JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

Technisch-Naturwissenschaftliche
Fakultät

Indexing Content-Based Music Similarity Models for Fast Retrieval in Massive Databases

DISSERTATION

zur Erlangung des akademischen Grades

Doktor

im Doktoratsstudium der

Technischen Wissenschaften

Eingereicht von:
Dipl.-Ing. Dominik Schnitzer

Angefertigt am:
Institut für Computational Perception

Beurteilung:
Univ.-Prof. Dr. Gerhard Widmer (Betreuung)
Dr. Simon Dixon

Linz, Oktober, 2011

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Kurzfassung

Die vorliegende Dissertation entwickelt ein industriell einsatzfähiges Musikempfehlungssystem. Um dieses Ziel zu erreichen, werden im Zuge der Arbeit drei Probleme gelöst. Diese Probleme verhindern den Einsatz der zur Zeit besten inhaltsbasierten Algorithmen zur Berechnung von Musikähnlichkeit in Systemen, die mit Millionen von Musikstücken operieren:

Zuerst wird gezeigt, wie man die nicht-vektoriellen Musikähnlichkeitsmodelle mit ihren nicht-metrischen Divergenzen korrekt in Algorithmen, die den Zentroiden berechnen, verwendet. Alle bisher veröffentlichen Arbeiten mussten die Daten dazu vorher künstlich vektorisieren.

Zweitens wird gezeigt, wie man das "Hub"-Problem dieser Algorithmen verringern kann. Hubs sind Objekte in einem Empfehlungssystem, die ungewöhnlich oft als nächste Nachbarn gefunden werden. Die besprochenen Musikähnlichkeitsalgorithmen sind besonders stark von dem Problem betroffen, das ihre Retrieval-Qualität signifikant vermindert. Hubs sind ein generelles Problem des maschinellen Lernens. In Untersuchungen zeigt die vorgestellte Methode positive Effekte auch auf einer großen Zahl öffentlicher Datenbanken zur Evaluierung von Methoden des maschinellen Lernens.

Drittens wird eine neue Methode zur Beschleunigung von Musikempfehlungsberechnung vorgestellt. Die Methode verwendet ein Filter- und Verfeinerungs-System. Sie erreicht sehr hohe Retrieval-Genauigkeit und beschleunigt Anfragen um einen Faktor 10–40 im Vergleich zu einer linearen Suche. Die Methode ermöglicht es, alle vorgestellten Musikähnlichkeitsalgorithmen auf großen Datenbanken mit Millionen von Musikstücken zu verwenden.

Schlussendlich werden alle drei entwickelten Methoden in einem großen Musikempfehlungsprototypen zusammengefasst: Das System berechnet (i) ein natürliches Clustering der Musikähnlichkeitsmodelle um (ii) die vorgestellte Methode gegen das Hub-Problem zu verwenden, und verwendet (iii) das entwickelte Filter- und Verfeinerungs-System für schnelle Abfragen in großen Datenbanken. Der Prototyp heißt "Wolperdinger", arbeitet mit 2.3 Millionen Musikstücken und ist imstande, Empfehlungen in einem Bruchteil einer Sekunde zu berechnen. Es ist das zur Zeit größte publizierte System zur inhaltsbasierten Musikempfehlung.

Abstract

This thesis develops a large-scale music recommendation system. To achieve this goal we solve three problems preventing the currently top-performing class of content-based music similarity algorithms from being used as recommendation engine in huge databases with millions of songs:

First, we show how to correctly use the non-vectorial music similarity features with their non-metric divergences in centroid-computing algorithms. All previous approaches had to artificially vectorize the data before they were able to work with the features.

Second, we show how the problem of "hubs" can be alleviated. Hubs are objects in a recommendation system which are unwontedly often retrieved as nearest neighbors. The examined music recommendation methods are especially prone to hubs, significantly decreasing their retrieval quality. We also identify hubs as a problem of machine learning and show the beneficial effects of our method on a large number of general public machine learning collections.

Third, we present a new method to speed up music recommendation queries. The method uses a filter-and-refine systems layout. It achieves a very high retrieval accuracy and speeds up queries by a factor of 10–40 compared to a linear scan. The method enables us to use the music similarity methods with very large databases.

We finally merge all three introduced methods in a large-scale, high-quality music recommendation prototype: the system computes (i) a natural clustering of the music similarity features to (ii) apply the introduced hub-reducing method and (iii) use the filter-and-refine method to allow for fast retrieval. The prototype is called "Wolperdinger", it operates on a collection of 2.3 million songs and it is able to answer recommendation queries in a fraction of a second. It is the largest content-based music recommendation system published to date.

Acknowledgments

This work was carried out while I was working as a researcher at the Department of Computational Perception (CP) at the Johannes Kepler University (JKU), Linz and at the Austrian Research Institute for Artificial Intelligence (OFAI), Vienna. I want to thank Gerhard Widmer for giving me the great opportunity to work at the JKU and OFAI.

I want to thank my colleagues at the OFAI: Arthur Flexer, Martin Gasser, Andre Holzapfel, Rainer Typke, Thomas Grill, Volkmar Klien and David Meichle. Many thanks to the research group in Linz for their hospitality and help whenever I was visiting them: Peter Knees, Tim Pohle, Markus Schedl, Klaus Seyerlehner, Andreas Arzt, Josef Scharinger, Sebastian Flossman, Harald Frostel, Bernhard Niedermayer, Reinhard Sonnleitner, Maarten Grachten, Sebastian Böck and Claudia Kindermann.

Thanks to Elias Pampalk for introducing me to MIR. Special thanks to Peter Knees for helping me with my first paper. Kudos to Arthur Flexer for his long patience and helpful comments improving all our papers. A cordial Danish "Tak" to John Hammer Madsen for the interesting discussions we had. I also want to thank Robert Trappl, the head of OFAI, and the colleagues from the other groups at OFAI. I would have never managed to finish this thesis without your help.

Thanks to Caroline, my wonderful love, who always stood by me. Thanks for all your love and infinite support. Thanks to my beloved parents, fantastic family and all my friends.

This research was supported by the Austrian Research Fund (FWF) under grants L511-N15 ("Music Retrieval Beyond Simple Audio Similarity") and Z159 ("Wittgenstein Award"). The Austrian Research Institute for Artificial Intelligence (OFAI) is supported by the Austrian Federal Ministry for Transport, Innovation, and Technology.

Contents

1 Introduction **1**
- 1.1 The Digital Music Revolution . . . 1
- 1.2 Scope of the Thesis . . . 2
- 1.3 Organization . . . 2
- 1.4 Contributions . . . 4

2 Computational Measures of Music Similarity **5**
- 2.1 Music Information Retrieval . . . 6
- 2.2 What is Music Similarity? . . . 7
  - 2.2.1 Music Similarity in Commercial Applications . . . 8
  - 2.2.2 Music Similarity in Research . . . 9
  - 2.2.3 Common Definition . . . 11
- 2.3 Related Work . . . 12
  - 2.3.1 Context-based Music Similarity . . . 12
  - 2.3.2 Content-based Music Similarity . . . 14
    - 2.3.2.1 Summary . . . 16
- 2.4 Algorithms . . . 17
  - 2.4.1 Generic Music Similarity Algorithm . . . 17
  - 2.4.2 Mandel-Ellis (ME) . . . 18
  - 2.4.3 Elias Pampalk (EP) . . . 20
  - 2.4.4 Pohle-Schnitzer (PS) . . . 21
- 2.5 Evaluation . . . 22
  - 2.5.1 Collections . . . 23
  - 2.5.2 Genre Classification . . . 26
- 2.6 Problems Using the Methods on a Large Scale . . . 27
  - 2.6.1 Non-Vectorial Representation, Non-Metric Music Similarity . . . 27
  - 2.6.2 The Hub Problem . . . 29
  - 2.6.3 Missing Search Algorithms . . . 31
- 2.7 Summary . . . 32

3 Working with Gaussian Music Similarity Models **33**
3.1 Introduction . . . 34
3.2 The Exponential Class of Distributions . . . 35
3.2.1 The Multivariate Normal Distribution . . . 36
3.3 Bregman Divergences . . . 37
3.3.1 Linking the Exponential Family . . . 38
3.3.2 Bregman Centroids . . . 39
3.3.2.1 Centroids for Symmetrized Bregman Divergences . . . 39
3.3.3 K-means Clustering . . . 40
3.4 Bregman Divergences and Music Similarity Measures . . . 40
3.4.1 Kullback-Leibler Divergences for Multivariate Gaussians . . . 40
3.4.1.1 Kullback-Leibler Divergence . . . 41
3.4.1.2 Symmetrized Kullback-Leibler Divergence . . . 41
3.4.1.3 Jensen-Shannon Divergence . . . 42
3.4.1.4 A Jensen-Shannon-like Divergence . . . 42
3.4.2 Kullback-Leibler Centroids for the Multivariate Gaussian . . . 43
3.4.2.1 Left and Right Centroids . . . 43
3.4.2.2 Symmetrized Centroids . . . 43
3.5 Self-Organizing Maps with Gaussian Music Similarity Models . . . 44
3.5.1 The SOM Algorithm . . . 45
3.5.2 The Generalized SOM . . . 46
3.5.2.1 Weighted Gaussian Centroids . . . 47
3.5.2.2 Generalized Algorithm . . . 47
3.5.3 Evaluation . . . 48
3.5.3.1 Results . . . 49
3.5.3.2 Subjective Evaluation . . . 50
3.6 Multivariate Normal (MVN) Matlab Toolbox . . . 55
3.6.1 Initialization . . . 55
3.6.2 Divergences . . . 55
3.6.3 Centroids . . . 56
3.6.4 Clustering . . . 56
3.6.5 Additional Functions . . . 57
3.6.6 Usage Examples . . . 57
3.7 Summary . . . 61

4 Reducing Hubs with Mutual Proximity **63**
4.1 Introduction . . . 64
4.2 Related Work . . . 65
4.3 Scaling Methods . . . 67
4.3.1 Local Scaling . . . 67
4.3.2 Global Scaling - Mutual Proximity . . . 68
4.3.2.1 Related Formulation . . . 70
4.3.2.2 Approximation of the Normal Distribution Parameters . . . 70
4.3.2.3 Linear Combination of Distance Measures . . . 71
4.4 Evaluation . . . 72

4.4.1 Benchmarks . . . 72
4.4.2 Public Machine Learning Datasets . . . 73
4.4.3 Results . . . 74
4.4.3.1 Mutual Proximity Approximation . . . 81
4.4.4 Summary of results . . . 81
4.5 Mutual Proximity and Content-Based Music Similarity . . . 81
4.5.1 Algorithms . . . 83
4.5.2 Evaluation . . . 85
4.5.2.1 Hubness . . . 88
4.5.2.2 Leave-one-out nearest neighbor classification accuracy . . . 88
4.5.3 Discussion . . . 88
4.6 Summary . . . 90

5 Indexing Computer Music Similarity Algorithms 91
5.1 Introduction . . . 92
5.2 Related Work . . . 93
5.3 A Fast Approximative Search Method . . . 94
5.3.1 Preliminaries . . . 94
5.3.2 Original FastMap . . . 95
5.3.3 A Filter & Refine Search Method Using FastMap . . . 96
5.3.4 Modifications . . . 97
5.3.4.1 Rescaling . . . 97
5.3.4.2 Pivot Object Selection . . . 98
5.3.5 Evaluation of the Modifications . . . 98
5.4 Evaluation with Music Similarity Algorithms . . . 100
5.4.1 Single Divergence Measure . . . 100
5.4.2 Linear Combinations . . . 102
5.4.3 Mutual Proximity . . . 102
5.5 Retrieval Quality . . . 105
5.6 Summary . . . 106

6 Dealing with the Music of the World 109
6.1 Introduction . . . 109
6.2 2.3 Million Songs . . . 110
6.3 The Prototype System: Wolperdinger . . . 112
6.3.1 Music Similarity Features . . . 112
6.3.2 Music Similarity Search . . . 113
6.3.2.1 Parameters . . . 113
6.3.3 Qualitative Evaluation . . . 115
6.3.4 Implementation and Server Specification . . . 116
6.3.5 Performance . . . 117
6.3.6 User Interface . . . 118
6.4 Summary . . . 122

7 Conclusions 123

A Applications of Music Similarity **125**
A.1 Music Recommendation in the Banshee Music Player 125
A.1.1 Discussion 127
A.2 The Intelligent iPod 127
A.2.1 Towards the Interface 127
A.2.1.1 Last.fm-based Artist Clustering 127
A.2.1.2 Playlist Generation 129
A.2.1.3 Automatic Region Labeling 130
A.2.2 Prototype 130
A.3 BeoSound 5 Stereo with MOTS 131

Bibliography **133**

Chapter 1

Introduction

Contents

1.1 The Digital Music Revolution 1
1.2 Scope of the Thesis 2
1.3 Organization 2
1.4 Contributions 4

1.1 The Digital Music Revolution

The digital music revolution started silently, without much furor but changed the rules of music consumption, sales and distribution overnight. It began in 1993 when the audio compression technique, MPEG-1 Audio Layer 3, commonly referred as MP3, was standardized and ISO approved [ISO93]. MP3 reduced the storage size of audio files to a fraction of their original size on CD. Along with the Internet and capable computers, easy transfer and storage of music was suddenly possible without requiring any additional physical music medium.

From that point, the size of private digital music collections exploded from few privately owned CDs sitting on the shelf, to thousands of MP3 files stored on hard-disks and portable music players. Ten years after the standardization of MP3, in 2003, the first on-line music stores opened, among them the Apple iTunes[1] on-line music store. Today this on-line store is one of the most popular on-line digital-media stores worldwide hosting over 12 million tracks and accounting for 75% of the on-line music sales in the US, according to the IFPI [IFP10].

Over the last decade more on-line music stores have opened all over the world. In 2009 over 400 on-line music stores in 60 different countries were open, accounting for 27% of the music sold world-wide. According to a Forrester study it is expected that 2012 the music sold on-line

[1] `http://www.apple.com/itunes/`, visited August 12th, 2011

will surpass CD sales[2]. At this point, for the first time in history, more people will shop music on-line instead of buying it in a retail store.

As ubiquitous Internet connectivity is starting to be default, in addition to on-line music stores, music services in "the cloud" are spreading fast. The idea of these services is to pay a monthly fee to get access to their whole multi-million music catalog. The difference to standard on-line music stores is that music is never downloaded to a storage device, but it is only allowed to stream from their servers.

With the possibility to access vast music archives almost anywhere and anytime it will be essential to offer new ways to navigate and interact with these large music collections. To create such services one of the most basic requirements is to understand how and why people perceive two music pieces as similar, related or fit. A number of methods trying to automatically measure "music similarity" have already been developed in the past years. However, today's state-of-the-art techniques do not work with the massive music collections accessible today.

The question of how to scale music similarity measures to work with millions of songs sets the cornerstones of this thesis. In fact this thesis is set in the gap between automatic music recommendation techniques and their commercial use, focusing on the question of how to design a scalable system for state-of-the-art music recommendation algorithms.

1.2 Scope of the Thesis

This thesis proposes solutions to three major problems which prevent using a whole class of standard music similarity algorithms in large scale applications for high-quality music recommendation: (i) the non-vectorial music similarity features and their non-metric similarity functions can not be clustered using traditional methods, (ii) the employed algorithms exhibit very high hubness leading to strong imbalances in the their retrieval results, (iii) there do not exist viable indexing solutions which can be used with the covered methods to scale the algorithms.

These problems define the key points to be solved in this thesis. This thesis does not define a new music similarity algorithm, but it solves the identified major problems for a whole class of existing music recommendation algorithms to scale these methods. A large scale prototype music recommendation system is finally presented using all techniques introduced in this thesis. We evaluate the system and show that it offers a high retrieval quality, short answer time and works with millions of music pieces.

Although the focus of this work is clearly music similarity algorithms, its theoretical results are not limited to music similarity algorithms. In the case of the hubness problem we explicitly show that our method to remedy the effect, also applies for general machine learning problems.

1.3 Organization

Chapter 2 introduces the reader to the research area of music information retrieval and automatic content-based music similarity measures. The general notion of music similarity, as used in the music information retrieval literature, is reviewed and explained. We present the currently

[2] `http://www.forrester.com/ER/Press/Release/0,1769,1200,00.html`, visited August 12th, 2011

standard and top-ranked music similarity measures which are all from a class of algorithms that use multivariate Gaussians to represent their similarity features. We identify three problems which prevent the algorithms from being used for high-quality music recommendations in large-scale scenarios. These problems and the development of possible solutions form the topics of the next chapters and the core of the thesis.

Working with Gaussian Music Similarity Models

The first problem we identify are the uncommon non-vectorial music similarity models (parametric multivariate Gaussians) and their attached non-metric divergences (Kullback-Leibler divergences). Chapter 3 introduces the parametric Gaussian, their related divergences and puts that in the appropriate information theoretic context. We show how the multivariate Gaussian centroids of the Kullback-Leibler divergence are defined. With the centroids, any centroid-computing algorithm like the standard k-means algorithm can be natively used with these non-vectorial features and divergences. On top of that we develop a modified self organizing map algorithm which can be used to cluster parametric multivariate Gaussians as opposed to the standard vector space it has been developed for. The method is shown to work better than previous algorithms using vector space approximations. A freely available, high-performance Octave/Matlab toolbox to ease working with multivariate Gaussians is presented.

Reducing Hubs with Mutual Proximity

A general problem of recommendation algorithms using high dimensional data is identified in Chapter 4 – hubness. Hubs are data points which keep appearing unwontedly often as nearest neighbors of a large number of other data points. This effect is particularly visible in the examined music similarity algorithms and in general problematic in algorithms for similarity search. Algorithms with high hubness find the same similar objects over and over again while never recommending certain other objects. This effect significantly degrades the performance of retrieval algorithms. We present a method that alleviates the hubness problem while at the same time increasing the quality of the retrieval results. The method is shown to be effective on general machine learning databases, and on music similarity algorithms exhibiting high hubness. A highly efficient approximation of the method which just requires a k-means clustering is introduced so the proposed method can be used for very large collections.

Fast Search with the Music Similarity Measures

Chapter 5 introduces a filter-and-refine technique to enable searching for nearest neighbors in sub-linear time using any of the examined music similarity measures. To achieve that we use a modified FastMap technique to compute a vector space representation of the features. A query is processed by first roughly filtering the collection for possible candidate nearest neighbors in the vector space. In a second step the filtered set is re-ranked according to the original similarity measure only using the filtered set of features. The method is shown to be very accurate and achieves speedups by 10–40 orders of magnitude, compared to a linear scan. A US patent (12/458,230) based on this method is pending.

After tackling all problems preventing the use of music similarity algorithms in large scale applications, Chapter 6 merges all methods which have been developed and presents "Wolperdinger", a prototype large scale music recommendation system. The Wolperdinger system works with 2.3 million music pieces and is able to process recommendation queries in less than a second on a standard computer.

Chapter 7 concludes this thesis. Finally Appendix A reviews three applications of content based music similarity algorithms which have been developed in the years leading to this thesis.

1.4 Contributions

The main contributions of this thesis are introduced in the three core chapters (3, 4, 5) of this work. They include:

- The development of a novel filter-and-refine indexing method which can be used to for fast music similarity search in large collections. We show how the proposed method can be used to query several million songs for their acoustic neighbors instantly while producing almost the same results that would be returned by a linear scan over the whole database [SFW10, SFW09].

- The development of a generalized self-organizing maps algorithm which is able to cluster Gaussian music similarity features [SFWG10] using their Bregman centroids. We can show for the first time how Gaussian music similarity features can be clustered natively using a self-organizing map. The development of the freely available Matlab MVN (Multivariate Normals) toolbox open-sources the algorithms for further research. It implements the information theoretic methods which are introduced in the thesis to easily deal with multivariate Gaussians.

- The development of a method which alleviates the problem of hubness in high dimensional spaces [SFSW11]. The method is called Mutual Proximity (MP) and combines mutual object neighborhood information in a probabilistic way. We show that the method reduces hubness while at the same time significantly increasing the retrieval quality. We show these surprising effects on a large number of standard machine learning databases. We also develop an approximate method of Mutual Proximity which can be used on large collections.

  We show that the standard music similarity algorithms we employ exhibit very high hubness and as a result of this MP can increase their retrieval accuracy significantly without changing the features.

All three main contributions of this thesis are required to build an industrial-strength content-based music recommendation system – the objective of this thesis – which is finally developed in Chapter 6.

Chapter 2

Computational Measures of Music Similarity

Contents

2.1 Music Information Retrieval 6
2.2 What is Music Similarity? 7
2.2.1 Music Similarity in Commercial Applications 8
2.2.2 Music Similarity in Research 9
2.2.3 Common Definition 11
2.3 Related Work 12
2.3.1 Context-based Music Similarity 12
2.3.2 Content-based Music Similarity 14
2.4 Algorithms 17
2.4.1 Generic Music Similarity Algorithm 17
2.4.2 Mandel-Ellis (ME) 18
2.4.3 Elias Pampalk (EP) 20
2.4.4 Pohle-Schnitzer (PS) 21
2.5 Evaluation 22
2.5.1 Collections 23
2.5.2 Genre Classification 26
2.6 Problems Using the Methods on a Large Scale 27
2.6.1 Non-Vectorial Representation, Non-Metric Music Similarity 27
2.6.2 The Hub Problem 29
2.6.3 Missing Search Algorithms 31
2.7 Summary 32

This chapter examines computational measures of music similarity. Section 2.1 briefly introduces to the general music information retrieval research domain. It leads over to Section 2.2 which attempts to define the term "music similarity" from a music information retrieval perspective. Section 2.3 reviews related work and research. Besides discussing the state of the art and applications, this section also examines the difference between context- and content-based music similarity. The focus of this thesis is on algorithms for the latter one, that is content-based music similarity measures using features computed solely the audio signal to compute similarity. Section 2.4 introduces a class of de-facto standard algorithms to compute music similarity. Section 2.5 evaluates and benchmarks three algorithms which will be used throughout this thesis. Eight different public music collections are used to benchmark and evaluate the algorithms.

In the final section of this chapter (Section 2.6) we identify major problems with the class of computer music similarity methods we have introduced. These problems hinder the development of a truly scalable music recommendation system. The section discusses the individual problems faced by the algorithms. In the next chapters we will gradually develop methods to resolve these problems to arrive at a truly scalable music recommendation system.

2.1 Music Information Retrieval

This doctoral thesis is set in the music information retrieval research domain. One of the first publications mentioning music information retrieval (MIR, [Kas66]) dates back as far as 1966. In his publication Kassler proposed a special purpose programming language to describe a music piece as a well defined program. Things have changed a lot since then as music information retrieval has evolved into a much wider-spread research field. Hand in hand with the growing success of digital media and the new ways to process audio information, research interest expanded significantly.

In 2003 Downie [Dow03] characterized music information retrieval as a much wider domain. Downie distills seven basic music information "facets" defining the domain: the *pitch*, *temporal*, *harmonic*, *timbral*, *editorial*, *textual*, and *bibliographic* facet. These facets describe the main music characteristics MIR tries to extract, learn predict or classify using any information source available. While Downie's definition is in general still valid today, the scope of MIR expanded certainly at least to an eighth facet: the *social* facet. This facet covers popularity, discussions and opinions of music and emerged from today's ubiquitous social networks where this information all of a sudden could be gathered.

The ISMIR [DBC09] (International Society for Music Information Retrieval) conference is one of the main international conferences for music information retrieval and was established in 2000. The conference is certainly a hub point for MIR research. To get a snapshot of the current focus of MIR we show a table which lists the topics of ISMIR as they were advertised in the call for papers for the 2011 conference[1] (Table 2.1) and created a tag-cloud visualization [BBF06] using the full ISMIR 2010 conference proceedings (Figure 2.1). Tag clouds are a simple method to visualize the frequency distribution of keywords extracted from text. The words are resized according to their frequency distribution in the text, i.e., words printed in large font sizes occur

[1] `http://ismir2011.ismir.net/callforpapers.html`, visited August 12th, 2011

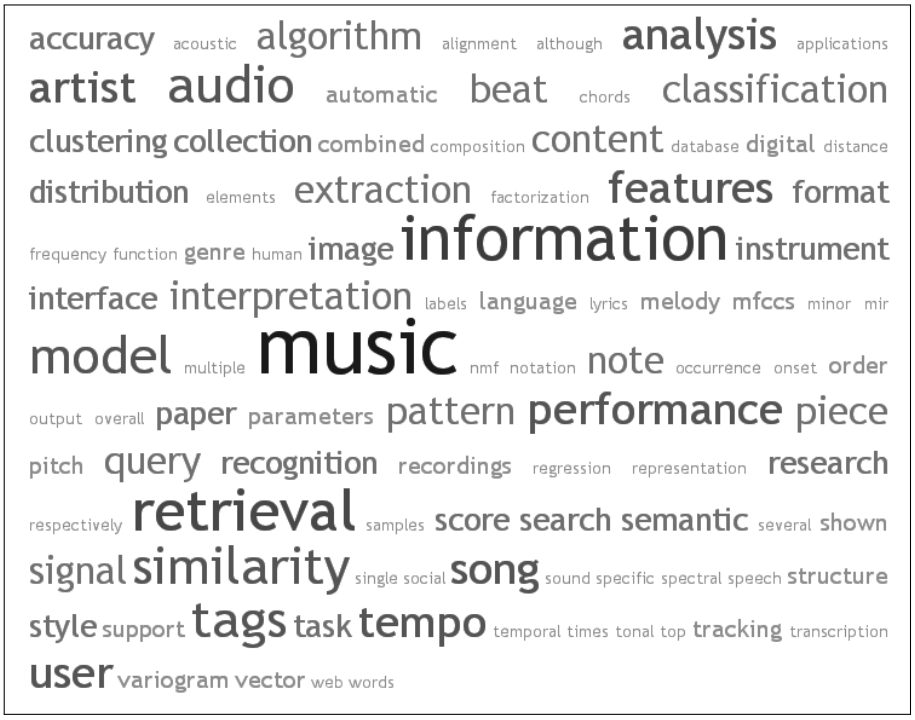


Figure 2.1: A Tag-Cloud representation of the full conference proceedings of the ISMIR trying to visualize the current hot topics.

more often in the text and can be interpreted as more important than words printed in small font. Tag clouds have been made popular by the photo sharing website flickr.com and have found widespread use. To create the ISMIR tag-cloud we removed stop-words from the text and used the one hundred most frequent words of the proceedings (Figure 2.1). The tag-cloud shows a high relevance of "retrieval" and "similarity" in the ISMIR publications of 2010.

To finally put this thesis into MIR context: in terms of Downie's facets, we extract *timbral* and *temporal* features from music pieces to compute *music similarity*, in terms of ISMIR conference topics this thesis develops an *indexing method* to build a large scale *content-based music recommendation* and *playlist generation system* using computer *music similarity* methods.

2.2 What is Music Similarity?

The definition of a general similarity measure between two arbitrary music pieces is a challenging exercise as the concept of music similarity is highly manifold and subjective. However in many scenarios where music is played, the concept of similarity is an important aspect of music consumption. Subsequent pieces should somehow fit together. It may be music on the radio, in a bar, at a dance event or in a concert where the whole impression should harmonize.

MIR research topics	• meta-data • user modeling • automatic transcription and annotation of music • optical music recognition • libraries • archives • digital collections • database systems • indexing • digital rights management • content-based querying and retrieval • fingerprinting • music signal processing • user interfaces • score following and audio alignment • music recommendation • music similarity • playlist generation • performing arts and multimedia • social tagging systems • music summarization • automatic classification • modification and transformation of music data • automatic composition • computational musicology • text and web mining • evaluation and annotation issues • methodological and philosophical issues • social, legal, ethical, and business issues

Table 2.1: Characterizing the field of music information retrieval using the ISMIR call for papers topic list.

In each of these settings the idea behind the selection of matching music may be different, but still, it is there. A radio DJ may select songs according to the genre of his show, at a dance event only music of a selected time period may be played or a department store may wish to play ambient background music preferably with unnoticeable transitions between different songs.

An algorithm which tries to simulate that, first needs to identify core attributes of music which describe the desired aspects. But what are these natural attributes which should be used to describe music for computing general music similarity?

2.2.1 Music Similarity in Commercial Applications

To get a sense of what seem to be reasonable choices, at least for popular music, we first look at three successful large commercial applications which collect music attributes to work with them.

All Music Guide

An expert source for music information is the *All Music Guide*[2]. The AMG categorizes music, artists and bands according to main attributes that are defined by music experts. They categorize music pieces into genre (rock, pop, electronic,...), style (contemporary, political, psychedelic,...), mood (rousing, bitter, playful,...) and theme (late night, road trip, relaxing,...). Of course also standard information like the decade, chart positions, artist, band and composer information is stored. The guide started 1995 as a small site for music fans to read more about the music they like. Today the AMG is licensing their music information to large companies who need this meta-data. MIR research is also often (re-)using their mature taxonomies [WL02, PC00, BEL03,

[2] `http://www.allmusic.com/`, visited August 12th, 2011

EWBL02]. Recently they have engaged in creating a music discovery engine (Rovi[3]) where their own data is used to power a recommendation engine.

Pandora Internet Radio

Similarly to this, the "Music Genome Project" operated by *Pandora Internet Radio*[4] defines its own attributes to classify music. However its attributes[5] are much more detailed than the ones from AMG and include music structure, rhythm, tonality, instrumentation, feelings, influences, lyric content, vocals and many more. The individual attributes are hand-picked by music experts and need to be set manually for each music piece. That, unfortunately, does not scale well. Having an individual data vector of music attributes for every music piece is the basis for Pandora to compute music similarity. In their case it is used to stream personalized radio.

Last.fm

Last.fm[6] is a gigantic source of unstructured attributes each characterizing artists and their music. It is an Internet radio station offering personalized radio streams. It works by collecting arbitrary tags for artists and music pieces from its community. The tags are used to dynamically generate personalized music streams for their users. To get a feeling which tags are mainly used we created Figure 2.2 which shows a tag cloud [BBF06] visualization. We generated the figure from a Last.fm tag dataset of seven million artist tags. The data-set was created in 2007 [GLA+09] and is available freely on the Internet.[7]

Looking at the figure it can be seen that besides decades (00s, 90s, 80s, 70s, 60s) or language/origin (French, Finnish, German, Polish) clearly genre names (rock, jazz, indie, electronic, etc.) dominate the plot. Last.fm data is widely used in works examining music similarity [ELBMG07, Lam08, KPSW07].

2.2.2 Music Similarity in Research

Similarly to the commercial applications discussed in the previous section, the concept of music genres as a basic aspect of music similarity is prevalent in the research literature too. Aucouturier and Pachet [AP02a] and Tzanetakis et al. [TC02] were one of the first to use a (coarse) musical genre taxonomy to evaluate their music similarity measure. Typical musical genres which they used were: Classical, Country, Disco, Hip-Hop, Jazz or Rock. Music from these basic genres is usually clearly separated in terms of its sound. The observation that musical timbre is a dominant factor also led to the development of music similarity measures based solely on music timbre, one of the first being Logan [Log00] in 2000. Today almost all common music similarity measures still use a timbre component similarly to the one introduced by Logan (see Section 2.4).

Besides musical timbre another important aspect defining the notion of a musical genre is rhythm. Work in this direction has for example been done by Pampalk et al. [PFW05] or

[3] `http://www.allrovi.com/`, visited August 12th, 2011
[4] `http://www.pandora.com/`, visited August 12th, 2011
[5] `http://en.wikipedia.org/wiki/List_of_Music_Genome_Project_attributes`, visited August 12th, 2011
[6] `http://www.last.fm/`, visited August 12th, 2011
[7] `http://musicmachinery.com/2010/11/10/lastfm-artisttags2007/`, visited August 12th, 2011

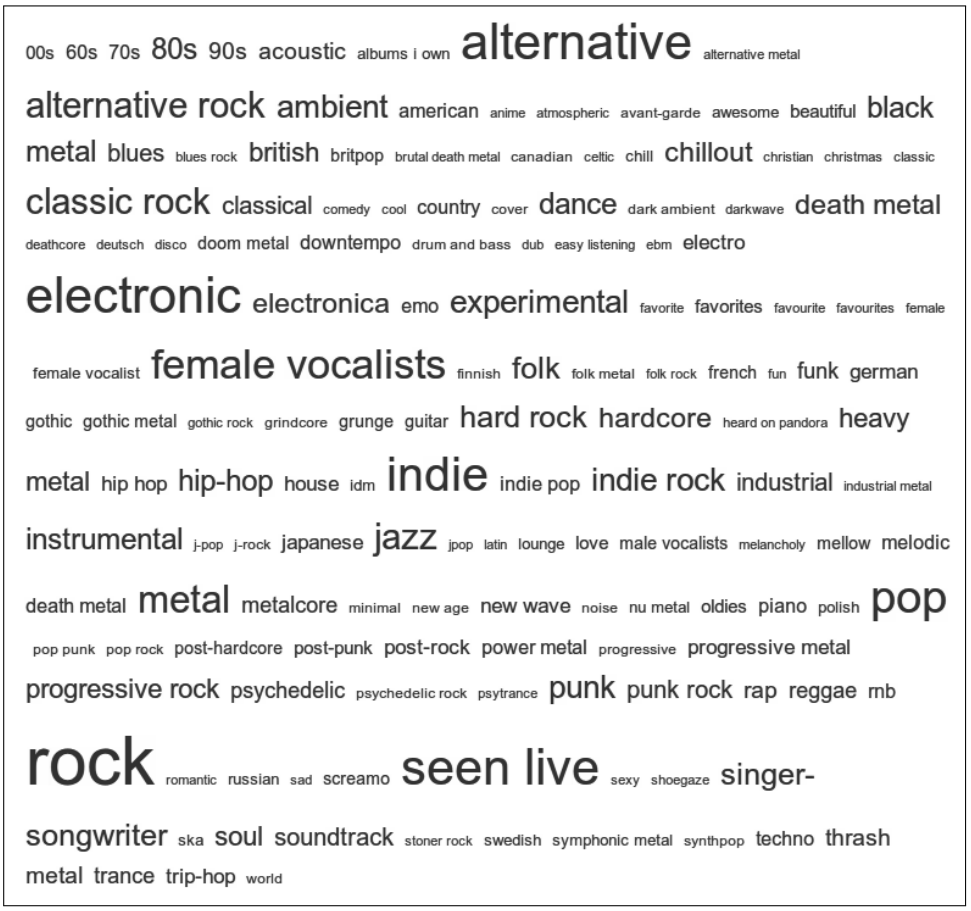


Figure 2.2: A tag cloud visualization of a publicly available dataset of seven million Last.fm tags. Tags on Last.fm can be chosen freely. The tag-cloud visualization plots the top 150 tags, arranged in alphabetical order. The font size is determined by the term frequencies of the individual tag.

Gouyon et al. [GDPW04]. They use music pieces from classical dances (Waltz, Rumba, Foxtrot, Salsa, etc.) to evaluate the performance of their algorithms. Rhythm is a crucial element in non-western music where musical genres are often predominantly defined by a certain repeating rhythmic pattern. This is for example the case with Malay music [NDW05] or traditional Crete (Greek) music [HS09]. In these cases rhythm is the most important aspect to identify musical genres.

In a growing number of publications automatic mood and emotion detection is used to compute similarity between two music pieces. This is done by analyzing the raw audio signal (see Liu et al. [LLZ03] or Li et al. [LO04]) as well as by using information like the music lyrics (see Laurier et al. [LGH08]). Hu et al. [HBD07] built a music mood ground-truth set. Usually the basic moods which are used are based on the two dimensional valence-arousal graph from Russell [Rus80]. Methods to classify music emotions typically use the adjective groups (fearful, sad, quiet, relaxing, happy, amazed,...) introduced by Farnsworth [Far58].

Other methods which have been used to construct a music similarity measure take into account cultural and context information about music. This information can be mined from web-pages [KPW04, CF00] or social networks [SPKW06]. Music context information can include the artist/band biography, news, trivia, lead singer and band member information, lyrics, the music video or even fashion linked to a certain type of music.

2.2.3 Common Definition

Looking at all these different views on what defines computational music similarity, we try to distill common main aspects. Although all introduced taxonomies (in commercial applications as well as in research) seem to differ a lot, nevertheless a single common main music taxonomy is clearly visible. The notion of musical genres seem to be the dominant similarity aspect for popular music. Its notion is found in a large number of the presented works. If two music pieces are from the same genre, they are likely to have more similarity than two pieces from different genres. However as we have seen, a music genre can be defined by an array of features: musical style, instrumentation or even by the typical sound of a decade. This makes the concept of genres rather ill defined as Aucouturier and Pachet [AP03] remarked. Still genres appear to be an important and widely accepted concept to describe similar music.

A second main aspect defining the term computer music similarity can be derived from context information. If two music pieces share similar context information like the artist, decade, instruments or both even have a similar music video they can be considered similar.

In the small survey a third interesting facet of music similarity emerged. Feelings, mood and emotions coupled to music can seem to play an important role defining the similarity of two music pieces. Sometimes feelings and mood can even be found again in musical sub-genres like in “Psychodelic Rock” or “Happy Hardcore” (an electronic music sub-genre).

Based on these observations we define the term Computational Music Similarity Measure *to refer to a method which is capable of computing a similarity value between two music pieces according to their musical genre, context and mood. A method for computing a similarity value between two music pieces may use any possible information available.*

A mixture of the three ingredients which we distilled would likely define a good path towards a general computer music similarity measure. This thesis focuses on music similarity which is optimized to return similar songs from the same genre and develops a scalable method for that. More specifically, the similarity measures we will use are based on models of timbre and rhythm. Context is hard to assess because relevant information is usually not available. Aspects of mood could be captured by our timbre and rhythm models.

2.3 Related Work

There is high interest in creating a meaningful music similarity algorithm, as numerous possible applications require one to work:

- Services to automatically recommend new unknown music to listeners, e.g., in a streaming service, an on-line music store or even a small personal music device to automatically create playlists on-the-go [Log02, AP02b, AH06, FSGW08].
- Methods to simplify retrieval of relevant similar audio files from expert databases, like Pampalk et al. [PHH04] who propose a system to organize drum sample libraries.
- Methods to automatically generate meaningful maps of music collections [Pam03, KSPW07, SFWG10] to create better ways to browse music.
- Automatic classifcation of unknown music according to specific learned labels [ELBMG07, TC02, MF04]
- Duplicate or near duplicate [HK03, CS07] detection of music pieces.

All these scenarios use or implement a similarity algorithm. Based on their goal and the data they need to operate on, the algorithms may differ. To discuss the main algorithms which are used to compute music similarity, we first distinguish between context- and content-based music similarity algorithms.

2.3.1 Context-based Music Similarity

Although only vaguely defined in the literature, the term "context-based features", "cultural features" and "community meta-data" are commonly used interchangeably. We use the term *context-based music similarity* to denote a music similarity measure which does not use information which is extracted in some way from the audio file itself, but rather uses external sources. Sources which are used in context-based music similarity methods are the Web, social networks, expert or survey data. Context-based methods do not require the audio file and are able to capture a multitude of information about the music pieces. However they suffer from a few shortcomings which need to be considered before using such an algorithm (see Celma [Cel06]): (i) The "cold start problem". This problem arises for newly released songs or new artists/bands, where usually not enough (or any) information for a method to work properly can be found. That is, for newly released songs there are usually no discussions, music reviews or websites with information an algorithm could process. (ii) The "long tail problem". The "long tail" includes

unknown artists and songs where barely any information can be found. The sparsity of information leads to the effect that only songs/artists where sufficient information is available, i.e. popular songs/artists, are recommended. The "long tail" is never recommended. This effect is sometimes also referred to as a "popularity bias".

While context-based methods can capture a multitude of different information criteria, but suffer from the described problems, content-based methods (Section 2.3.2) working solely with the audio signal have the inverse problem. These methods can process any music piece, but can not extract all the desired information from the audio signal. This fact led to many works which tried to combine the two methods to profit from their respective advantages (see for example Knees et al. [KPSW07] or Aucouturier et al. [APRB07]).

In the next paragraphs we try to give an overview of the different methods which are used in context-based music similarity algorithms. The methods are grouped by the data source which is used. A comprehensive study on context-based methods for music similarity estimation was published by Schedl and Knees [SK09].

Web

Methods which use data from the Web need to infer knowledge about music from unstructured data. Usually these methods work on top of a Web search engine or need to implement their own focused crawler to retrieve data. If a search engine is used, usually a query for the artist name together with a key phrase like "music review" [WL02] or "music genre style" [KPW04] is performed. Standard text retrieval methods, i.e., the bag-of-words representation and standard tf · idf weighting (term frequency · inverse document frequency, see Baeza-Yates and Ribeiro-Neto [BYRN99a]) is used.

Whitman and Lawrence [WL02] use data from the Web to learn artist similarities. To do so they use a list of 400 artists which they collected from a Peer-to-Peer network (see paragraph below), they search for the artists names and the key phrase "music review" in a search engine and crawl the top 50 retrieved Web-pages. Part-of-Speech (PoS) tagging and standard tf · idf are used to create an artist similarity matrix. Other methods use co-occurrence models to infer knowledge about artists. Schedl et al. [SPKW06], for example, use Web co-occurrence analysis to predict the music genre of an artist. Usually Web analysis is limited to extracting band or artist information, as information about individual songs is very sparse.

Lyrics

Song lyrics are a possible source to obtain information about individual music pieces. Logan et al. [LKM04] were one of the first to try to use lyrics for artist similarity measurements, but did not succeed; simple content-based methods outperformed their method, which just used the song lyrics. They propose to mix lyrics similarity with content-based information as both methods showed their strength in different cases. Mahedero et al. [MMC+05] find three special aspects where music lyrics analysis outperforms other methods: music lyrics are important if (i) the language of a music piece needs to be identified, (ii) the general song theme is of interest and (iii) music lyrics are useful to detect cover pieces of songs. Using lyrics to identify the mood of music pieces is proposed by Laurier et al. [LGH08].

Annotations

Methods which rely on manually annotated data to compute context-based music similarity were already discussed in the introductory section about music similarity. Crowd or explicit expert data is required in services using annotation data. Popular examples are the All Music Guide (experts), last.fm (crowd) and Pandora (experts). The tags from last.fm are openly accessible and have found widespread use in research. For example, Green et al. [GLA+09] implement a collaborative-filtering method using last.fm data to recommend music. Their algorithm compares weighted artist tag-clouds to define an artist similarity measure.

As it is very difficult for researchers to get manual music annotations it was proposed to use games to get users to tag music. One of the first games was Tagatune by Law et al. [LVADC03] which plays a song to two users. The users are asked to describe the song and then they have to figure out (based on their descriptions) if they are both listening to the same tune.[8]

User Collections/Data

To retrieve user collection data, usually Peer-to-Peer (P2P) networks with users sharing their collections are employed. A number of approaches using user collection data from P2P networks have been published using the (now closed) OpenNap network (see for example [WL02, BLEW04]). A more recent system developed by Shavitt and Weinsberg [SW09] tries to create an artist and song similarity measure using data from the Gnutella file sharing network. They build a user-song relation network from the crawled data to compute the similarities.

Besides P2P networks, co-occurrence analysis of user-generated music playlists data (e.g., downloaded from the "Art of the Mix"[9] homepage) has also been used for music similarity research [CCKB06].

2.3.2 Content-based Music Similarity

Content-based music similarity measures only use the audio signal to build similarity models. The scheme of a content-based music similarity measures along with an in-depth look into three specific algorithms is presented in Section 2.4. The following paragraphs give an overview of algorithms computing content-based music similarity. We group the methods by the type of features they extract.

What is common to all these methods is that they are often referred to as "bag of frames" methods as their music features are computed independently from short audio slices (c.f., "bag of words" approaches in classic text information retrieval). As music happens in time, these methods potentially ignore a lot of relevant information. So far no satisfactory ways of modeling temporal aspects of music have been found.

Music Timbre

Presumably the first method for computing content-based music similarity was developed by Logan and Salomon [LS01]. They use the Mel frequency cepstrum coefficients (MFCCs) to compute music similarity. MFCCs have their origin in speech processing (see for example Ganchev

[8] `http://www.gwap.com/gwap/gamesPreview/tagatune/`, visited August 12th, 2011
[9] `http://www.artofthemix.com`, visited August 12th, 2011

et al. [GFK05]). They are a noise-robust, compressed representation of the spectrum. Logan proposes to compute 13 MFCCs for every 26 ms of audio (with a hop size of 13 ms). A k-means clustering of the MFCC vectors defines the so called signature of each analyzed music piece. As solely an average spectrum of MFCCs is used in the algorithm, methods based of this idea are said to model the overall music timbre of a song. The earth movers distance is used to compute the similarity between two music pieces.

