th International Conference on Music Information Retrieval* Uni [117], pages 228–233.

[92] T. Pohle, E. Pampalk, and G. Widmer. Generating similarity-based playlists using traveling salesman algorithms. In *Proceedings of the 8th International Conference on Digital Audio Effects (DAFxŠ05)*, Madrid, Spain, September 2005.

[93] A. Pople. *Theory, Analysis and Meaning in Music*. Cambridge University Press, 1994.

[94] R. Ragno, C.J.C. Burges, and C. Herley. Inferring similarity between music objects with application to playlist generation. In *MIR '05: Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 73–80, 2005.

[95] A. Rauber and M. Frühwirth. Automatically analyzing and organizing music archives. In *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries*, pages 402–414, September 2001.

[96] H. Rauhe, H. Reinecke, and W. Ribke. *Hören und Verstehen. Theorie und Praxis handlungsorientierten Musikunterrichts.* München, 1975.

[97] J. Reed and C. Lee. A study on music genre classification based on universal acoustic models. In *Proceedings of the 7th International Conference on Music Information Retrieval*, pages 89–94, October 2006.

[98] J. Rissanen. Stochastic Complexity in Statistical Inquiry, volume 15 of Series in Computer Science. *World Scientific*, 1989.

[99] E. H. Rosch. Natural categories. *Cognitive Psychology*, (4):328–350, 1973.

[100] E. H. Rosch. Cognitive representations of semantic categories. *Journal of Experimental Psychology: General*, 104(3):192–233, 1975.

[101] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. Technical report, Stanford University, 1998.

[102] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2), November 2000.

[103] A. Rushinek and S. F. Rushinek. What makes users happy? *Commun. ACM*, 29(7):594–598, 1986.

[104] A. Sankar. Experiments with a gaussian merging splitting algorithm for hmm training for speech recognition. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, February 1998.

[105] E. D. Scheirer. *Music Listening Systems.* PhD thesis, MIT MediaLab Cambridge, 2000.

[106] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[107] M. Song and H. Wang. Highly efficient incremental estimation of Gaussian mixture models for online data stream clustering. In K. L. Priddy,

editor, *Proceedings of the SPIE conference on Intelligent Computing: Theory and Applications III*, volume 5803, pages 174–183, March 2005.

[108] C. Stauffer and W.E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, Fort Collins, CO, USA, 1999.

[109] C. Stauffer and W.E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):747–757, 2000. ISSN 0162-8828.

[110] H. Terasawa, M. Slaney, and J. Berger. The thirteen colors of timbre. In *Proceedings of the 2005 IEEE Workshop on Applications of Signal Porcessing to Audio and Acoustics*, pages 323–326, October 2005.

[111] R. E. Thayer. *The Biopsychology of Mood and Arousal*. Oxford University Press, 1989.

[112] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Elsevier Academic Press, second edition, 2003.

[113] M. Tolos, R. Tato, and T. Kemp. Mood-based navigation through large collections of musical data. In *Second IEEE Consumer Communications and Networking Conference*, pages 71–75, January 2005.

[114] G. Tzanetakis and P. R. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5), July 2002.

[115] A. L. Uitdenbogerd. Music ir: Past, present, and future. In *Proceedings of the First International Conference on Music Information Retrieval (ISMIR 2000)* Uni [116].

[116] *First International Conference on Music Information Retrieval*, Plymouth, Massachusetts, USA, October 2000. University of Massachusetts, University of Massachusetts.

[117] *ISMIR 2006*, Victoria, BC, Canada, October 2006. University of Victoria, University of Victoria.

[118] N. Vasconcelos. On the complexity of probabilistic image retrieval. In *ICCV*, pages 400–407, July 2001.

[119] F. Vignoli. Digital music interaction concepts: A user study. In *Proceedings of the 5th International Conference on Music Information Retrieval*, October 2004.

[120] K. West and S. Cox. Features and classifiers for the automatic classification of musical audio signals. In *Proceedings of the 5th International Conference on Music Information Retrieval*, pages 96–102, October 2004.

[121] K. West and S. Cox. Finding an optimal segmentation for audio genre classification. In *Proceedings of the 6th International Conference on Music Information Retrieval*, pages 680–685, 2005.

[122] B. Whitman and P. Smaragdis. Combining musical and cultural features for intelligent style detection. In *Proceedings of the 3rd International Conference on Music Information Retrieval*, pages 47–52, October 2002.

[123] D. L. Wiesenthal, D. A. Hennessy, and B. Totten. The influence of music on mild driver aggression. *Transportation Research Part F*, (6):125–134, 2003.

[124] E. Wold, T. Blum, D. Keislar, and J. Wheaton. Content-based classification, search, and retrieval of audio. *IEEE MultiMedia*, 3(3), September 1996.