A number of algorithms were already published before Logan and Salomon to find or identify similar audio signals (e.g., Wold et al. [WBKW96] or Feiten et al. [FG94]). Logan and Salomon however were presumably the first to investigate the specific question of the similarity between two music pieces.

Methods based on the ideas of Logan and Salomon were developed by Aucouturier and Pachet [AP02a]. They train Gaussian Mixture Models (GMMs) on MFCC vectors to model the similarity features. To compute similarity between two GMMs they use a Monte-Carlo sampling method which randomly samples from one GMM to compute the likelihood that the samples could have been generated from the other GMM.

Mandel and Ellis [ME05] propose a very straightforward variant to compute music similarity. Again based on Logan/Salomon [LS01] they use MFCCs, but contrary to Aucouturier/Pachet [AP02a] who train a GMM, they only estimate a single multivariate Gaussian model on the MFCC vectors. That way they can use closed from solutions of the Kullback-Leibler divergence to compute the similarity between two music models. Besides being very fast compared to older algorithms, their method also outperformed other music similarity measures in terms of quality. It is since then one of the de-facto standard methods to compute music similarity. We will discuss the algorithm in Section 2.4.2.

Rhythm

The beat-spectrum is an early algorithm to model (and compare) rhythm in music [FU01]. It works in three steps: (i) the audio is transformed into an MFCC representation, (ii) all frames are compared and their pairwise distances are recorded in a distance matrix and (iii) the beat spectrum is found by finding periodicities in the similarity matrix using auto-correlation.

An approach which uses rhythm features to compute music similarity was published by Pampalk et al. [PRM02a]. They compute so called rhythm- or fluctuation patterns. Fluctuation patterns are a two dimensional representation of the periodicities found in the audio signal for each frequency band. They use the Sone scale, a subjective loudness measure originally proposed by Stanley Smith Stevens [Ste36], and compute a fast Fourier transform (FFT) on 128 Sone-frames to find the periodicities in the spectrum. To compute the similarity between two fluctuation patterns the squared Euclidean distance is computed.

Pohle et al. [PSS+09] improve the fluctuation patterns and develop the so called onset patterns. In comparison to the fluctuation patterns, onset patterns (i) use the Cent- instead of the Sone-scale, (ii) emphasize onsets before computing the onset pattern for a song and (iii) represent the periodicities on a logarithmic scale. Their method is shown to outperform the fluctuation patterns on the standard "Ballroom dances" dataset.

Multiple Features

Besides algorithms explicitly modeling rhythm and timbre, a number of music similarity measures combine multiple features to improve their performance.

Tzanetakis and Cook [TC02] create a similarity measure partly based on timbre, rhythm and pitch features so that a similarity signature of each music piece is a 30-dimensional vector. The timbre part of the feature vector is computed from the MFCC representation, the rhythm features use beat histogram [TEC01] features and the vector elements describing the pitch are computed using a multi-pitch detection algorithm. Tzanetakis and Cook are one of the first to propose a similarity function combining various aspects (predominantly musical timbre and rhythm) into a single music similarity feature. Their similarity algorithm is integrated in MARSYAS [TC99] an open-source framework for audio processing where it has been enhanced since then.

Similarly to Tzanetakis and Cook, Neumayer et al.[NLR05] also mix timbre and rhythm features in their music similarity measure. Their rhythm features are derived from the Fluctuation Patterns, the musical timbre features are statistical descriptors computed from a Bark spectrum, another psychoacoustically motivated spectrum.

The success of Mandel and Ellis's work [ME05] inspired a number of derivative methods which improve the measure but keep the general idea, to use MFCCs and a single Gaussian model. A successful method was published by Pampalk [Pam06]. The similarity method combines the (rhythm-) fluctuation pattern similarity with the single Gaussian timbre features of Mandel and Ellis. This algorithm variant is also published as an opensource Matlab toolbox [Pam04] (see Section 2.4.3). Another derivative method is the algorithm published by Pohle et al. [PSS+09]. By improving the rhythm component, the similarity measure could be improved substantially. The method of Pohle et al. is discussed in Section 2.4.4.

A method which currently competes with the single Gaussian representation of music similarity features was proposed by Seyerlehner et al. [SWP10] in 2010. They call their method Block-Level Features/Similarity. The Block-Level features consist of a number of different music features, describing musical timbre, rhythm, onsets or toneness. What sets this method apart from previous approaches, is that each of the features is iteratively computed for a predefined audio block length instead of very short frames as it has been done in most methods before. The individual feature blocks which are computed for each time-step are aggregated into a single block for each music piece.

2.3.2.1 Summary

The methods by Pohle et al. [PSS+09] and Seyerlehner et al. [SWP10] are at present seen as two of the top performing music similarity measures according to the yearly music information retrieval evaluation exchange (MIREX, [Dow08]). The MIREX[10] is a community-based framework for the formal evaluation of Music Information Retrieval systems and algorithms, similar to TREC[11] in text retrieval research. Since 2006 (with the exception of 2008) there has been a yearly *Audio Music Similarity and Retrieval* (AMSR) task, which evaluates music similarity measure. Contrary to all other evaluations, this evaluation task is performed by anonymous human listeners who rate music playlists generated by the different algorithms.

[10] `http://music-ir.org/mirex`, visited August 12th, 2011
[11] `http://trec.nist.gov/`, visited August 12th, 2011

Year	Team Ranked #1	Algorithm
2006	Elias Pampalk (EP)	[Pam06]
2007	Pohle-Schnitzer, (PS)	[Pam06]
	George Tzanetakis (GT)	[TEC01]
2009	Pohle-Schnitzer (PS)	[PSS+09]
2010	Seyerlehner-Schedl-Pohle-Knees (SSPK)	[SWP10]
	Pohle-Schnitzer (PS)	[PSS+09]

Table 2.2: MIREX: Music Information Retrieval Evaluation eXchange - *Audio Music Similarity and Retrieval* (AMSR) results: top ranked algorithms. There was no AMSR task in 2008

Table 2.2 lists the top-ranked teams and algorithms of the MIREX AMSR evaluation task starting with 2006. We will use the MIREX team name acronyms (in brackets) to refer to a particular allgorithm in this thesis.

2.4 Algorithms

This section presents the design of a generic music similarity computing algorithm before describing the three similarity computing algorithms which will accompany us throughout the thesis. First we introduce the de-facto standard music similarity computing algorithm by Mandel and Ellis [ME05] (hereafter denoted with *ME*). ME defines a commonly used, straightforward and still powerful algorithm to compute music similarity. It is the basis of the other two algorithms: the MIREX 2006 top performing algorithm by Elias Pampalk (denoted with EP) and the MIREX 2009/2010 top performing algorithm by Pohle-Schnitzer (hereafter denoted with *PS*).

2.4.1 Generic Music Similarity Algorithm

A generic music similarity algorithm typically is defined by two functions: (i) the *Feature Extraction* and (ii) the *Similarity function*.

To use the *Similarity* function first the music pieces need to be analyzed to extract the features. Typically after the *Feature Extraction* these features are stored to be reused. Usually the features are computed in four generic steps:

- *Decoding*: The encoded music file (MP3, AAC) is decompressed and transformed to its discrete time-domain representation (see Figure 2.3a). In the time-domain representation audio is described by a number of uniformly spaced, discrete real numbers $x(t)$ (samples) at each specified time step t. The number of samples per second is called the sample rate of the audio. Standard sample rates which are used in music similarity algorithms are 22 050 Hz – 44 100 Hz in a mono-channel signal.

- *Frequency Domain Representation*: A Short-Time Fourier Transformation (STFT) computes the discrete frequency-domain representation of the audio. In the frequency-domain

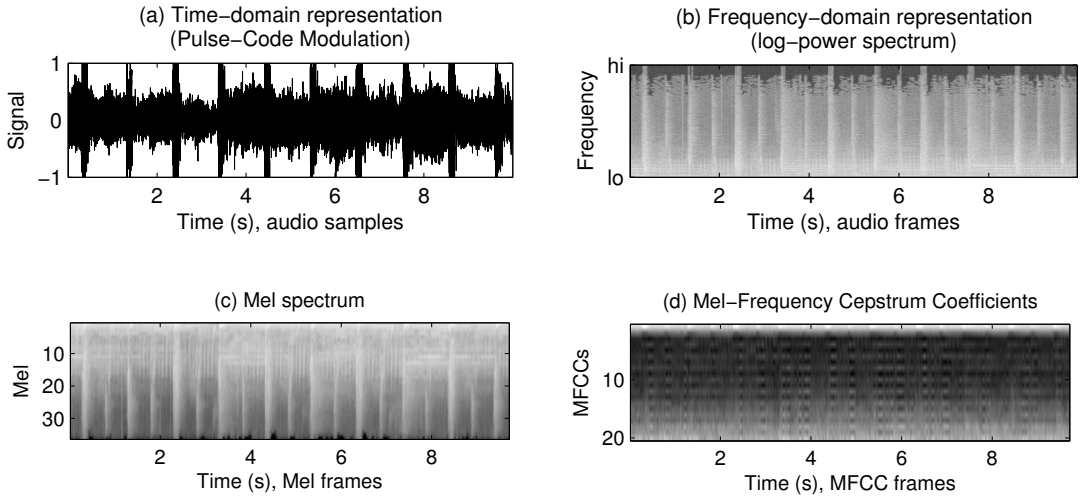


Figure 2.3: Visualization of the typical *Feature Extraction* steps of a music similarity algorithm showing the various spectrum representations of a 10 s audio snippet.

the signal is represented by the spread of its frequencies $f_b(t)$ over a given number of frequency bands b per analyzed time interval t (window size). Music similarity algorithms typically use $1024 - 2048$ samples per STFT. In most cases the power spectrum representation is computed, which represents the signal with the energy per frequency-band and time interval (see Figure 2.3b).

- *Psychoacoustic Modeling*: The frequency spectrum computed in the previous section does not take into account any characteristics of the natural human ear. That is why similarity algorithms use psychoacoustically motivated representations prior to computing the similarity features. Common representations which are used include MFCCs or the Sone/Bark/Mel/Cent frequency scales.
- *Feature Computation*: The final music similarity model (or signature) are computed from the representation in more or less complex ways. We will describe three different techniques in the next sections.

To compute an estimate of the similarity between two songs, a music similarity computing algorithm also defines a *Similarity function*. To find the music pieces most similar to a given song, in the simplest case a retrieval algorithm linearly scans and compares all music pieces to the query according to the similarity function.

2.4.2 Mandel-Ellis (ME)

Mandel and Ellis [ME05] published their music similarity measure in 2005. It uses very simple features which still seem to capture some notion of general (timbre) music similarity. In the

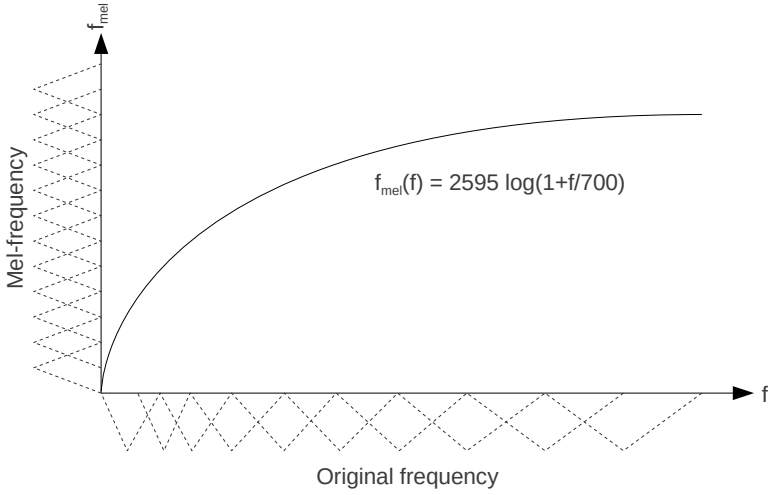


Figure 2.4: Schematic plot of the non-uniformly spaced triangular filterbank for the original spectrum (x-axis) to obtain a Mel-wrapped spectrum. It is also possible to obtain a Mel-wrapped spectrum from interpolation of the power spectrum using an equally spaced filterbank (y-axis).

feature extraction phase, it first *decodes* the music to a mono 22 050 Hz signal. A window size of 1 024 samples with a hop size of 512 samples is used to compute the power spectrum of the signal (c.f. the *Frequency Domain Representation* step). The power spectrum is transformed to 20 MFCCs per frame. To compute the MFCCs the power spectrum is converted to a 36-band Mel spectrum using a non-uniformly, Mel-spaced triangular filterbank in the original spectrum. Figure 2.4 schematically depicts this procedure. Figure 2.3c shows a Mel-spectrum computed from a 10 s music snippet. The filterbank output is cosine transformed using a Discrete Cosine Transform (DCT) to obtain the MFCC representation (Figure 2.3d). A popular reference implementation to compute MFCCs was published by Slaney [Sla93].

To finally compute the song model features from the MFCCs, a single multivariate (20-dimensional) Gaussian $X \backsim \mathcal{N}(\hat{\mu}, \hat{\Sigma})$ is estimated from the MFCC vectors m_i of each music piece.

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} m_i, \qquad \hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(m_i - \hat{\mu})(m_i - \hat{\mu})^T. \tag{2.1}$$

A song is thus modeled by a 20-dimensional mean vector $\hat{\mu}$ and a 20×20 covariance matrix $\hat{\Sigma}$. To compute a similarity value between two Gaussians the symmetrized Kullback-Leibler divergence is used. The Kullback-Leibler divergence (KL) has a closed form for two d-dimensional Gaussians $X_{1,2}$. It is defined in Chapter 3, Section 3.4.1.1. The symmetrized Kullback-Leibler divergence (SKL) which is used by ME to estimate timbre music similarity (d_t) is defined as:

$$d_t(X_1, X_2) = SKL(X_1|X_2) = \frac{1}{2}KL(X_1|X_2) + \frac{1}{2}KL(X_2|X_1). \tag{2.2}$$

Section 3.4.1.2 in Chapter 3 shows a simplified closed form solution.

ME shows a very stable performance in classification experiments and listening tests (c.f. Table 2.2). The feature extraction and similarity estimation can be implemented quite efficiently. Unfortunately the SKL is not a metric distance and the similarity features are non-vectorial, which makes it difficult to use the features and measure in standard methods for clustering or indexing. We evaluate ME in Section 2.5. ME has been improved in many further works, two of which are presented next.

2.4.3 Elias Pampalk (EP)

An algorithm that builds on ME is Pampalk's method [Pam06]. A full Matlab implementation is available on-line in the Music Analysis Toolbox (MA Toolbox) [Pam04].

EP combines the single Gaussian musical timbre features of ME with the fluctuation patterns (and features computed from these) to include rhythm information in the similarity measure. The fluctuation patterns are computed from the Mel spectrum of a song. That way it fits nicely into the *Feature Extraction* process of ME.

Fluctuation Patterns are computed for every 3 seconds of music. This is done by computing an FFT on the corresponding 3s segment in the Mel spectrum to get an amplitude modulation frequency representation. The rows of the resulting matrix, $FP = (fp(i,j))$, correspond to the frequency bands, the columns to the modulation frequencies ranging from 0-10Hz. A value in the matrix describes the strength of a repeating fluctuation in a frequency band at a given periodicity. The modulation frequencies are weighted and filtered to arrive at the final Fluctuation Pattern. The pattern at the median is selected as representative Fluctuation Pattern for the whole music piece. The similarity d_{FP} between two Fluctuation Patterns $FP_{1,2}$ is computed by a simple pointwise squared Euclidean distance:

$$d_{FP}(FP_1, FP_2) = \sum_i \sum_j \left(fp_1(i,j) - fp_2(i,j)\right)^2 . \tag{2.3}$$

For the rhythm similarity measure two additional scalar rhythm descriptors are derived from the fluctuation patterns: "Bass" (b, total periodicities in the two lowest frequency bands) and "Gravity" (g, describes the most dominant modulation frequency by computing the center of gravity of the Fluctuation Pattern matrix columns). The similarity in terms of Bass (d_{FPb}) and Gravity (d_{FPg}) is defined using the absolute difference:

$$d_{FPg}(FP_1, FP_2) = |g_1 - g_2|, \quad d_{FPb}(FP_1 \| FP_2) = |b_1 - b_2|. \tag{2.4}$$

To linearly combine all rhythm similarities, EP normalizes the individual distances using static (precomputed) normalization values to zero mean and unit variance. The ME musical timbre similarity (weighted with 70%) is then linearly combined with the FP rhythm similarity (weighted 30%) using static normalization values which were precomputed on training collections. The linear combination EP used to compute the similarity between two models m_1 and m_2 is

defined as [Pam06]:

$$\begin{aligned} d_{EP}(m_1, m_2) =& 0.7 \exp(-d_t(X_1, X_2)/450) + 0.7950)/0.1493 + \\ & 0.1\,(d_{FP}(FP_1, FP_2) - 1688.4)/878.23 + \\ & 0.1\,(d_{FPg}(FP_1, FP_2) - 1.2745)/1.1245 + \\ & 0.1\,(d_{FPb}(FP_1, FP_2) - 1064.8)/932.79. \end{aligned} \tag{2.5}$$

The static normalization values ("magic numbers") are a weak point of the EP method, as the precomputed numbers may not fit for arbitrary collections. In Chapter 4 we present a much better way to do the linear combination. As EP extends the method ME with a rhythm component it has the same problems like ME when it comes to reusing the features in clustering or indexing methods. EP is evaluated in Section 2.5.

2.4.4 Pohle-Schnitzer (PS)

Similarly to EP, PS combines a timbre and rhythm component into a single general music similarity measure. The method is also closely related to ME and EP as it also uses the single Gaussian modeling to represent its features. PS uses a rhythm component which improves the fluctuation patterns used in EP. PS was top-ranked in the MIREX 2009 and 2010 evaluations (see Table 2.2).

Rhythm

To compute the rhythm features (called *onset coefficients*), PS performs the following steps:

1. From the STFT (using a 15.5 ms window size) a cent-scaled representation of the STFT spectrum is computed (with 85 bands, each band spaced 103.6 cents).
2. To emphasize onsets, the sliding mean over 0.25 ms is subtracted from each frame of the cent-scaled spectrum, which is then resampled from 85 to 35 bands.
3. As in the fluctuation patterns (see Section 2.4.3), an FFT is used on a 2.63 s window (zero padded to 6 s) to detect periodicities. Unlike with FPs, the periodicities are represented on a log-scale. To do that a log-filterbank is applied. The result, a two-dimensional matrix (36×25), is called *onset pattern*.
4. In the next step the so called *onset coefficient* representation is computed by applying a DCT on each onset pattern, discarding higher-order coefficients. From the periodicity dimension eight (0–7), in the frequency dimension three (0–2) coefficients are kept.
5. Finally all onset coefficients of a song are summarized by estimating a single Gaussian representation OC from the features. Thus the rhythm features use the same representation like ME uses for its music timbre model (Section 2.4.2).

To compare two onset coefficient signatures $OC_{1,2}$, PS uses a Jensen-Shannon divergence approximation (JS, see Chapter 3, Section 3.4.1.3), another variant of symmetric Kullback-Leibler divergences:

$$d_r(OC_1, OC_2) = JS(OC_1|OC_2). \tag{2.6}$$

Timbre

The timbre component consists of three sub-components which are concatenated into a large vector. The sub-components are computed for each audio frame of 26 ms length: (i) 16 MFCC coefficients (ii) 16 spectral contrast coefficient values [JLZ+02] and (iii) the frame's harmonic and percussive portions, computed according to Ono et al. [OMKS08].

All timbre vectors which are computed for a music piece are summarized by estimating a single Gaussian timbre model T. As similarity function the Jensen-Shannon approximation (JS, see Chapter 3, Section 3.4.1.3) is used:

$$d_t(T_1, T_2) = JS(T_1|T_2). \tag{2.7}$$

Similarity

To compute overall similarity PS uses, similarly to EP, a linear combination of the rhythm (d_r) and timbre (d_t) components, each component weighted with 50%. To linearly combine the two measures, PS computes the full similarity matrix for each measure (using the Jensen-Shannon approximation as similarity measure for both components) and normalizes each row of the distance matrix using z- (or standard-score) normalization. Since this row-wise normalization yields a non symmetric measure, it is symmetrized again by averaging both distance pairs. For now we denote this normalization technique with $z()$:

$$d_{PS}(m_1, m_2) = 0.5\, z(d_t(T_1, T_2)) + 0.5\, z(d_r(OC_1, OC_2)). \tag{2.8}$$

The normalization technique to linearly combine similarity measures in PS is a preliminary variant of the probabilistic method for distance normalization we will introduce in Chapter 4. We will show that this method has applications with beneficial effects beyond music similarity.

2.5 Evaluation

This section introduces evaluation strategies and public collections which we use to evaluate the performance of the three introduced computer music similarity measures. The collection and methods to evaluate the similarity measures are used throughout the thesis.

As music similarity is a very subjective concept, the best way to evaluate the performance of an algorithm are of course listening tests. In such tests listeners would have to manually rate the similarity between two songs based on their subjective feelings. Such tests are, for example, done in the yearly MIREX AMSR evaluations, which we have already discussed in Section 2.3.2. Unfortunately it is costly and difficult to conduct listening tests. To measure the quality of a music similarity algorithm it is therefore common to conduct automatic genre classification experiments. These experiments only require a collection where each song is labeled with a genre. In Section 2.2 we have already shown that the musical genre is a prevalent attribute to define the similarity of two music pieces. A convincing argument for automatic genre classification experiments was made by Pohle [Poh10] who has shown a very strong correlation between the similarity ratings users assigned in listening tests, and the genres of the songs.

Leave-One-Out Classification

To perform automatic genre classification experiments, commonly a leave one out k-nearest neighbor classification experiment is done. In such an experiment:

1. the k-nearest neighbors for each song in the collection are computed,
2. for each song its class/genre is determined using a majority vote in the k nearest neighbors (if a tie is detected, the first nearest neighbor decides the class),
3. if the determined class matches the actual class of the song, the song was classified correctly,
4. the average number of all correctly classified songs is computed and called k-nearest neighbor classification accuracy.

We denote the k-nearest neighbor classification accuracy with C^k. Consistently high genre classification accuracies on multiple different collections point to a high-quality music similarity measure.

Artist Filter

In addition to computing the raw classification accuracy, Pampalk et al. [PFW05] propose to use an "artist filter" (AF) in all genre classification experiments to prevent songs by the same artist from appearing in both the training and test sets. Pampalk et al. argue that without using an artist filter, algorithms focus on identifying artists instead of general music similarity. We have extensively studied the effects of artist filters on very large music databases [FS10] and follow our recommendation to conduct the genre classification experiments with an artist filter. We use an artist filter for the collections where artist information is available and adopt the leave-one-out classification experiments: before computing the nearest neighbors for each song x (Section 2.5, step 1), we filter the database from songs having the same artist as x.

2.5.1 Collections

To evaluate the computer music similarity algorithms we use eight public different music collections. The collections will be reused and referenced throughout the thesis in further tests and evaluations to improve or point at problems in algorithms.

To make the evaluation reproducible all collections are standard collections which have already been used in the literature and are in most instances freely available for research. The collections are characterized in the following paragraphs by listing their size, structure, genres and citing the relevant publications. If applicable, a location to download the files is given. Based on their genre the eight collections can be separated in two groups:

1. General (commonly used) genres: *ISMIR 2004 Collection*, *GTzan*, *Homburg*, *1517 Artists*
2. Genres strongly related to rhythm: *Ballroom Dances*, *Cretan Dances*, *Latin Music Database*, *Popular Rhythms*.

A good music similarity algorithm should have high classification rates in both groups.

ISMIR 2004 Collection

Training Set

Songs	729
Genres	metal_punk (45/8), pop_rock (101/26), electronic (115/30), classical (320/40), world (122/19), jazz_blues (26/5).

Full Set (Training & Development)

Songs	1458
Genres	metal_punk (90), rock_pop (203), electronic (229), classical (640), world (244), jazz_blues (52).

The ISMIR 2004 collection is one of the most widely used music collections to evaluate music similarity algorithms. The collection was created for the genre classification contest of the ISMIR 2004 conference where it still can be downloaded from the website.[12] It was subsequently used, for example, by Holzapfel et al. [HS09], Pohle et al. [PSS+09] or Seyerlehner et al. [SWP10]. The collection is split in a *training* and a *development* set, as it was originally introduced as a benchmark collection to evaluate automatic genre classification algorithms. It is notable that the genre "classical" comprises 43.9% of the collection.

As it is done in a number of other publications, we will use the ISMIR 2004 collection in two separate configurations: in a training- and a full (training and development) configuration.

GTzan

Songs	1000
Genres	country (100), rock (100), reggae (100), blues (100), disco (100), hiphop (100), jazz (100), pop (100), classical (100), metal (100).

The "GTzan" collection was assembled in 2002 by George Tzanetakis [TC02]. It consists of 1 000 audio tracks (each 30 s length) evenly spread over 10 music genres. According to Tzanetakis, the files are collected from a number of different sources (including CDs, radio and microphone recordings). "GTzan" is available freely from the Marsyas webpage[13]. The dataset was for example used by Lidy et al. [LR05] or Panagakis et al. [PBK08].

Homburg

Songs	1886
Genres	funksoulrnb (47/39), alternative (145/121), rock (504/448), raphiphop (300/210), folkcountry (222/177), blues (120/80), electronic (113/97), jazz (319/214), pop (116/106).

The "Homburg" [HMM+05] collection consists of 1 886 songs. Each music piece is only 10 s, which is quite short. The short length is sometimes problematic for algorithms trying to estimate rhythm features. The music snippets are available for download on the website of the Artificial Intelligence Group of the University of Dortmund[14].

[12] `http://ismir2004.ismir.net/genre_contest/index.htm`, visited August 12th, 2011
[13] `http://marsyas.info/download/data_sets`, visited August 12th, 2011
[14] `http://www-ai.cs.uni-dortmund.de/audio.html`, visited August 12th, 2011

1517 Artists

Songs	3180
Genres	Soundtracks & More (150/72), R&B & Soul (175/112), Classical (125/46), Religious (172/71), Country (187/102), Easy Listening & Vocals (175/98), Folk (185/98), HipHop (155/87), Comedy & Spoken Word (134/68), New Age (175/82), Rock & Pop (181/117), Reggae (172/83), Children's (164/74), Blues (186/100), Jazz (177/103), Alternative & Punk (182/116), Latin (163/85), World (158/76), Electronic & Dance (164/92),

The "1517 Artists" collection was introduced by Seyerlehner et al. [SWK08, SWP10]. It consists of 3 180 freely available songs from 1 517 artists. Each song is assigned to one of the 19 genres. It is notable that there is a non-music genre included: Comedy & Spoken Word. The dataset is available freely from Seyerlehner's website[15].

Ballroom Dances

Songs	698
Genres	Quickstep (82), Rumba (98), Samba (86), VienneseWaltz (65), Waltz (110), Jive (60), ChaCha (111), Tango (86).

The "Ballroom Dances" collection was introduced by Gouyon [GDPW04]. It is a collection with music pieces assigned to classical ballroom dance genres. Each music piece has a duration of 30 s. The collection was subsequently used for the ISMIR 2004 rhythm classification contest, where it can be downloaded[16].

Cretan Dances

Songs	180
Genres	sous (30), mal (30), pent (30), syrt (30), kal (30), kont (30).

This collection consists of six different traditional dances commonly encountered on the island of Crete (Greece). The genres are predominantly differentiated by their main rhythmic elements. The collection was created by Holzapfel et al. [HS09].

Latin Music Database

Songs	3637
Genres	Pagode (307/17), Forro (313/33), Bachata (313/72), Tango (816/20), Merengue (315/103), Bolero (315/102), Salsa (311/64), Gaucha (312/104), Sertaneja (322/10), Axe (313/40).

The Latin Music (LMD) database was created by Silla et al. [SJKCK08]. 3 637 songs are classified into ten Latin dance genres. According to Silla et al., the songs were carefully selected by professionals. More information about the LMD can be found on the website of the authors[17].

[15] `http://www.seyerlehner.info/`, visited August 12th, 2011
[16] `http://mtg.upf.edu/ismir2004/contest/rhythmContest/`, visited August 12th, 2011
[17] `http://www.ppgia.pucpr.br/~silla/lmd/`, visited August 12th, 2011

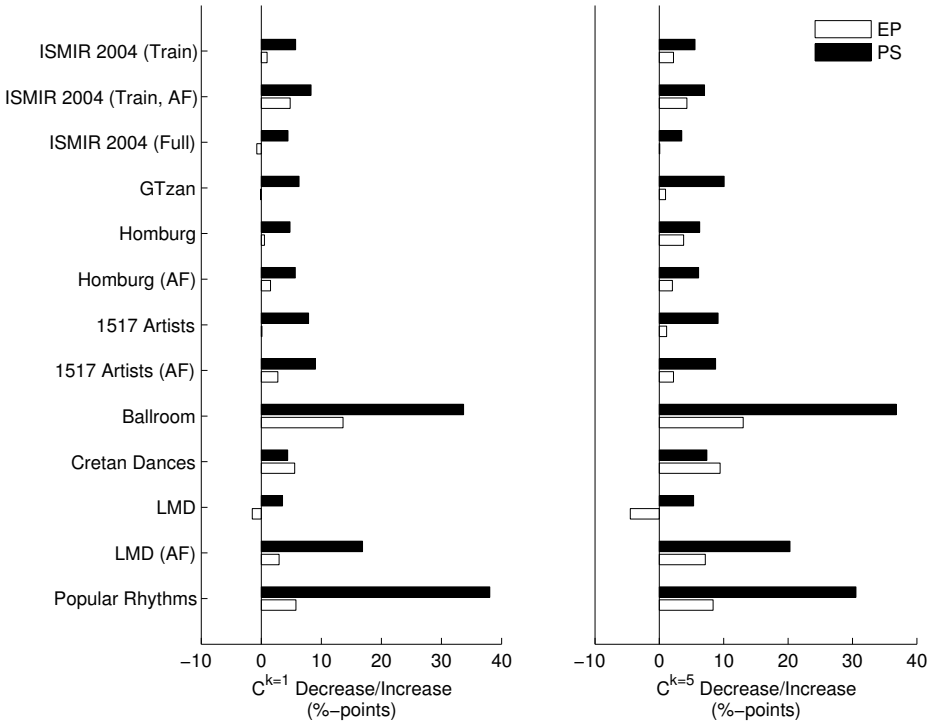


Figure 2.5: Increase/Decrease of the $C^{k=1,5}$ classification accuracies of EP and PS compared to the standard music similarity algorithm ME.

Popular Rhythms

Songs	347
Genres	Jive (30), Jig (16), Minuet (20), Quickstep (26), DrumnBass (14), ChaChaCha (18), BreakBeat (29), Tango (25), ViennesseWaltz (25), Merengue (16), Rumba (15/, Slowfox (25), Salsa (30), Samba (30), DiscoFox (28).

The "Popular Rhythms" database was created by Holzapfel [AAG11] who classified 347 songs into 14 genres.

2.5.2 Genre Classification

We conduct an initial genre classification experiment on all collections we have introduced in the previous section. Where applicable we use an artist filter (denoted by AF). Table 2.3 shows the individual classification accuracies of each music similarity algorithm by collection. We computed

the classification accuracies for $k = 1, 5, 10, 20$. From the table we can see that PS is performing excellently on all collections and is, in terms of C^k, notably better than the basic music similarity algorithm ME. Another observation is that the classification accuracy of ME drops sharply if an artist filter is used. This is an indication that in many cases ME is merely doing well in identifying artists instead of computing music similarity. EP and PS consistently perform better than ME even if an artist filter is used, although their performance drops sharply too.

To compare PS with EP we now look at the increase of each algorithm's classification accuracy relative to ME (see Figure 2.5). The first observation is that using a rhythm similarity component in PS leads to a significant increase of classification accuracy in rhythmic collections compared to ME (i.e., *Ballroom* (+33.7%), *Cretan Dances* (+4.4%), *LMD* (+3.1%, +16.9%) and *Popular Rhythms* (+38.0%)) while at the same time all other classification rates increase too. Apparently the rhythm component also helps to increase the algorithms performance on collections where the genres are defined by timbre. PS also outperforms EP (with the exception of the *Cretan Dances*) especially on rhythmic genre collections, which makes PS a top-choice as a music similarity measure.

2.6 Problems Using the Methods on a Large Scale

We have introduced and evaluated three content-based computer music similarity algorithms. However three major shortcomings common, to all these algorithms, prevent them from being used widely and on large scale collections. The following section identifies and discusses these issues.

2.6.1 Non-Vectorial Representation, Non-Metric Music Similarity

A predominant problem of the algorithms are, simply put, the very uncommon music similarity models. In all introduced music similarity algorithms parametric multivariate Gaussians $\mathcal{N} \sim (\Sigma, \mu)$ are used to represent a song: ME, EP and PS use the multivariate Gaussian for the timbre features. PS also uses a Gaussian for the rhythmic features. The single Gaussian representation is rather uncommon in other research fields, and usually some kind of vector representation is used. An example where probability distributions are used as features is image retrieval where color histograms are commonly used (c.f. Pérez et al. [PHVG02]). However, as opposed to the continuous distributions which are used in the case of the music similarity algorithms, these distributions are in discrete form.

As a consequence of the Gaussian representation non-metric Kullback-Leibler divergences have to be used to estimate the similarity between two Gaussians, as opposed to a standard vector norm (e.g. ℓ^2).

While a linear arrangement by similarity can easily be computed with one of the aforementioned similarity algorithms (ME, EP, PS), more efficient algorithms for indexing, clustering or visualization are just not designed to work with Gaussian distributions and non-standard metrics like the Kullback-Leibler divergence induces. For instance, how would a centroid be computed amongst a number of Gaussians?

To work around that limitation the feature data is often artificially vectorized. In the domain of content-based music similarity techniques we have seen approaches computing the full distance

			C^k			
Collection	Algorithm	Artist Filter	1	5	10	20
ISMIR 2004 (Train)	ME	no	80.4%	67.9%	60.6%	57.1%
	EP	no	81.3%	70.1%	64.5%	60.2%
	PS	no	**86.1%**	**73.5%**	**71.1%**	**63.3%**
	ME	yes	64.2%	61.9%	60.2%	57.8%
	EP	yes	69.0%	66.2%	63.2%	60.8%
	PS	yes	**72.5%**	**69.0%**	**66.2%**	**63.5%**
ISMIR 2004 (Full)	ME	no	85.0%	75.6%	68.9%	62.3%
	EP	no	84.2%	75.7%	71.6%	64.6%
	PS	no	**89.4%**	**79.1%**	**76.5%**	**71.8%**
GTzan	ME	no	73.8%	54.1%	48.4%	35.3%
	EP	no	73.7%	55.1%	48.6%	39.6%
	PS	no	**80.1%**	**64.2%**	**54.1%**	**46.1%**
Homburg	ME	no	43.8%	39.8%	36.3%	36.6%
	EP	no	44.4%	43.6%	39.4%	38.7%
	PS	no	**48.6%**	**46.1%**	**44.3%**	**41.9%**
	ME	yes	41.8%	40.7%	39.3%	38.1%
	EP	yes	43.3%	42.7%	41.6%	40.2%
	PS	yes	**47.5%**	**46.8%**	**45.5%**	**44.6%**
1517 Artists	ME	no	41.9%	24.4%	19.8%	15.6%
	EP	no	42.0%	25.5%	21.9%	17.6%
	PS	no	**49.8%**	**33.5%**	**27.8%**	**22.7%**
	ME	yes	22.1%	19.7%	18.5%	17.0%
	EP	yes	24.9%	21.9%	20.7%	19.1%
	PS	yes	**31.2%**	**28.4%**	**26.7%**	**24.9%**
Ballroom Dataset	ME	no	54.3%	42.1%	36.4%	31.5%
	EP	no	67.9%	55.2%	49.6%	41.3%
	PS	no	**88.0%**	**78.9%**	**73.2%**	**58.3%**
Cretan Dances	ME	no	26.1%	18.9%	20.6%	22.2%
	EP	no	**31.7%**	**28.3%**	16.1%	17.2%
	PS	no	30.5%	26.3%	**28.0%**	**22.9%**
Latin Music Database	ME	no	92.4%	84.5%	76.6%	65.4%
	EP	no	90.9%	80.0%	73.1%	66.3%
	PS	no	**95.9%**	**89.8%**	**84.4%**	**77.8%**
	ME	yes	65.2%	57.4%	56.7%	53.9%
	EP	yes	68.2%	64.5%	61.9%	58.3%
	PS	yes	**82.1%**	**77.7%**	**75.4%**	**71.6%**
Popular Rhythms	ME	no	35.2%	19.3%	18.7%	9.2%
	EP	no	40.9%	27.7%	20.7%	18.4%
	PS	no	**73.2%**	**49.9%**	**38.0%**	**23.3%**

Table 2.3: Results of a genre classification experiment on all eight datasets. The classification accuracies $C^{k=1,5,10,20}$ for three music similarity algorithms (ME, EP, PS, where applicable with Artist Filter) are reported. In all but one collection PS returns consistently the best classification accuracies.

matrix and using each row of the matrix as a feature vector [KSPW07, PRM02a], or more venturesome ones reshaping the Gaussian covariance matrix and mean vector into a single long vector [ME07]. The first solution becomes very expensive to compute as the collection grows, and the latter one, although fast, takes away the sense of using Gaussians.

Having a good music similarity measure, but not being able to even use it for native clustering, is a big problem for a real applied large scale music recommendation system. In such an application clustering is essential to build a fast search index or to create a visualization. Computing a full similarity matrix for one of these tasks is impossible for collections with millions of tracks.

No method except for our own work [SFWG10] has been published so far that deals with the specific problem of clustering with these music similarity measures. We introduce these methods in greater detail in Chapter 3 and show how to correctly handle multivariate Gaussians and their attached divergences in clustering algorithms. We also present a modified Self Organizing Map (SOM) visualization/clustering algorithm that works directly and naturally with Gaussian features.

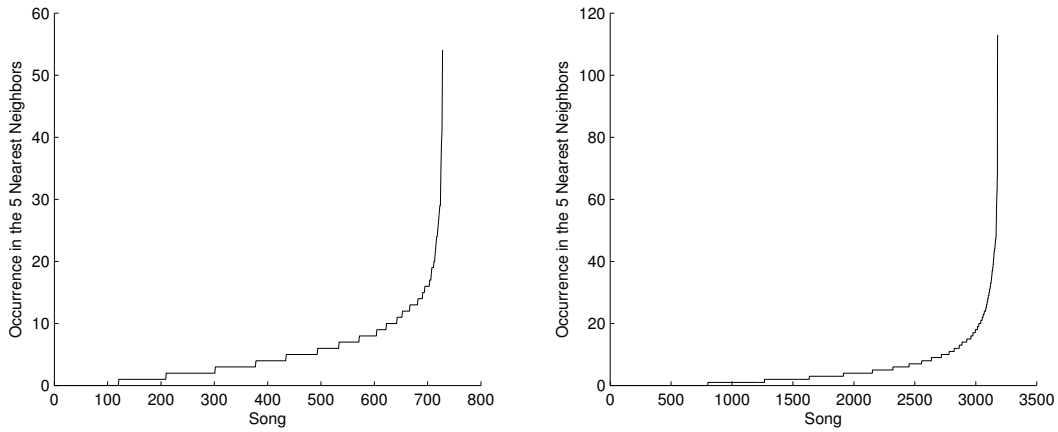
2.6.2 The Hub Problem

The introduced music similarity algorithms are affected by a problem which negatively influences the retrieval quality of the algorithms. It turns out that that specific songs (so-called "hubs") appear unwontedly often as nearest neighbors for a large portion of songs in the collection. This in turn causes the effect that some songs are never recommended at all (these are called "orphans"). In MIR literature the problem is referred to as the "hub problem" [PA04, GFS10]. This is especially problematic in similarity based recommendation or browsing systems since it would be highly desirable that all songs are equally discoverable (or discoverable at all).

To illustrate the impact of the "hub problem" on our collections, we show how often each song occurs in other top five nearest neighbors lists for two selected collections in Figure 2.6. We use the *ISMIR 2004 (Train)* and *1517 Artists* collections with the default ME music similarity algorithm. From the plots it can be seen that using ME leads to the effect that over 100 songs (ISMIR 2004 (Train)) or 800 songs (1 517 Artists) are never found in any nearest neighbor lists, thus never recommended – orphaned. On the other hand a few hub songs appear in almost 10% (ISMIR 2004 (Train)) or 5% (1517 Artists) of all nearest neighbor lists. Similar adverse effects can be observed for all music collections.

We have examined the effect of hubs and orphans in a real music recommendation service on the web in a previous work [FGS10]. The web music recommendation service we created is called FM4 Soundpark[18]. The FM4 Soundpark hosts around 11 000 free music tracks from independent bands and artists. To help exploring the music database, an automatic music recommendation system (based on the ME similarity algorithm) was developed by our research team. Whenever a user listens to a song from the database, they are presented with lists of most similar songs. These lists can be explored interactively by going from song to song and expanding the similarity lists. In theory, any song from the data base should be discoverable from the similarity lists, however the "hub problem" unfortunately prevents this from happening. We show that when using ME, only 65% percent of all songs can be reached using an arbitrary start song in the recommendation

[18] `http://fm4.orf.at/soundpark`, visited August 12th, 2011



(a) ISMIR 2004 (Train)

(b) 1517 Artists

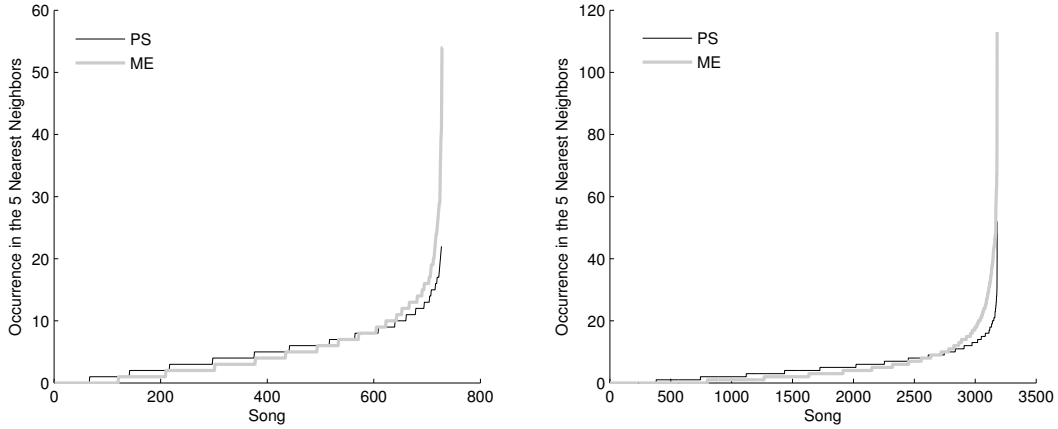
Figure 2.6: Limitations of music similarity algorithms using the example of ME on two collections: The figure plots the number of occurrences a song has in all $k = 5$ nearest neighbor lists of a collection. A large number of songs never occur as nearest neighbor ("orphans"), while at the same time a few songs are appearing in a large number of nearest neighbor lists ("hubs").

interface, while there exist a few hub songs which appear in almost 5% of all recommendations. This is a grave limitation of the recommendation service and the audio similarity algorithm.

In another paper [FSGP10a] we show that a linear combination of multiple similarity measure can alleviate the hub problem. More precisely we examine the EP method (a linear combination of ME with the Fluctuation Patterns see Section 2.4.3) and show that the number of hubs in the Soundpark music collection halves.

The hub problem has been examined for PS by Pohle [Poh10], who observes a sharp decline in orphans and hubs when using PS. Figure 2.7 shows the positive effects in terms of hubs and orphans for the two previously examined collections, where in both cases hubs and orphans are reduced in PS. Chapter 4 will show that this positive effect comes from the way PS normalizes the distances. As already mentioned in Section 2.4.4, PS uses a preliminary variant of the method which we present in Chapter 4. It was subsequently also used by Seyerlehner et al. [SWP10] (termed distance space normalization) to linearly combine multiple similarity measures.

Chapter 4 introduces a general method, which we call Mutual Proximity, to alleviate the problem of hubs. The approach is fully unsupervised, scalable and transforms an arbitrary distance or similarity to a new probabilistic similarity. The chapter investigates the method and its effect on hubs. Using it like Pohle [Poh10] or Seyerlehner et al. [SWP10] only for linear combination of multiple measures is only a side aspect of the method. We evaluate Mutual Proximity using 30 general public machine learning datasets. In these experiments we can show that using the method leads to a significant decrease of hubs and at the same time to an increase of k-nearest neighbor classification accuracy with high dimensional data spaces.



(a) ISMIR 2004 (Train)

(b) 1517 Artists

Figure 2.7: Comparing the distribution of $k = 5$-occurrences of ME and PS. We can see that PS reduces the number of orphans (songs which never occur as nearest neighbor) and hubs significantly.

Besides presenting a general method to alleviate the hubs problem, we also show that the quality of all music similarity algorithms (ME, EP in addition to PS) benefits from using the proposed method. Mutual Proximity can be easily approximated so that it can be used with large scale collections.

2.6.3 Missing Search Algorithms

A major problem which prohibits the introduced music similarity algorithms from being used for very large collections, is the lack of indexing algorithms that are capable to build an efficient search index for the uncommon multivariate Gaussian features together with their associated Kullback-Leibler divergences. The non-vectorial features and the uncommon divergences prevent us from using one of the many standard search algorithms such as *kd*-trees or locality sensitive hashing.

In addition to these basic problems, the high dimensionality of the features to be indexed poses another obstacle to a search algorithm. In very high dimensional spaces classic search structures like space partitioning algorithms tend to lose their efficiency. This effect has been termed the curse of dimensionality [Bel61a, MP88]. As a result of that, even with appropriate partitioning/clustering methods, we can not achieve high retrieval accuracies in sub-linear time.

Chapter 5 proposes a method for fast nearest neighbor retrieval in large databases which works well for the class of introduced music similarity algorithms. In its core the proposed method first rescales the Kullback-Leibler divergence to make it more "metric". It then uses a modified FastMap implementation to compute a vector-space approximation of the Gaussian

features. We use this vector approximation to speed up nearest-neighbor search using a filter-and-refine system layout. In our method first a coarse scan quickly pre-filters all objects in the database for good candidate nearest neighbors in the vector space. In a second step the candidates are refined using the original similarity method.

Overall the method that way accelerates the search for similar music pieces by a factor of 10–40. Although the proposed method is an approximate search algorithm it still yields high recall values of 95–99% compared to a standard linear search.

2.7 Summary

This chapter has given an introduction to computational models of music similarity and presented three different content-based computer music similarity algorithms: ME, EP, PS. PS is currently one of the top performing music similarity algorithms, which we confirm in our genre classification experiments comparing the performance of all three algorithms. We introduced eight different public music collections which will be used throughout the thesis to evaluate our algorithms.

We identified three major problems which need to be tackled before using any of the music similarity methods in an effective and large scale music recommendation application: (i) handling Gaussian features and their associated Kullback-Leibler divergences, (ii) finding an efficient solution to the hub problem, and (iii) building an indexing solution for the similarity algorithms to allow rapid answers to recommendation queries on large collections.

These three problems will be investigated in the next chapters of the thesis to finally build at a truly scalable, high-quality music recommendation system.

Chapter 3

Working with Gaussian Music Similarity Models

Contents

3.1 Introduction 34
3.2 The Exponential Class of Distributions 35
3.2.1 The Multivariate Normal Distribution 36
3.3 Bregman Divergences 37
3.3.1 Linking the Exponential Family 38
3.3.2 Bregman Centroids 39
3.3.3 K-means Clustering 40
3.4 Bregman Divergences and Music Similarity Measures 40
3.4.1 Kullback-Leibler Divergences for Multivariate Gaussians 40
3.4.2 Kullback-Leibler Centroids for the Multivariate Gaussian 43
3.5 Self-Organizing Maps with Gaussian Music Similarity Models . . 44
3.5.1 The SOM Algorithm 45
3.5.2 The Generalized SOM 46
3.5.3 Evaluation 48
3.6 Multivariate Normal (MVN) Matlab Toolbox 55
3.6.1 Initialization 55
3.6.2 Divergences 55
3.6.3 Centroids 56
3.6.4 Clustering 56
3.6.5 Additional Functions 57
3.6.6 Usage Examples 57
3.7 Summary 61

This chapter defines the multivariate Gaussian as a part of the exponential class of distributions (Section 3.2), introduces Bregman divergences, their link to the exponential family of distributions and shows how all Bregman divergences can be natively used in a generalized k-means clustering algorithm (Section 3.3). These basic definitions enable us to use Gaussian features and their Bregman divergences in centroid-computing algorithms. In Section 3.4 we refine these general definitions for the multivariate Gaussian as they are used in the music similarity algorithms (ME, EP, PS). Sections 3.2 – 3.4 review existing literature, to, for the first time, show the link between the recent research results in the filed of Bregman divergences and recent music similarity measures.

In Section 3.5 we build on that and propose a generalized self-organizing map (SOM) algorithm. We show how the SOM algorithm can be changed to naturally work with Gaussians as features. SOMs are popular to compute two- and three-dimensional visualizations of music collections. In previous visualizations the Gaussian features have always been artificially vectorized before they could be used in the SOM algorithm. We show that using the SOM algorithm directly with the Gaussians and their divergences leads to higher-quality visualizations of music archives. We also retain the nice scalability characteristics of the general SOM algorithm. We published the method in 2010 [SFWG10]. The section extends the published results.

Finally Section 3.6 introduces the MVN (Multivariate Normals) GNU Octave/Matlab toolbox which includes all methods which have been presented here to simplify further research working with multivariate Gaussians. The toolbox is available freely on the Internet [1].

3.1 Introduction

One of the basic foundations of content-based music recommendation systems is the ability to compute music similarity with a similarity function. We have presented three algorithms (ME, EP, PS) to compute music similarity in Chapter 2. All three variants use multivariate Gaussians as their song models and a Kullback-Leibler divergence as similarity function.

We see the single Gaussian representation of the music features as a very powerful way to describe the variability of the measured features (e.g. timbre) in a song. By using a Gaussian representation the song's characteristics can be stored efficiently, while at the same time the features seem to be smoothed enough to allow modeling of general music similarity. With the Kullback-Leibler divergences there also exist well founded ways to compute a distance/similarity value between the models. For simple similarity computation the algorithms are widely used.

But things get interesting when we leave the path of simple feature extraction and similarity computation. Standard algorithms for indexing, clustering or visualization are usually just not designed to work with Gaussian distributions and non-standard metrics like the one the Kullback-Leibler divergence defines.

To work around that limitation and still use the well performing features in clustering algorithms, works dealing with this problem usually artificially vectorize the features. In the domain of content based music recommendation, we have seen approaches computing the full distance

[1] `http://www.ofai.at/~dominik.schnitzer/mnv`, visited August 12th, 2011

matrix and using each row of the matrix as a feature vector [Pam06, KSPW07], or more venturesome ones that reshape the Gaussian covariance matrix and mean vector into a single long vector [ME07]. The first solution requires the full similarity matrix to be computed and gets more expensive the larger the collection grows, and the latter one, although fast, takes away the sense of using Gaussians. None of these uses is ideal.

In [MUNS05], where an algorithm for visualizing and clustering music collections using an Emergent SOM is described, the authors even argue that they cannot use Gaussian music similarity features as "they can not be used with datamining algorithms requiring the calculation of a centroid".

Solving that problem is the focus of this chapter. The basis for our research was published by Banerjee et al. [BMDG05] where it was shown how the k-means clustering algorithm can be generalized to the broad class of Bregman divergences. This generalization practically opened all centroid-based algorithms to the wide range of Bregman divergences, which the Kullback-Leibler divergence is part of.

The results of Banerjee's work have since then already been used widely in different areas, like data mining, indexing, image retrieval or general machine learning. For example, Garcia et al. [GNN10] use the findings about Bregman divergences to design an image segmentation algorithm; Cayton [Cay08] uses k-means clustering with Bregman divergences to build one of the first indexing structures for these divergences. Cayton's indexing structure is called "Bregman-ball tree" and is shown to achieve sub-linear query times and high retrieval accuracies for low dimensional data.

The following Sections (3.2–4) summarize published definitions and methods in literature, to show how to correctly use Bregman divergences in music similarity measures, before arriving at Section 3.5 and 3.6, the main contributions of this chapter: a generalized SOM algorithm for Gaussians, and an open Octave/Maltab toolbox implementing all methods presented in this chapter.

3.2 The Exponential Class of Distributions

This broad family of probability distributions includes distributions like the gamma, exponential, multinomial or the normal distribution. In fact all distributions which can be reduced to the canonical form of the exponential class [NBN07] are included in that family. The canonical form is defined by:

- $F(\theta)$, the log normalizer function, a strictly convex and differentiable function that uniquely specifies the exponential family,

- θ, the natural parameters associated with the sufficient statistics $t(x)$. The original (source) parameters of a distribution are denoted with λ. The natural parameters, λ, can be computed from the source parameters, θ, and vice-versa.

- $C(x)$, a Lebesgue/carrier measure.

So that the canonical exponential form $p_F(x;\theta)$ of a probability distribution $p(x;\lambda)$ is defined as ($< \cdot, \cdot >$ in the equation denotes the inner dot-product of two vectors):

$$p(x;\lambda) = p_F(x;\theta) = \exp\left(< \theta, t(x) > -F(\theta) + C(x)\right). \tag{3.1}$$

For example we can look at the univariate normal distribution $\mathcal{N}(\mu, \sigma^2)$. The source parameters of $\mathcal{N}$ are $\lambda = (\mu, \sigma^2)$. Its probability density function (pdf) is defined as:

$$p(x;\mu,\sigma^2) = (2\pi)^{-\frac{1}{2}}(\sigma^2)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^2(\sigma^2)^{-1}\right) \tag{3.2}$$

It can be rewritten to the canonical exponential form and thus is a member of the exponential family:

- The log normalizer $F(\theta) = -\frac{\theta_1^2}{4\theta_2} + \frac{1}{2}\log(\frac{-\pi}{\theta_2})$
- Its natural parameters $\theta = (\theta_1 = \frac{\mu}{\sigma^2}, \theta_2 = \frac{1}{2\sigma^2})$
- The sufficient statistic $t(x) = (x, x^2)$
- The carrier/Lebesque measure $C(x) = 0$

Nielsen and Nock [NN09] have more details and examples of different exponential family decompositions.

3.2.1 The Multivariate Normal Distribution

The multivariate normal distribution (MVN) as it is used in all content-based music similarity algorithms we employ, is a natural member of the exponential family of distributions. We define its canonical decomposition and two important aspects (entropy and the estimation from data).

Canonical Decomposition

A multivariate normal (or Gaussian) distribution is defined by two source parameters, its mean vector $\mu \in \mathbb{R}^d$ and its covariance matrix Σ ($d \times d$, symmetric positive semi-definite variance-covariance matrix). These two define its probability density function (pdf):

$$p(x;\mu,\Sigma) = (2\pi)^{-\frac{d}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)\right). \tag{3.3}$$

We denote a multivariate Gaussian with $X \sim \mathcal{N}(\mu, \Sigma)$. According to Nielsen et al. [NBN07] its canonical exponential class decomposition is given by:

- The log normalizer $F(\theta) = \frac{1}{4} tr\left(\theta_1^{-1}\theta_2\theta_2^T\right) - \frac{1}{2}\log|\theta_1| + \frac{d}{2}\log\pi$,
- Its natural parameters $\theta = (\theta_1 = \Sigma^- 1\mu, \theta_2 = \frac{1}{2}\Sigma^{-1})$,
- The sufficient statistic $t(x) = (x, -x^T x)$,
- The carrier/Lebesque measure $C(x) = 0$.

Entropy

The entropy of a multivariate Gaussian distribution X is denoted with $H(X)$. It is defined as:

$$H(X) = \frac{1}{2} \ln \left((2\pi e)^d \left| \Sigma \right| \right) \tag{3.4}$$

Estimation from Data

To estimate the parameters of a Gaussian $X \sim \mathcal{N}(\hat{\mu}, \hat{\Sigma})$ from a number of data vectors or observations $\mathbf{x}_{i=1..n}$, their maximum likelihood estimator is used:

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i, \qquad \hat{\Sigma} = \left(\frac{1}{2n} \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T \right) - \hat{\mu}\hat{\mu}^T \tag{3.5}$$

This parameter estimation method was already used in all computer music similarity methods introduced in Chapter 2. All methods (ME, EP, PS) fitted a single multivariate Gaussian to d-dimensional spectrum vectors to build a music similarity model.

3.3 Bregman Divergences

Bregman divergences [Bre67] are defined for every strictly convex and differentiable function $F(x)$:

$$B_F(p, q) = F(p) - F(q) - \langle p - q, \nabla F(q) \rangle \,, \tag{3.6}$$

where $\nabla F(q)$ denotes the gradient of F at q and $\langle \cdot, \cdot \rangle$ the inner dot-product of two vectors (i.e., $\langle x, y \rangle = x^T y$).

A simple widespread example for a Bregman divergence is the squared Euclidean distance [BMDG05]. Its generating log normalizer is $F(x) = \langle x, x \rangle$ and the gradient at any point x is $\nabla F(x) = 2x$. Using the generating function $F(x)$ in the definition of a Bregman divergence we can easily derive the original definition of the squared Euclidean distance from that:

$$\begin{aligned} B_F(p, q) &= \langle p, p \rangle - \langle q, q \rangle - \langle p - q, 2q \rangle & (3.7) \\ &= \langle p, p \rangle - \langle q, q \rangle - 2 \langle p, q \rangle + 2 \langle q, q \rangle & (3.8) \\ &= \langle p - q, p - q \rangle = \| p - q \|^2. \end{aligned}$$

Figure 3.1 depicts the squared Euclidean distance between two points p, q as a Bregman divergence $B_F(p, q)$ in $\mathbb{R}^1$.

There exists a large number of divergences which belong to the family of Bregman divergences. For example the Mahalanobis distance, the Ikatura-Saito distance or the Kullback-Leibler divergence [BMDG05].

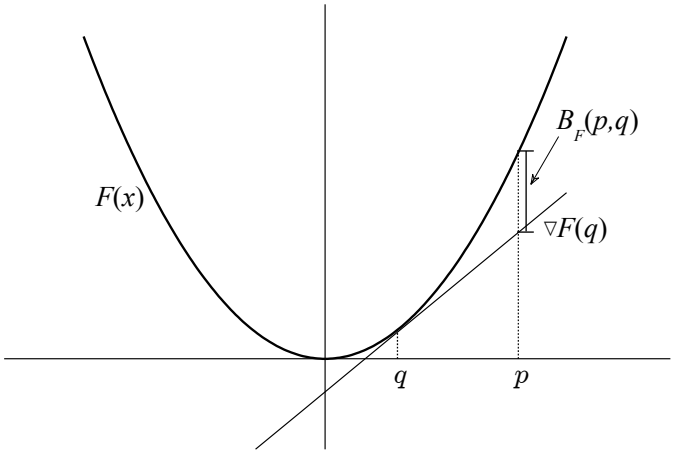


Figure 3.1: Schematic plot of the squared Euclidean distance as a Bregman divergence B_F. $\nabla F(x)$ is the gradient tangent-line at point q. The Bregman divergence $B_F(p,q)$ is the distance between the tangent and $F(x)$ at p.

Properties

Bregman divergences have the following properties:

- Non-Negativity: $B_F(p,q) \geq 0,\ \forall p,q$
- Asymmetry: $B_F(p,q) \neq B_F(q,p),\ \forall p \neq q$
- Convexity: in the first argument p of the function $B_F(p,q)$
- Duality: $B_F(p,q) = B_{F^*}(\nabla F(q), \nabla F(p))$. B_{F^*} is the Bregman divergence defined with the convex conjugate function of $F(x)$:

$$F^*(\nabla F(x)) = \langle x, \nabla F(x) \rangle - F(x) \tag{3.9}$$

$$\nabla F^* = (\nabla F)^{-1} \tag{3.10}$$

Bregman divergences do not necessarily fulfill the triangle inequality, thus are per definition non metric divergences, i.e. they are no distances.

3.3.1 Linking the Exponential Family

Banjeree et al. [BMDG05] showed an important link between Bregman divergences and the exponential family of distributions. In their *equivalence theorem* they prove that the Kullback-Leibler divergence between two distributions from the exponential family is the same as the Bregman divergence of their log normalizer generator function ($F(x)$, see Section 3.2) but with *swapped* parameters:

$$KL(p(x;\lambda_1)|p(x;\lambda_2)) = B_F(\theta_2,\theta_1) \tag{3.11}$$

The parameters θ denote the natural parameters and λ the source distribution parameters (Section 3.2) of the distributions. The importance of the theorem formulated by Banjeree et al. is clear, as it practically links all methods developed for Bregman divergences – via the Kullback-Leibler divergence – to the broad family of exponential-class of distributions.

3.3.2 Bregman Centroids

Every Bregman divergence (see again Banjeree et al. [BMDG05]) defines a unique right-type centroid c_R of a point-set $\mathcal{P} = \{p_1, ..., p_n\}$. The right-type centroid is the minimizer of the centroids right divergences to all points $p_i \in \mathcal{P}$: $\arg\min \sum_{p_i \in \mathcal{P}} B_F(p_i, c_R)$:

$$c_R = \frac{1}{n} \sum_{i=1}^{n} p_i \tag{3.12}$$

The Bregman right-type centroid is always the center of mass and thus independent of the Bregman divergence. Bregman divergences also define a unique left-type centroid, c_L. It is defined using the right-type centroid of its dual Bregman divergence B_{F^*} (see Section 3.3).

The right type-centroid c'_R of B_{F^*} is again simply the center of mass of the gradient point space [NBN07]. To compute c_L the right type centroid c'_R has to be inverted with the inverse gradient function $\nabla F^{-1}(x)$:

$$c_L = (\nabla F)^{-1}(c'_R), \tag{3.13}$$

$$c'_R = \frac{1}{n} \sum_{i=1}^{n} p'_i, \qquad p'_i = \nabla F(p_i) \tag{3.14}$$

3.3.2.1 Centroids for Symmetrized Bregman Divergences

Symmetrized Bregman divergences are defined as $B_{F_{sym}}(x, y) = \frac{1}{2} B_F(x, y) + \frac{1}{2} B_F(y, x)$. A popular symmetrized Bregman divergence is the symmetrized Kullback-Leibler divergence (see Section 3.4.1.2).

There exists no closed form solution to the centroid c of symmetrized Bregman divergences:

$$c = \arg\min_{c} \sum_{p_i \in \mathcal{P}} \frac{1}{2} \left(B_F(p_i, c) + B_F(c, p_i) \right), \tag{3.15}$$

Although c can not be computed directly form both the right (c_R) and left-type centroid (c_L), c is still geometrically uniquely defined. Nielsen and Nock [NBN07] present a bisection search algorithm to find the centroid c:

1. Initialization: set $\lambda_0 = 0$ and $\lambda_1 = 1$.
2. Mid-Point: Compute the mid-point $\lambda_c = \frac{\lambda_0 + \lambda_1}{2}$ and the corresponding point c on the geodesic linking:

$$c = (\nabla F)^{-}1((1 - \lambda_c)\nabla F(c_R) + \lambda_c \nabla F(c_L)). \tag{3.16}$$

3. Branching: If $B_F(c_R, c) < B_F(c, c_L)$ set $\lambda_0 = \lambda_c$, else set $\lambda_1 = \lambda_c$.

4. Repeat step 2 and 3, until a threshold (ϵ) is reached, i.e., $|\lambda_0 - \lambda_1| < \epsilon$.

Essentially the bisection search algorithm iteratively computes a weighted mix between the left and right-type centroid to find c (Equation 3.15).

3.3.3 K-means Clustering

With the definitions of the centroids Banerjee et al. [BMDG05] showed that it is possible to cluster a point-set $\mathcal{P} = \{p_1, ..., p_n\}$ using the standard k-means algorithm with the new centroids, either using the left, right or symmetrized centroid. The generalized k-means algorithm has four steps:

1. Random initialization (once): randomly assign one of the k centers c_j a point p_i.

2. Assign each p_i to their closest center c_j.

3. Recompute the centers c_j according to the points p_i which have been assigned to them.

4. Repeat step 2 and 3 until a convergence criterion is reached.

The only difference to the classic k-means algorithm is that in step 3, a Bregman centroid is computed. Step 2, of course, uses the same Bregman divergence to assign the points to their centers which is used to compute the centroids. To use different initialization methods for k-means (step 1) selected initialization method needs to be compatible to work with Gaussians and their divergences.

3.4 Bregman Divergences and Music Similarity Measures

With these definitions all is in place to return to the computer music similarity methods and their multivariate Gaussian similarity features. The next sections define the divergences and, based on that, the centroids of the Gaussian features as they are used in the ME, EP, PS music similarity algorithms. These definitions enable us to natively use the features in centroid-computing algorithms like the k-means algorithm (Section 3.3.3).

3.4.1 Kullback-Leibler Divergences for Multivariate Gaussians

We have shown the direct connection of Bregman divergences to the exponential family of distributions and the Kullback-Leibler divergence. In Chapter 2 we have introduced three music similarity algorithms which all use Kullback-Leibler divergences as they are defined for multivariate Gaussians. All divergences which are presented here are implemented in a freely available Octave/Matlab toolbox, to be presented in Section 3.6.

This section defines all divergences for the multivariate Gaussian, as they have been used in the music similarity algorithms (Chapter 2) and sets them into a common context. All these divergences are closely related to the Kullback-Leibler divergence. To use any of the divergences in a music similarity algorithm, the divergences need to be used in their closed form for multivariate parametric Gaussians. For example, the music similarity algorithms ME and EP

both use a symmetrized version of the Kullback-Leibler divergence to compute similarity, whereas PS uses a Jensen-Shannon-like divergence. The divergences in the different music similarity algorithms are interchangeable, as the song models are all of the same form, i.e. multivariate Gaussians.[2]

The Kullback-Leibler divergence is known under a number of synonyms like *relative entropy* or *information gain* and was introduced by Kullback and Leibler [KL51]. The divergence computes the difference between two distributions p and q:

$$KL(P|Q) = \int p(x) \log \frac{p(x)}{q(x)} \, \mathrm{d}x. \tag{3.17}$$

It is non-symmetric and non-negative. Although the divergence is sometimes misleadingly named a distance, it is none – the triangle inequality does not hold. From the definition of the Kullback-Leibler divergence its close relation to the Shannon-entropy, $H(x)$, and thus its synonym *relative entropy*, can be seen:

$$H(P) = \int p(x) \log p(x) \, \mathrm{d}x. \tag{3.18}$$

The Kullback-Leibler divergence forms the basis of a number of different related divergences. The next sections define and discuss the closed form solutions of Kullback-Leibler divergences for multivariate Gaussians. We denote a multivariate Gaussian with $X \sim \mathcal{N}(\mu, \Sigma)$.

3.4.1.1 Kullback-Leibler Divergence

The Kullback-Leibler divergence and its closed form for two multivariate Gaussians X_1, X_2 is defined as (see for example Hershey and Olsen [HO07]):

$$KL(X_1|X_2) = \tag{3.19}$$
$$\frac{1}{2}\left(\mathrm{tr}\left(\Sigma_2^{-1}\Sigma_1\right) + (\mu_2 - \mu_1)^\top \Sigma_2^{-1}(\mu_2 - \mu_1) - \log_e \frac{|\Sigma_1|}{|\Sigma_2|} - d\right),$$

where $\mathrm{tr}\,(M)$ is the trace of a matrix M.

3.4.1.2 Symmetrized Kullback-Leibler Divergence

The symmetric or symmetrized Kullback-Leibler (SKL) divergence is the most widely used variant of the divergence. Symmetry is an intuitive property which is expected from many divergences. The SKL symmetrizes the Kullback-Leibler divergence by swapping arguments and averaging both divergence values:

$$SKL(X_1, X_2) = \frac{1}{2}\left(KL(X_1|X_2) + KL(X_2|X_1)\right). \tag{3.20}$$

[2] We noted in our experiments that the Jensen-Shannon-like divergence as it is used in PS leads to higher-quality results in the ME and EP algorithms, which originally use the symmetrized Kullback-Leibler divergence

The closed form for the multivariate Gaussian simplifies to:

$$SKL(X_1, X_2) = \frac{1}{4}\left(\mathrm{tr}\left(\Sigma_1\Sigma_2^{-1}\right) + \mathrm{tr}\left(\Sigma_2\Sigma_1^{-1}\right) + \mathrm{tr}\left(\left(\Sigma_1^{-1} + \Sigma_2^{-1}\right)(\mu_1 - \mu_2)^2\right) - 2d\right). \tag{3.21}$$

Implementation Details

If the SKL is used in music similarity algorithms, fast implementations (i) pre-compute the inverted covariance matrix (Σ^{-1}) of each song and (ii) use the fact that the matrix trace of a matrix product can be computed very efficiently (only the matrix diagonal of the product needs to be computed). We present a fast implementation in [Sch07].

3.4.1.3 Jensen-Shannon Divergence

The Jensen-Shannon (JS) divergence is another symmetric divergence derived from the Kullback-Leibler divergence. To compute it, a mixture X_m of the two distributions is defined:

$$X_m = \frac{1}{2}(X_1 + X_2) \tag{3.22}$$

The JS divergence is the average of the Kullback-Leibler divergence of X_1 and X_2 to the mixture X_m:

$$JS(X_1, X_2) = \frac{1}{2}\left(KL(X_1|X_m) + KL(X_2|X_m)\right), \tag{3.23}$$

There exists a dual definition of the Jensen-Shannon divergence which permits its estimation through the Shannon entropy [Lin91]:

$$JS(X_1, X_2) = H(X_m) - \frac{1}{2}H(X_1) - \frac{1}{2}H(X_2), \tag{3.24}$$

Unfortunately the mixture of two Gaussians is not a Gaussian anymore, but a Gaussian mixture model. The Kullback-Leibler divergence between a Gaussian and a mixture model, as well as the Shannon entropy of a Gaussian mixture model has no closed form solution for the multivariate Gaussian. However, a number of approximations exist which can be used instead [HO07].

3.4.1.4 A Jensen-Shannon-like Divergence

To use the Jensen-Shannon divergence (cf. Section 3.4.1.3) to estimate similarities between Gaussians, an approximation of X_m as a single multivariate Gaussian can be used:

$$\mu_m = \frac{1}{2}\mu_1 + \frac{1}{2}\mu_2 \tag{3.25}$$

$$\Sigma_m = \frac{1}{2}\left(\Sigma_1 + \mu_1{\mu_1}^T\right) + \frac{1}{2}\left(\Sigma_2 + \mu_2{\mu_2}^T\right) - \mu_m{\mu_m}^T. \tag{3.26}$$

This approximation of X_m is exactly the same as the left-type Kullback-Leibler centroid of the two Gaussian distributions (see for example Xu et al. [XDDD98]). Other approximations require

Monte-Carlo sampling or even the use of an expectation maximization algorithm to iteratively approximate the divergence. Although the other approximations may be more accurate, their computational overhead is too big to be used in any large application.

The proposed X_m approximation conserves the important divergence properties of symmetry, non-negativity and identity. With its clear definition an approximated JS divergence can be computed easily for the multivariate Gaussian using one of the two introduced Equations (3.24 or 3.23). For the multivariate Gaussian the JS divergence can be simplified to (from Equation 3.24):

$$JS(X_1, X_2) = \frac{1}{2}\log|\Sigma_m| - \frac{1}{4}\log|\Sigma_1| - \frac{1}{4}\log|\Sigma_2|. \tag{3.27}$$

Implementation Details

In implementations of this variant using high dimensional Gaussians, problems can occur with skyrocketing values of determinants which lead to inaccurate results. In this case the logarithm of the determinant should be computed as the sum of the upper-triangular matrix of a Cholesky decomposition. This alternative formulation is much more accurate on standard floating point architectures.

3.4.2 Kullback-Leibler Centroids for the Multivariate Gaussian

This section defines and presents algorithms to compute the centroids for the multivariate Gaussian and all Kullback-Leibler divergences we use in the music similarity algorithms. All methods are implemented in a freely available Octave/Matlab toolbox, to be presented in Section 3.6.

3.4.2.1 Left and Right Centroids

Both, the left- and right-type centroids are explicitly defined for the multivariate Gaussian and can be computed from the definition of the Kullback-Leibler divergence as a general Bregman divergence. Nielsen and Nock [NN09] computed the transformation of the source and natural parameters for the multivariate Gaussian. Using these results we can directly compute the left-type ($C_L(\mathcal{X}) \sim \mathcal{N}(\mu_L, \Sigma_L)$) and right-type ($C_R(\mathcal{X}) \sim \mathcal{N}(\mu_R, \Sigma_R)$) Kullback-Leibler centroids of a set of multivariate Gaussians $\mathcal{X} = \{X_1, \ldots, X_n\}$, $X_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ (^{T} denotes the transpose of a matrix or vector):

$$C_L(\mathcal{X}): \qquad \mu_L = \frac{1}{n}\sum_{i=1}^{n}\mu_i \qquad\qquad \Sigma_L = -\mu_L\mu_L^T + \frac{1}{n}\sum_{i=1}^{n}\left(\mu_i\mu_i^T + \Sigma_i\right) \tag{3.28}$$

$$C_R(\mathcal{X}): \qquad \mu_R = \Sigma_R\left(\frac{1}{n}\sum_{i=1}^{n}\Sigma_i^{-1}\mu_i\right) \qquad \Sigma_R = \left(\frac{1}{n}\sum_{i=1}^{n}\Sigma_i^{-1}\right)^{-1} \tag{3.29}$$

3.4.2.2 Symmetrized Centroids

With the exact definition of the two separate Kullback-Leibler centroids, the centroid for the symmetrized Kullback-Leibler divergence can be computed in a straightforward way, using $C_L(\mathcal{X})$

and $C_R(\mathcal{X})$ in the bisection search algorithm for symmetrized Bregman divergences as described in Section 3.3.2.1.

Another way to quickly find a centroid approximation is to use a mid-point approximation instead of computing the exact centroid. The mid-point empirically proved to be a good approximation of the true centroid of the symmetrized Kullback-Leibler divergence [Vel02]. The approximation merges the weighted left and right centroids in one step:

$$C_M(\mathcal{X}): \qquad \mu_M = \frac{1}{2}\left(\mu_L + \mu_R\right) \qquad\qquad \Sigma_M = \frac{1}{2}\sum_{i=\{L,R\}}\left(\mu_i\mu_i^T + \Sigma_i\right) - \mu_M\mu_M^T \tag{3.30}$$

To estimate the Jensen-Shannon divergence centroid of a set of multivariate Gaussians $\mathcal{X} = \{X_1, ..., X_n\}$, two Kullback-Leibler divergences would need to be minimized (Note that the arguments of the KL divergence are swapped compared to a Bregman divergence B_F)):

$$C_{JS} = \arg\min_C \sum_{i=1}^{n} KL\left(\frac{X_i + C}{2}|X_i\right) + KL\left(\frac{X_i + C}{2}|C\right). \tag{3.31}$$

The minimizer would define the centroid of the Jensen-Shannon divergence as used in the PS music similarity algorithm. There exists no direct centroid estimation method to compute that centroid. However, the symmetrized Kullback-Leibler centroid can be used instead as an approximation.

3.5 Self-Organizing Maps with Gaussian Music Similarity Models

This section is an extension of the work published under the title "Islands of Gaussians: The Self Organizing Map and Gaussian Music Similarity Features" [SFWG10].

There already exists a wide range of publications dealing with visualizing the similarity structure of digital music collections on self-organizing maps (SOMs). The SOM [Koh01] is an unsupervised neural network that organizes multivariate data on a two dimensional map and is suited well for visualizations. It maps items which are similar in the original high-dimensional space onto locations close to each other on the map.

One of the first to use the SOM algorithm to visualize music according to a computer music similarity measure were Rauber and Frühwirth [RF01], who use a basic music similarity feature and a simple tabular grid to display the clustered song titles on the map. This idea was extended by Pampalk et al. who use Fluctiation Patterns (see Section 2.3.2) and the smoothed data histogram (SDH) visualization to draw the SOM [PRM02b]. Their visualization is inspired by geographical maps: blue regions (oceans) indicate areas onto which very few pieces of music are mapped, whereas clusters containing a larger quantity of pieces are colored in brown and white (mountains and snow). It was published under the name "Islands of Music" [PRM02a].

"Neptune" [KSPW07], developed by Knees et al., improved the "Islands of Music" visualization by taking the two dimensional map into the third dimension. They added information and pictures from the web and allow a 3D walk through a music collection. They use EP as their music similarity measure (see Section 2.4.3) to arrange songs on the map.

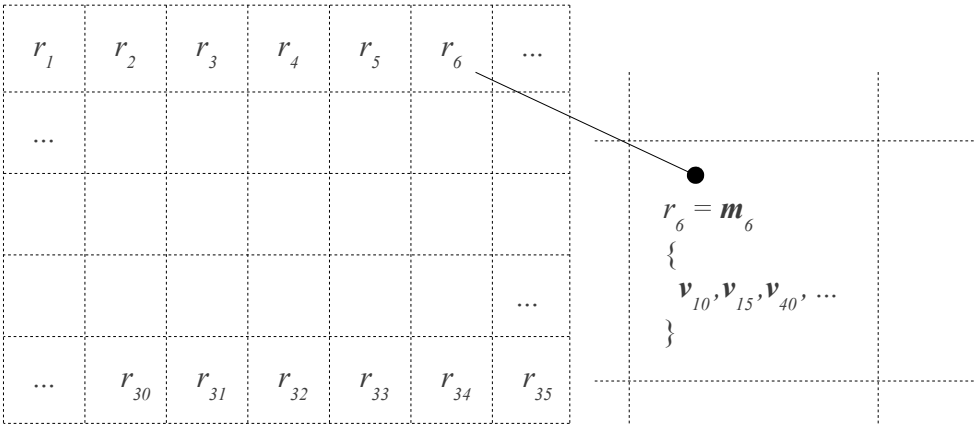


Figure 3.2: This plot shows a sketch of 5×7 SOM grid. Each map unit on the grid is denoted with r_i, its unique model vector with $\boldsymbol{m}_i$. Each map unit is assigned a number of data vectors ($\boldsymbol{v}_j$).

The "Globe of Music" [LT07] by Leitich and Topf uses a GeoSOM [WT06] to map the music collection onto a globe for exploration. Lübbers developed the "SoniXplorer" [Lüb05] to navigate through music archives. They use a multimodal navigation model where the music is synchronized to the position on the map to support the user in the SOM navigation of the music collection.

Mörchen et al. use the Emergent SOM algorithm to visualize and cluster music collections in their "Music Miner" [MUNS05] system. For music similarity they use a large set of low-level features similarly to the features used in MARSYAS (see Chapter 2, Section 2.3.2).

What all the publications of SOM visualizations of music archives have in common, is that they could not directly be used with any of the music similarity measures discussed in this work (ME, EP, PS) using Gaussians similarity models and Kullback-Leibler based divergences as similarity function. With the previously presented centroid computation methods using these music similarity measures is now possible. This section introduces the generalized SOM algorithm which is capable to natively cluster multivariate Gaussians using the Bregman centroids. We show that the new SOM maps which are generated have a higher quality than SOM maps created with approaches artificially vectorizing the data.

3.5.1 The SOM Algorithm

The SOM consists of an ordered set of so-called *map units* r_i, each of which is assigned a *reference vector* (or *model vector*) $\boldsymbol{m}_i$ in the feature space. The set of all reference vectors of a SOM is

called its *codebook*. In the simplest case the codebook is initialized by a random strategy. Each r_i is assigned the data vectors $\boldsymbol{v}_i$ which are most similar to $\boldsymbol{m}_i$. The map units r_i are usually put on a two dimensional grid. The distance of two map units on the grid can be computed with the Euclidean or Manhattan distance. Figure 3.2 shows an example of a simple rectangular SOM grid. There exists a multitude of SOM visualization techniques like the U-matrix visualization. A popular way to visualize SOM grids trained with music similarity models is the smoothed data histogram (SDH) [PRM02b]. SDHs visualizes clusters in the data set by estimating the probability density of the data samples on the SOM.

To compute a SOM, first the map dimensions and the number of training iterations (t) have to be fixed. The training is done in four basic repeating steps:

1. At iteration t select a random vector $\boldsymbol{v}(t)$ from the set of features.
2. Search for the best matching map unit c on the SOM by computing the distance of $\boldsymbol{v}(t)$ to all $\boldsymbol{m}_i$ in feature space distance.
3. The codebook is updated by calculating a weighted centroid between $\boldsymbol{v}(t)$ and the model vector $\boldsymbol{m}_c$ of the best-matching unit r_c. Based on a neighborhood weighting function $h_{ci}(t)$ all map units participate in the adaptations depending on their distance on the two-dimensional output map. Equation 3.32 uses a standard Gaussian neighborhood function.

$$h_{ci}(t) = \alpha(t) \exp\left(-\frac{||r_c - r_i||}{2\alpha^2(t)}\right) \tag{3.32}$$

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) + h_{ci}(t)\left[\boldsymbol{v}(t) - \boldsymbol{m}_i(t)\right] \tag{3.33}$$

4. The adaptation strength $\alpha(t)$ is decreased gradually with the iteration cycle t. This supports the formation of large clusters in the beginning and a fine-tuning toward the end of the training.

Usually, the iterative training is continued until a convergence criterion is fulfilled or a preselected number of training iterations is finished. In the final step all data items $\boldsymbol{v}_j$ are assigned to the map unit r_i whose model vector they are most similar to.

Although originally SOMs were defined for Euclidean feature vectors only, the algorithm is not limited to the vector space. Kohonen himself mentions this in the most recent edition of his standard work on self-organizing maps [Koh01]. This observation will be the basis for extending the SOM algorithm to the "distribution space".

3.5.2 The Generalized SOM

A closer look at the SOM algorithm sketched in the last section, shows that there is only a single step where the algorithm in fact depends on vectors. It is the computation of the weighted centroid in Equation 3.33[3]. We now rewrite Equation 3.33 so that it is more obvious that a centroid (a weighted mean of several vectors) is computed:

$$\boldsymbol{m}_i(t+1) = (1 - h_{ci}(t))\,\boldsymbol{m}_i(t) + h_{ci}(t)\boldsymbol{v}(t) \tag{3.34}$$

[3]The distance $||r_c - r_i||$ in Equation 3.32 is computed in 'map space' and does not need to be modified.

By rewriting Equation 3.33 this way we discover that if a weighted centroid can be computed for a divergence, the SOM algorithm could be adopted to work with these divergences – and, in the case of this work, with the features and divergences used in any music similarity measure a weighted centroid can be computed.

3.5.2.1 Weighted Gaussian Centroids

To use the Kullback-Leibler divergences and Gaussian features of the music similarity algorithms we modify the Gaussian centroids defined in Section 3.4.2 and add a per-Gaussian weighing term ω_i (with $\sum_{i=1}^{n} \omega_i = 1$). Thus the left-type and right-type weighted Kullback-Leibler centroid definitions are:

$$W_L^{\omega}(\mathcal{X}): \qquad \mu_L = \sum_{i=1}^{n} \omega_i \mu_i, \qquad \Sigma_L = -\mu_L \mu_L^T + \sum_{i=1}^{n} \omega_i \left(\mu_i \mu_i^T + \Sigma_i\right), \tag{3.35}$$

$$W_R^{\omega}(\mathcal{X}): \qquad \mu_R = \Sigma_R \left(\sum_{i=1}^{n} \omega_i \Sigma_i^{-1} \mu_i\right), \qquad \Sigma_R = \left(\sum_{i=1}^{n} \omega_i \Sigma_i^{-1}\right)^{-1}. \tag{3.36}$$

To compute weighted symmetrized Kullback-Leibler centroids, first the weighted left and right centroids are to be computed. The bisection search algorithm (Section 3.3.2.1) or the mid-point approximation (Section 3.4.2.2) can then be used to find the weighted symmetrized Kullback-Leibler centroid.

3.5.2.2 Generalized Algorithm

With the definition of the weighted Kullback-Leibler centroids everything is in place to use the SOM algorithm with Gaussian music similarity models:

- The initialization of the SOM and its $\boldsymbol{m}_i$ is done by selecting random Gaussians from the music similarity models. An initialization with a PCA is not possible in this case due to the non-vectorial features.
- The iterative computation of the weighted centroid during the training of the codebook can now be replaced with the respective weighted Kullback-Leibler centroids.
- The learning rate adaptation and neighborhood functions do not need to be changed. They are not dependent on the features.
- In the final step the Gaussians are assigned to the nearest map units according to their Kullback-Leibler divergence.

The generalized SOM algorithms now works directly with the Gaussian features; clustering is done natively with the data and divergences as the algorithm was originally intended to.

3.5.3 Evaluation

We now evaluate the generalized SOM with the ME and EP music similarity measures. As the original PS algorithm uses a new normalization technique (Mutual Proximity, the main topic of Chapter 4) to combine timbre and rhythm similarity and get rid of hubs, a centroid computation is impossible. To use PS in a generalized SOM we use a workaround and only use the timbre component in the SOM algorithm.

We plot and discuss two large SOMs which have been computed for the GTzan music collection: one is computed using a vectorization approach, the other with the generalized SOM algorithm natively using Gaussians.

To test how the SOM algorithm performs operating directly on the Gaussians we use all music collections and similarity algorithms we have introduced. We compare the quality of the SOMs generated with our approach to SOMs generated with vectorized features, which are computed as follows: for each Gaussian model we build a vector by computing the distance to all other model and normalize this distance vector to zero mean and unit variance. This is equivalent to computing the full similarity matrix and using each (normalized) row as a feature vector (as done e.g., in [KSPW07, Pam06]).

As a baseline for our experiments we show a randomly initialized SOM without any training. In our experiments we vary the SOM size according to the size of the music collection. We use a SOM grid size of 7×7 for collections with less than 1 000 songs, and a grid size 14×14 for larger collections. To ensure a fair evaluation we took the following two precautions:

- In each run the same random seed is used for the random, vectorized and Gaussian SOM. This ensures identical random initialization and use of the same randomly chosen features during the training phase.
- Each unique experiment configuration is repeated ten times. The results reported here are average values.

To quantitatively compare SOMs computed using the different methods, we compute the average k-nearest neighbor rank distance and average map unit precision for each SOM. These measures are described in the next paragraphs.

Average Nearest Neighbor Rank Distance

To compare SOMs generated with different approaches, we quantify how well the original neighborhood topology is preserved in a SOM mapping. As we need to compare SOMs using different metrics it is not possible to use standard SOM mapping quality measures like the quantization error or clustering quality measures like the Dunn's Index or the Davies-Bouldin Index [BP98].

Because of these limitations we define a rank measure: the *average nearest neighbor rank distance*, which allows to cross-compare SOMs which were computed using different features and divergence measures.

To do that, we search for the k-nearest neighbors of every item x_i in the original space and check their location on the SOM. Ideally the nearest neighbors should also be mapped close to each other on the SOM.

For a given k and a Gaussian x_i this will be measured as the *k-nearest neighbor rank distance*:

1. Assign all Gaussians to their corresponding map unit on the two dimensional SOM grid according to the similarity measure.

2. Assign all Gaussians x_i the coordinates of their map unit on the two-dimensional SOM grid (references as s_i).

3. For a Gaussian x_i, use its two dimensional representation on the SOM (s_i) and compute the Euclidean distance to all other Gaussians x_j using s_j, with $j \neq i$.

4. Sort the list of Euclidean distances in ascending order and transform it into a list of ranks.

5. Find the k nearest neighbors of x_i in the original space, and average across their corresponding ranks on the SOM (step 4) to get the *k-nearest neighbor rank distance* for x_i.

The *average k nearest neighbor rank distance*, computed over all x_i, measures how close objects and their neighborhoods are mapped on the SOM, according to the original similarity measure. The lower its value, the better the preservation of the original neighborhoods on the SOM. In the experiments we compute the average k-nearest neighbor rank distance for $k = 1, 5, 10, 20$.

Average Map Unit Precision

The label of a map unit is usually set by a majority vote of the class of the objects which are assigned to it. We use the label of each map unit i to compute the percentage of objects on the unit with the same class. We call that map unit precision (p_i).

We average the precision over all map units which have objects assigned to them, and call that the average map unit precision ($\bar{p}$). By averaging only over the map units which have objects assigned to them, tighter clustering on the SOM is rewarded. The average map-unit precision should always be considered together with the average nearest-neighbor rank distance. High values of $\bar{p}$ alone can be misleading as they do not tell anything about the mapping of the similarity space. But if the average nearest-neighbor rank distance is low too, and thus the SOM algorithm was successful in creating a good mapping of the the songs according to the similarity measure, a good SOM according to the collection's genres could be created.