[125] A. Wright. Anatomy and ultrastructure of the human ear. *Scott-Brown's Otolaryngology*, 1:6–1, 1997.

[126] C.F.J. Wu. On the convergence properties of the EM algorithm. *Ann Statist*, 11:95–103, 1983.

[127] S.J. Young, P.C. Woodland, and W.J. Byrne. HTK: Hidden Markov Model Toolkit V1. 5. *Entropic Research Laboratories Inc, Decembre*, 1993.

[128] A. Zils and F. Pachet. Automatic extraction of music descriptors from acoustic signals using eds. In *Proceedings of the 116th AES Convention*, May 2004.

[129] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 2, pages 28–31, 2004.

[130] Z. Zivkovic and F. van der Heijden. Recursive unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):651–656, 2004.

Curriculum Vitae

Personal Data

Name:	Jan Wietse Balkema
Address:	Mittelallee 16
	31139 Hildesheim
	Germany
Phone:	+49 5121 2 898 798
Mobile:	+49 176 233 057 09
Email:	wietse@wietsebalkema.nl
Date of Birth:	November 9, 1978
Place of Birth:	Arnhem, The Netherlands
Nationality	Dutch

Professional experience

Robert Bosch GmbH - Corporate sector research, advance engineering multimedia

- Research Engineer	2007 - present
- PhD 'Design and realisation of an efficient content based music playlist generation system'	2004 - 2007

Education

Study

- Electrical engineering, major in telecommunications, University of Twente, Enschede, NL — 1997 - 2004

School

- Dorenweerdcollege, Doorwerth, NL — 1991 - 1997
- Montessori school 'de Eekmolen', Wageningen, NL — 1983 - 1991

Extracurricular activities

Student Union

- Board of supervisors, delegate representing cultural associations — 2001 - 2003

Umbrella organization 'Apollo', representation of 20 cultural associations

- Secretary, chairman (fulltime) — 2000 - 2001

Students drama society 'Nest'

- Head of stage design and realisation — 2000 - 2004
- Vice-chair — 1998 - 1999

Other activities

Sailing instructor, sailing school 'It Beaken' — 2002 - 2004

Rowing

- Coach — 1998 - 1999
- Race rowing, freshmen's eight — 1997 - 1998

Acknowledgments

Writing a PhD thesis is like sailing a long journey over the sea. When the decision is made to start the research, the end is still far away and how it will look like when one arrives at the end is still unknown. During the trip the weather may vary, sometimes the winds are good, sometimes it rains and the wind comes from the wrong direction. The waves may be high which makes one seasick and wonder what was the fun of starting the trip at all. One might wish not to be on a boat and just to sit at the couch at home. The nights with clear sky and a view at the bright stars however are amazingly beautiful, and the stops at harbours are a big relief.

During my travel through this PhD, I have spent many nights at sea, sometimes with good winds and sometimes with bad. I was more than lucky to have a good boat to sail on: I felt comfortable and very welcome at my working group, CR/AEM1 at Robert Bosch GmbH in Hildesheim. Thank you for the three wonderful years, the interesting discussions and the nice social events. Special thanks go to Hartwig, being responsible for the project I was part of the first two years.

One cannot sail without having learnt how to do it and without one who keeps an eye on the progress. This role was taken by several persons. Thanks to Paul Regtien for being my supervisor during the first few months, to Prof. Kees Slump and Prof. Karlheinz Brandenburg for taking over the supervision afterwards, to Ferdi van der Heijden for the interesting discussions on pattern recognition and the people at the Fraunhofer IDMT in Ilmenau for sharing their expertise on MIR in interesting discussions and presentations.

Without having harbours to get fresh water, food and have a good night's rest, there would be no fun in travelling. I am lucky to have many harbours to go to and to know that I am welcome there all the time. In random order: Thanks to my friends of the Parkweg for being the place like home to return to, wherever that is at that moment. Without the 'Kernteam' in Hildesheim, life in Hildesheim would not have been as nice as it was with them. Thanks to

Christina, for sharing so much time with me the first two years. To Frederieke for the nice evenings in Utrecht. To the people of the choir having no name and not having a clue what they are singing, to the people from the salsa-community in Hildesheim for having a good mood everytime and to all those people that I forgot to mention here.

And last but not least, it is essential to know that there are people that care about you, all your life, no matter what. A very warm 'thank you' to my dear family. Papa, Mama, Jikke, thanks for being there.

While writing these last few sentences of this thesis, land is in sight, a long journey was made and it was great. I hope to return to the visited harbours many times and enjoy each visit as much as I enjoyed them the past years.

Wietse Balkema
Hildesheim, Germany
August 2009