3.5.3.1 Results

Figure 3.3 and Figure 3.4 show the results of the evaluation computing SOMs using the music similarity algorithms ME and EP, (a) with vectors from the distance matrix and (b) natively using Gaussians in the generalized SOM algorithm.

From the bar-plots in both figures we can see that in all cases for $k \geq 5$ the average kNN rank distance is smaller for the Gaussian variant than using vectors from the full distance matrix - an indication that the intermediate neighborhoods of objects is more accurately mapped in the Gaussian SOMs than in the SOMs computed from vectors. The average map unit precision $\bar{p}$ shows a similar positive picture for the Gaussian SOMs; simultaneously to the decreasing kNN rank distance, $\bar{r}$ is increasing all collections with ME (Figure 3.3) and in eight of nine collections with EP (Figure 3.4). We interpret that as an increase of consistency in the map units.

Taking for granted that the music similarity measure is of high quality, these results admit the conclusion that SOMs computed directly with Gaussians produce higher-quality mappings and this method should be preferred.

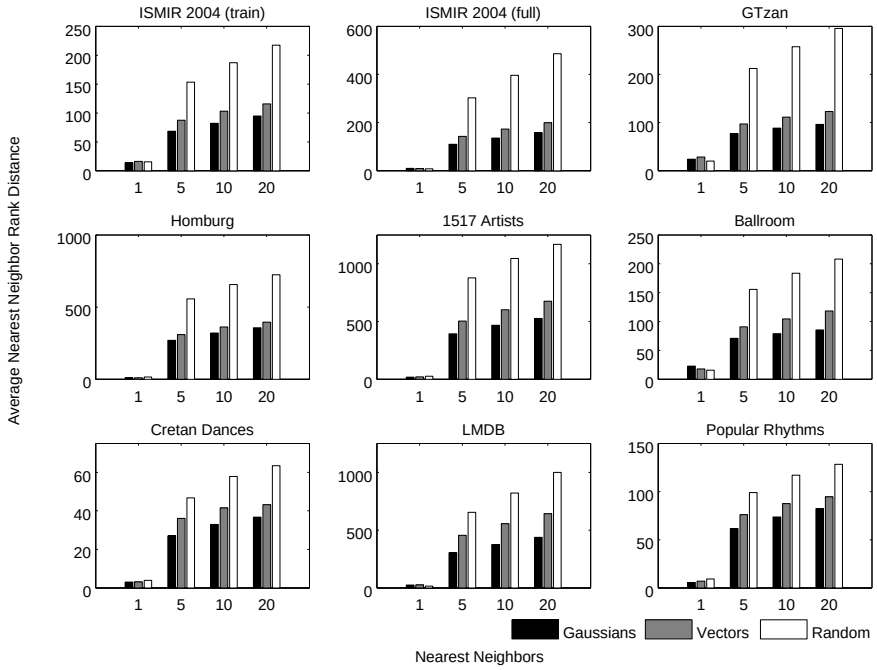
Besides producing higher-quality SOMs, we emphasize that this approach is also far less complex to compute than a variant working with vectorized features: (1) it is almost impossible to compute the full similarity matrix on a large collection of songs (i.e. over 100 000 songs) and (2) a SOM with 100 000-dimensional (or larger) vectors would be very expensive to compute. By using random projections [BM01] one can overcome that, but that would probably come with a loss of mapping quality. A SOM computed directly with the Gaussians, on the other hand, requires only a fraction of the computational effort, as the full similarity matrix does not need to be computed and the original features are used as intended.

3.5.3.2 Subjective Evaluation

To demonstrate the benefits of our method in an example, we use the generalized SOM algorithm to natively compute a SOM using the timbre component of the PS music similarity measure. We compare the result to a SOM computed using the rows of the distance matrix as feature vectors. The SOM grid size is set to 20×20 and the GTzan music collection (1 000 songs from 10 music genres, Chapter 2, Section 2.5.1) is used. The SDH visualization which is used in the following plots creates "islands" (the gray areas) where a high density of objects was found and water (white) where a low number of objects were found. The labels in the SDH visualization are set in a majority vote per map unit.

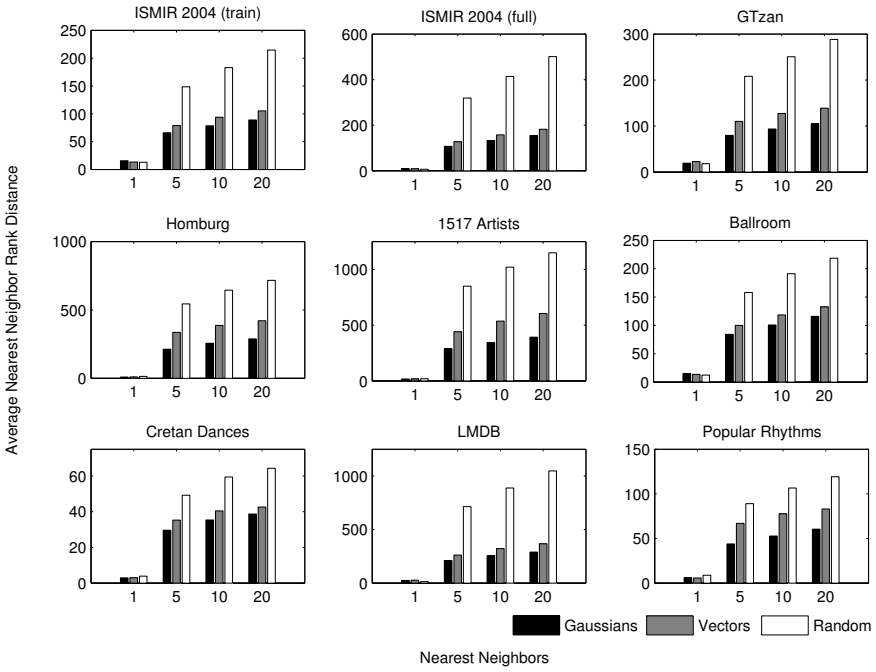
Figure 3.5 shows the result of 300 training iterations for the Gaussian SOM. The SOM has, at $k = 5$, an average NN rank distance of 65.3. For comparison, the randomly initialized SOM (using the same random seed) has a much higher average rank distance (331.1). When looking at the SOM in Figure 3.5, we can identify six very well clustered areas (highlighted in the figure): (1) clusters classic music and jazz, (2) country music, (3) reggae and hip-hop, (4) jazz and classic music, (5) metal and rock and (6) country and blues.

As in the quantitative evaluations, we use the same random seed and compute a SOM using the vectors from the full distance matrix as features. The resulting SOM has, at $k = 5$, an average NN rank distance of 87.8, which is a clear indication of a worse clustering than in the Gaussian SOM. When looking at the SOM in Figure 3.5 and comparing it to the Gaussian SOM we can only identify two coherent regions: (1) metal and rock and (2) a large classic and jazz region. The rest of the map seems rather mixed and spread around the map, further confirming the results of the quantitative evaluation that our native algorithm yields better SOMs than vectorizing approaches.



			Nearest Neighbors				
Collection	SOM	*Type*	1	5	10	20	$\bar{p}$
ISMIR 2004 (train)	7×7	Gaussians	**0.88**	**0.44**	**0.44**	**0.44**	**67.0%**
		Vectors	1.05	0.57	0.55	0.53	63.4%
ISMIR 2004 (full)	14×14	Gaussians	1.16	**0.36**	**0.34**	**0.32**	**76.2%**
		Vectors	**1.09**	0.47	0.44	0.41	69.9%
GTzan	7×7	Gaussians	**1.19**	**0.36**	**0.34**	**0.33**	**52.6%**
		Vectors	1.43	0.46	0.43	0.42	50.9%
Homburg	14×14	Gaussians	0.73	**0.48**	**0.49**	**0.49**	**56.5%**
		Vectors	**0.67**	0.55	0.55	0.55	51.7%
1517 Artists	14×14	Gaussians	**0.68**	**0.45**	**0.45**	**0.45**	**36.0%**
		Vectors	0.79	0.57	0.58	0.58	32.2%
Ballroom	7×7	Gaussians	1.44	**0.46**	**0.43**	**0.41**	**56.5%**
		Vectors	**1.13**	0.58	0.57	0.57	51.3%
Cretan Dances	7×7	Gaussians	**0.78**	**0.58**	**0.57**	**0.58**	**59.5%**
		Vectors	0.79	0.77	0.72	0.68	59.0%
LMDB	14×14	Gaussians	**1.69**	**0.47**	**0.46**	**0.44**	**64.5%**
		Vectors	1.74	0.70	0.68	0.64	47.5%
Popular Rhythms	7×7	Gaussians	**0.60**	**0.62**	**0.63**	**0.64**	**47.9%**
		Vectors	0.75	0.77	0.75	0.74	45.0%

Figure 3.3: Results of the evaluation of the nine music collection using ME is used as music similarity algorithm. The bar-plots show the average k-nearest neighbor rank distance for $k = 1, 5, 10, 20$ for each collection comparing the random, vector and Gaussian SOMs. The table below shows the average kNN rank distance in relation to a randomly initialized one and the average map unit precision $\bar{p}$ for the vector and Gaussian SOMs.



			Nearest Neighbors				
Collection	SOM	*Type*	1	5	10	20	$\bar{p}$
ISMIR 2004 (train)	7×7	Gaussians	1.21	**0.44**	**0.43**	**0.41**	**69.8%**
		Vectors	**1.04**	0.53	0.51	0.49	66.7%
ISMIR 2004 (full)	14×14	Gaussians	1.35	**0.34**	**0.32**	**0.31**	**76.0%**
		Vectors	**1.29**	0.40	0.38	0.36	74.3%
GTzan	7×7	Gaussians	**1.06**	**0.38**	**0.37**	**0.36**	**52.1%**
		Vectors	1.27	0.53	0.51	0.48	46.0%
Homburg	14×14	Gaussians	**0.67**	**0.39**	**0.40**	**0.40**	**55.5%**
		Vectors	0.73	0.62	0.60	0.59	53.9%
1517 Artists	14×14	Gaussians	**0.89**	**0.34**	**0.34**	**0.34**	**35.8%**
		Vectors	0.92	0.52	0.52	0.53	34.4%
Ballroom	7×7	Gaussians	1.23	**0.53**	**0.53**	**0.53**	**55.1%**
		Vectors	**1.10**	0.63	0.62	0.61	50.9%
Cretan Dances	7×7	Gaussians	**0.76**	**0.60**	**0.59**	**0.60**	**60.8%**
		Vectors	0.79	0.72	0.68	0.66	58.9%
LMDB	14×14	Gaussians	**1.76**	**0.29**	**0.29**	**0.28**	**65.7%**
		Vectors	2.00	0.37	0.36	0.35	61.6%
Popular Rhythms	7×7	Gaussians	0.68	**0.49**	**0.49**	**0.51**	46.0%
		Vectors	**0.64**	0.75	0.73	0.70	**48.4%**

Figure 3.4: Results of the evaluation of the nine music collection using EP is used as music similarity algorithm. The bar-plots show the average k-nearest neighbor rank distance for $k = 1, 5, 10, 20$ for each collection comparing the random, vector and Gaussian SOMs. The table below shows the average kNN rank distance in relation to a randomly initialized one and the average map unit precision $\bar{p}$ for the vector and Gaussian SOMs.

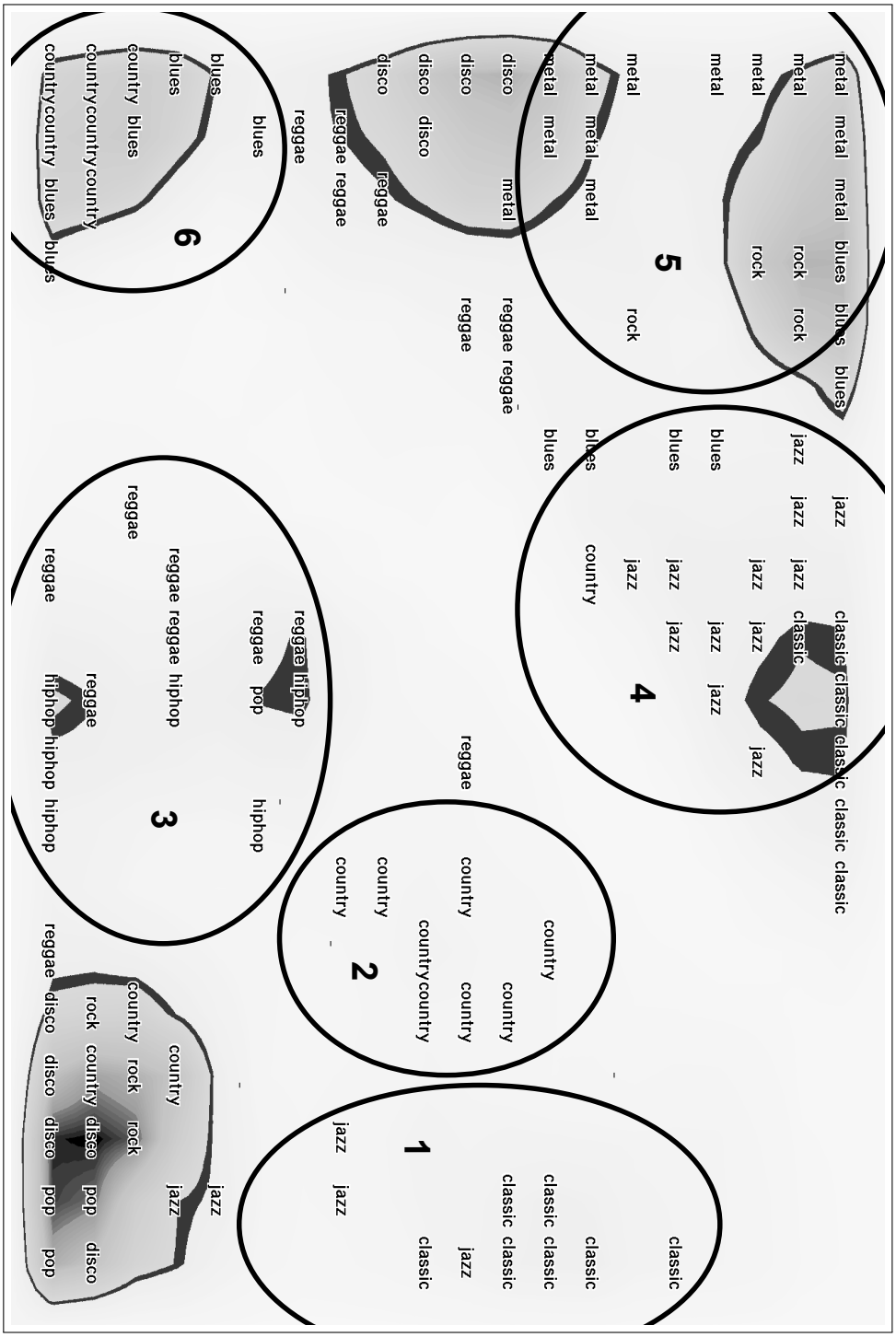


Figure 3.5: An SDH visualization of a 20×20 SOM for the GTzan collection and PS music similarity. It is using multivariate Gaussian features and Bregman centroids. Six contiguous regions can be seen.

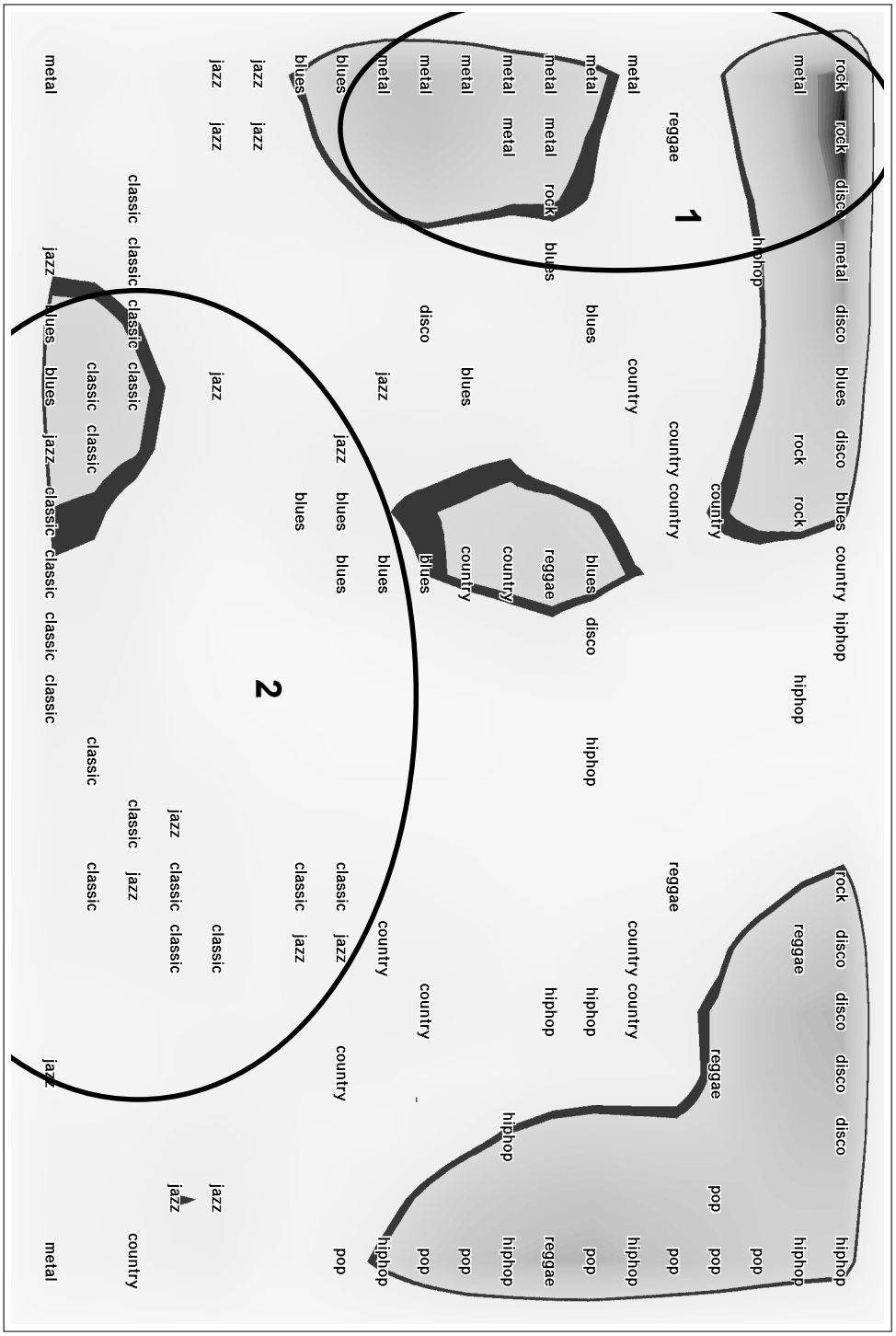


Figure 3.6: An SDH visualization of a 20×20 SOM for the GTzan collection and PS music similarity. Euclidean vectors (from the distance matrix) are used as features. Two contiguous regions can be seen.

3.6 Multivariate Normal (MVN) Matlab Toolbox

This section describes the MVN (MultiVariate Normal) Matlab toolbox which implements all divergences, centroids and algorithms for the multivariate Gaussian which were described in the previous sections of this chapter. The toolbox is freely available on the Internet.[4].

To see how the toolbox works for example with Elias Pampalk's music analysis (MA) matlab toolbox [Pam04], skip to Section 3.6.6 where some examples are shown. Of course usage is not limited to music similarity models, any multivariate Gaussian features can be processed with this toolbox.

3.6.1 Initialization

Initialization of MVN Gaussians is done using a single function call:

- `mvn1 = mvn_new(co, m)`. To instantiate a new n-dimensional multivariate Normal (MVN) object with the toolbox, pass the covariance matrix (`co`) and mean vector (`m`) of your Gaussian to the `mvn_new()` function. The return value `mvn1` is a Matlab structure with the following attributes:
  - `co`: The full $n \times n$ covariance matrix. If the matrix is not positive-definite an exception is thrown and execution aborted.
  - `m`: The n-dimensional mean vector of the Gaussian.
  - `ico`: For speed reasons we also pre-compute the inverse of the covariance matrix $(n \times n)$ using the fast Cholesky decomposition.
  - `logdet`: We store the logarithm of the determinant of the covariance matrix `co`. `logdet` is computed from the Cholesky decomposition.

  To estimate a n-dimensional Gaussian from an $m \times n$ data matrix (like a matrix of MFCC vectors) just use the built-in Matlab functions **`cov`**`(data)` and **`mean`**`(data)`.

3.6.2 Divergences

The toolbox implements a variety of divergences which can be used to compute a distance or similarity between two MVN models. The divergences implemented are all discussed in Section 3.4.1. The most common divergence is the Symmetrized Kullback-Leibler divergence.

- `d = mvn_div_kl(m1, m2)`

  Computes the (asymmetric) Kullback-Leibler divergence between the MVN `m1` and `m2`. See Section 3.4.1.1 for the properties of the KL divergence.

- `d = mvn_div_skl(m1, m2)`

  Computes the symmetric Kullback-Leibler divergence between the MVN `m1` and `m2`. See Section 3.4.1.2 for the properties of the symmetric KL divergence.

[4] `http://www.ofai.at/~dominik.schnitzer/mnv`, visited August 12th, 2011

- `d = mvn_div_js(m1, m2)`

  Computes the Jensen-Shannon-like divergence between MVN `m1` and `m2`. Since it is not possible to compute the Jensen-Shannon divergence between two multivariate Normals, we use an approximation which works quite well in our approximations (Section 3.4.1.4).

- `D = mvn_divmat(models, divergence)`

  This is a meta-function to quickly compute the whole similarity matrix for the given `models` and `divergence`. The parameter `models` is a struct-array of MVN models and `divergence` is a string specifying the divergence to use: `'kl_left'`, `'kl_right'`, `'skl'`, `'js'`. The full distance matrix `D` is returned in single precision to save memory.

3.6.3 Centroids

- `c = mvn_bregmancentroid_kl_left(models)`

  Computes the left-sided Kullback-Leibler centroid of the given MVN models as defined in Section 3.4.2.1.

- `c = mvn_bregmancentroid_kl_right(models)`

  Computes the right-sided Kullback-Leibler centroid of the given MVN models as defined in Section 3.4.2.1.

- `c = mvn_bregmancentroid_kl_skl(models, approx)`

  Computes the symmetrized Kullback-Leibler centroid of the given MVN models (Section 3.4.2.2). If the parameter `approx` is not set, the centroid is computed using a geodesic walk algorithm. If `approx` is set to the value 1, an approximative centroid is returned. The approximative centroid is computed very fast, and is in many cases sufficiently exact.

3.6.4 Clustering

- `[centers, assig, qe] = mvn_kmeans(models, k, divergence)`

  This function implements the k-means clustering algorithm for MVN models. It does that in regard to the selected divergence. Similar to the `mvn_divmat()` function the `divergence` parameter selects the divergence to use for the k-means clustering: `'kl_left'`, `'kl_right'`, `'skl'`, `'skl_mid'`.

  Return values are a struct-array with the centers in `centers`, a vector which assigns each MVN to a centroid (`assig`) and a quantization error (`qe`) according to the divergence chosen.

- `[mapunits D] = mvn_som(models, n, divergence, sel)`

  This function trains a basic self-organizing map with the generalized SOM algorithm (defined in Section 3.5. The dimensions are specified by the parameter `n` which in turn creates a square SOM with the dimension $n \times n$. To compute the SOM the function uses the MVN models in parameter `models` and the divergence selected by `divergence`. The divergence paramter can be one of these divergences: `'kl_left'`, `'kl_right'`, `'skl'`, `'skl_mid'`.

To restrict the models which should be used to train the SOM, the optional parameter `sel` can be set. Set it to the indices you want to include in SOM computation.

The first return value `mapunits` is a $n \times n$ SOM grid where each map unit is represented by an MVN model. The whole SOM has n^2 map units. The structure array has the following structure:

- `x, y` the x/y-axis position of the map unit
- `n` An array which stores the indices of the models which are assigned to this map unit.

The return value `D` is an $(n * n) \times m$ matrix, where m is the number of `models` used during computation.

3.6.5 Additional Functions

- `h = mvn_entropy(m1)`

  Computes the entropy of the given an N-dimensional MVN `m1` with the covariance Matrix Σ. The entropy h is computed as:

$$h = \frac{1}{2}\left(N + N\ln\left(2\pi\right) + \ln|\Sigma|\right) \tag{3.37}$$

- `p = mvn_ismetric(D)`

  Given a divergence matrix `D`, returns the fraction of triples elements obeying the triangle inequality.

3.6.6 Usage Examples

Feature Extraction We use the Music Analysis (MA) Matlab toolbox[5] by Elias Pampalk to extract audio music similarity features for the ISMIR2004 *Genre/Artist Identification/Classification* collection which is available freely on the web[6]. To do so we extract the features with the command:

```
1  ma_g1c_FeatureExtraction('ismir04_filelist.txt', 'features_dir/');
```

The the text file `ismir04_filelist.txt` lists all filenames of the ISMIR 2004 dataset introduced before. The above command does the feature extraction for all songs given in the text file and writes the features in the specified directory.

[5] `http://www.pampalk.at/ma/`, visited August 12th, 2011
[6] `http://ismir2004.ismir.net/genre_contest/index.htm`, visited August 12th, 2011

Loading the Features To load the features which, we first load the features from the "G1C_features.mat" file to memory:

```
1  load features_dir/G1C_features.mat
```

After that we prepare the features for MVN processing and load the filenames which encode the music genre in their path.

```
1  models = mvn_new(squeeze(data.feat.g1.co(1,:,:)),...
2                   squeeze(data.feat.g1.m(1,:,:)));
3
4  for i = 2:length(data.filenames)
5    models(i) = mvn_new(squeeze(data.feat.g1.co(i,:,:)),...
6                        squeeze(data.feat.g1.m(i,:,:)));
7  end
8
9  filenames = importdata('ismir04_files.txt');
10 [genres grene_assignment] = mvn_fn2class(filenames, 4, '/');
```

Lines 1-7 initialize the MVN models, line 9-10 loads the filenames and extracts the genre names from the filenames. We will use the genres of the individual files for classification experiments later on.

Similarity and K-Means Clustering In this step we compute a divergence matrix using the Symmetric Kullback-Leibler divergence (line 1). For completeness and to check if everything was done correctly, we compute the 1-nearest neighbor classification accuracy (line 2, `nn1_accuracy = 0.7819` in the example).

```
1   D = mvn_divmat(models, 'skl');
2   nn1_accuracy = mvn_knnclass(D, genre_assignment, 1);
3
4   [centroids c_assignment] = mvn_kmeans(models, 10, 'skl');
```

In line 4 we compute a randomly initialized k-means clustering of our MVN models. In the return value `centroids` we return the 10 centroids found with the k-means clustering. The variable `c_assignment` stores the index of the centroid a MVN model is assigned.

Figure 3.7 displays the cluster/music genre assignment confusion matrix which is generated from the k-means clustering. We clustered the collection into 10 clusters. In the figure it can be seen that the clustering which was emerging has a jazz/blues cluster (3), multiple classical/world clusters (1, 2, 4, 5, 6, 8), a strong metal/punk cluster (9), a pop/rock/electronic cluster (7) and an electronic/pop/rock/jazz cluster (10).

Computing and visualizing a SOM In the next code snippet we compute the SOM and select the SOM unit labels.

```
1    [som_grid, som_D] = mvn_som_skl(models, [20 20]);
2
```

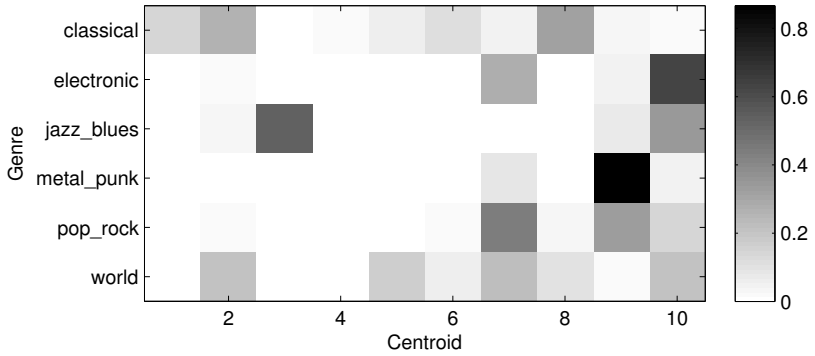


Figure 3.7: Confusion matrix displaying the genre to k-means cluster assignment. The various clusters with their relative genre composition can be seen.

```
3       % Compute the labels for the SOM
4       labels = cell(length(som_grid), 1);
5       for i=1:length(som_grid)
6           if (isempty(som_grid(i).n)) continue; end
7           ng = genre_assignment(som_grid(i).n);
8           check = unique(ng);
9           maxct = 0; maxj = 0;
10          for j=1:length(check)
11              nct = length(find(ng == check(j)));
12              if (nct > maxct)
13                  maxj = j;
14                  maxct = nct;
15              end
16          end
17          if (maxct < 3) continue; end
18          labels{i} = genres{check(maxj)};
19      end
```

Finally we visualize the SOM using the Smoothed-Data Histograms Matlab Toolbox[7] and label it according to the genre label names we just prepared in the last code snippet (line 4-19)

```
1       M.dist_codebook = 1:(20*20);
2       M.topol.msize = [20 20];
3       S = sdh_calculate(som_D, M, 'spread', 10);
4
5       % Visualize SOM using the SDH Toolbox
6       sdh_visualize(S, 'sofn', 0, 'labels',  labels);
```

Figure 3.8 shows the Matlab Figure displaying the SOM using the ME music similarity measure. The Smoothed-Datagram visualization which was first presented for Music Collections in [Pam03] is called *Islands of Music*. When looking at the map we can see that large *Classical Music* islands emerged. On the bottom of the visualization we can see an *Electronic Music* island

[7] `http://www.ofai.at/~elias.pampalk/sdh/`, visited August 12th, 2011

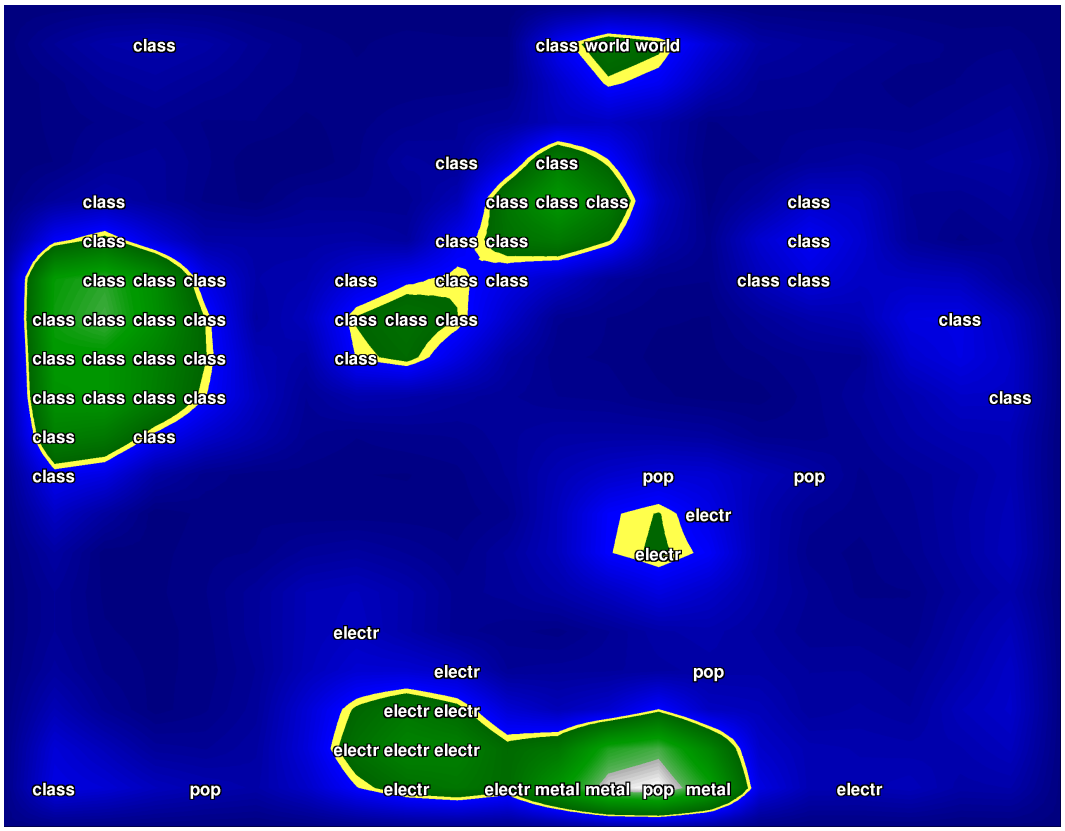


Figure 3.8: *Islands Of Music* visualization of the ISMIR 2004 collection using standard timbre music similarity features extracted with the MA Matlab toolbox.

which is connected to a *Metal* island. On top of the map there is a *World Music* island.

3.7 Summary

This chapter gave an introduction to Bregman divergences and their centroids to define the prerequisites for working natively with multivariate Gaussian models and their divergences. We reviewed existing literature and linked the recent research in the field of Bregman divergences to music similarity measures. With that knowledge, any centroid-computing algorithm can be used natively with Gaussians and their divergences. The remainder presented the main contributions of this chapter:

We developed a generalized SOM algorithm which uses weighted centroids to directly and natively cluster Gaussians. In contrast to previous solutions no vectorization of features needs to be done to compute a SOM. We evaluated the generalized algorithm on all music collections and show that the algorithm, besides being very efficient, yields higher quality clusterings than previous approaches using vectorization.

Second, we presented a freely available Octave/Matlab toolbox which was developed to to ease working with multivariate Gaussian features. The toolbox is available freely on the Internet and was developed with performance in mind. It includes implementations of all methods which have been presented in this chapter.

Chapter 4

Reducing Hubs with Mutual Proximity

Contents

4.1 Introduction 64
4.2 Related Work 65
4.3 Scaling Methods 67
4.3.1 Local Scaling 67
4.3.2 Global Scaling - Mutual Proximity 68
4.4 Evaluation 72
4.4.1 Benchmarks 72
4.4.2 Public Machine Learning Datasets 73
4.4.3 Results 74
4.4.4 Summary of results 81
4.5 Mutual Proximity and Content-Based Music Similarity 81
4.5.1 Algorithms 83
4.5.2 Evaluation 85
4.5.3 Discussion 88
4.6 Summary 90

This chapter presents an unsupervised method which is able to remove hubs in retrieval algorithms. Hubs are data points which keep appearing unwontedly often as nearest neighbors of a large number of other data points.

Section 4.1 and 4.2 give a general introduction to the hub problem and nearest neighbor search in high dimensional spaces. In Section 4.3 we show that neighborhood scaling methods are able to deal with the problem of hubs, and we present a novel global neighborhood scaling method called Mutual Proximity. Contrary to the local scaling methods, Mutual Proximity has the advantage

that it can be used with large databases using a straightforward approximation. We do not limit our evaluation of Mutual Proximity and local scaling in Section 4.4 to music similarity algorithms, but use 30 public machine learning databases to show the general applicability of the method. Section 4.5 returns to computer music similarity to show that using Mutual Proximity increases the retrieval quality (in terms of k-nearest neighbor accuracy and hubness) of all three introduced similarity algorithms (ME, EP, PS) significantly.

We published the Mutual Proximity method in 2011 [SFSW11]. An adopted version of this chapter was submitted for publication after finishing this thesis.

4.1 Introduction

In a recent publication Radovanović et al. [RNI10] describe the so-called 'hubness' phenomenon and explore it as a general problem of machine learning in high-dimensional data spaces. Hubs are data points which keep appearing unwontedly often as nearest neighbors of a large number of other data points. This effect is particularly problematic in algorithms for similarity search (e.g., similarity-based recommenders), as the same similar objects are found over and over again and other objects are never recommended. The effect has been shown to be a natural consequence of high dimensionality and as such is yet another aspect of the curse of dimensionality [Bel61b].

A direct consequence of the presence of hubs is that a large number of nearest neighbor relations in the distance space are asymmetric, i.e., object y is amongst the nearest neighbors of x but not vice versa. A hub is by definition the nearest neighbor of a large number of objects, but these objects cannot possibly all be the nearest neighbor of the hub. This observation connects the hub problem to methods that attempt to symmetrize nearest neighbor relations, such as 'shared near neighbors' [JP73] and 'local scaling' [ZP05]. While these methods require knowledge of the local neighborhood of every data point, we propose a global variant that combines the idea of 'shared near neighbor' approaches with a transformation of distances to nearest neighbor 'probabilities' to define a concept we call *Mutual Proximity*. The approach is fully unsupervised and transforms an arbitrary distance function to a new probabilistic similarity (distance) measure. Contrary to the local variants, this new approach lends itself to fast approximation even for very large data bases and enables easy combination of multiple distance spaces due to its probabilistic nature.

In experiments with a large number of public machine learning databases we show that both local and global scaling methods lead to:

1. *Significant decrease of hubness*

2. *Increase of k-nearest neighbor classification accuracy*

3. *Strengthening of the pairwise class stability of the nearest neighbors*

To permit other researchers to reproduce the results of this chapter, all databases and the main evaluation scripts have been made publicly available.[1]

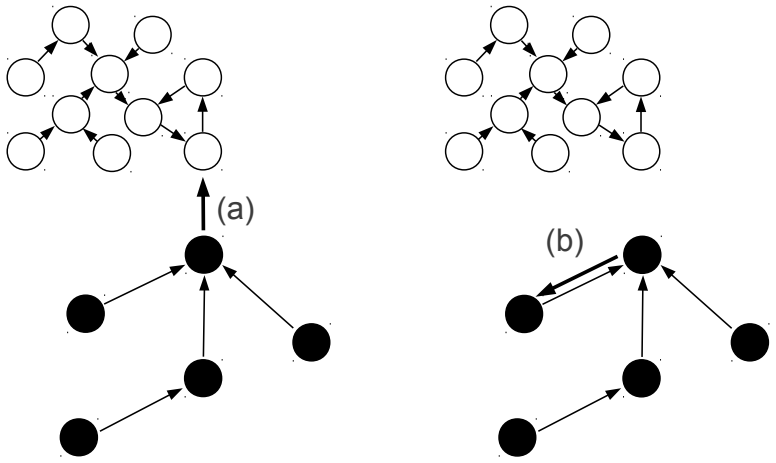
[1] `http://www.ofai.at/~dominik.schnitzer/mp`

4.2 Related Work

The starting point for our investigations are Aucouturier and Pachet [AP04], who found that some songs, according to their audio similarity function, were similar to very many other songs and therefore kept appearing unwontedly often in recommendation lists, preventing other songs from being recommended at all. They called these songs hub songs. The songs that do not appear in any recommendation list have been termed 'orphans'. Similar observations about false positives in music recommendation that are not perceptually meaningful have been made elsewhere [PDW03, FSGP10b, KRNI10]. The existence of the hub problem has also been reported for music recommendation based on collaborative filtering instead of on audio content analysis [Cel08]. Similar effects have been observed in image [DLM$^+$98, HUW05] and text retrieval [RNI10], making this phenomenon a general problem in multimedia retrieval and recommendation.

In the MIR literature, Berenzweig [Ber07] first suspected a connection between the hub problem and the high dimensionality of the feature space. The hub problem was seen as a direct result of the curse of dimensionality [Bel61b], a term that refers to a number of challenges related to to the high dimensionality of data spaces. Radovanović et al. [RNI10] were able to provide more insight by linking the hub problem to the property of *concentration* [FWV07] which occurs as a natural consequence of high dimensionality. Concentration is the surprising characteristic of all points in a high dimensional space to be at almost the same distance to all other points in that space. It is usually measured as a ratio between spread and magnitude, e.g., the ratio between the standard deviation of all distances to an arbitrary reference point and the mean of these distances. If the standard deviation stays more or less constant with growing dimensionality while the mean keeps growing the ratio converges to zero with dimensionality going to infinity. In such a case it is said that the distances concentrate. This has been studied for Euclidean spaces and other ℓ^p norms [AHK01, FWV07]. Radovanović et al. [RNI10] presented the argument that in the finite case, some points are expected to be closer to the center than other points and are at the same time closer, on average, to all other points. Such points closer to the center have a high probability of being hubs, i.e., of appearing in nearest neighbor lists of many other points. Points which are further away from the center have a high probability of being 'orphans', i.e., points that never appear in any nearest neighbor list.

The general setting we are considering here is 'nearest neighbor search' (NNS). NNS is essential in many areas of computer science, such as pattern recognition, multimedia search, vector compression, computational statistics and data mining [SDI06] and, of course, information retrieval and recommendation. It is a well defined task: given an object x find the most similar object in a collection of related objects. In the case of recommendation, the k most similar objects are retrieved with $k << n$ (n being the number of all objects in the data base). Since hubs appear in very many nearest neighbor lists, they tend to render many nearest neighbor relations asymmetric, i.e., a hub y is the nearest neighbor of x, but the nearest neighbor of the hub y is another point a ($a \neq x$). This is because hubs are nearest neighbors to very many data points but only k data points can be nearest neighbors to a hub since the size of a nearest neighbor list is fixed. This behavior is especially problematic in classification or clustering if x and y belong to the same class but a does not, violating what Bennett et al. [BFG99] called the *pairwise stability* of clusters. Radovanović et al. [RNI10] coined the term *bad hubs* for points that show a disagreement of class information for the majority of data points they are nearest



(a) Original nearest neighbor relations

(b) Desired nearest neighbor relations

Figure 4.1: Schematic plot of two classes (black/white filled circles). Each circle has its nearest neighbor marked with an arrow: (a) violates the *pairwise stability* clustering assumption, (b) fulfills the assumption. In many classification and retrieval scenarios, (b) would be the desired nearest neighbor relation for the dataset.

neighbors to. Figure 4.1 illustrates the effect: although a is, in terms of the distance measure, the correct answer to the nearest neighbor query for y, it may be beneficial to use a distance measure that enforces symmetric nearest neighbors. Thus a small distance between two objects should be returned only if their nearest neighbors concur.

This links the hub problem to 'shared near neighbor' (SNN) approaches, which try to symmetrize nearest neighbor relations. The first work to use common near neighbor information dates back to the 1970s. Jarvis and Patrick [JP73] proposed a 'shared near neighbor' similarity measure to improve the clustering of 'non-globular' clusters. As the name suggests, the shared near neighbor (SNN) similarity is based on computing the overlap between the k nearest neighbors of two objects. Shared near neighbor similarity was also used by Ertöz et al. [ESK03] to find the most representative items in a set of objects. Pohle et al. [PKSW06] define a related similarity measure based on the rank of nearest neighbors. They call their method 'proximity verification' and use it to enhance audio similarity search. Jin et al. [JTHW06] use the reverse nearest neighbor (RNN) relation to define a general measure for outlier detection.

Related to SNN approaches are local scaling methods, which use local neighborhood information to rescale distances between data points. The intention is to find specific scaling parameters for each point, to be used to tune the pairwise distances in order to account for different local densities (scales) of the neighborhoods. Local scaling in this sense was first introduced as part of a spectral clustering method by Lihi and Pietro [ZP05]. It transforms arbitrary distances using the distance between object x and its k'th nearest neighbor (see Section 4.3.1 below). In the

context of image retrieval, Jegou et al. [JSHV10] describe a related method called 'contextual dissimilarity measure' (CDM) and show that it reduces the error rates of the retrieval algorithm significantly, observing "*the neighborhood symmetry rate increases*", while at the same time "*the percentage of never seen images decreases*", and in addition that "*the most frequent image is returned 54 times in the first 10 positions with the CDM, against 1062 times using the standard L1 distance*". While they do not explicitly make reference to the notion of hubs, their observations indicate the potential of local distance scaling to mitigate hub-related problems.

4.3 Scaling Methods

In what follows we describe a local scaling method plus our own global variant (which, as we will show, has certain advantages) and evaluate and compare them in Section 4.4.3. All methods we describe require a divergence measure with the following properties:

Definition 1 *Given a non-empty set M with n objects, each element $m_x \in M$ is assigned an index $x = 1 \dots n$. We define a divergence measure $d : M \times M \to \mathbb{R}$ with the following properties:*

- *non-negativity:* $d(m_x, m_y) \geq 0, \quad m_x, m_y \in M,$
- *identity:* $d(m_x, m_y) = 0, \iff m_x = m_y, \quad m_x, m_y \in M,$
- *symmetry:* $d(m_x, m_y) = d(m_y, m_x), \quad m_x, m_y \in M.$

Individual objects $m_x \in M$ are referenced in the text by their index x. The distance between two objects x and y is denoted as $d_{x,y}$.

4.3.1 Local Scaling

Local scaling [ZP05] transforms arbitrary distances to so-called *affinities* (that is, similarities) according to:

$$LS(d_{x,y}) = \exp\left(-\frac{{d_{x,y}}^2}{\sigma_x\,\sigma_y}\right), \tag{4.1}$$

where σ_x denotes the distance between object x and its k'th nearest neighbor. $LS(d_{x,y})$ makes neighborhood relations more symmetric because it includes local statistics of both data points x and y. The exponent in equation 4.1 can be rewritten as ${d_{x,y}}^2/\sigma_x\,\sigma_y = (d_{x,y}/\sigma_x)(d_{x,y}/\sigma_y)$: only when both parts in this product are small will the locally scaled similarity $LS(d_{x,y})$ be high. That is, x and y will be considered close neighbors only if the distance $d_{x,y}$ is small relative to both local scales σ_x and σ_y. Jegou et al. [JHS07] introduce a closely related variant called non-iterative contextual dissimilarity measure (NICDM). Instead of using the distance to the k'th nearest neighbor to rescale the distances, the average distance of the k nearest neighbors is used. This should return more stable scaling numbers and will therefore be used in all our evaluations. The non-iterative contextual dissimilarity measure (NICDM) transforms distances according to:

$$NICDM(d_{x,y}) = \frac{d_{x,y}}{\sqrt{\mu_x \, \mu_y}},$$

where μ_x denotes the average distance to the k nearest neighbors of object x. The iterative version of this algorithm performs the same transformation multiple times until a stopping criterion is met. Since these iterations yield only very minor improvements at the cost of increased computation time, we used the non-iterative version in our evaluations.

4.3.2 Global Scaling - Mutual Proximity

In this section we introduce a global scaling method that is based on: (i) transforming a distance between points x and y into something that can be interpreted as the probability that y is the closest neighbor to x given the distribution of all distances to x in the data base; and (ii) combining these probabilistic distances from x to y and y to x via the product rule. The result is a general unsupervised method to transform arbitrary distance matrices to matrices of probabilistic *mutual proximity* (MP). The first step of transforming distances to probabilities re-scales and normalizes the distances much like a z-transform. The second step combines the probabilities to a mutual measure in a way similar to shared near neighbor approaches. In contrast to local scaling methods MP does not need to determine the local nearest neighbors, but uses global information about the distances in a collection – that is, a distance matrix – which lends itself to fast approximation (see Section 4.3.2.2 below).

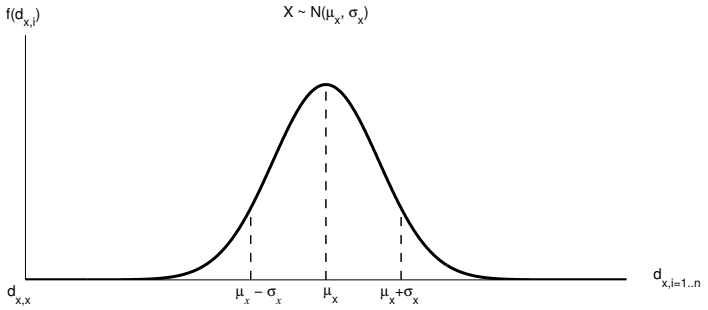
To convert the distances to mutual nearest-neighbor probabilities, in a first step, for each object x the average distance $\hat{\mu}_x$ and the standard deviation $\hat{\sigma}_x$ of its distances $d_{x,i=1\ldots n}$ to all objects in M are computed, in effect estimating a Gaussian distance distribution $X \backsim \mathcal{N}(\hat{\mu}_x, \hat{\sigma}_x)$ for each element x. This is based on the assumption that the distances are normally distributed due to the central limit theorem. The estimated normal distribution X thus models the spread of distances from x to all other elements in M:

$$\hat{\mu}_x = \frac{1}{n}\sum_{i=1}^{n} d_{x,i}, \qquad \hat{\sigma}_x^2 = \frac{1}{n}\sum_{i=1}^{n}(d_{x,i} - \hat{\mu}_x)^2. \tag{4.2}$$

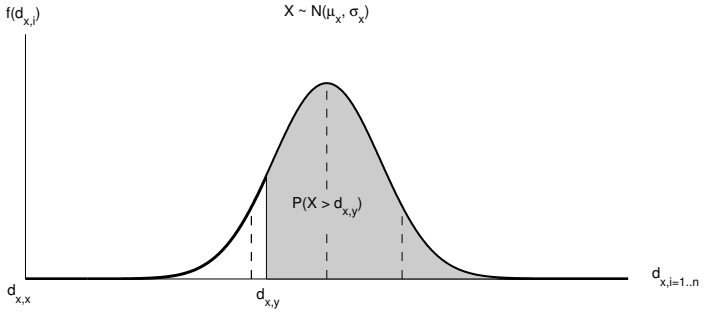
Figure 4.2a shows a schematic plot of the probability density function (pdf) that was estimated for the distances of some object x. The mean distance ($\hat{\mu}_x$) is in the center of the density function. Objects with a small distance to x (that is, objects with high similarity in the original space) find their distance towards the left of the density function. Note that the leftmost possible distance in this sketch is $d_{x,x} = 0$.[2]

By estimating a normal distribution X from the distances $d_{x,i=1\ldots n}$, it is possible to reinterpret any distance $d_{x,y}$ as the probability that y is the nearest neighbor of x, given the distance $d_{x,y}$

[2]Strictly speaking, then, our interpretation of this as a normal distribution is incorrect, since distances < 0 are not possible. However, we find the interpretation useful as a metaphor that helps understand why it makes sense to combine different views as we will do it in this section.



(a) The closer other elements are to x, the more to the left is their distance located on the x-axis of the density function plot. The leftmost possible observation in the data is the distance $d_{x,x} = 0$.



(b) The shaded area shows the probability that y is the nearest neighbor of x based on the distance $d_{x,y}$ and X. The closer y is to x (the smaller $d_{x,y}$) the higher the probability.

Figure 4.2: Schematic plot of the probability density function of a normal distribution which was estimated from the distances $d_{x,i=1\ldots n}$: $X \backsim \mathcal{N}(\hat{\mu}_x, \hat{\sigma}_x)$.

and the normal X (that is, the probability that a randomly drawn element z will have a distance $d_{x,z} > d_{x,y}$):

$$P(X > d_{x,y}) = 1 - P(X \leq d_{x,y}) = 1 - \mathcal{F}_x(d_{x,y}).$$

$\mathcal{F}_x$ denotes the cumulative distribution function (cdf) of the normal distribution defined by X. The probability of an element being a nearest neighbor of x increases the farther left its distance is on the x-axis of the pdf (see Figure 4.2a). To illustrate that, Figure 4.2b plots the probability of y being the nearest neighbor of x given $d_{x,y}$ (the gray filled area).

Transforming all original distances into the probability that any point y is a nearest neighbor of x offers a convenient way to combine this with the opposite view (the probability that x is the nearest neighbor of y) into a single expression. The combined probability should be high only if both views concur, which ensures that neighborhood relations become more symmetric since again local statistics of both data points x and y are included into one overall measure.

Definition 2 *Under the assumption of independence, we compute the probability that y is the nearest neighbor of x given X (the normal defined by the distances $d_{x,i=1\ldots n}$) and x is the nearest neighbor of y given Y (the normal defined by the distances $d_{y,i=1\ldots n}$) and call the resulting probability* ***Mutual Proximity (MP)****:*

$$\begin{aligned} MP(d_{x,y}) &= P(X > d_{x,y} \cap Y > d_{x,y}) \\ &= P(X > d_{x,y}) \cdot P(Y > d_{x,y}), \qquad \forall d_{x,y} > 0. \end{aligned}$$

Clearly, the assumption of independence of $P(X)$ and $P(Y)$ is not warranted in the given scenario. Still, as we will empirically show in the next section, MP has, similarly to local scaling methods, very beneficial effects especially in high dimensional data spaces with high hubness.

4.3.2.1 Related Formulation

Besides the Gaussian view on the distances which was pursued in the original MP formulation, we have found that similar results can also be achieved using a Discrete Uniform Distribution to define the Mutual Proximity. In a discrete uniform distribution each object (distance) has equal probability ($\frac{1}{n}$), thus MP_{uni} can be formulated as:

$$MP_{uni}(d_{x,y}) = \frac{|\{d_{x,i} | d_{x,i} > d_{x,y}\}|}{n} \cdot \frac{|\{d_{y,i} | d_{y,i} > d_{x,y}\}|}{n}, \quad i = 1..n.$$

This formulation is simpler to implement but more expensive to compute, as the cardinalities in the numerators need to be re-estimated for each distance to transform.

4.3.2.2 Approximation of the Normal Distribution Parameters

The computational cost of estimating the normal distribution parameters for MP grows quadratically with the size of the dataset. The original formulation (Definition 2) requires the full distance matrix (that is, all distances) to be computed. Given a fixed data set X^{n-1} (consisting of $n-1$ instances), if we are presented with a new object x_i to analyze (not contained in X^{n-1}), we

would have to compute the full distance matrix over $X^{n-1} \cup \{x_i\}$ in order to then be able to compute MP over this matrix. (This is because MP needs information about all distances to and from x_i.)

To avoid this shortcoming we propose a method that estimates the distance distribution parameters using the centers of a k-means clustering: MP_{approx}. We set the number of clusters to $k = 3$ as to estimate the average distances and their standard deviations a rough clustering should suffice. We use the following nomenclature:

k	...	number of clusters (3),	c_j	...	a cluster found with k-means,
w_j	...	number of objects assigned to c_j,	$v_{j,i}$	...	i^{th} object assigned to c_j.

With the clusters from k-means we estimate the mean ($\hat{\mu}_x$) and standard-deviation ($\hat{\sigma}_x$) for each model m_x using a per-cluster weighted distance:

$$\hat{\mu}_x = \frac{1}{n}\sum_{j=1}^{k} w_j d(m_x, c_j), \qquad \hat{\sigma}_x^2 = \frac{1}{n}\sum_{j=1}^{k} w_j \left(\hat{\mu}_x - d(m_x, c_j)\right)^2. \tag{4.3}$$

The difference to the original estimation of the parameters in Equation 4.2 (Section 4.3.2) is that only a small fraction of distances ($k \times n$) needs to be computed. However using a weighted distance from every object to the centroids (Equation 4.3) is biased, as it systematically underestimates the correct values. To correct the bias we introduce a constant $corr_j$ which we compute for each cluster j.

To do that first we select an object $u_j \in M$ for each cluster which has a median distance to cluster j. We define $corr_j$ for each cluster j as the difference between (a) the average distance of all objects assigned to a cluster to u_j ($\hat{\mu}_{u_j}$) and (b) the average distance of all objects to the cluster center ($\hat{\mu}_j$):

$$corr_j = \hat{\mu}_{u_j} - \hat{\mu}_j = \frac{1}{w_j}\sum_{i=1}^{w_j} d(u_j, v_{j,i}) - \frac{1}{n}\sum_{i=1}^{n} d(c_j, m_i). \tag{4.4}$$

The correction constant $corr_j$ which is computed once for each cluster in Equation 4.4 is added to the per-cluster distance $d(m_x, c_j)$ in Equation 4.3 to compute the distribution parameters required for MP. We evaluate MP_{approx} in Section 4.4.3.1.

4.3.2.3 Linear Combination of Distance Measures

MP yields $[0, 1]$-normalized similarities. Thus, MP transformation can easily be used to linearly combine multiple different distance measures $d1$ and $d2$:

$$d = \omega_1\, MP(d1) + \omega_2\, MP(d2).$$

for some combination weights $\omega_{1,2}$. Similar to a global zero-mean unit-variance normalization, each object's distances are also standardized by their respective mean and standard deviation. Thus, no distance measure can dominate the other in this combination This property is useful in scenarios where multiple different distance measures (describing different aspects of a phenomenon) need to be linearly combined. We will be using MP to linearly combine multiple similarity measures in Section 4.5 with music similarity measures.

4.4 Evaluation

To investigate the effects of using local neighborhood scaling methods and MP, we evaluate them on 30 public machine learning datasets. Each dataset is characterized by the following parameters: name/origin, number of classes, size/number of items n and data dimensionality d. For each dataset we evaluate the original distance space and compare it to the distances that are generated by the local scaling method and by MP.

4.4.1 Benchmarks

To quantify the impact of the two methods, a number of properties and quality measures are computed for the original and the new distances. The characteristics which we compute for each dataset are:

Leave-One-Out k-Nearest Neighbor Classification Accuracy (C^k) We report the k-nearest neighbor (kNN) classification accuracy using leave-one-out classification, where classification is via majority vote among the k nearest neighbors, with the class of the nearest neighbor used for breaking ties. We denote the k-NN accuracy as C^k. In the context of a retrieval problem, higher values would indicate better retrieval quality.

Hubness (S^k) We also compute the *hubness* for each object x according to Radovanović et al. [RNI10]. Hubness is defined as the skewness of the distribution of k-occurrences (i.e., number of k-nearest neighbor lists in which x appears) N_k:

$$S^k = \frac{\mathrm{E}\left[(N_k - \mu_{N_k})^3\right]}{\sigma_{N_k}^3}$$

Positive skewness indicates high hubness, negative values low hubness. The hubness of an entire dataset X is defined as the average S^k over all objects in X.

Goodman-Kruskal Index (I_{GK}) The Goodman-Kruskal Index [GB03] is a clustering quality measure that relates the number of *concordant* (Q_c) and *discordant* (Q_d) tuples ($d_{i,j}$, $d_{k,l}$) of a distance matrix.

- A tuple is concordant if its items i, j are from the same class, items k, l are from different classes and $d_{i,j} < d_{k,l}$.
- A tuple is discordant if its items i, j are from the same class, items k, l are from different classes and $d_{i,j} > d_{k,l}$.
- A tuple is not counted if it is neither concordant nor discordant, that is, if $d_{i,j} = d_{k,l}$.

The Goodman-Kruskal Index (I_{GK}) is defined as:

$$I_{GK} = \frac{Q_c - Q_d}{Q_c + Q_d}.$$

I_{GK} is bounded to the interval $[-1, 1]$, and the higher I_{GK}, the more concordant and fewer discordant quadruples are present in the dataset. Thus a large index value indicates a good clustering (in terms of *pairwise stability* – see section 4.2).

Other indices to compare clustering algorithms like the classic Dunn's Index or Davies-Bouldin Index [BP98] cannot be used here as their values do not allow a comparison across different distance measures.

Intrinsic Dimensionality (d_{mle}) To further characterize each dataset we compute an estimate of the intrinsic dimensionality of the feature spaces. Whereas the embedding dimension is the actual number of dimensions of a feature space, the intrinsic dimension is the – often much smaller – number of dimensions necessary to represent a data set without loss of information. It has also been demonstrated that hubness depends on the intrinsic rather than embedding dimensionality [RNI10]. We use the maximum likelihood estimator proposed by Levina and Bickel [LB05], which was also used by Radovanović et al. [RNI10] to characterize the datasets.

Percentage of symmetric neighborhood relations We call a nearest neighbor relation between two points x and y 'symmetric' if the following holds: object x is a nearest neighbor of y if and only if y is also the nearest neighbor of x. As both examined methods aim at symmetrizing neighborhood relations, we report the percentage of symmetric relations at different neighborhood sizes k.

4.4.2 Public Machine Learning Datasets

We evaluate the proposed method by applying it to 30 different public machine learning datasets. The datasets include problems from the general machine learning field, and the bio-medical, image, text and music retrieval domains. We use the following datasets:

- The UCI Machine Learning Repository[3] (*UCI*, see [FA10]) datasets: *arcene*, *gisette*, *mfeat-pixels/karhunen/factors*, *dexter*, *mini-newsgroups*, *dorothea*, *reuters-transcribed*.
- The Kent Ridge bio-medical datasets[4] (*KR*): *amlall*, *lungcancer* and *ovarian-61902*.
- The LibSVM datasets[5] (*LibSVM*, see [CL01]): *australian*, *diabetes*, *german numbers*, *liver-disorders*, *breast-cancer*, *duke (train)*, *heart*, *sonar*, *colon-cancer*, *fourclass*, *ionosphere*, *splice*.
- The Music Information Retrieval Evaluation eXchange[6] (MIREX) datasets (*Mirex*, see [Dow08]): *ballroom* and *ismir2004*.
- Two music artist web pages and tweets datasets[7] (*CP*, see [Sch10]): *c1ka-twitter* and *c224a-web*.

[3] http://archive.ics.uci.edu/ml/
[4] http://datam.i2r.a-star.edu.sg/datasets/krbd/
[5] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/
[6] http://www.music-ir.org/mirex
[7] http://www.cp.jku.at/people/schedl/datasets.html

For the general machine learning datasets from the statistical or biological domains no feature extraction is necessary. The feature vectors can be downloaded directly from the respective repositories. However, before using the features we standardized the datasets with their collection mean and variance. These general machine learning datasets use the standard Euclidean distance (denoted as ℓ^2) as similarity measure.

The text retrieval datasets (*reuters-transcribed*, *c224a-web*, *movie-reviews*, *dexter*, *mini-newsgroups*, *c1ka-twitter*[8]) need to be preprocessed before evaluating them. To this end we employ stop-word removal and stemming. They are transformed into the bag-of-words representation, and standard $tf \cdot idf$ (term frequency · inverse document frequency) weights are computed (see for example [BYRN99b]). The word vectors are normalized to the average document length. Individual document vectors are compared with the cosine distance (denoted as *cos*).

For the image retrieval dataset (*corel*) normalized color histograms are computed as features. They show reasonable classification performance despite their simplicity, as Chapelle et al. [CHV99] show. The three 64-dimensional color histograms are concatenated into a single vector and compared using the Euclidean distance (ℓ^2).

To extract the features for the two music information retrieval datasets (*ismir2004*, *ballroom*) we use the ME and the symmetrized Kullback-Leibler divergence (denoted with *skl*).

4.4.3 Results

Tables 4.1 and 4.2 show the results of the evaluations conducted on the afore introduced 30 public datasets. The collections have very diverse properties. There are collections like *fourclass* or *liver-disorders* with very low dimensionality ($d = 2$ and $d = 6$), as well as datasets with very high dimensionality, such as *dorothea* ($d = 100\,000$) or *c1ka-twitter* ($d = 49\,820$). Related to that, column d_{mle} lists the intrinsic dimensionality according to the maximum likelihood estimator. Using the intrinsic dimensionality estimate we can see that there are datasets where the data is originally represented using high-dimensional feature vectors, although the data's intrinsic dimensionality is quite low. For example the *ovarian_61902* dataset has an embedding dimension of $d = 15\,154$ but its estimated intrinsic dimension is only $d_{mle} = 9$.

The evaluation results in Tables 4.1 and 4.2 are sorted by the hubness $S^{k=5}$ of the original distance space (printed in bold). In subsequent plots individual collections are referenced by their numbers as given in Tables 4.1 and 4.2. The columns $C^{k=1}/C^{k=5}$ show the k-nearest neighbor classification rates of the collections. The classification rates with the original distances, the local scaling (NICDM) and the global scaling (MP) are documented. For convenience the column +/- shows the difference in classification accuracy, in terms of absolute percentage points, between the original distances and NICDM/MP. All improvements compared to the original distances are printed in bold. Statistically significant differences are marked with an asterisk (McNemar's test, $df = 1$, $\alpha = .05$ error probability, see e.g., Salzberg [Sal97]).

Looking at the tables a first observation is that very high-dimensional data sets (in terms of their intrinsic as well as their actual embedding dimensionality) also tend to have high hubness. This is in agreement with the results of Radovanović et al. [RNI10] and the theory that hubness is a consequence of high dimensionality.

[8]Set *c1ka-twitter* equals *c3ka-twitter* from *CP*, omitting artists classified as 'rock' to balance the data.

Name/Src.	Cls.	n	d	d_{mle}	Dist.	$C^{k=1}$	+/-	$C^{k=5}$	+/-	$S^{k=5}$	I_{GK}
fourclass (sc)	2	862	2	2	ℓ^2	100%		100%		**0.15**	0.22
1. *LibSVM*					NICDM	100%	0	100%	0	0.06	0.21
					MP	100%	0	100%	0	0.07	0.22
liver-disorders (sc)	2	345	6	6	ℓ^2	62.6%		60.6%		**0.39**	0.00
2. *UCI*					NICDM	63.2%	**0.6**	65.8%	**5.2***	-0.04	0.03
					MP	61.7%	-0.9	62.6%	**2.0**	0.46	0.01
australian	2	690	14	3	ℓ^2	65.5%		68.8%		**0.44**	0.13
3. *LibSVM*					NICDM	65.7%	**0.2**	69.4%	**0.6**	-0.09	0.14
					MP	65.1%	-0.4	69.6%	**0.8**	0.41	0.14
diabetes (sc)	2	768	8	6	ℓ^2	70.6%		74.1%		**0.49**	0.20
4. *UCI*					NICDM	69.8%	-0.8	74.1%	0	0.04	0.15
					MP	70.2%	-0.4	73.7%	-0.4	0.22	0.19
heart (sc)	2	270	13	7	ℓ^2	75.6%		80.0%		**0.50**	0.35
5. *LibSVM*					NICDM	75.9%	**0.3**	79.3%	-0.7	-0.00	0.27
					MP	75.9%	**0.3**	80.7%	**0.7**	0.11	0.36
ovarian-61902	2	253	15 154	10	ℓ^2	95.3%		93.7%		**0.66**	0.20
6. *KR*					NICDM	95.7%	**0.4**	93.3%	-0.4	-0.10	0.19
					MP	94.5%	-0.8	93.7%	0	-0.03	0.20
breast-cancer (sc)	2	683	10	5	ℓ^2	95.6%		97.4%		**0.71**	0.89
7. *LibSVM*					NICDM	95.8%	**0.2**	97.1%	-0.3	0.19	0.42
					MP	95.3%	-0.3	97.2%	-0.2	0.44	0.91
arcene	2	100	10 000	23	ℓ^2	81.0%		77.0%		**0.78**	0.02
8. *UCI*					NICDM	80.0%	-1.0	74.0%	-3.0	0.31	0.06
					MP	81.0%	0	75.0%	-2.0	0.54	0.04
mfeat-factors	10	2 000	216	7	ℓ^2	95.0%		94.7%		**0.79**	0.71
9. *UCI*					NICDM	94.8%	-0.2	94.7%	0	0.15	0.76
					MP	94.7%	-0.3	94.7%	0	0.43	0.77
colon-cancer	2	62	2 000	11	ℓ^2	72.6%		77.4%		**0.81**	0.19
10. *LibSVM*					NICDM	69.4%	-3.2	82.3%	**4.9**	0.08	0.18
					MP	69.4%	-3.2	83.9%	**6.5**	-0.08	0.20
ger.num (sc)	2	1 000	24	8	ℓ^2	67.5%		71.7%		**0.81**	0.07
11. *LibSVM*					NICDM	66.8%	-0.7	72.0%	**0.3**	0.32	0.03
					MP	67.8%	**0.3**	70.9%	-0.8	0.34	0.07
amlall	2	72	7 129	32	ℓ^2	91.7%		93.1%		**0.83**	0.31
12. *KR*					NICDM	93.1%	**1.4**	97.2%	**4.1**	0.56	0.33
					MP	91.7%	0	94.4%	**1.3**	-0.00	0.35
mfeat-karhunen	10	2 000	64	15	ℓ^2	97.4%		97.4%		**0.84**	0.76
13. *UCI*					NICDM	97.2%	-0.2	97.6%	**0.2**	0.27	0.74
					MP	97.0%	-0.4	97.3%	-0.1	0.48	0.79
lungcancer	2	181	12 533	60	ℓ^2	98.9%		100.0%		**1.07**	0.56
14. *KR*					NICDM	99.4%	**0.5**	98.9%	-1.1	0.31	0.50
					MP	100.0%	**1.1**	99.4%	-0.6	0.21	0.55
c224a-web	14	224	1 244	41	cos	86.2%		89.3%		**1.09**	0.79
15. *CP*					NICDM	87.9%	**1.7**	92.4%	**3.1***	0.42	0.89
					MP	87.5%	**1.3**	92.0%	**2.7**	0.51	0.89

Table 4.1: Evaluation results of public machine learning datasets ordered by ascending hubness $S^{k=5}$ of the original distance space. Results are given for data sets with small hubness, see section 4.4.3 for details.

Name/Src.	Cls.	n	d	d_{mle}	Dist.	$C^{k=1}$	+/-	$C^{k=5}$	+/-	$S^{k=5}$	I_{GK}
mfeat-pixels	10	2 000	240	12	ℓ^2	97.4%		97.8%		**1.18**	0.73
16. *UCI*					NICDM	97.2%	-0.2	97.3%	-0.5	0.24	0.74
					MP	97.4%	0	97.1%	-0.7*	0.44	0.78
duke (train)	2	38	7 129	16	ℓ^2	73.7%		68.4%		**1.37**	0.02
17. *UCI*					NICDM	81.6%	**7.9**	68.4%	0	0.43	0.06
					MP	76.3%	**2.6**	68.4%	0	0.43	0.04
corel1000	10	1 000	192	9	ℓ^2	70.7%		65.2%		**1.45**	0.33
18. *Corel*					NICDM	72.9%	**2.2***	72.0%	**6.8***	0.39	0.47
					MP	71.1%	**0.4**	67.6%	**2.4***	0.68	0.50
sonar (sc)	2	208	60	11	ℓ^2	87.5%		82.2%		**1.54**	0.07
19. *UCI*					NICDM	87.0%	-0.5	87.0%	**4.8**	0.47	0.08
					MP	89.9%	**2.4**	85.1%	**2.9**	0.50	0.09
ionosphere (sc)	2	351	34	13	ℓ^2	86.9%		85.5%		**1.55**	0.31
20. *UCI*					NICDM	92.3%	**5.4***	94.3%	**8.8***	0.28	0.07
					MP	92.6%	**5.7***	90.3%	**4.8***	0.87	0.26
reuters-transcribed	10	201	2 730	70	cos	44.3%		49.3%		**1.61**	0.38
21. *UCI*					NICDM	45.3%	**1.0**	52.7%	**3.4**	0.63	0.32
					MP	44.8%	**0.5**	50.7%	**1.4**	0.86	0.44
ballroom	8	698	820	12	*skl*	54.3%		48.1%		**2.98**	0.15
22. *Mirex*					NICDM	57.2%	**2.9**	51.6%	**3.5***	1.09	0.20
					MP	56.2%	**1.9**	53.3%	**5.2***	1.16	0.19
ismir2004	6	729	820	25	*skl*	80.4%		74.1%		**3.20**	0.37
23. *Mirex*					NICDM	83.8%	**3.4***	79.0%	**4.9***	0.77	0.21
					MP	82.6%	**2.2***	77.4%	**3.3***	1.09	0.42
movie-reviews	2	2 000	10 382	28	cos	71.1%		75.7%		**4.07**	0.05
24. *PaBo*					NICDM	72.0%	**0.9**	76.0%	**0.3**	1.22	0.07
					MP	72.5%	**1.4**	76.1%	**0.4**	0.85	0.07
dexter	2	300	20 000	161	cos	80.3%		80.3%		**4.22**	0.10
25. *UCI*					NICDM	84.3%	**4.0**	86.0%	**5.7***	2.02	0.13
					MP	84.3%	**4.0**	88.0%	**7.7***	1.83	0.14
gisette	2	6 000	5 000	149	ℓ^2	96.0%		96.3%		**4.48**	0.16
26. *UCI*					NICDM	97.2%	**1.2***	98.1%	**1.8***	0.78	0.20
					MP	97.3%	**1.3***	97.9%	**1.6***	0.89	0.21
splice (sc)	2	10 00	60	27	ℓ^2	69.6%		69.4%		**4.55**	0.07
27. *LibSVM*					NICDM	73.3%	**3.7***	79.3%	**9.9***	1.51	0.11
					MP	72.5%	**2.9**	77.8%	**8.4***	0.57	0.09
mini-newsgroups	20	2 000	8 811	188	cos	64.4%		65.6%		**5.14**	0.47
28. *UCI*					NICDM	67.2%	**2.8***	68.5%	**2.9***	1.32	0.52
					MP	67.7%	**3.3***	68.3%	**2.7***	1.01	0.58
dorothea	2	800	100 000	201	ℓ^2	90.6%		90.2%		**12.91**	0.21
29. *UCI*					NICDM	92.2%	**1.6**	93.0%	**2.8***	11.72	0.21
					MP	92.4%	**1.8**	92.6%	**2.4***	10.26	0.21
c1ka-twitter	17	969	49 820	46	cos	31.9%		26.6%		**14.63**	0.08
30. *CP*					NICDM	47.8%	**15.9***	53.0%	**26.4***	2.94	0.33
					MP	48.7%	**16.8***	50.5%	**23.9***	3.39	0.16

Table 4.2: Evaluation results of public machine learning datasets ordered by ascending hubness $S^{k=5}$ of the original distance space. Results are given for data sets with large hubness, see section 4.4.3 for details.

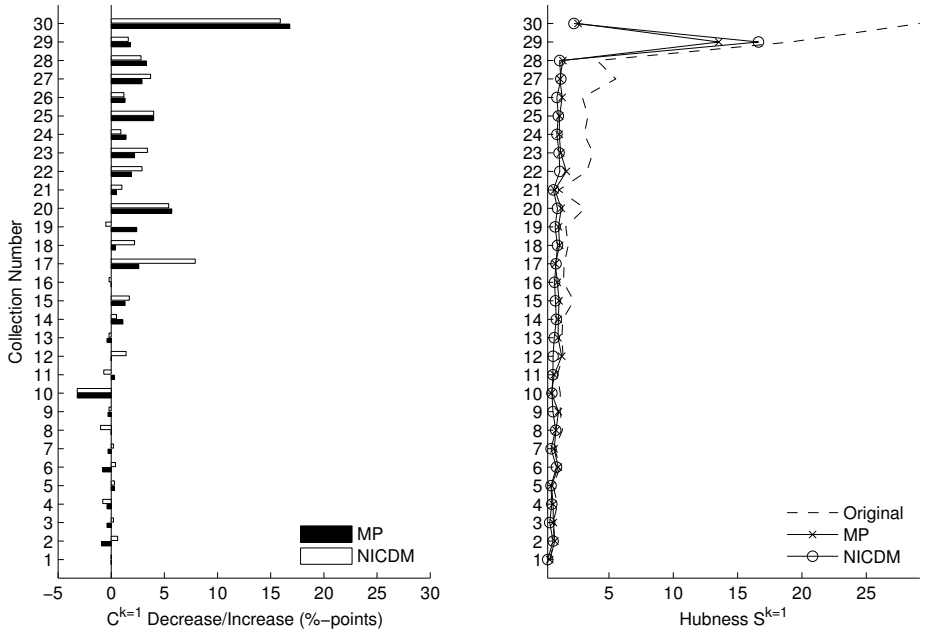


Figure 4.3: Improvements in accuracy (absolute percentage points) and hubness evaluated with $k = 1$

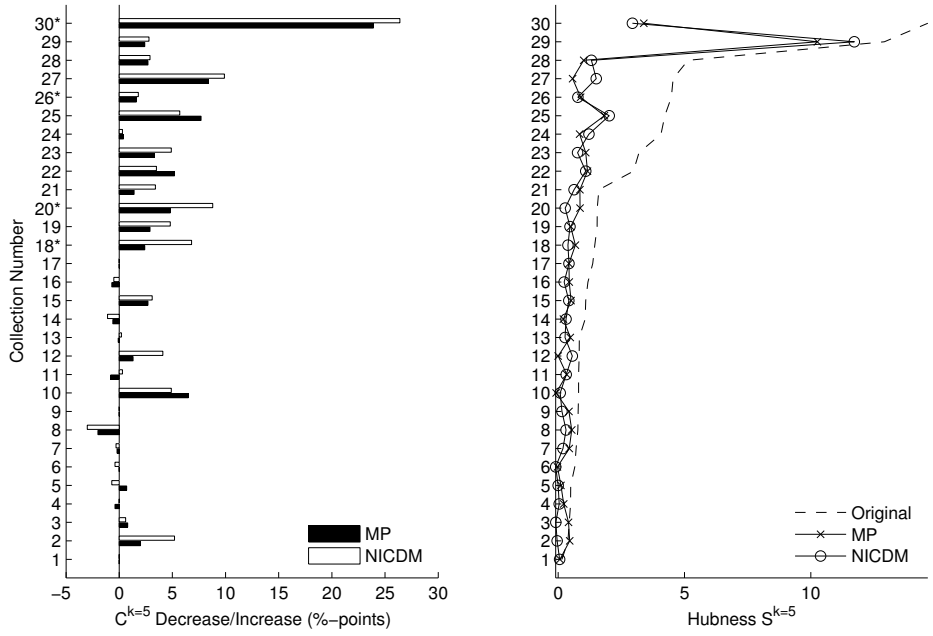


Figure 4.4: Improvements in accuracy (absolute percentage points, significant differences marked with an asterisk) and hubness evaluated with $k = 5$

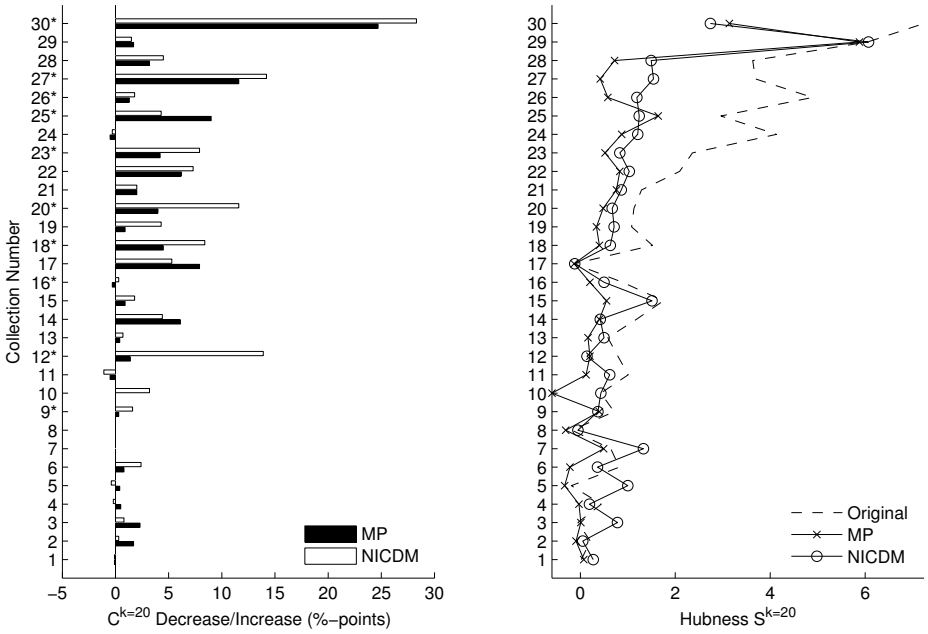


Figure 4.5: Improvements in accuracy (absolute percentage points, significant differences marked with an asterisk) and hubness evaluated with $k = 20$

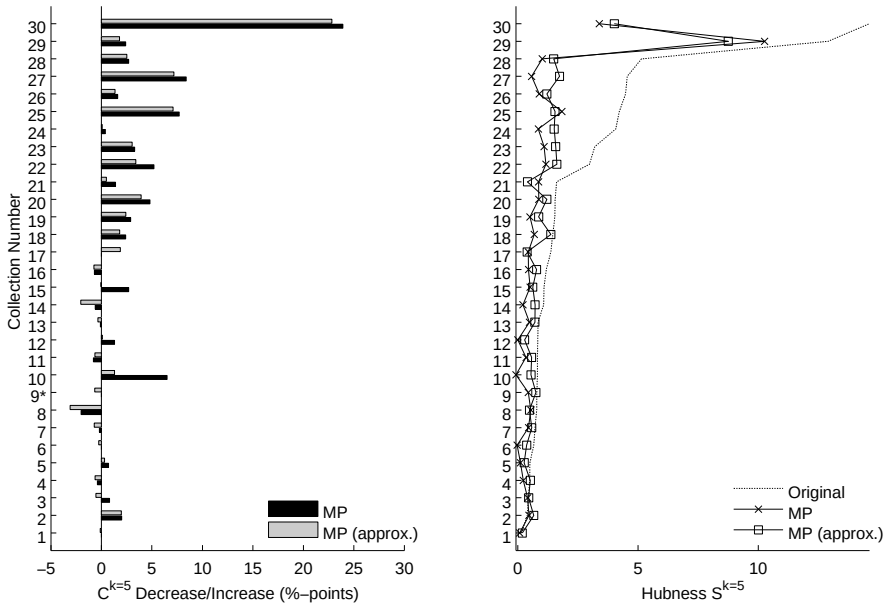


Figure 4.6: Improvements in accuracy (absolute percentage points, significant differences marked with an asterisk) and hubness evaluated with $k = 5$ for MP (black) and its approximative variant MP_{approx} (gray).

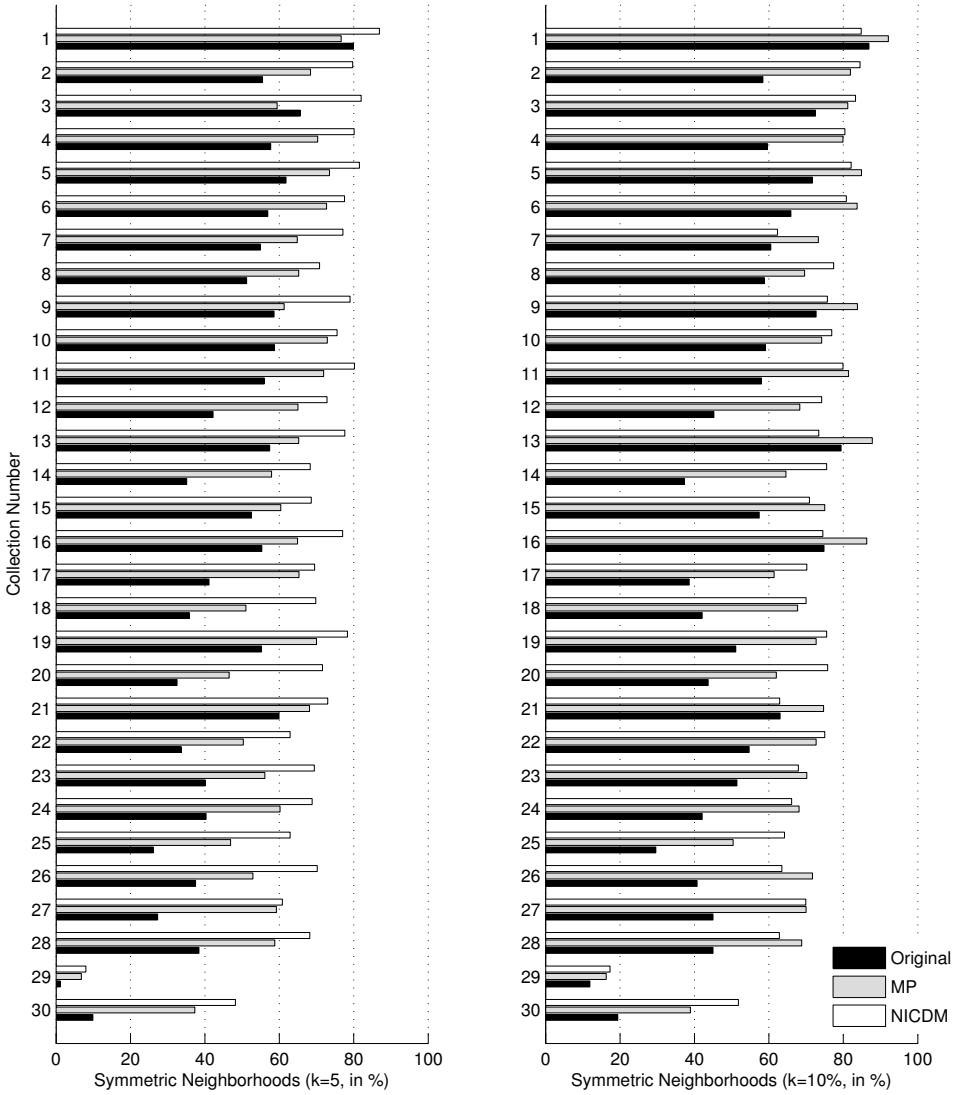


Figure 4.7: Percentage of symmetric neighborhood relations at $k = 5$ and $k = 10\%$ of the respective collection size.

By looking at the classification rates (columns $C^{k=1}$ and $C^{k=5}$) it can also clearly be observed that the higher the hubness and intrinsic dimensionality, the more beneficial, in terms of classification accuracy, is NICDM and MP. For datasets with high hubness (in the collections we used, a value above 1.4 seems to be a limit) the increase in classification rates is notable. For $C^{k=1}$, the accuracy gain ranges from rather moderate 1 to up to $7-8$ percentage points, and in the case of *c1ka-twitter* it is 15.9 percentage points for NICDM and 16.8 percentage points for MP. For $C^{k=5}$ the trend is even clearer. Whereas only three changes in accuracy (relative to the original distances) are significant for data sets with low hubness ($S^{k=5} <= 1.4$, data sets 1 – 17), 32 changes in accuracy are significant for data sets with high hubness ($S^{k=5} > 1.4$, data sets 18 –30). There is only a single and rather small negative change (*mfeat-pixels*, distance based on MP, -0.7%) that is statistically significant.

Figures 4.3, 4.4 and 4.5 (left hand sides) present these results in bar plots where the y-axis shows the index of the data sets (ordered according to hubness as in Tables 4.1 and 4.2) and the bars show the increase or decrease of classification rates. The bar plots also directly show how MP compares to NICDM in terms of classification accuracy for $k = 1, 5, 20$. Generally speaking, results for MP and NICDM are very comparable especially for lower values of k. As for $k = 1$, MP and NICDM perform equally well and there is no statistically significant difference between MP and NICDM (McNemar's test, $df = 1$, $\alpha = .05$ error probability). Based on the same statistical testing procedure, results for NICDM and $k = 5$ are significantly better than for MP for data sets 18, 20, 26 and 30 (marked with asterisks in Figure 4.4). Results for NICDM and $k = 20$ are significantly better than for MP for 9 out of 30 data sets and MP is superior to NICDM for data set 25 (all significant differences are marked with asterisks in Figure 4.5). Despite these significant differences, the general tendency of both MP and NICDM is comparable in the sense that if there is an improvement compared to the original distances, it can be seen for both MP and NICDM.

Another observation from the results listed in Tables 4.1 and 4.2 is that both NICDM and MP reduce the hubness of the distance space for all data sets to relatively low values. The hubness $S^{k=5}$ decreases from an average value of 2.5 (original) to 0.95 (MP) and 0.94 (NICDM), indicating a well balanced distribution of nearest neighbors. The impact of MP and NICDM on the hubness per data set is plotted in Figures 4.3, 4.4 and 4.5 (right hand sides). It can be seen that both MP and NICDM lead to lower hubness (measured for $S^{k=1,5,20}$) compared to the original distances. The effect is more pronounced for data sets having large hubness values according to the original distances.[9]

More positive effects in the distances can also be seen in the increase of concordant (see Section 4.4 for the definition) distance quadruples indicated by higher Goodman-Kruskal index values (I_{GK}). This index improves or remains unchanged for 26 out of 30 data sets in the case of using MP. The effect is not so clear for NICDM, which improves the index or leaves it unchanged for only 17 out of 30 data sets. The effect of NICDM on I_{GK} is especially unclear for data with low hubness (data sets 1 – 17).

Finally we also checked whether both MP and NICDM are able to raise the percentage of symmetric neighborhood relations. Results for $k = 5$ and k set to 10% of the collection size (denoted with $k = 10\%$) are shown in Figure 4.7. As can be seen, with the single exception of

[9] A notable exception is dataset 29 ("dorothea") where the reduction in hubness is not so pronounced. This may be due to the extremely skewed distribution of its two classes (9:1).

data set 30, the symmetry for all data sets with both MP and NICDM increases. The average percentage of symmetric neighborhoods across all data sets for $k = 5$ is 45 for the original distances, 60 for MP and 70.5 for NICDM. The numbers for $k = 10\%$ are 53 (original), 71.1 (MP) and 70.9 (NICDM).

4.4.3.1 Mutual Proximity Approximation

We also compared MP and its approximation MP_{approx} (see Section 4.3.2.2) regarding their performance in terms of accuracy and hubness. Results for MP_{approx} depicted in Figure 4.6 are averages over ten approximations. Accuracy results are very comparable, there is only one statistically significant difference (data set 9, McNemar's test, $df = 1$, $\alpha = .05$ error probability). The decrease in hubness is also equally strong for MP and MP_{approx}.

Figure 4.8 shows how well the approximation of the parameters has worked in the experiment. We can see that the mean ($\hat{\mu}$) parameter can be estimated very well. In contrast to the standard deviation ($\hat{\sigma}$) parameter which is in many cases estimated off by more than 30%. A more accurate approximation of parameters can be obtained if the number of clusters is increased. However, as hubness and classification rates are already very close to the original MP method, the proposed parameter estimation with only three clusters already seems to suffice.

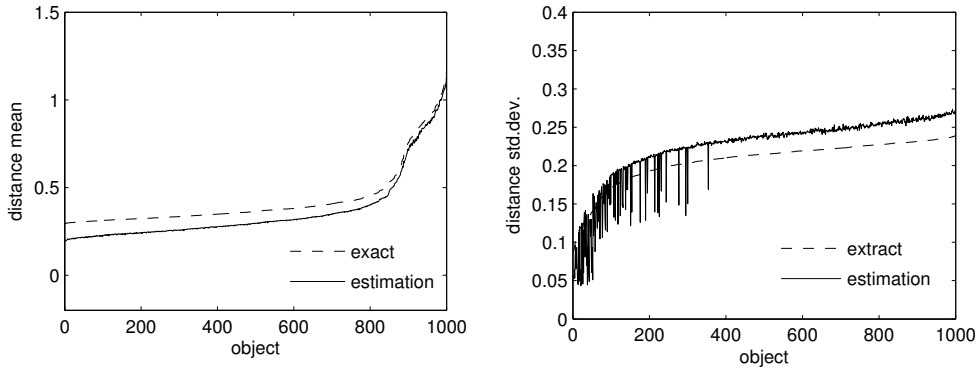
4.4.4 Summary of results

Our main result is that both global (MP) and local (NICDM) scaling show very beneficial effects concerning hubness on data sets that exhibit high hubness in the original distance space. Both methods are able to decrease the hubness, raise classification accuracy and improve other indicators like percentage of concordant distance quadruples or symmetric neighborhood relations. In case of MP, its approximation MP_{approx} is able to perform at equal level with substantially less computational cost ($O(n)$, as opposed to $O(n^2)$ for both MP and local scaling). As a global approximation of scaling values is not possible for NICDM, the approximation of MP will be essential if it is used in large databases.

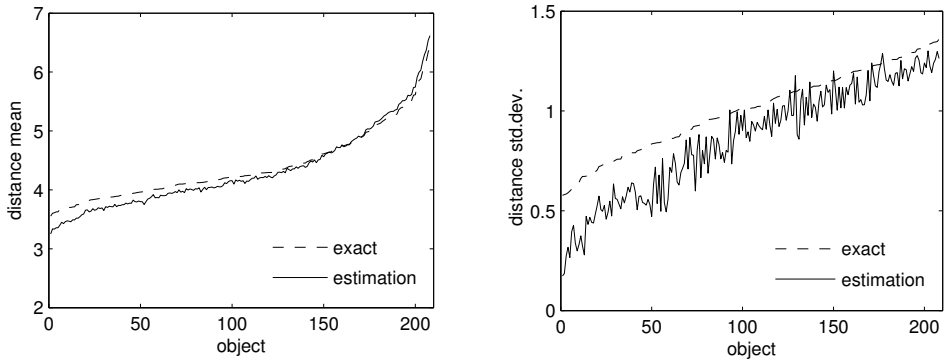
For data sets exhibiting low hubness in the original distance space, improvements are much smaller or non-existent but there is no degradation of performance.

4.5 Mutual Proximity and Content-Based Music Similarity

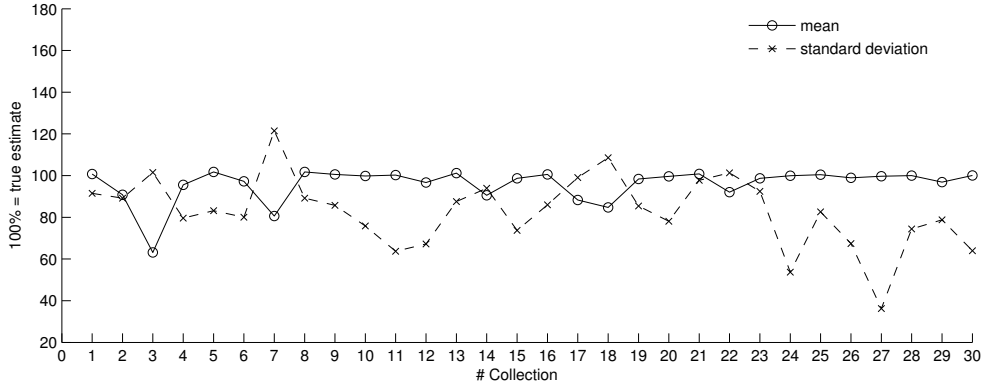
This section uses Mutual Proximity, its approximation (Section 4.3.2.2) and the linear combination (Section 4.3.2.3) of multiple similarity measures with the music similarity algorithms investigated throughout this thesis. The features which are used in the music similarity algorithms are of high dimensionality and exhibit all very high hubness in their nearest neighbors. The high hubness of the music similarity algorithms has a notable impact on the retrieval quality of the algorithms: the same songs are retrieved for a large number of query songs (hubs) and simultaneously those hub songs push out other songs from the nearest neighbor lists so that these songs are never recommended at all.



(a) Corel 1000 database



(b) Sonar database



(c) All databeases

Figure 4.8: Plot (a) and (b) show a visualization of the approximation of the normal distribution parameters on two selected databases. The objects are sorted by the exact values to compare the estimation with the exact value. A perfect estimate would overlap with the exact values. Plot (c) displays the averaged mean and standard deviation estimates for all databases.

Collection	*Orphaned* ME	EP
ISMIR 2004 (Train)	16.5%	15.1%
ISMIR 2004 (Full)	16.3%	12.7%
GTzan	15.2%	12.5%
Homburg	31.5%	25.3%
1517 Artists	25.2%	20.1%
Ballroom Dataset	21.2%	15.8%
Cretan Dances	18.9%	14.4%
Latin Music Database	11.3%	9.1%
Popular Rhythms	28.5%	25.9%

Table 4.3: Percentage of orphaned songs in the different databases in the $k = 5$ nearest neighbor lists.

To illustrate this particularly aspect we compute the number of songs which are never retrieved in a music recommendation system using the top 5 nearest neighbors. Table 4.3 shows the percentage of songs of each collection which are never found in the top 5 nearest neighbors. These songs are orphaned. From the table we can see that the percentage of orphaned songs ranges from 9.1% to 31.5%, i.e., in the most extreme case of the *Homburg* collection we can see that over 31% of the songs are never occurring as nearest neighbors of any song. Thus a music recommendation system relying solely on these techniques would never recommend almost a third of the songs available in the database. This points to a severe limit of the performance of a system which was designed to discover new music. We have published an analysis of the limitations of a content-based music recommendation service where these problematic aspects are studied in a real application [FGS10]. We found that hubness can be reduced in linear combinations of multiple independent similarity measures. Accordingly we observe in Table 4.3 that EP (a linear combination of ME similarity with a rhythmic similarity) exhibits lower numbers of orphaned objects in all collections. But still, the number of orphaned songs is very high.

The remainder of this section investigates if MP can also improve the retrieval quality of the music similarity algorithms which are affected by high hubness. We measure the same characteristics as we have done in the previous section. To study the effects of using MP with the selected music similarity algorithms we evaluate them in terms of their hubness and classification/retrieval rates and finally show how the orphaned objects evolved.

4.5.1 Algorithms

Using MP with one of the content-based music similarity algorithms is straight-forward. Each algorithm is discussed shortly in the following paragraph. To use the MP approximation (MP_{approx}), we use the approximation routine as it was defined in Section 4.3.2.2. The k-means routine for Gaussians and Kullback-Leibler divergences (SKL, JS) which is required in all algorithms the approximation was defined in Chapter 3.

Collection	Algorithm	Artist Filter	Increase in C^k with MP 1	5	10	20
ISMIR 2004 (Train)	ME	no	**2.2**	**0.4**	**3.6**	**0.0**
	ME (JS)	no	**3.4**	**1.2**	**4.0**	**1.8**
	EP	no	**1.1**	**2.6**	**2.5**	**0.7**
	ME	yes	**1.0**	**1.7**	**1.4**	**1.2**
	ME (JS)	yes	**3.2**	**2.3**	**1.9**	**1.3**
	EP	yes	**1.2**	**1.2**	**1.5**	**1.5**
ISMIR 2004 (Full)	ME	no	**1.9**	**2.4**	**4.7**	**1.9**
	ME (JS)	no	**3.7**	**2.4**	**2.8**	**2.9**
	EP	no	**1.6**	-0.5	**0.5**	**3.2**
GTzan	ME	no	**3.7**	**6.7**	**0.6**	**4.3**
	ME (JS)	no	**3.4**	**6.2**	**6.6**	**2.2**
	EP	no	**1.8**	**2.8**	**0.9**	**0.6**
Homburg	ME	no	**4.5**	**1.3**	**3.8**	**1.5**
	ME (JS)	no	**3.8**	**2.9**	**0.7**	**2.0**
	EP	no	**2.9**	**0.8**	**2.8**	**1.5**
	ME	yes	**4.3**	**2.8**	**3.1**	**2.5**
	ME (JS)	yes	**3.6**	**3.0**	**3.0**	**3.1**
	EP	yes	**2.7**	**2.3**	**2.4**	**2.4**
1517 Artists	ME	no	**2.4**	**2.9**	**2.3**	**1.8**
	ME (JS)	no	**3.8**	**3.7**	**2.8**	**2.2**
	EP	no	**2.0**	**3.0**	**1.9**	**1.6**
	ME	yes	**1.3**	**2.2**	**2.2**	**2.0**
	ME (JS)	yes	**1.9**	**2.8**	**2.3**	**2.4**
	EP	yes	**1.5**	**2.1**	**1.7**	**1.8**
Ballroom Dataset	ME	no	**2.0**	**3.9**	**4.7**	**2.0**
	ME (JS)	no	**5.0**	**5.4**	**9.7**	**2.4**
	EP	no	**3.4**	**4.6**	**1.4**	**3.6**
Cretan Dances	ME	no	**1.7**	**10.6**	-3.3	-4.4
	ME (JS)	no	**7.2**	**6.1**	**2.8**	-1.7
	EP	no	**7.2**	**0.0**	**4.4**	**3.9**
Latin Music Database	ME	no	**1.1**	**2.1**	**2.9**	**3.4**
	ME (JS)	no	**1.0**	**2.1**	**4.2**	**5.1**
	EP	no	**0.4**	**1.7**	**2.3**	**1.8**
	ME	yes	**3.5**	**4.6**	**3.4**	**3.9**
	ME (JS)	yes	**3.6**	**4.3**	**5.0**	**5.2**
	EP	yes	**3.6**	**2.3**	**2.5**	**2.2**
Popular Rhythms	ME	no	**2.9**	**3.2**	**1.7**	**8.1**
	ME (JS)	no	**6.9**	-0.3	**4.9**	-2.6
	EP	no	**0.6**	-1.2	**5.2**	**1.4**

Table 4.4: Increase in Classification Accuracies with MP, given in percentage-points

Mandel-Ellis (ME)

MP can directly be used with the ME music similarity algorithm as the similarity is a single divergence computed using either the symmetrized Kullback-Leibler (SKL) divergence or the Jensen-Shannon divergence (JS). We have already introduced these divergences in Chapter 3 and we just apply the MP method (either approximated or not) to the divergence (denoted as d_t):

$$ME_{MP} = MP(d_t). \tag{4.5}$$

In its original implementation ME uses the SKL divergence, but in the evaluation we also evaluate the algorithm with the JS divergence.

Elias Pampalk (EP)

In contrast to ME, EP uses a linear combination of multiple different similarity measures in its similarity measure. The SKL is used to compute similarity for the music timbre component d_t and the Euclidean distance is used to compute similarity between the three rhythm components (d_{FP}, d_{FPg}, d_{FPb}). In the original algorithm precomputed normalization values for each component are used to allow linearly combining the four measures. When using MP to linearly the normalization is not necessary to combine the measures. MP already normalizes the similarities. So the transformed measure looks like this:

$$EP_{MP} = 0.7\,MP(d_t) + 0.1\,MP(d_{FP}) + 0.1\,MP(d_{FPg}) + 0.1\,MP(d_{FPb}). \tag{4.6}$$

To use the MP approximation the k-means clustering has to be done for each component. Each similarity measure requires its own MP parameter estimation.

Pohle-Schnitzer (PS)

PS already uses a preliminary variant of MP. In the following experiments we will benchmark all other algorithms with MP against PS. PS uses MP in its linear combination of timbre (d_t) and rhythm (d_r) similarities:

$$PS_{MP} = 0.7\,MP(d_t) + 0.3\,MP(d_r). \tag{4.7}$$

4.5.2 Evaluation

To evaluate the impact of MP on the content based music similarity algorithms we look at their change in hubness and classification accuracy. In all forthcoming evaluations we use MP_{approx} and report results averaged over 10 independent runs. To compute MP_{approx} we used three clusters.

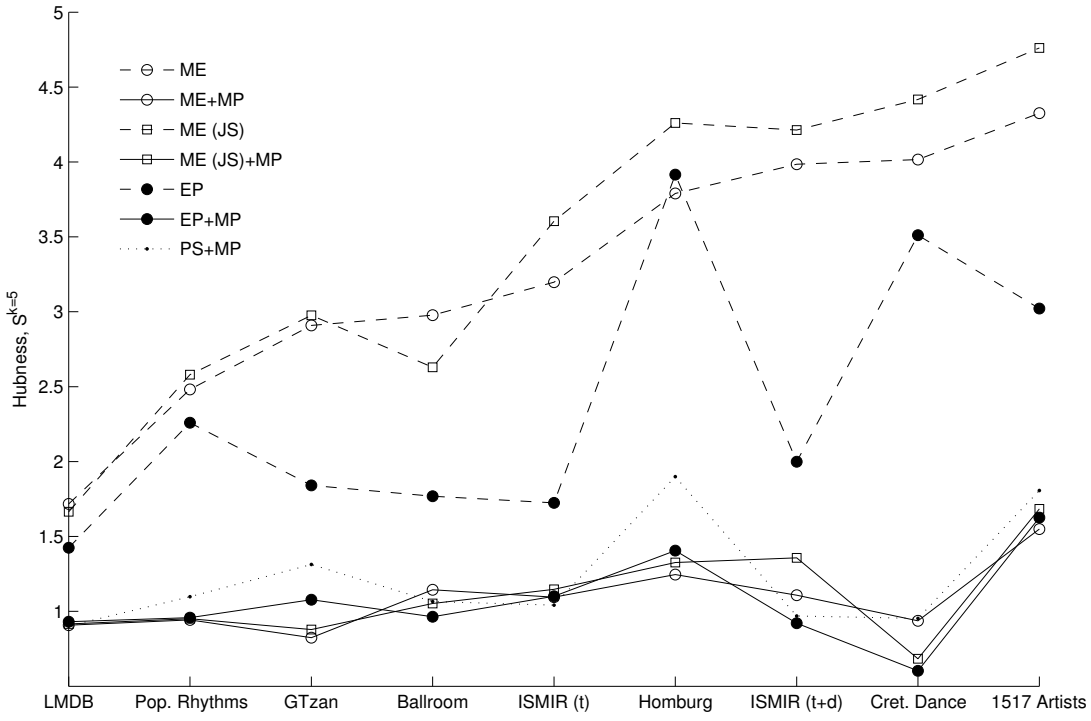
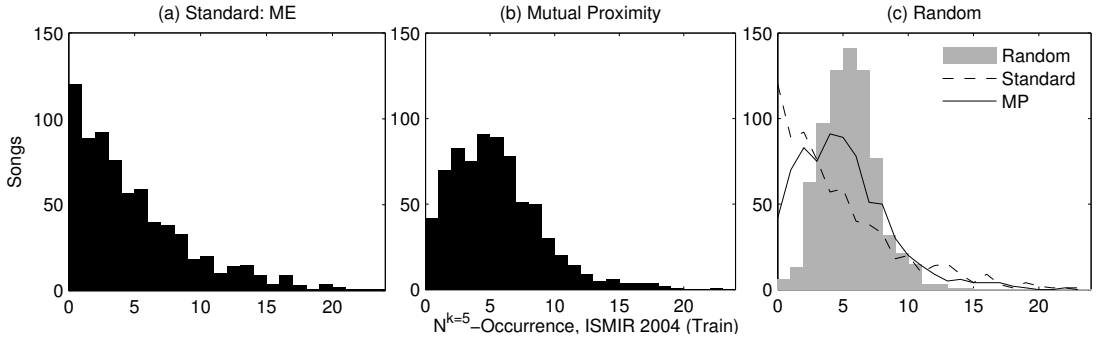
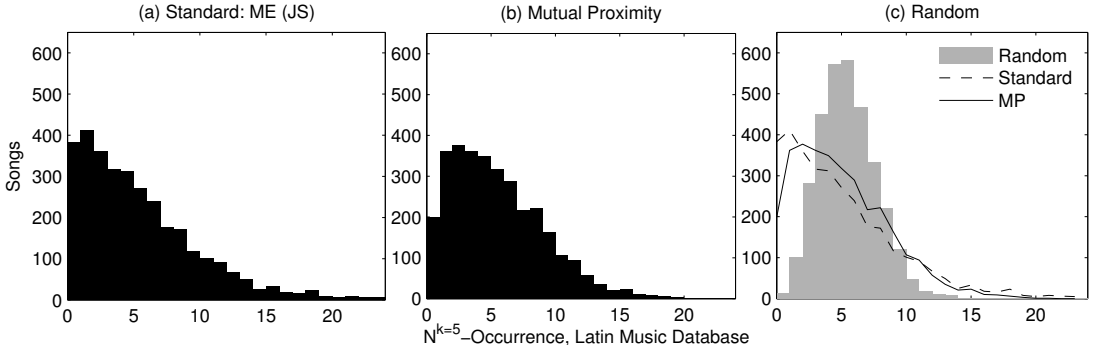


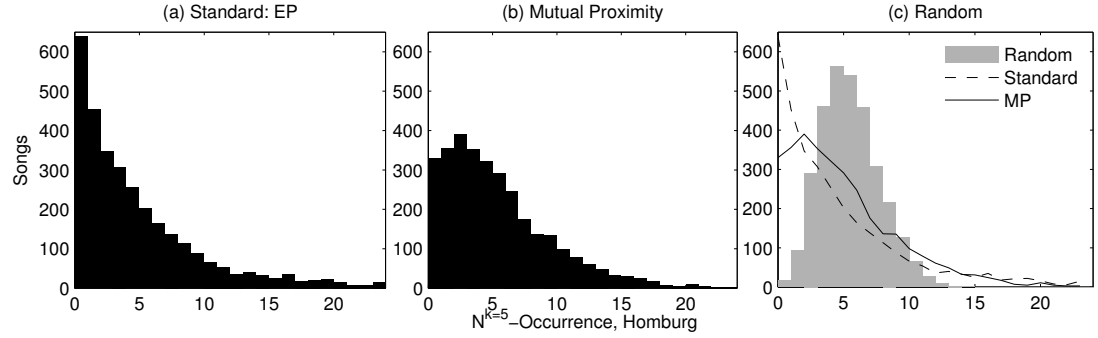
Figure 4.9: Hubness values ($S^{k=5}$) decline notably when using the MP. The original algorithm hubness is depicted with the dotted lines. MP produces nearest neighbors with notably lower hubness values around 1 and 1.5. The collections are sorted according to their hubness values computed for their standard MP algorithm. We use a different line style for PS as, contrary to ME and EP, it is only defined with MP.



(a) ME, ISMIR 2004 (Train)



(b) ME (JS), Latin Music Database



(c) EP, Homburg

Figure 4.10: Comparing the $k = 5$-occurrences of the ME/ME (JS)/EP content based audio similarity algorithm with the $N^{k=5}$ MP produces. Objects are more uniformly distributed with MP than with the original algorithm. For further comparison the $k = 5$ occurrence of a random distance matrix is also shown.

4.5.2.1 Hubness

Figure 4.9 shows the hubness exhibited by the three algorithms on the music collections we use throughout the thesis. In the original algorithms (the dotted lines) the hubness values are very high. Hubness in the original algorithms has a minimum value of 1.48 and a maximum of 4.7. It is notable that EP has in almost all cases a lower hubness than ME. These high hubness values sharply decrease in all collections if MP is applied. The minimum hubness value decreases to 0.6 and the maximum hubness value to 1.6.

High hubness means that the skewness of the k-occurrence (N^k) in the nearest neighbors is high. Figure 4.10 plots three histogram of the $k = 5$ occurrences for ME. The first column of plots shows the $N^{k=5}$ distribution for the original nearest neighbors. In all three example cases (ISMIR 2004 (Train), Latin Music Database and Homburg) we see that this distribution is highly skewed for ME, ME (JS) and EP. We can see the high number of orphaned songs in the first bin of the histogram. The skewness of the $N^{k=5}$ distribution visibly flattens and the number of orphans is notably reduced when using MP (second column of plots in Figure 4.10). The third column of plots in Figure 4.10 shows randomly generated $N^{k=5}$, exhibiting the most symmetric N^k distribution of objects.

4.5.2.2 Leave-one-out nearest neighbor classification accuracy

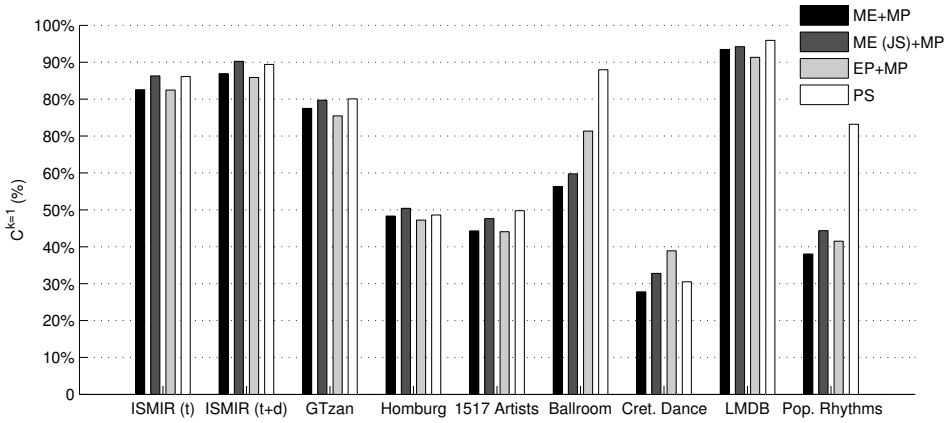
The detailed results of the classification experiments with MP are shown in Table 4.4. The table compares the original classification accuracies with the classification accuracies obtained with MP. It can be clearly seen that in almost all cases the classification accuracy increases with MP, from 3 (on average) up to 7.2 percentage points. Two exceptions are the *Cretan Dances* and *Popular Rhythms* collections where the classification accuracies for $k = 10, 20$ decrease with MP. In these collections the size of classes is sometimes smaller than k. Figure 4.11a compares the classification accuracies of the algorithms ME, ME (JS) and EP with PS. PS already uses MP in its original implementation. Interestingly, if we compare these results with the performance of the results of their original algorithms (Chapter 2, Section 2.5) PS does not clearly outperform the other algorithms any more.

On the contrary – in the case of the collections *ISMIR 2004 (Train)/(Full)*, *GTzan*, *Homburg*, *1517 Artists*, *LMDB* and *Cretan Dances* the original ME algorithm with the Jensen-Shannon divergence (JS) achieves the same results as PS. However PS still performs notably better on two rhythmic collections: *Ballroom* and *Popular Rhythms*. If we look at the performance of the music similarity algorithms if an artist filter is used in the evaluation, PS can still outperform the other algorithms, but no longer of the huge differences we have seen in Chapter 2.

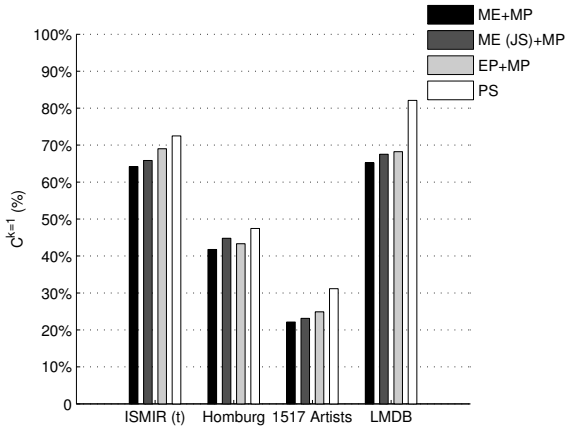
4.5.3 Discussion

Similarly to the results we have found for a variety of machine learning datasets in Section 4.4, using MP for a music similarity algorithm to transform music similarity spaces leads to lower hubness and higher classification rates, improving the overall performance of the music similarity algorithm.

In parallel with the decrease of hubness (the flattening of the N^k distribution) we can also observe that the number of orphaned songs in the nearest neighbor lists is significantly reduced.



(a) No artist filter



(b) Artist filter

Figure 4.11: Music genre classification rates $C^{k=1}$ of all algorithms using MP. Using MP reduces the difference in classification performance between PS and the other music similarity algorithms if Compared this to the original results of Section 2.5

	Orphaned songs in $N^{k=5}$				
Collection	$\boldsymbol{ME_{MP}}$	ME	$\boldsymbol{EP_{MP}}$	EP	$\boldsymbol{PS_{MP}}$
ISMIR 2004 (Train)	**5.8%**	16.5%	**7.7%**	15.1%	**9.1%**
ISMIR 2004 (Full)	**6.6%**	16.3%	**7.4%**	12.7%	**7.6%**
GTzan	**6.7%**	15.2%	**6.9%**	12.5%	**8.0%**
Homburg	**9.1%**	31.5%	**10.4%**	25.3%	**13.6%**
1517 Artists	**10.0%**	25.2%	**10.4%**	20.1%	**11.9%**
Ballroom Dataset	**8.7%**	21.2%	**7.0%**	15.8%	**8.5%**
Cretan Dances	**6.7%**	18.9%	**5.6%**	14.4%	**11.0%**
Latin Music Database	**6.3%**	11.3%	**5.5%**	9.1%	**5.9%**
Popular Rhythms	**10.4%**	28.5%	**8.4%**	25.9%	**8.6%**

Table 4.5: Percentage of orphaned songs in the different databases in the $k = 5$ nearest neighbor lists.

Table 4.5 compares the number of orphaned songs in $N^{k=5}$ of the different algorithms. We can see that MP reduces the number of songs which are never reached on average, by almost 60%. In the *Homburg* collection where over 31% of the songs could never make it in the $k = 5$ nearest neighbors, only 9.1% are orphaned after using MP.

As in addition to these aspects the classification accuracies increased we consider MP an important aspect of the music similarity algorithms we have evaluated. However without the approximation method, MP could not be used in applications.

4.6 Summary

We have presented a possible remedy for the 'hubness' problems, which tends to occur when learning in high-dimensional data spaces. Considerations on the asymmetry of neighbor relations involving hub objects led us to evaluate a recent local scaling method, and to propose a new global variant named 'Mutual Proximity' (MP). In a comprehensive empirical study we showed that both scaling methods are able to reduce hubness and improve classification accuracy as well as other performance indices. Local and global methods perform at about the same level. Both methods are fully unsupervised and very easy to implement. Our own global scaling variant, Mutual Proximity, presented in this chapter has the additional advantage of being easy to approximate for large data sets. Its probabilistic formulation also makes it straightforward to combine multiple distance spaces.

We showed that MP is ideal to be used for the content-based music similarity measures examined in this thesis, as they all exhibit very high hubness in their original implementations and the use of MP led to a significant increase of their retrieval quality. By using the fast, approximate MP method we also presented a way of how it could be used in large scale music recommendation systems.

Chapter 5

Indexing Computer Music Similarity Algorithms

Contents

5.1 Introduction 92
5.2 Related Work 93
5.3 A Fast Approximative Search Method 94
5.3.1 Preliminaries 94
5.3.2 Original FastMap 95
5.3.3 A Filter & Refine Search Method Using FastMap 96
5.3.4 Modifications 97
5.3.5 Evaluation of the Modifications 98
5.4 Evaluation with Music Similarity Algorithms 100
5.4.1 Single Divergence Measure 100
5.4.2 Linear Combinations 102
5.4.3 Mutual Proximity 102
5.5 Retrieval Quality 105
5.6 Summary 106

This chapter presents a filter-and-refine method to speed up content-based music similarity search in very large music collections. The introduced method achieves speedups of 10–40 times compared to an exhaustive scan, while at the same time returning 90% to 99% of the true nearest neighbors, depending on the divergence and parameter settings.

We introduce large scale music similarity search methods in Section 5.2 and present our new search method in Section 5.3. The filter-and-refine search method is evaluated in Section 5.4 where all introduced music similarity algorithms (ME, EP, PS) are used. The impact on the retrieval quality using the filter-and-refine method is investigated in Section 5.5.

Chapter 6 uses the presented method and builds a real prototype large-scale music recommendation service working with 2.3 million tracks. A query request to the system is answered in less than a second with a nearest neighbor retrieval accuracy of over 90%.

The method was published in 2009 [SFW09] ("A Filter-and-Refine Indexing Method for Fast Similarity Search in Millions of Music Tracks") and extended in 2010 [SFW10] ("A fast audio similarity retrieval method for millions of music tracks").

5.1 Introduction

Automatic content-based music recommendation systems usually operate with acoustic music similarity algorithms and work as a query-by-example system: (i) the user selects a song she/he likes, (ii) the system searches its databases for similar songs, (iii) according to the similarity measure the k nearest neighbors are returned to the user as possible recommendations. To use content based music recommendation algorithms on these large databases, the search strategy usually needs to be adjusted to scale, as a query on millions of songs should be answered quickly.

Content-based music similarity algorithms that represent their features as vectors and use the Euclidean distance (e.g., Tzanetakis and Cook [TC02] or Neumayer et al.[NLR05]) can use a wide array of standard techniques for the indexing. General approaches to search use binary space partitioning (BSP) trees, like Kd-Trees [Ben75] or vantage-point trees [Yia93]. These work well for moderately high-dimensional features using common metrics. For very high-dimensional data locality sensitive hashing (LSH, [AI06]) could be used, as the aforementioned algorithms are likely to perform worse or equal than a linear scan with very high dimensional data in terms of computational efficiency.

LSH is an approximate nearest neighbor retrieval algorithm for the metric spaces ℓ_1 and ℓ_2, it scales sub-linearly in the number of items and comes with a probabilistic guarantee on accuracy. It is possible to use LSH for non-metric divergences if the features can be embedded in the $\ell_{1,2}$ spaces.

However, all content-based music similarity methods we focus on, in this thesis are non-vectorial (i.e., use multivariate Gaussians) and use non-metric divergence measures (i.e., Kullback-Leibler divergences) so that these indexing methods can not be used directly. This is a major obstacle for systems that want to use any high-quality content-based music similarity algorithm at a large scale.

This chapter introduces a method to embed the Gaussian features in ℓ_2 using a modified FastMap implementation. Based on the embedding we propose a filter and refine algorithm to speed up nearest neighbor queries, so that all three content-based music similarity algorithms we use throughout this thesis (ME, EP, PS, see Chapter 2) can be used in large-scale systems. Overall the method accelerates the search for similar music pieces by a factor of $10 - 40$ and yields high nearest neighbor recall values of $90 - 99\%$ compared to a standard linear search, depending on which similarity measure is used.

5.2 Related Work

Scalable music recommendation systems have been the subject of a number of publications. One of the first content-based music recommendation system working on large collections (over 200 000 songs) was published by Cano et al. [CKW05] in 2005. They present a music recommendation system relying on a diverse range of audio features including rhythmic as well as timbre parameters. Together these features form a feature vector. They report artist identification rates of 24% for their music similarity measure. They do not report on special indexing techniques.

In 2007 Cai et al. [CZZM07] presented a music recommendation system which uses LSH to scale their acoustic music similarity algorithm. A single song is represented by about 30 high-dimensional vectors. Those are obtained using techniques from fingerprinting algorithms [BPJ03]. Their music similarity algorithm is evaluated using a ground truth of twenty playlists but is never compared with other established methods. They report an average query time of 2.5 seconds on a collection of about 100 000 tracks, which is very high, as the LSH index needs to be searched multiple times to compute similarity using their method.

Casey and Slaney [CS06] describe a large scale system to scan music collections for possible copyright infringements. They represent their song features as high dimensional vectors and, like Cai et al. [CZZM07], use LSH to scale their system. Although their system was not designed for music recommendation, it clearly shows that LSH can be effectively applied to build large scale MIR systems if the features and metrics permit.

Roy et al. [RAPB05] were the first to present a music recommendation system which could be used for large databases using Gaussian timbre features. They use a Monte-Carlo approximation of the Kullback-Leibler (KL) divergence to measure music similarity. In principle, their method also resembles a filter-and-refine method similar to the one proposed here. To pre-filter their results, they steadily increase the sampling rate of the Monte-Carlo approximation. As the divergence values converge they are able to reduce the number of possible nearest neighbors. In comparison to the closed form of the KL divergence, which is used in recent music similarity algorithms, this method is far more expensive to compute and yields worse results [ME07].

Another method to better cope with multivariate Gaussian timbre models was proposed in a publication by Levy and Sandler [LS06] a year later in 2006. They propose to use Gaussians with a diagonal covariance matrix, instead of a full one to compute music similarity. They report a ten-fold speedup compared to the full Kullback Leibler divergence. However, the quality of this simpler similarity measure is degraded, which can be seen, for example from worse genre classification rates.

With mufin.com there also exists a first commercial content-based music recommendation service that computes acoustic audio similarities for personal collections. Their website gives no information on how their service works[1].

Besides these approaches to build large scale music recommendation systems, a number of general methods from the class of distance-based indexing methods are relevant for indexing Gaussian features. These methods usually just require a dissimilarity function. A member of this class are vantage-point (VP) Trees [Yia93], which build a tree structure that can be searched efficiently, but require a metric distance measure. Another interesting and novel method, distance-based hashing (DBH), was presented in 2008 by Athitsos et al. [APPK08]. They use

[1] `http://www.mufin.com`, visited August 12th, 2011

FastMap [FL95] to randomly embed arbitrary features in ℓ_2 and use LSH to index that mapping.

The idea of using FastMap-related techniques for computationally heavy, non-metric similarity measures and nearest neighbor retrieval was already demonstrated by Athitsos et al. [AASK04] in 2004 to speed up classification of handwritten digits. In MIR research, FastMap was also used by Cano et al. [CKGB02] to map the high dimensional music timbre similarity space into a 2-dimensional space for visualization purposes.

Finally, we would like to mention an approach to deal with high dimensional statistical distance measures pursued by Garcia [GDB08]. He uses modern graphics processors (GPUs) to compute the divergences, as they offer very high floating-point performance and parallelism. Garcia shows how a linear brute force nearest neighbor scan can be accelerated on a GPU by a factor of about 40, compared to a plain C implementation on a standard CPU.

5.3 A Fast Approximative Search Method

To build our filter-and-refine method for fast similarity queries we use an adapted version of FastMap [FL95], a Multidimensional Scaling (MDS) technique. MDS [CHU+08] is a widely used method for visualizing high-dimensional data. FastMap takes the distance matrix of a set of items as input and maps the data to vectors in an arbitrary-dimensional Euclidean space. It was developed in the mid 1990s and was since then extended in various ways like BoostMap [AASK04] or MetricMap [WWSZ05]. These extensions were designed to improve the quality of the mapping of the objects. For our purposes, the original FastMap algorithm produces excellent results.

FastMap is straightforward to use even for large databases since it requires only a fixed number of rows of the similarity matrix to compute the vector mapping. However, FastMap requires the distances to adhere to metric properties.

5.3.1 Preliminaries

To evaluate the indexing method we propose, we introduce the evaluation dataset and the quality measure we will use in the evaluation.

Evaluation Dataset

Throughout this chapter we use a collection of 100 000 songs (30 s center-snippets) to evaluate the performance of the proposed method. These 100 000 songs are a random subset of a larger 2.3 million songs dataset which will be presented in the next chapter. The full dataset will be used in the next chapter to demonstrate the performance of our method in a large-scale prototype system.

Nearest Neighbor Recall

To compare the effectiveness of the nearest neighbor retrieval methods, we use what we call nearest neighbor (NN) recall. We define it as the ratio of true nearest neighbors found by some algorithm (NN_{found}) to the real number of true nearest neighbors (NN_{true}) as computed by an

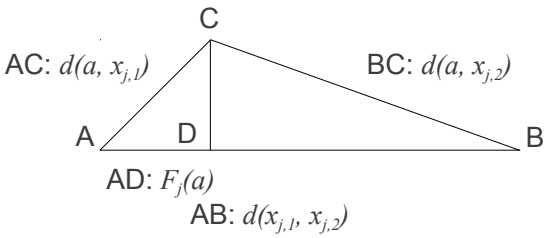


Figure 5.1: Visualizing Equation 5.2 as a triangle ABC. The lengths of sides AC and BC are given by the distances of the pivot points $x_{1,2}$ to the mapped point a, the length of side AB is the distance between the two pivot points. Thus the distance AD can be computed using $F_j(a)$ (Equation 5.2).

exhaustive search.

$$recall = \frac{|NN_{found} \cap NN_{true}|}{|NN_{true}|} \tag{5.1}$$

The NN-recall can be computed for various sizes of neighborhoods (k). We denote the NN recall of a size of k nearest neighbors as *k-NN recall*.

5.3.2 Original FastMap

The original FastMap [FL95] algorithm uses a simple mapping formula (Equation 5.2) to compute a k-dimensional projection of objects into the Euclidean vector space. The dimension k is arbitrary and can be chosen as required. Usually higher dimensions yield a more accurate mapping of the original similarity space.

To project objects into a k-dimensional Euclidean vector space, first two pivot objects have to be selected for each of the k dimensions. The original algorithm uses a simple random heuristic to select those pivot objects: for each dimension ($j = 1..k$), (i) choose a random object x_r from the database, (ii) search for the object most distant from x_r using the original distance measure $d()$ and select it as the first pivot object $x_{j,1}$ for the dimension, (iii) the second pivot object $x_{j,2}$ is the object most distant to $x_{j,1}$ in the original space.

After the $2k$ pivot objects have been selected, the vector representation of an object a is computed by calculating $F_j(x)$ for each dimension ($j = 1..k$):

$$F_j(a) = \frac{d(a, x_{j,1})^2 + d(x_{j,1}, x_{j,2})^2 - d(a, x_{j,2})^2}{2d(x_{j,1}, x_{j,2})} \tag{5.2}$$

Figure 5.1 visualizes the computation of $F_j(a)$, which clearly depends on metric properties of d to produce meaningful mappings. However, it has been noted that FastMap works surprisingly well also for non-metric divergence measures [AASK04].

Essentially FastMap only requires a distance function d and pivot objects to compute a vector mapping. Therefore it can be instantly applied to map the Gaussian music similarity models with their Kullback-Leibler divergences to the Euclidean vectors (ignoring the fact that they are non-metric divergences).

Divergence	*flops*	$flops/flops_{ME}$	$flops/flops_{EP}$	$flops/flops_{PS}$
ME	3 552	1	-	-
EP	4 636	-	1	-
PS	35 223	-	-	1
$d^2, k = 20$	60	0.017	0.013	0.002
$d^2, k = 40$	120	0.034	0.026	0.003
$d^2, k = 60$	180	0.051	0.039	0.005

Table 5.1: The computational complexity (in floating point operations, flops) of computing the squared Euclidean distance (d^2) is, even for high mapping dimensions like $k = 60$, much lower than the cost of a single similarity computation using ME or EP. When PS is used a comparison is only a $\frac{5}{1000}$th at $k = 60$.

5.3.3 A Filter & Refine Search Method Using FastMap

To use FastMap to quickly process music recommendation queries, we initially map the Gaussian timbre models to k-dimensional vectors. In a two step filter-and-refine process we then use those vectors as a pre-filter: given a query object we first *filter* the whole collection in the vector space (with the squared Euclidean distance) to return a number (=*filter-size*) of possible nearest neighbors. We then *refine* the result by computing the exact divergence on the candidate subset to return the nearest neighbors. By using the exact/original divergence to refine the results, correct nearest neighbor ranking is ensured. We set the parameter *filter-size* to a fraction of the whole collection.

The complexity of a single exact divergence computation is much higher than a simple vector comparison, so using the squared Euclidean distance to pre-filter the data results in large speedups compared to a linear scan using the exact divergence. To put that into context to the examined music similarity algorithm, Table 5.1 compares the computational cost (in floating point operations, flops) of one similarity comparison with ME, EP and PS to the squared Euclidean distance d^2 using different vector dimensions (k). From the table we can clearly see the computational advantage of using a Euclidean mapping, as a comparison in the vector space costs only a fraction of computational power compared to any other music similarity function. The high complexity (in terms of floating point operations) of PS stands out, which comes from the fact that a matrix inverse needs to be computed in the Jensen-Shannon divergence we use (Chapter 3, Section 3.4.1.4).

Unfortunately, as the evaluation in Section 5.3.5 will show, applying FastMap to map the music similarity models without any modifications yields very poor results.

Divergence	% triangle inequality
SKL	90.08%
$1 - e^{\lambda SKL}, \lambda = -\frac{1}{100}$	94.27%, used in [Pam06]
$1 - e^{\lambda SKL}, \lambda = -\frac{1}{50}$	97.54%, used in [Pam06]
$\sqrt{SKL}$	99.32%, used in [SFW09]
$\log(1 + SKL)$	99.99%
JSD	97.87%
$\sqrt{JSD}$	100%, used in [ES03]

Table 5.2: Percentage of Gaussian object triples fulfilling the triangle inequality ($D(x, z) \leq D(x, y) + D(y, z)$) with and without rescaling. The triangle inequality was checked for all possible triples in a collection of 10 000 randomly selected Gaussian music similarity models.

5.3.4 Modifications

We propose two modifications to improve the quality of FastMap mappings for approximate nearest neighbor retrieval. The modifications are centered around two thoughts:

1. A metric divergence measure would produce better vector mappings.

2. A more specialized heuristic for pivot object selection could produce better mappings especially for the near neighbors, which are at the center of our interest.

5.3.4.1 Rescaling

The symmetrized Kullback-Leibler divergence (denoted with SKL, Chapter 3) as it is used in the ME and EP algorithm already has the important metric properties of being symmetric and non-negative. It fails to fulfill the triangle inequality. The Jensen-Shannon Divergence (denoted with JSD, Chapter 3) is no metric either. The square-root of the JSD is, however, proven to be a metric distance [ES03]. In this section we test various rescaling methods and try to make the divergences more metric, i.e., make them obey the triangle inequality.

Table 5.2 shows the rescaling variants which were used to rescale the SKL and JSD. We experimentally verified the effect of rescaling on a collection of 10 000 randomly drawn Gaussian song models and checked the triangle inequality for all possible divergence triples. As PS cannot use the Jensen-Shannon divergence (it is undefined for multivariate Gaussians), we use an approximation in the experiment (see Chapter 3, Section 3.4.1.3). We denote the Jensen-Shannon approximation as it is used in the similarity measures with JSD.

For the SKL, the table shows that using $\log(1 + SKL)$ to rescale the divergence makes the divergence obey the triangle inequality in more than 99.9% of the cases. Thus we suggest using $\log(1 + SKL)$ to rescale the SKL for the ME and EP music similarity algorithms.

For the Jensen-Shannon-like divergence as it is used in the PS music similarity algorithm $\sqrt{JSD}$ is clearly the best rescaling variant. In 100% of all triples tested the triangle inequality was fulfilled.

5.3.4.2 Pivot Object Selection

To select the pivot objects needed to map an object x to a vector space, the original algorithm uses two objects for each dimension which lie as far away from each other as possible (see Section 5.3.2). In contrast to the original heuristic we propose to select the pivot objects using an adapted strategy:

1. First we randomly select an object x_r and compute the distance to all other objects.
2. We then select the first pivot object x_1 to be the object lying at the distance median, i.e. the object at the index $i = \lfloor N/2 \rfloor$ on the sorted list of divergences.
3. Likewise, the second pivot object x_2 is selected to be the object with the distance median of all divergences from x_1 to all other objects.

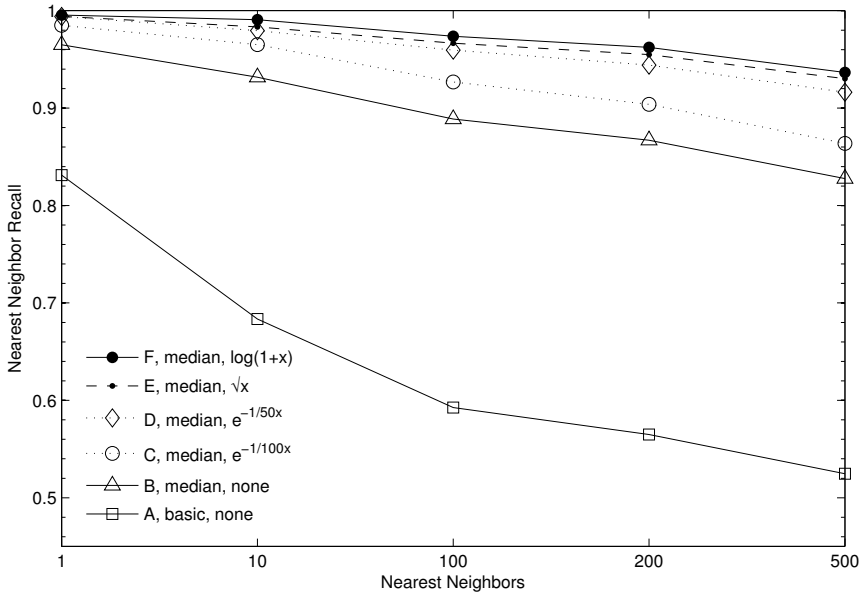
By using pivot objects at the median distance we avoid using objects with extremely high divergence values which often occur in the divergence tails when using any Kullback-Leibler divergence. Additionally, as we are particularly interested in optimally mapping the near neighbors and not the whole divergence space, this strategy also helps preserving the neighborhoods in the mapping method.

5.3.5 Evaluation of the Modifications

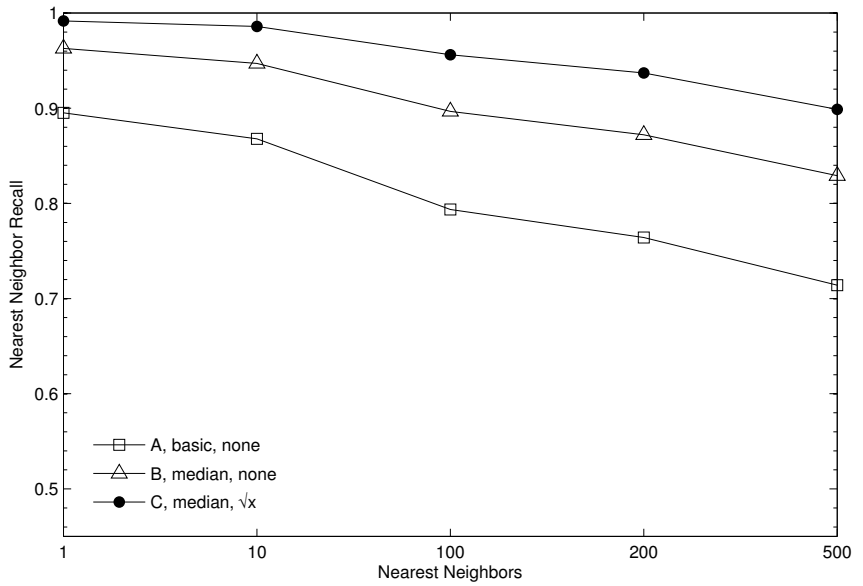
In a first experiment we measure the impact of each proposed modification on the filter-and-refine method. To do that, we compute the nearest neighbor (NN) recall on the collection of 100 000 songs using standard ME similarity models. We use the symmetrized Kullback-Leibler divergence (SKL) as well as the Jensen-Shannon-like divergence (JSD). Based on the results of the experiment, we select the modifications yielding the best results to be used in our algorithm. The experiment is set up as follows:

We use $k = 40$ as the mapping dimension for mapping the Gaussians into the vector space. In the mapping step the different divergence rescaling variants are applied. In addition to the rescaling of divergences, we also test with (i) the original FastMap pivot object strategy and (ii) our median-object selection strategy to compute the mapping. We use a fixed *filter-size* of 10% (= 10 000 objects) in the refine step of the search for this experiment. All unique experiment configurations are rerun ten times, the results are averaged.

Figure 5.2a shows the result of the experiment for the SKL. A huge improvement in the nearest neighbor recall can be seen for all strategies which use the median pivot object selection heuristic (B, C, D, E) compared to the original FastMap heuristic (A). The figure also shows that rescaling the SKL values helps to further increase the NN recall. The suggested pivot object selection strategy together with log-rescaling gives the best results: over 99% of the 10, and over 93% of the 500 nearest neighbors can be found using this filter-and-refine configuration, while computing only 10% of the SKL divergences that a linear scan would require.



(a) Symmetrized Kullback-Leibler divergence (SKL, used in ME, EP)



(b) Jensen-Shannon divergence (JSD, used in PS)

Figure 5.2: Nearest neighbor recall of two pivot object selection methods (*median*: the proposed pivot object selection heuristic, *basic*: the original Fastmap heuristic) in combination with four divergence rescaling methods (*no-rescaling*, $e^{\lambda x}$, $\sqrt{x}$, $\log(1+x)$) evaluated for the raw SKL and JSD divergences. NN recall is averaged over ten independent evaluation runs (10 000 random queries per run). Parameters: $k = 40$, *filter-size* $= 10\%$, *collection size*$= 100\,000$.

Figure 5.2b shows the result of the experiment for the JSD. Again using the median pivot-point selection heuristic (B, C) leads to a significant better mapping of the true nearest neighbors, measured by the NN-recall. The suggested pivot object selection strategy together with $\sqrt{x}$-rescaling gives the best results for the JSD: over 99% of the 10, and over 90% of the 500 nearest neighbors can be found found using this filter-and-refine configuration, while computing only 10% of the JSD divergences that a linear scan would require.

5.4 Evaluation with Music Similarity Algorithms

The experiments in the previous section have shown that the filter-and-refine process we propose is able to return the nearest neighbors very accurately while computing the exact divergence in the refine step for only a fraction of the full collection. However, in a system using this approach there are two free parameters which need to be chosen: (i) the mapping dimension (k), and (ii) the *filter-size*. Both have a direct impact on the nearest neighbor recall and computing power needed to process a query.

It is obvious that a larger *filter-size* results in better NN recall values but higher computational costs. Likewise, a higher k used for the vector mapping results in a more accurate mapping of the divergence space, but with each dimension the computational costs to compute the squared Euclidean distance in the pre-filter steps are increased.

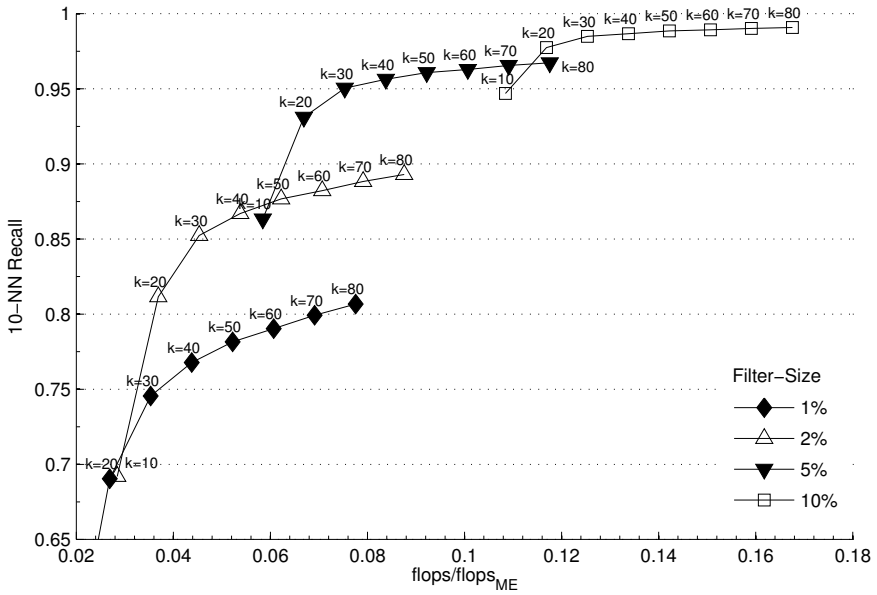
The next sections discusses each of the computer music similarity measures (ME, EP, PS) and evaluates their nearest neighbor recall under different parameter settings. These evaluations show how the method performs on the actual music similarity measures in terms of computational complexity and nearest neighbor recall. Based on the evaluations a system implementing the method should select its parameters.

We discuss ME as a system using a single divergence (Section 5.4.1), EP as a similarity measure using a linear combination of multiple measures (Section 5.4.2) and PS as a system using Mutual Proximity (Section 5.4.3) to enhance its performance.

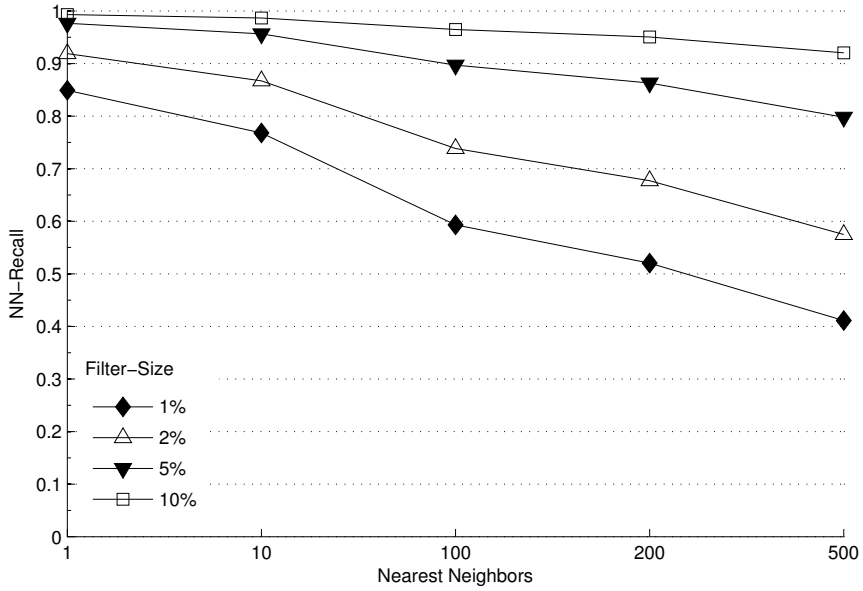
5.4.1 Single Divergence Measure

The ME music similarity algorithm can be used with the proposed filter-and-refine method without any further changes, as it only uses a single divergence measure. Figure 5.3 evaluates the filter-and-refine method with ME using different parameter combinations of k and *filter-size*. The original algorithm using SKL as similarity function is used. The same experiment could of course be performed using the JSD as divergence measure; in that case the speedups compared to a linear scan would be even greater as the computational complexity of the divergence is far greater.

The first plot (Figure 5.3a) shows the fraction of floating point operations needed to answer a query compared to a full scan. The plot also shows how the 10-NN recall develops with changing *filter-size* and k. From this plot we can see that in comparison to a full scan only 8.5% of flops are required to find 95% of the 10 nearest neighbors with a *filter-size* of 5% and $k = 40$. Looking at Figure 5.3b we observe that this configuration would still return over 90% of the true 100 nearest neighbors that a linear scan would return. This configuration would result in an 11× speedup compared to a linear scan.



(a) ME 10-NN recall for $k = 10-80$, *filter-size*$= 1-10\%$ and their computational complexity compared to a full scan.



(b) ME NN-recall with different filter-sizes. Parameters: $k = 40$

Figure 5.3: Evaluation of different parameters settings for the original ME algorithm. All results are averaged over ten runs. (a) shows the computational complexity under various settings, (b) shows the development of the nearest neighbor recall.

If a 10-NN recall of 81% is acceptable a parameter combination requiring only 3.3% the computational cost of a linear scan is possible ($k = 20$ and *filter-size* = 2%), yielding a 30× speedup). Almost perfect 10-NN recall values (> 99%) can be reached when setting *filter-size* to about 10% of the collection size, which still requires only 10% of the time a linear scan would need (10× speedup).

5.4.2 Linear Combinations

In this section we want to show the applicability of the previously proposed method to the EP music similarity measure. EP linearly combines multiple measures (music timbre and rhythm) in a single music similarity function. The music timbre component is the same as in ME (with the SKL as similarity function), and the rhythm component uses the Fluctuation Patterns (see Chapter 2, Section 2.4.3 for a more detailed discussion of the similarity measure). The similarities of the rhythm components are computed using a standard Euclidean distance.

In the case of EP we refrain from using the log-rescaling as the original similarity measure already rescales the divergence with e^{λ}. When using the log rescaling with the SKL, the static normalization parameters of this similarity measure would need to be recomputed.

The proposed filter-and-refine method can be applied to the combined similarity measure EP without any modifications, as the triangle inequality ($d(x, z) \leq d(x, y) + d(y, z)$) also holds for any linear combination of two metric distance measures d_1 and d_2:

$$\begin{aligned} \alpha_1 d_1(x, z) + \alpha_2 d_2(x, z) \leq \alpha_1 (d_1(x, y) + d_1(y, z)) \\ + \alpha_2 (d_2(x, y) + d_2(y, z)). \end{aligned} \tag{5.3}$$

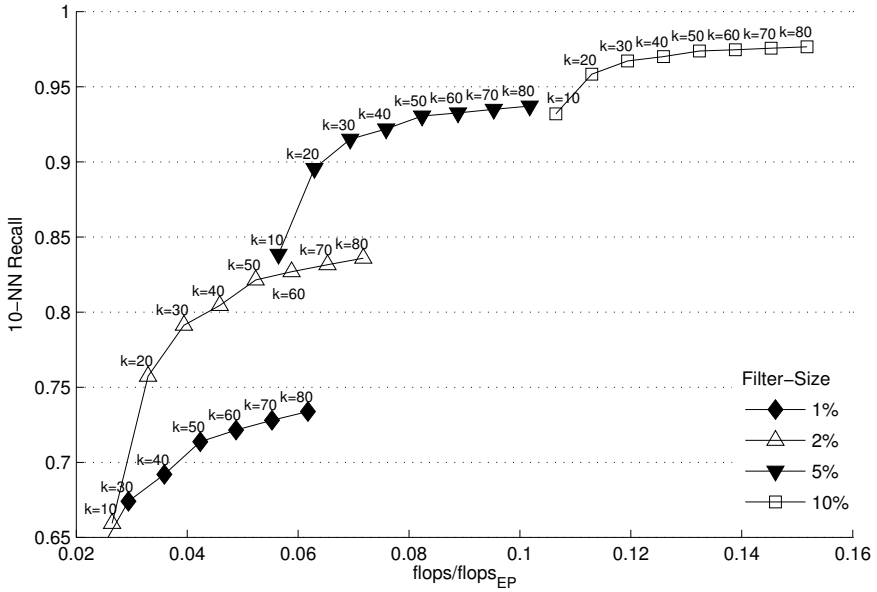
Therefore a linear combination of a non-metric (e.g., the SKL) and a metric distance (e.g., the Euclidean distance) could only violate the triangle inequality where the non-metric measure does. Since we have experimentally shown that rescaling the SKL can make the divergence almost metric, the linear combination inherits this property and should deliver comparable results with FastMap.

Figure 5.4 shows the same evaluation for the EP similarity measure as we have done for the single divergence measure in Section 5.4.1. The results are similar to the evaluation with ME. With only 7.5% of the computational complexity ($k = 40$, *filter-size*= 5%) over 91% of the true 10 nearest neighbors can be retrieved. From Figure 5.4b we see that this configuration retrieves 96% of the true first nearest neighbor and over 86% of the true 100 nearest neighbors. This configuration speeds up search 13× compared to a full exhaustive scan.[2]

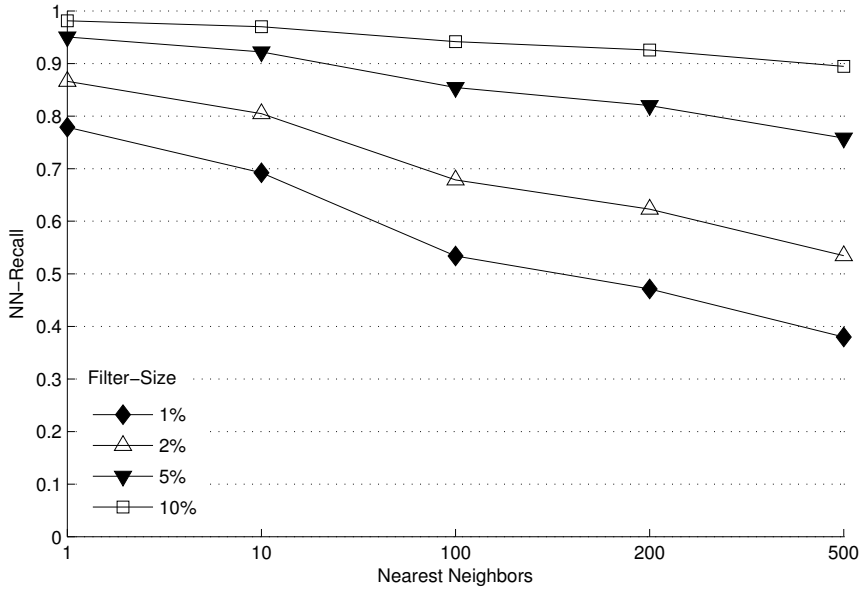
5.4.3 Mutual Proximity

Similarly to EP, the PS music similarity measure also uses a linear combination of multiple similarity measures, however the PS similarity values can not be used in the filter-and-refine method in a meaningful way. PS uses Mutual Proximity (Chapter 4) to combine the individual similarities. As Mutual Proximity yields probabilities, all metric properties of a divergence

[2]Note that we achieved higher nearest neighbor recall when using log-rescaling of the SKL. However, this requires changing the original EP similarity measure to adopt the global normalization factors.

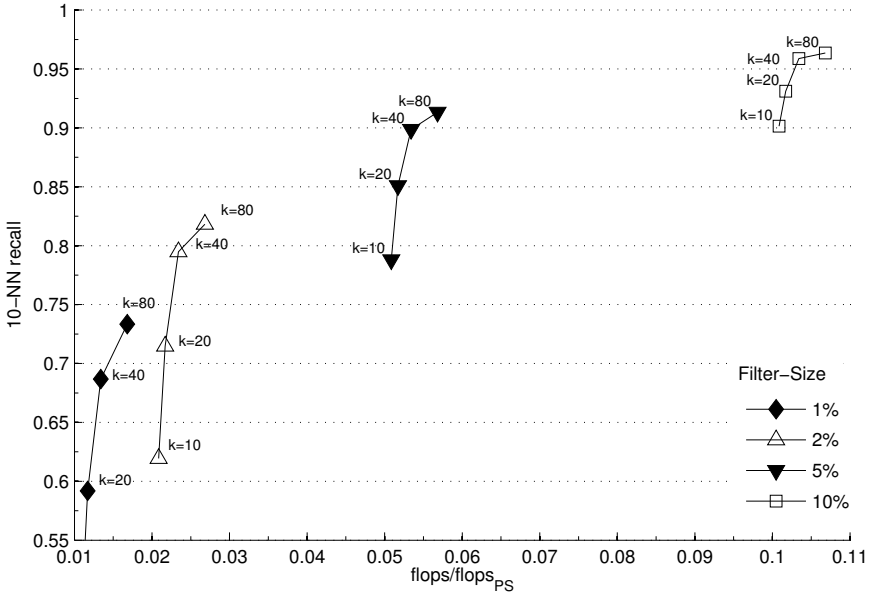


(a) EP 10-NN recall for $k = 10-80$, *filter-size*$= 1-10\%$ and their computational complexity compared to a full scan.

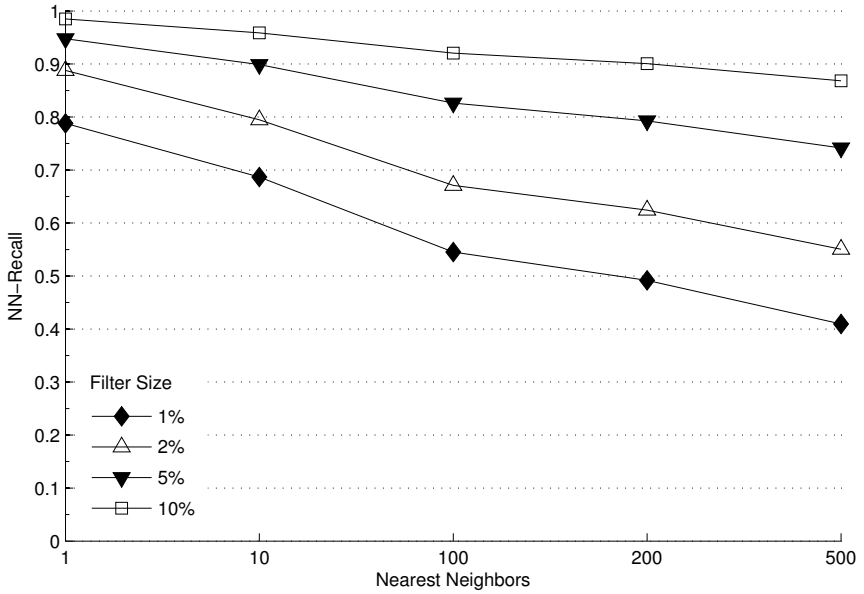


(b) EP NN-recall with different filter-sizes. Parameters: $k = 40$

Figure 5.4: Evaluation of different parameters settings for the original EP algorithm. All results are averaged over ten runs. (a) shows the computational complexity under various settings, (b) shows the development of the nearest neighbor recall.



(a) PS 10-NN recall for $k = 10-80$, *filter-size*$= 1-10\%$ and their computational complexity compared to a full scan.



(b) PS NN-recall with different filter-sizes. Parameters: $k = 40$

Figure 5.5: Evaluation of different parameters settings for the original PS algorithm using Mutual Proximity. All results are averaged over ten runs. (a) shows the computational complexity under various settings, (b) shows the development of the nearest neighbor recall.

measure are destroyed. Using the similarities returned by MP to create the mapping vectors with Fastmap would not yield usable results, i.e., very low nearest neighbor recall values.

To nevertheless use a similarity measure with Mutual Proximity and the filter-and-refine method, we propose to use the original divergence of the similarity measure to create the vector mapping for the filter step. Only the refine step would use the divergence together with Mutual Proximity. In the case of ME that would mean using the (rescaled) JSD or SKL to compute the vector mappings, and using MP during the refine phase. That way a good vector mapping could still be computed from the distances, while the quality of the refined results still benefit from using Mutual Proximity.

In the case of PS, where there is no original similarity measure, we propose using a unit-variance normalization, instead of MP to linearly combine the rhythm (d_r) and timbre (d_t) similarity measure. Thus the PS music similarity measure used to compute the vector mapping would be defined as:

$$d_{PS}^{F\&R}(m_1, m_1) = 0.5 \frac{d_r(m_1, m_2)}{\sigma^r} + 0.5 \frac{d_t(m_1, m_2)}{\sigma^t}, \tag{5.4}$$

In the refine step the standard PS similarity measure is be used. To compute the global normalizers $\sigma_{t,r}$ we simply average the standard deviation of distances σ_i computed for each song to use MP:

$$\sigma^r = \frac{1}{n} \sum_i^n \sigma_i^r, \qquad \sigma^t = \frac{1}{n} \sum_i^n \sigma_i^t \tag{5.5}$$

Of course the approximate variant of MP (MP_{approx} as described in Chapter 4, Section 4.3.2.2) would need to be used for large collections. Computing the full similarity matrix to estimate the MP parameters is not feasible in a large scale scenario.

Figure 5.5 evaluates PS with Mutual Proximity. The results are similar to the previous evaluations, although the speedup factor is even higher due to the higher complexity of the original divergence. The steeper curves in the plots (compared to ME and EP) come from the fact that even a high number of mapping dimensions (k) does not add much computational complexity relative to the complexity of the full PS similarity.

From Figure 5.5a it can be seen that with only 5.4% of the computational complexity ($k = 40$, *filter-size*= 5%) 90% of the true 10 nearest neighbors can be retrieved. From Figure 5.4b we see that this configuration retrieves 96% of the true first nearest neighbor and over over 82% of the true 100 nearest neighbors. This configuration speeds up search 18× compared to a full exhaustive scan. If a NN-recall of 90% for the first nearest neighbor and 80% for the 10 nearest neighbors is acceptable (at $k = 40$ and *filter-size*= 2%) a speedup of 43× can be achieved.

5.5 Retrieval Quality

An aspect of the filter-and-refine method which was not yet evaluated, is how falsely reported nearest neighbors (false positives) affect the quality of the music recommendations. Missing a nearest neighbor does not necessarily need to lead to worse retrieval quality.

To try to quantify the effect on the retrieval quality, we conduct a genre classification experiment with the filter-and-refine method. We evaluate all eight music collections which we

use throughout this thesis (cf. Chapter 2, Section 2.5.1), again with all three music similarity algorithms (ME, EP, PS), in a genre classification experiment. As in previous experiments we use four different neighborhood sizes $C^{k=1,5,10,20}$ in the experiment. The filter-and-refine method uses a mapping dimension of $k = 40$, and a *filter-size* of 5%.

Table 5.3 summarizes the results. For $C^{k=1}$ the increases and decreases in genre classification rates balance each other. For larger $C^{k>1}$ the classification rates increase notably with the filter-and-refine technique in cases where no artist filter is used. However, if an artist filter is used, slight decreases of classification accuracies can be observed. The maximum decrease in a larger collection with more than 1 000 songs is 1.3 percentage points (LMDB collection at $C^{k=10}$). As the method also seems to work quite well for small collections, it could be an indication that small devices with limited computational power could also use the method to generate playlists. The results also suggest that the false positives (in terms of their nearest neighbor position) do not seem to adversely affect the classification accuracies.

5.6 Summary

We have presented a filter-and-refine method for fast music similarity search in large collections. The method is primarily designed and evaluated for music similarity algorithms which use Kullback-Leibler divergences with multivariate Gaussian features to compute acoustic similarity. It works by first mapping the Gaussian similarity features into an intermediate Euclidean vector space using a modified FastMap algorithm. To increase the quality of this vector mapping, the divergences are rescaled and a median pivot object selection heuristic is used within the FastMap algorithm. We use this vector-space approximation to quickly filter for possible nearest neighbors. In a second (refine) step the results are reordered according to their original divergence.

As a search in the vector space has low computational complexity compared to the Kullback-Leibler divergences, the two-tiered search is shown to be very effective. The introduced method achieves speedups of 10–40 times compared to an exhaustive scan, while at the same time returning 90% to 99% of the true nearest neighbors, depending on the divergence and parameter settings. By accelerating similarity queries we show how a large scale music recommendation service using recent music similarity algorithms could be operated.

			Filter & Refine C^k				Difference to Exact C^k			
Collection	Algorithm	AF	1	5	10	20	1	5	10	20
ISMIR 2004 (Train)	ME	no	79.2%	71.7%	67.6%	63.1%	-1.2	**3.8**	**7.0**	**6.0**
	EP	no	80.6%	72.9%	68.2%	62.9%	-0.7	**2.8**	**3.7**	**2.7**
	PS	no	86.8%	78.2%	73.9%	68.3%	**0.7**	**4.7**	**2.8**	**5.0**
	ME	yes	63.7%	62.8%	60.7%	58.2%	-0.5	**0.9**	**0.5**	**0.4**
	EP	yes	68.7%	66.0%	62.9%	60.1%	-0.3	-0.2	-0.3	-0.7
	PS	yes	72.4%	68.9%	65.7%	63.4%	-0.1	-0.1	-0.5	-0.1
ISMIR 2004 (Full)	ME	no	85.2%	78.7%	75.1%	70.2%	**0.2**	**3.1**	**6.2**	**7.9**
	EP	no	84.5%	78.7%	74.8%	70.1%	**0.3**	**3.0**	**3.2**	**5.5**
	PS	no	90.5%	84.4%	80.7%	76.5%	**1.1**	**5.3**	**4.2**	**4.7**
GTzan	ME	no	74.7%	61.7%	54.8%	46.7%	**0.9**	**7.6**	**6.4**	**11.4**
	EP	no	75.1%	63.9%	56.6%	48.1%	**1.4**	**8.8**	**8.0**	**8.5**
	PS	no	80.3%	71.6%	65.3%	56.9%	**0.2**	**7.4**	**11.2**	**10.8**
Homburg	ME	no	43.9%	41.2%	39.9%	38.6%	**0.1**	**1.4**	**3.6**	**2.0**
	EP	no	45.8%	43.4%	41.6%	40.0%	**1.4**	-0.2	**2.2**	**1.3**
	PS	no	48.0%	46.1%	44.6%	41.9%	-0.6	**0.0**	**0.3**	-0.0
	ME	yes	41.7%	40.4%	39.4%	38.3%	-0.1	-0.3	**0.1**	**0.2**
	EP	yes	44.5%	42.8%	41.2%	39.7%	**1.2**	**0.1**	-0.4	-0.5
	PS	yes	47.9%	46.6%	45.3%	44.7%	**0.4**	-0.2	-0.2	**0.1**
1517 Artists	ME	no	42.8%	31.1%	25.9%	21.5%	**0.9**	**6.7**	**6.1**	**5.9**
	EP	no	41.7%	30.7%	26.1%	21.9%	-0.3	**5.2**	**4.2**	**4.3**
	PS	no	50.8%	39.4%	33.8%	28.3%	**1.0**	**5.9**	**6.0**	**5.6**
	ME	yes	22.7%	19.9%	18.7%	17.2%	**0.6**	**0.2**	**0.2**	**0.2**
	EP	yes	24.8%	21.7%	20.2%	18.4%	-0.1	-0.2	-0.5	-0.7
	PS	yes	31.1%	27.7%	25.7%	23.4%	-0.1	-0.7	-1.0	-1.5
Ballroom Dataset	ME	no	53.3%	45.0%	42.0%	38.3%	-1.0	**2.9**	**5.6**	**6.8**
	EP	no	67.6%	56.4%	51.2%	43.4%	-0.3	**1.2**	**1.6**	**2.1**
	PS	no	89.4%	83.1%	79.2%	72.6%	**1.4**	**4.2**	**6.0**	**14.3**
Cretan Dances	ME	no	25.8%	23.2%	21.4%	21.1%	-0.3	**4.3**	**0.8**	-1.1
	EP	no	34.9%	25.0%	21.9%	20.3%	**3.2**	-3.3	**5.8**	**3.1**
	PS	no	37.4%	31.4%	26.2%	18.9%	**6.9**	**5.1**	-1.8	-4.0
LMDB	ME	no	92.9%	88.4%	84.4%	77.1%	**0.5**	**3.9**	**7.8**	**11.7**
	EP	no	92.4%	86.8%	82.4%	76.1%	**1.5**	**6.8**	**9.3**	**9.8**
	PS	no	95.0%	90.8%	87.4%	82.5%	-0.9	**1.0**	**3.0**	**4.7**
	ME	yes	64.2%	56.4%	55.4%	53.8%	-1.0	-1.0	-1.3	-0.1
	EP	yes	67.3%	63.6%	61.1%	58.0%	-0.9	-0.9	-0.8	-0.3
	PS	yes	81.9%	77.8%	75.1%	70.8%	-0.2	-0.1	-0.3	-0.8
Popular Rhythms	ME	no	33.4%	26.6%	22.6%	18.9%	-1.8	**7.3**	**3.9**	**9.7**
	EP	no	40.8%	30.3%	23.9%	17.4%	-0.1	**2.6**	**3.2**	-1.0
	PS	no	72.6%	55.7%	45.8%	33.9%	-0.6	**5.8**	**7.8**	**10.6**

Table 5.3: Results of a genre classification experiment on all eight datasets. The classification accuracies $C^{k=1,5,10,20}$ for three music similarity algorithms (ME, EP, PS, where applicable with Artist Filter) and a system using the approximate filter-and-refine system are compared to their original (exact) classification accuracies.

Chapter 6

Dealing with the Music of the World

Contents

6.1 Introduction 109
6.2 2.3 Million Songs 110
6.3 The Prototype System: Wolperdinger 112
6.3.1 Music Similarity Features 112
6.3.2 Music Similarity Search 113
6.3.3 Qualitative Evaluation 115
6.3.4 Implementation and Server Specification 116
6.3.5 Performance 117
6.3.6 User Interface 118
6.4 Summary 122

This chapter builds a prototype large-scale music recommendation service working with 2.3 million tracks. While we admit that 2.3 million tracks is not quite all the music of the world, the evaluation of the system is the largest published to date. The system requires all methods which have been introduced in this thesis. A query request to the music recommendation prototype is answered in less than a second with a nearest neighbor retrieval accuracy of over 90%. Section 6.2 describes the evaluation collection we use, and Section 6.3 builds, evaluates and presents a high-performance and high-quality music recommendation engine using all the methods presented in this thesis.

6.1 Introduction

Chapter 2 identified the problems related to building a large scale music recommendation system using any of the three content-based music similarity measures used in this thesis. Chapter 3,

Genres per Album	1	2	3	4	5	6	7	8
Albums	70 182	52 803	38 761	18 782	4 986	870	162	18
% of Collection	37.6%	28.3%	20.8%	10.1%	0.3%	0.5%	0.1%	0.01%

Table 6.1: Number of music genre names assigned per album. The majority of albums is assigned 1–3 genres.

4 and 5 subsequently presented solutions to each of the problems. We showed how to use the music similarity features natively and correctly in clustering algorithms, developed a method to alleviate the hub problem and created an indexing solution for the reviewed class of music similarity algorithms.

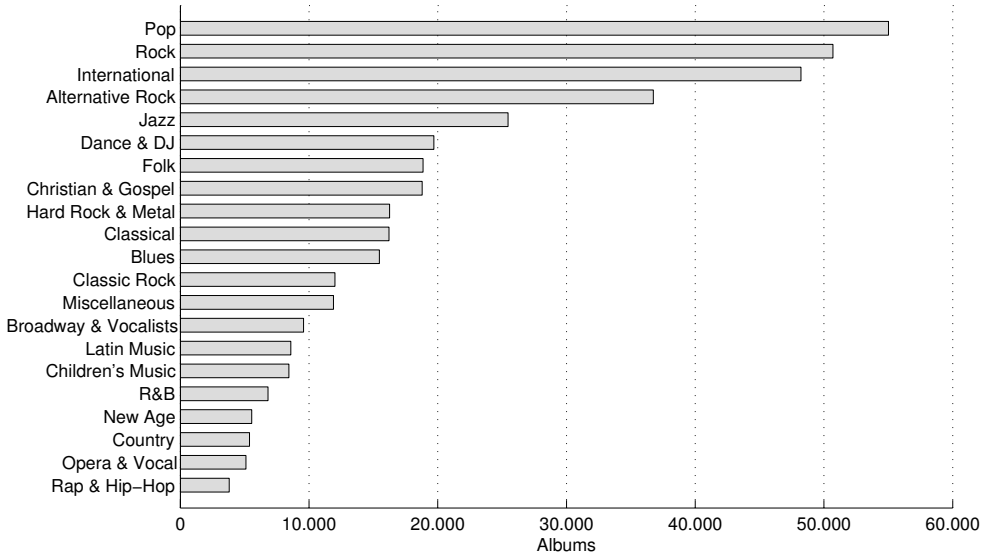
Now everything is in place to build a truly scalable music recommendation system. The prototype system we will develop alongside with the data we use for evaluation, is described in the next sections of this final chapter.

6.2 2.3 Million Songs

To build the prototype system we use real music data. The data is downloaded from an on-line MP3 store, which offers free 30 second snippets of songs for preview purposes. In total 2.3 million song-snippets (2 314 925) were downloaded over the course of three months. These songs are from 186 564 different albums from 96 397 different artists. In addition to the MP3s we also gathered their CD cover-art and parsed the meta data from the download web page. Meta data which we parsed includes: The artist/band name, the song title, the song number, release year, broad music genre tags and the publishing company.

Each album is assigned to one or more music genres, which enables us to perform a large scale genre-evaluation experiment. In total there are 21 different music genres in the collection. The distribution of the genres in the collection is shown in Figure 6.1. Almost two thirds of the albums are assigned to the genre pop or rock. When building the collection we noticed that the genre labels of the MP3 store seem to be selected to boost sales rather than by musical considerations. For example popular music albums would often also be labeled with multiple other genre names to ensure users will notice the album regardless of the category they click on in the shop. To get an impression of the music genre distribution, Table 6.1 lists the number of albums and the number of genres they are assigned to.

An interesting fact can be uncovered when looking at the number of songs per album in the collection. Figure 6.2 displays a histogram of the number of songs per album. The distribution seems quite Gaussian, with a clear peak at 10 and 12 tracks per album. However, a notable valley at 11 tracks per album is visible – a number which apparently some artists like to avoid on their album.



Genre	Albums	%			
Pop	55 013	29.5%	Classic Rock	12 013	6.4%
Rock	50 696	27.2%	Miscellaneous	11 902	6.4%
International	48 213	25.8%	Broadway & Vocalists	9 578	5.1%
Alternative Rock	36 743	19.7%	Latin Music	8 592	4.6%
Jazz	25 465	13.6%	Children's Music	8 434	4.5%
Dance & DJ	19 696	10.6%	R&B	6 813	3.7%
Folk	18 862	10.1%	New Age	5 551	3.0%
Christian & Gospel	18 791	10.1%	Country	5 383	2.9%
Hard Rock & Metal	16 258	8.7%	Opera & Vocal	5 106	2.7%
Classical	16 214	8.7%	Rap & Hip-Hop	3 803	2.0%
Blues	15 461	8.3%			
			Total	398 587	213.7%

Figure 6.1: Histogram plot of the music genres of the albums in the 2.3 million tracks collection. As multiple genres can be assigned to an album the totals sum up to more than 100% of the collection.

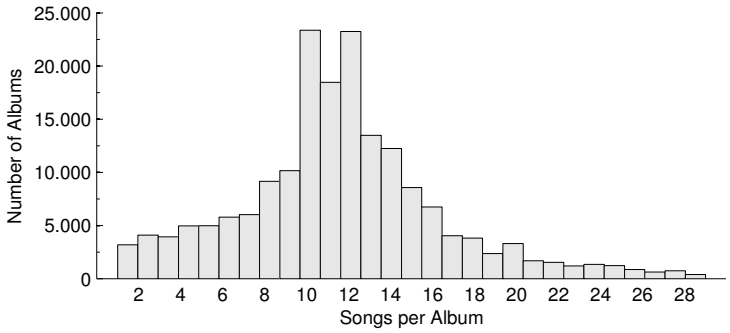


Figure 6.2: A histogram plot of the number of tracks per album.

6.3 The Prototype System: Wolperdinger

To build the prototype system we first select the music similarity algorithm to use, choose the parameters for the filter-and-refine method to achieve best retrieval results and evaluate the configuration according to the measures we have used throughout this thesis. After that we show how we implemented the prototype system, report query performance and show the web-based interface we built to allow easy browsing of the 2.3 million tracks with the "Wolperdinger" music discovery engine.[1]

6.3.1 Music Similarity Features

Throughout this thesis we have used three related content-based music similarity algorithms to demonstrate the methods we have developed: ME, EP and PS (see Chapter 2 for their review). For the prototype system we decide to use PS. It is currently one of the music similarity algorithms with the best qualitative results. Unfortunately it is also the most expensive one to compute, also in terms of feature extraction (see Section 2.4.4). In our prototype we use a modified feature extraction variant to speed up the analysis of songs. The changes are rather small, but have a big impact on the processing time:

- Instead of computing four different psychoacoustic spectra during the feature extraction (and thus requiring four STFTs), we compute a single STFT and compute only the Mel and a logarithmic Cent spectrum from it.

- We round all window sizes to the closest power of two. Especially when computing the FFT for the rhythm/onset coefficients this change reduces the runtime considerably as no corner-case FFT algorithms need to be used.

[1] A "Wolperdinger" is a horned rabbit, it can be (very rarely) seen in Munich in Bavaria during the Oktoberfest. The author shamelessly used the name for his music recommendation prototype system, after spotting one on the Oktoberfest in 2009. More information about Wolperdingers: `http://en.wikipedia.org/wiki/Wolperdinger`, visited August 12th, 2011

With these changes we are able to process 810 MP3 snippets per minute on a quad-core desktop system. The runtime includes the decoding and resampling of the sounds, which is usually required in any audio processing algorithm.

6.3.2 Music Similarity Search

The PS similarity function (computing two Jensen-Shannon-like divergences) is very expensive to compute. We have seen in Chapter 5 that ten times more floating point operations (35 223) are required to compute a single similarity compared to the simple ME measure, which requires only 3 552 *flops* per comparison. Additionally PS requires the dynamic Mutual Proximity normalization technique to linearly combine its two rhythm and timbre similarity measures.

However with the filter-and-refine method presented in Chapter 5 we have shown a way to use the complex similarity functions while still retrieving a very high percentage of true nearest neighbors. In Chapter 4 we have also presented an approximative method to compute the Mutual Proximity normalization factors requiring a coarse k-means clustering. In Chapter 3 we showed how a k-means clustering can be done for the Gaussian music similarity features. All of that enables us to finally use the PS algorithm with very large collections.

6.3.2.1 Parameters

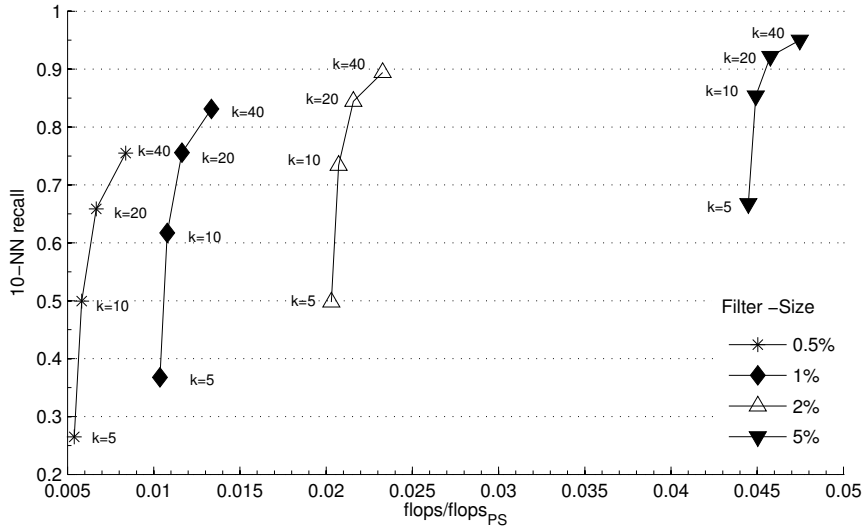
To use the filter-and-refine search algorithm two parameters need to be determined: the mapping dimension of the vectors (k) for the approximate filter step and the number of objects (*filter-size*) to search in the refine step of the method. Both parameters have a direct impact on the retrieval quality and the speed of the system.

In an experiment we evaluate how the approximate search method performs on the 2.3 million songs collection using different parameter configurations. This experiment is the basis to choose the optimum set of parameters for the recommendation system. To perform the experiment we randomly select 23 000 songs from the 2.3 million song snippets, compute their exact 1–500 nearest neighbors in the whole collection (of 2.3 million songs) and use the filter-and refine methods with different parameter settings to measure the impact on the retrieval quality, comparing it to an exact search. A larger part of the collection could not be tested as a single computation of an exact playlist takes about 19 s to complete. In the experiment we compute the nearest neighbor (NN) recall, measuring the percentage of true nearest neighbors compared to the exact solution (see Chapter 5 for a definition)

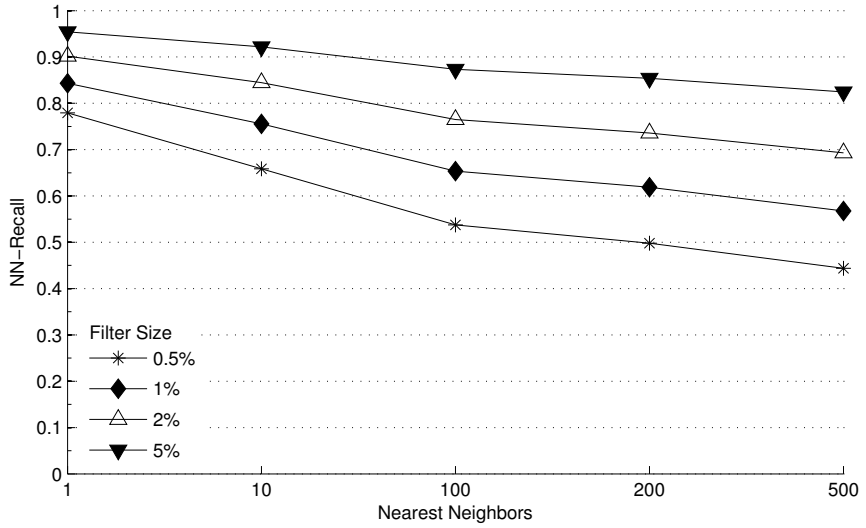
Figure 6.3 shows the result. As the exact nearest neighbors need to be determined to compute the NN recall values using a linear scan, the experiment took five days to complete. The first plot (Figure 6.3a) sets the computational complexity of a linear scan in relation to the mapping dimensions $k = 5, 10, 20, 40$, *filter-size*$= 0.5\%, 1\%, 2\%, 5\%$ and the resulting nearest neighbor recall. From the plot we can see that a smaller *filter-size* should be used to decrease the search time, rather than decreasing the number of mapping dimensions.

A mapping dimension of $k = 5$ and a filter size of 5% only returns 65% of the true nearest neighbors, while using a mapping dimension of $k = 40$ and a filter-size of 2% is twice as fast and still has a NN recall of 90%.

The second plot (Figure 6.3b) uses a fixed mapping dimension of $k = 40$ and looks at the impact on the {1,10,100,200,500}-NN-recall values. Using these two plots it is now easily possible



(a) PS 1-NN recall for $k = 5 - 40$, *filter-size*$= 0.5 - 5\%$ and their computational complexity compared to a full scan.



(b) PS NN-recall with different filter-sizes. Parameters: $k = 40$

Figure 6.3: Evaluation of different parameters settings for the PS algorithm on the 2.3 million song snippets. From the 2.3 million songs 23 000 have been randomly selected for this evaluation. (a) shows the computational complexity under various settings, (b) shows the development of the nearest neighbor recall.

Evaluation	*Overlap*		*Match*	
	$k = 1$	$k = 10$	$k = 1$	$k = 10$
Artist Filter	51.2%	46.7%	70.0%	67.0%
Album Filter	57.2%	49.3%	74.1%	69.0%
Standard	69.0%	55.0%	81.1%	72.2%
Random	*15.9%*	*15.9%*	*28.1%*	*28.1%*

Table 6.2: Genre retrieval accuracy evaluated for the 2.3 million songs collection.

to choose the optimal operating parameters for the filter-and-refine routine in the system. Based on that we use a static $k = 40$ and a *filter-size* of 1-2% for the system. That would result in a speedup of 80-40× compared to a linear scan, while still returning a 85-90% of true nearest neighbors.

A dynamic server solution could use a static $k = 40$ vector mapping dimension to pre-filter the results and vary the *filter-size* parameter according to the current load of the server, or the priority of the query. Implementing that strategy would make the system capable of answering queries with constant speed, at the cost of decreased accuracy under high load.

6.3.3 Qualitative Evaluation

After fixing the parameters, we perform a qualitative evaluation of the recommendation system. As we have (very coarse) genre labels available for all songs, we first evaluate the genre retrieval accuracy the system achieves for the whole collection. As there are multiple genres attached to each album (and thus to each song) we can compute two different measures:

The first measure (denoted as *Match*) counts the number of songs in the nearest neighbors which have at least one genre matching with the query song q, the second measure (denoted as *Overlap*) computes the relative overlap of the genres compared to the number of genres of query q. $[P]$ in Equation 6.1 denotes the Iversion bracket, which is 1 if P is true, and 0 otherwise:

$$Match(q) = \frac{1}{k} \sum_{i=1}^{k} ([|Genres(q) \cap Genres(NN_i)| > 0], \tag{6.1}$$

$$Overlap(q) = \frac{1}{k} \sum_{i=1}^{k} \frac{|Genres(q) \cap Genres(NN_i)|}{|Genres(q)|}. \tag{6.2}$$

We average both retrieval accuracy numbers over all songs and compare them to a random shuffle algorithm. The results for all 2.3 million songs using the full Wolperdinger system are shown in Table 6.2. The table also includes retrieval accuracy values using an artist as well as an album filter. Looking at the results we can see that both measures clearly outperform a random shuffle system. *Overlap* returns 69–46.7% genre retrieval overlap where a random shuffle system would return 15.9%. In the case of the *Match* measure, the results are similar: The system achieves 81.1–67% genre retrieval accuracy compared to 28.1% genre retrieval accuracy for the random system.

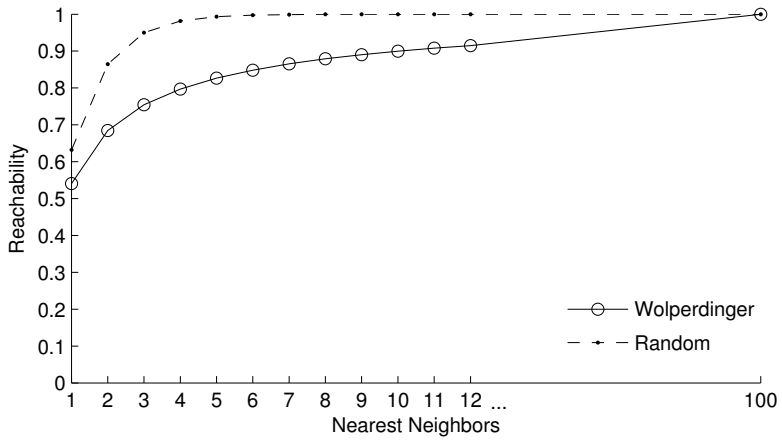


Figure 6.4: Reachability of all songs in the n nearest neighbors in the Wolperdinger prototype recommendation system compared to a random one.

Another qualitative measure we evaluate is the "reachability" of other songs in the nearest neighbors. Reachability measures the percentage of songs which occur in the nearest neighbor lists of at least one other song, i.e., the number of songs that can be reached/discovered by only looking at the nearest neighbor lists at a given k. The measure is closely related to hubness. A recommendation system exhibiting high hubness would have poor reachability. A good recommendation system would be desired to exhibit high reachability as well as high retrieval quality.

Figure 6.4 shows the reachability for increasing nearest neighborhood sizes. Again we compare it to the reachability of a random shuffle recommender system, which gives a possible upper limit. Our recommendation system, while returning qualitatively good results, can discover 90% of the 2.3 million songs even with a small neighborhood size of 10. At a neighborhood size of 100 all 2.3 million songs are reachable in the neighborhood graphs. The random variant reaches all songs in its recommendations at a neighborhood size of 5–6.

6.3.4 Implementation and Server Specification

The implementation of the actual recommendation system is now straightforward. In an initial step all music pieces are analyzed and their similarity models are computed. All similarity models are stored in a database. A single PS music similarity model is about 5 kilobytes. So all models of the 2.3 million song snippets (1 terabyte MP3s) require about 11 gigabytes of storage.

After the collection has been analyzed, two additional preprocessing steps are required before the system can answer queries:

- PS uses Mutual Proximity to linearly combine the rhythm and timbre component. To use Mutual Proximity, we estimate its parameters using a k-means clustering of the Gaussians and the approximative MP algorithm we introduced in this thesis. We use 5 centers for

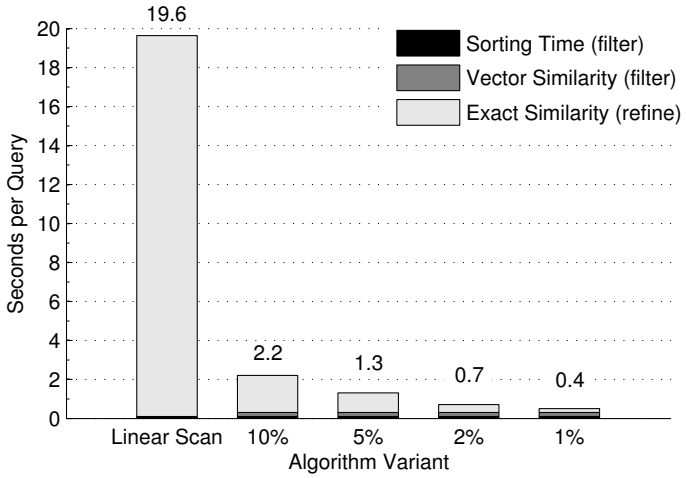


Figure 6.5: Comparison of the time it takes to query the 2.3 million song collection for nearest neighbors using a full scan compared to a scan using the proposed filter-and-refine method. A quad core CPU (2.5GHz) was used and all Gaussian similarity models were loaded to RAM.

the k-means clustering.

- Each song of the collection needs to be mapped to the vector space to use the filter-and-refine method. We use a mapping dimension of $k = 40$. To speedup the pivot object search we randomly sample 10% of the whole collection.

To allow a quick restarting of the system, we save to disk (i) the pivot objects for each dimension together with the vector mappings and (ii) the k-means centers together with each object's distance to the centers, and hold these in memory after the server was started. This allows fast restarting of the system and quick processing of new objects.

To query for similar objects we use the previously described filter-and-refine method, filtering out a predefined number (*filter-size*, a percentage of the collection size) of nearest neighbor candidates using the vector representation and refining the result with the exact PS similarity measure.

6.3.5 Performance

As high performance is a key objective, we finally measure the actual query response times of the system. The Wolperdinger server system is implemented in C++ and requires all similarity models to be loaded to RAM. In the case of the 2.3 million tracks 12 gigabytes of RAM are required to hold all music features in memory. To run the system we use a desktop PC with a quad core CPU (with 2.5 Ghz per core).

Figure 6.5 compares the query response times of four different *filter-size* settings (*filter-size*$= 10\%, 5\%, 2\%, 1\%$, $k = 40$) to a full linear scan using the PS music similarity measure on the 2.3 million songs collection. From that plot we can see that the final system is capable

of answering music recommendation queries in 0.68 s on a 2.3 million songs collection while returning about 90% of the correct nearest neighbors (Figure 6.3) compared to a linear scan, which takes 19.6 s on the same system. 92% 10-NN recall can be reached with 1.3 s queries and a *filter-size* of 5%. If 85%/75% 1/10-NN recall is acceptable, queries can be answered in 0.39 s.

At the point where a *filter-size* of 1% is used, the time to find and sort the vectors in the filter step is for the first time bigger (0.25 s) than the exact search step (0.14 s). Even faster query responses can be achieved if standard vector indexing techniques like kd-trees or locality-sensitive hashing are used to quickly search the vector space for candidates, thus reducing the time in the filter step.

6.3.6 User Interface

We finally use the server component which we developed to build a simple music recommendation web application on top of it. We integrate two functionalities on top of the recommendation system:

- a music and album recommendation service
- and an automatic radio station service.

The web application starts by displaying a text field which is used to search for music. A query searches for matching artists and album titles. Figure 6.6 shows a screenshot of the application using "blues" as a query. The search results are displayed in the bottom row as album covers in a cover flow-like arrangement. The mouse scroll wheel can be used to scroll through the covers. Music from the foremost album starts playing immediately.

When the button "Explore Album" is pressed, the music discovery screen loads. In our case we clicked on the album "Easy Blues" by Lafayette Leake. The screen shows all tracks from the album, a click on a song starts playing the selected track. If a song seems interesting to a user, the buttons "Album" (to use the album as seed) or "Playing Song" (to use the selected song as seed) can be used to discover more music using the recommendation engine.

Figure 6.7 shows a picture of the screen which is displayed if the user clicked on the "Playing Song" button. The explore screen shows eleven automatically recommended music pieces from different albums plus the initial album. By hovering over the album covers the recommended song from the albums is highlighted and can be played. A click on "Album" discovers similar albums, instead of individual songs. Any time a song is playing the user can click a button on the interface and use the currently playing song to dig deeper into the 2.3 million songs.

Figure 6.8 shows a screenshot the second functionality we integrated into the web application using the Wolperdinger service. When the "Radio" button is pressed on the explore screen the playing song is used to create a continuous radio station. The radio station uses the Wolperdinger server to select the songs on its playlist and tries to keep on playing music from the same style.

These functionalities show how easy high-performance content-based music recommendation could be integrated into a web-store for recommendations or used as a last.fm like radio station service.

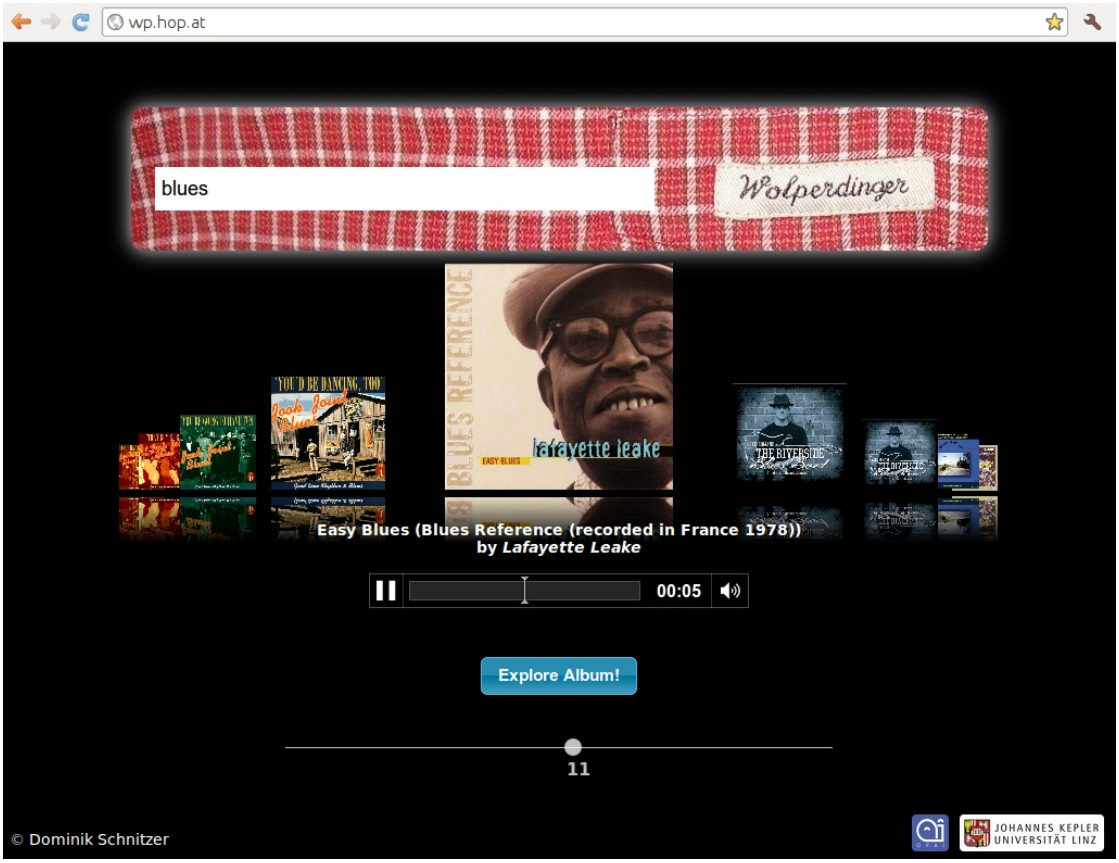


Figure 6.6: Screenshot of the Wolperdinger prototype music discovery. The search results are displayed in a cover flow like list in a standard web-browser.

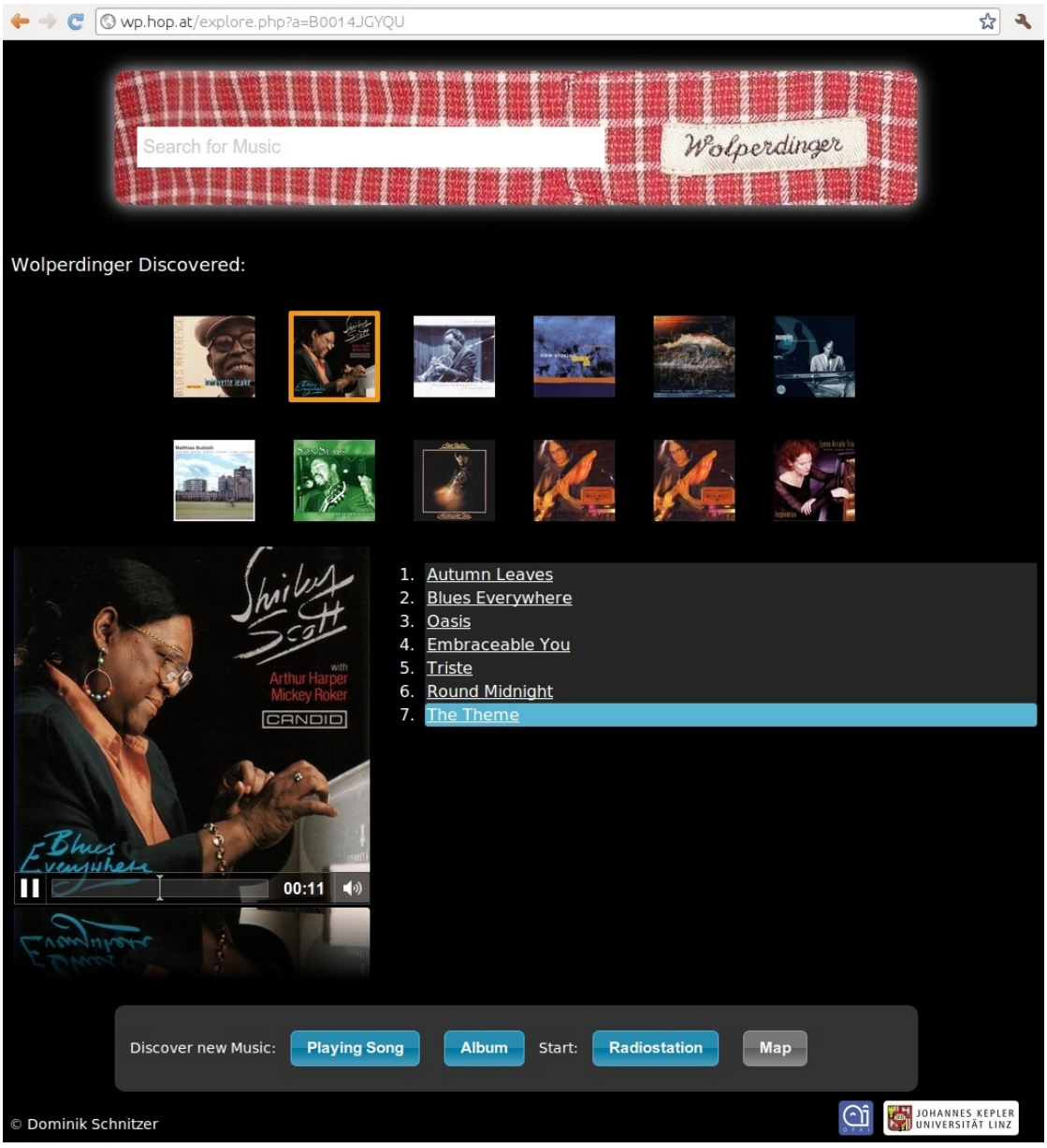


Figure 6.7: Screenshot of the Wolperdinger exploration screen. Twelve blues songs are recommended for a query song by Lafayette Leake.

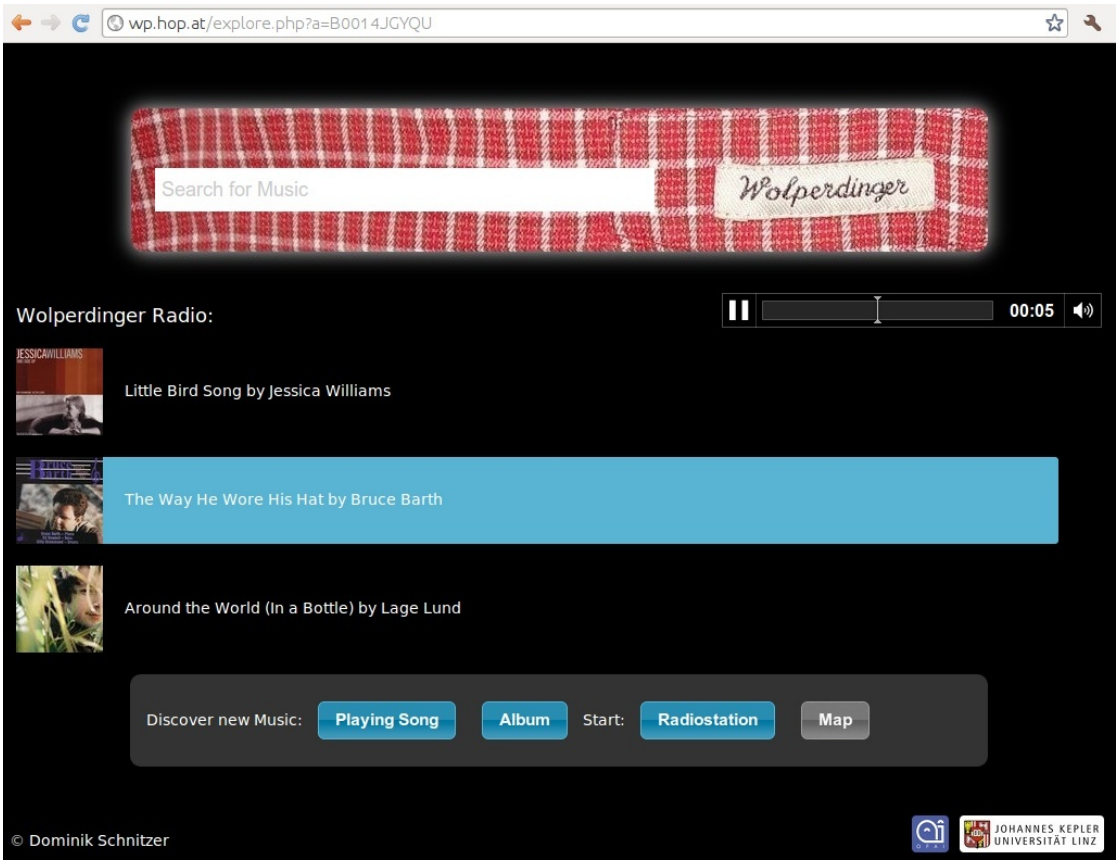


Figure 6.8: Screenshot of the Wolperdinger radio mode. A blues radio station was initialized.

6.4 Summary

This chapter integrated all methods presented in this thesis to finally build a music recommendation prototype engine working with 2.3 million songs. The proof-of-concept system is able to answer recommendation queries in less than one second, while still retrieving 90% of the true nearest neighbors using one of the currently top-rated music similarity measures. Finally we also presented a web application built on top of the recommendation service to demonstrate how the system could be used in commercial applications.

Chapter 7

Conclusions

This thesis identified three major problems prohibiting the use of state-of-the art music similarity measures in large scale recommendation systems: (i) the non-vectorial music similarity models and their non-metric similarity functions, (ii) the hub problem of the music similarity measures, and (iii) the lack of an employable indexing solution for fast similarity search. Solutions for these major problems defined the core of this thesis.

First the thesis showed the basics needed to understand how to properly use the multivariate Gaussian models and the divergences associated with this representation. We specifically show how the music similarity models can be natively and correctly used in standard centroid-computing algorithms, like the k-means algorithm. All previous works in music information retrieval avoided this step and artificially vectorized the features to use them in standard algorithms. Using these basic concepts, we developed a generalized self-organizing map algorithm which is able to natively cluster multivariate Gaussians. We showed that using the native features instead of a vector approximation yields higher quality maps with much less computational cost. We also published a toolbox which implements all methods introduced in this chapter and is freely available on the web.

In the second part we explored the problem of hubs and their impact on the covered class of music similarity algorithms. Hubs are objects which unwontedly keep appearing as nearest neighbors of a large part of objects. We showed that the problem of hubs is a general problem in machine learning and presented a method alleviating the problem. The method is called Mutual Proximity and works by symmetrizing nearest neighbor relations in a probabilistic framework. By using Mutual Proximity with a large number of standard machine learning databases we showed that the number of hubs is greatly reduced. At the same time we could show that measures like the classification accuracy of collections with previously high hubness increased.

In the third part we presented a filter-and-refine method for fast music similarity search. The method was specifically designed and evaluated for the music similarity algorithms which were presented in this thesis. The method modified the Fastmap algorithm so it can be efficiently used to map the features into the Euclidean vector-space which can be searched very fast in contrast to the original divergences. The method works by first searching the vector-space for possible nearest neighbors in the filter step, and then using the exact divergence to find the

nearest neighbors in the candidate objects. We could show that this method speeds up music similarity search by a factor of 10–40 compared to a linear scan, while still retrieving 90–99% of the true nearest neighbors.

The final chapter merges all the methods we have presented in this thesis to build a high-quality, large scale music recommendation prototype: "Wolperdinger". The prototype system uses an approximate variant of Mutual Proximity requiring a clustering of Gaussian features to improve the quality of the recommendation method. The filter-and-refine method is used to efficiently answer search queries. The completed prototype works on a demo collection of 2.3 million songs. It uses one of the top music similarity algorithms today and is able to answer queries in a fraction of a second on a standard computer.

This paragraph concludes the thesis which, as we hope, will contribute its share to a broader adoption of automatic music recommendation systems and consequently to novel ideas for interacting with large media archives.

Appendix A

Applications of Music Similarity

Contents

A.1 Music Recommendation in the Banshee Music Player 125
A.1.1 Discussion 127
A.2 The Intelligent iPod 127
A.2.1 Towards the Interface 127
A.2.2 Prototype 130
A.3 BeoSound 5 Stereo with MOTS 131

We present three working applications which use content-based music similarity technology. All three applications were developed in the course of this thesis and they show possible application scenarios for music similarity.

- *The first prototype shows how a content based playlist generation mechanisms can be integrated in a music player application.*
- *The second prototype shows how music similarity can be used on mobile devices with limited display sizes, processing power and interaction possibilities for navigation.*
- *The third application is a stereo audio player from Bang & Olufsen, which uses a content-based music recommendation engine we developed, to ensure that music never stops playing.*

A.1 Music Recommendation in the Banshee Music Player

Banshee[1] is an open-source desktop music player written in C#/Mono[2]. Although Banshee was designed for Unix-like environments, the audio player is already ported to multiple platforms.

[1] `http://banshee.fm/`, visited August 12th, 2011
[2] `http://www.mono-project.com/`, visited August 12th, 2011

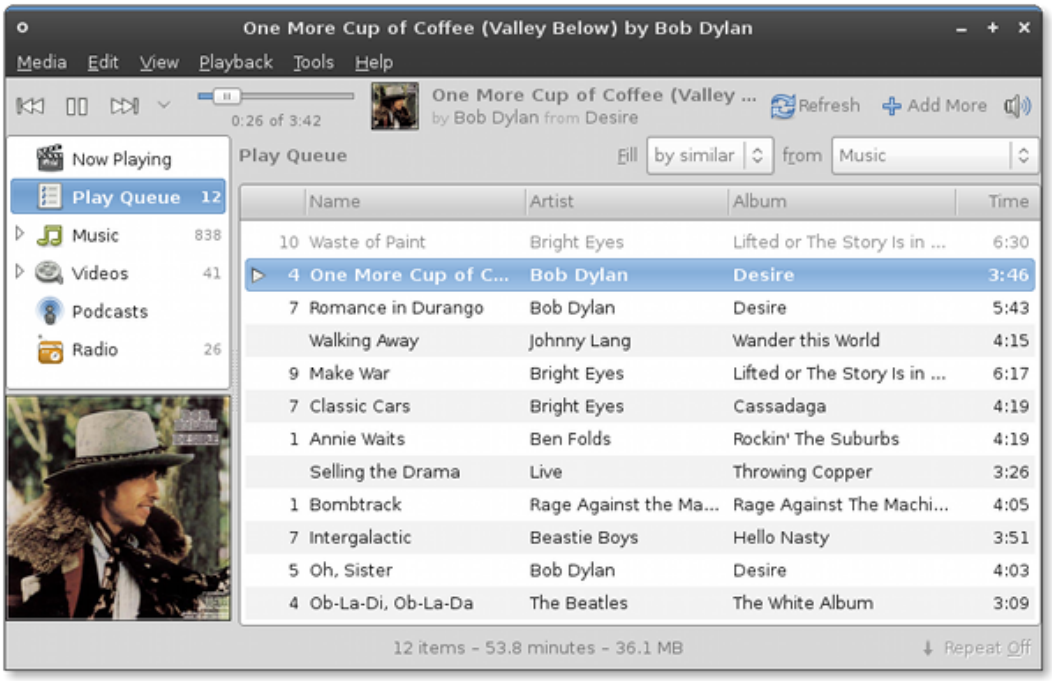


Figure A.1: A screen-shot of the Mirage Banshee extension. The playlist is continuously generated "by similarity".

To explore and demonstrate how the integration of a content-based music recommendation system into a standard digital audio player (DAP) application could work, a playlist generation plugin was developed. The Banshee DAP was chosen, as it comes with a powerful plugin architecture wherein the algorithms could be well integrated with the user-interface. The plugin is called *Mirage* [Sch07] and it is available freely on the web[3] under the terms of the GPL license:

The plugin was accepted in the *Banshee Community Extensions*[4] repository and is already available in many Linux distributions such as Ubuntu, Fedora (Red Hat) or openSUSE (Novell) Linux. Figure A.1 shows a screen-shot of Banshee during playlist generation with Mirage.

From a technical/MIR point of view Mirage implements the standard ME music similarity algorithm. The algorithm is discussed in depth in Chapter 2, Section 2.4.2 (see also [ME05]). To deal with the artist effect and avoid recommending only songs by the same artist, a partial artist filter is applied in the playlists. That is, if more than three songs are by the same artist as the seed song, songs by this artist are omitted in the playlist. Mirage supports two playlist generation techniques: standard and dynamic playlist generation.

- *Standard Playlist Generation*: Standard playlist generation creates a static playlist using a single seed song by searching the collection for similar music tracks using a simple linear

[3] `http://hop.at/mirage`, visited August 12th, 2011, since 2009 the plugin is co-maintained by Bertrand Lorentz.
[4] `http://banshee.fm/download/extensions/`, visited August 12th, 2011

scan. A static playlist consists of the twenty-five songs most similar to the query. If the collection does not change, the static playlists would not change either.

- *Dynamic Playlist Generation*: Dynamic playlist generation recomputes the playlist every time a previously recommended song finished playing. To compute the next recommendations the song heard last acts as new seed song. That way the similarity aspects of the last song are taken into account in contrary to a static standard playlist seeded with only one song. The plugin also takes skipping of songs into account. Songs which have been skipped are excluded from future playlists and are not used to advance the playlist.

A.1.1 Discussion

This first application shows a straightforward way to use content-based music similarity: A manually selected seed song is used to compute music playlists of similar music on-the-fly. These mixes can be saved or synced to mobile devices and make it easy for the user to effortless listen to music other than using random shuffle.

A.2 The Intelligent iPod

The music discovery prototype which is presented here is based on the metaphor of "The Wheel" from Pohle et al. [PPW05]. It shows how a user could be given a convenient and meaningful way to access her music on a mobile device. This section is based on our prototype and publication which presented the work in 2007 [SPKW07].

In Pohle's "Wheel" each position on a circular playlist-path is associated with a track in the collection. The circular arrangement is created by applying a *Traveling Salesman Problem (TSP)* algorithm to the collection's full music similarity matrix. As a result of the TSP, similar tracks, i.e., tracks with a small distance, are grouped together, so that certain regions of the wheel can be associated certain musical styles. This allows the user to select the kind of music she wants to listen to by simply turning the wheel into a certain region.

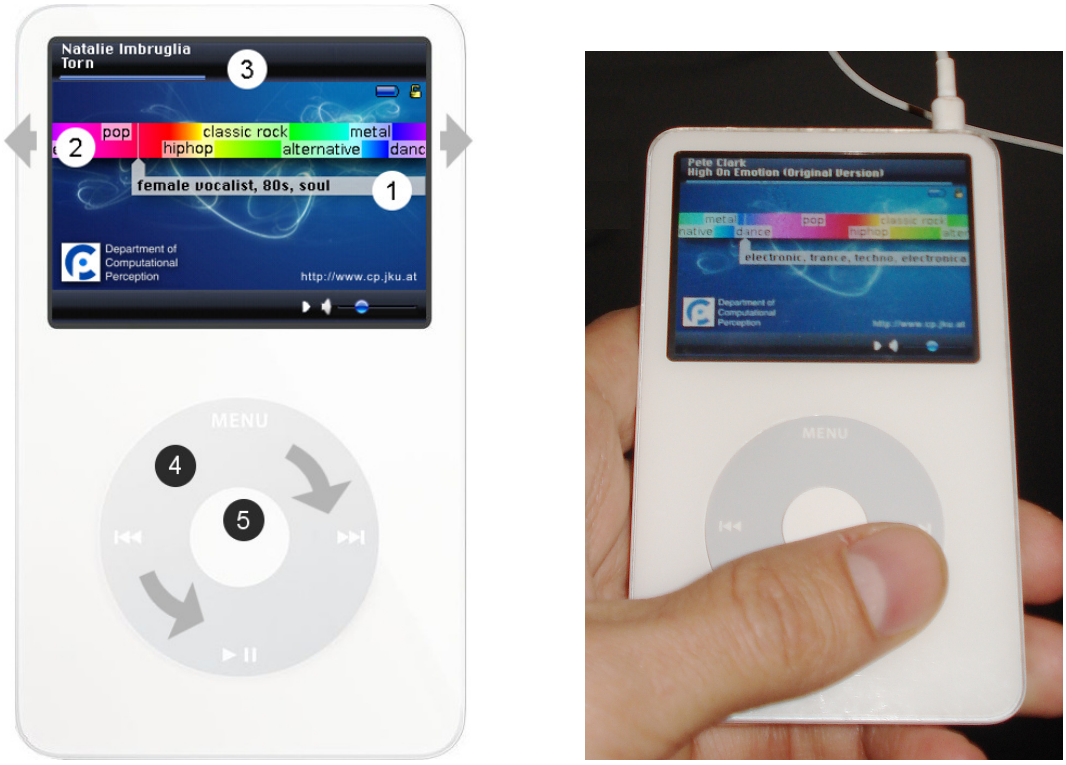
The prototype application presented in this section is implemented on an Apple iPod music player. It builds on the idea of a circular arrangement of songs by Pohle, and adds automatic labeling of the music in the playlist interface to support quick, one-touch navigation of music collections. The automatic labeling is done using community data gathered from last.fm.

A.2.1 Towards the Interface

This section describes the three steps required to build the proposed mobile music browsing interface: (i) the retrieval of artist-related community data, and (ii) the calculation of audio similarity between individual music pieces, to (iii) automatically construct and label the circular playlist.

A.2.1.1 Last.fm-based Artist Clustering

In a first step, we aim at obtaining descriptors from last.fm for all artists in the music collection on the mobile device. The artist tags collected from last.fm will be used to split the collection



(a) User Interface (b) Working prototype

Figure A.2: Pictures of the *Intelligent iPod*: In this interface the music is arranged on a continuous color bar. The bar displays the main music styles which were discovered for a music collection using last.fm artist information (2). The individual songs within a selected music style are arranged by acoustic music similarity. The color bar can be slid to the left/right using the scroll wheel (4). Additional music information is displayed in a text field (1) and updated while moving the bar. To listen to the music of a region, the user has to navigate to it and press the select button (5). A relevant song will start playing (3).

in large groups of similar artists.

For a specified artist, the last.fm tag data takes the form of a list of words with associated (normalized) weights. Based on this information, we cluster artists into (e.g., five) main clusters. Tracks with an unknown artist as well as tracks from artists that can not be found on last.fm, are put into the best-matching cluster based on the music similarity measure described in the next section. In each cluster the most frequent tag defines its name.

We implement a simple method to separate the artists into large clusters using the acquired tags from last.fm: In a first step, each artist is joined with artists sharing the same top ranked last.fm tag. Besides this initial assignment, each artist is also weighted by the number of associated songs in the collection. That way, weighted tag-bins of artists are formed. In a next step, the bin with the smallest weight is removed and its artists are spread among the remaining bins according to their second highest ranked tag. After that, again, the bin with the smallest weight is broken up. This procedure is repeated iteratively until the desired number of clusters is found. In the case of the prototype we are using five clusters.

An obvious drawback of this simple technique is that not all artists can be directly related to one of the main clusters since some artists are not assigned any of the top tags at all. The tracks by those artists can again be associated to a cluster using music similarity.

A.2.1.2 Playlist Generation

Once the top-level clusters are defined, and the association of each track to a particular cluster has been accomplished, two sub-steps need to be performed in order to create a circular arrangement of the collection. First, the placement of individual tracks within a particular cluster must be determined. Second, the different clusters have to be connected.

Both methods use the EP content-based music similarity measure (Chapter 2, Section 2.4.3) and require the pairwise distances between all music tracks in the collection to be computed.

Arranging Tracks in a Cluster via TSP

All tracks within a region should be arranged based on their audio similarity to obtain a smooth transition between consecutive tracks and subregions within a given cluster. Here, we adopt the idea from Pohle et al. [PKS+07] to use a Traveling Salesman Problem (TSP) algorithm to organize a given music collection. The "cities" to visit are the tracks, while distances between the tracks are calculated with the EP music similarity function. We use the *Minimum Spanning Tree (MST)*-based algorithm as proposed by Pohle et al. [PKS+07]. From the distance matrix of the tracks in a cluster, an MST is created. The tree is traversed in a depth-first order, and the tracks are written into a list in the order they are first visited. Finally, the first track is visited again to form a closed route. Thus, all tracks can be ordered into a circular playlist within each cluster.

Connecting the Intra-Cluster Playlists

Having a circular playlist for the tracks within each top-level cluster we are in need of a strategy to optimally connect them. The idea is to find an arrangement of clusters that reflects the

similarity of the contained music. Furthermore, we need to determine those tracks that are best suited as links between the clusters.

We decided to use a greedy approach: The output of the previous step is n circular routes, where n is the number of clusters. To merge them, each of the n circular playlists is broken at the largest "distance" (i.e., the two consecutive tracks in the playlist with the largest audio distance between them are searched). These two tracks become the endpoints of the (now linear) playlist. To connect these n linear playlists, the two most similar endpoints (according to audio similarity) from different clusters are connected, yielding $n - 1$ linear playlists. This procedure is repeated until all intra-cluster playlists are merged to one long circular playlist containing all tracks in the collection.

A.2.1.3 Automatic Region Labeling

To support an easier orientation within the music collection, the playlist created in the preceding steps is visualized and augmented with meaningful descriptors. For this purpose, we linearly arrange the whole playlist along a sliding bar that is colored according to the distribution of the previously identified clusters. Thus, different kinds of music are represented as different colors on the bar. Between clusters, colors are faded to support the impression of smooth transitions. Furthermore, the labels of the top-level clusters are displayed in the center of the corresponding cluster region on the color bar. While browsing, also additional tags that describe the currently viewed track are displayed. This allows for an immediate overview of the complete collection.

A.2.2 Prototype

By exploiting information on acoustic similarity and community-based music labels, a music collection can now be automatically structured and described to allow for easy orientation and navigation within the collection. As a consequence, regions of musical styles emerge. For demonstration, we implemented our prototype interface on an Apple iPod portable digital music player.

Our implementation of the music browsing interface directly integrates in the main mobile audio playback application and makes easy changing of music style possible, even during playback. Figure A.2 shows the finished prototype implementation and describes its main elements.

To understand how the player is used, the main focus should be put on the continuous color bar in the center of the screen. The color bar shows the six main music labels which were found to describe the collection. The labels are weighted and color-coded according to Section A.2.1.3. As such, the color bar alone allows a quick and coarse first glance on the music collection currently loaded. For clarity the number of labels displayed is limited to a maximum of six.

In the original Apple iPod player the scroll wheel is mainly used for controlling the playback volume. For our purpose we remapped the input of the scroll wheel to allow circular movement of the continuous color bar to select a specific area to play. A static pin overlaying the bar indicates the currently selected area. In addition to this, a text field displays last.fm tags which were found for the artists of the selected region.

As soon as the user selects a position on the color bar for playback, the selection is extrapolated on a song in the precomputed TSP playlist (see Section A.2.1.2). Since all songs in this playlist are arranged according to their maximum audio similarity, successive songs should fit together nicely, making the interface perfect for quick but matching on-the-go playlists.

Figure A.3: The Bang & Olufsen BeoSound 5 stereo in our office. The stereo integrates a content based music recommendation system to generate playlists.

A.3 BeoSound 5 Stereo with MOTS

A final example which we want to present here of how content-based music similarity can be used in applications is the Bang & Olufsen BeoSound 5 home stereo. A picture of the BeoSound 5 is shown in Figure A.3. The stereo uses a content-based music similarity to ensure that the music never stops playing. The similarity engine which was used is called MOTS, an acronym for "More Of The Same".[5]

The similarity engine was developed as a joint project of Bang & Olufsen and the Austrian research institute for artificial intelligence (OFAI)[6], my employer. I have been the main developer of the MOTS music similarity library. It is the final application presented in this work and shows how research results from the music information retrieval area are already picked up by industry and early adopters to create new products.

[5] `http://www.bang-olufsen.com/beosound5-digital-music-mots`, visited August 12th, 2011
[6] `http://www.ofai.at`, visited August 12th, 2011

Bibliography

[AAG11] Holzapfel A., Flexer A., and Widmer G. Improving tempo-sensitive and tempo-robust descriptors for rhythmic similarity. In *Proceedings of the 8th Sound and Music Computing Conference. SMC'11*, 2011.

[AASK04] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios. Boostmap: A method for efficient approximate similarity rankings. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, 2004.

[AH06] A. Andric and G. Haus. Automatic playlist generation based on tracking user's listening habits. *Multimedia Tools and Applications*, 29:127–151, 2006.

[AHK01] C. Aggarwal, A. Hinneburg, and D. Keim. On the Surprising Behavior of Distance Metrics in High Dimensional Space. In *Database Theory – ICDT 2001*, volume 1973 of *Lecture Notes in Computer Science*, pages 420–434. Springer Berlin / Heidelberg, 2001.

[AI06] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *47th Annual IEEE Symposium on Foundations of Computer Science. FOCS'06*, pages 459–468, 2006.

[AP02a] J. J. Aucouturier and F. Pachet. Music similarity measures: What's the use. In *3rd International Conference on Music Information Retrieval. ISMIR'02*, pages 157–163, 2002.

[AP02b] J. J. Aucouturier and F. Pachet. Scaling up music playlist generation. In *Proceedings of the IEEE International Conference on Multimedia and Expo. ICME '02*, 2002.

[AP03] J. J. Aucouturier and F. Pachet. Representing musical genre: A state of the art. *Journal of New Music Research*, pages 83–93, 2003.

[AP04] Jean-Julien Aucouturier and François Pachet. Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

[APPK08] V. Athitsos, M. Potamias, P. Papapetrou, and G. Kollios. Nearest neighbor retrieval using distance-based hashing. In *Proceedings of the IEEE 24th International Conference on Data Engineering. ICDE 2008*, pages 327–336, 2008.

[APRB07] J. J. Aucouturier, F. Pachet, P. Roy, and A. Beurivé. Signal + context = better classification. In *Proceedings of the 8th International Symposium of Music Information Retrieval. ISMIR'07*, 2007.

[BBF06] P. Bausch, J. Bumgardner, and C. Fake. *Flickr Hacks: Tips & Tools for Sharing Photos Online (Hacks)*. O'Reilly Media, Inc., 2006.

[Bel61a] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[Bel61b] Richard Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[BEL03] A. Berenzweig, D.P.W. Ellis, and S. Lawrence. Anchor space for classification and similarity measurement of music. *IEEE International Conference on Multimedia and Expo*, 1:29–32, 2003.

[Ben75] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18, 1975.

[Ber07] A. Berenzweig. *Anchors and hubs in audio-based music similarity*. PhD thesis, Columbia University, 2007.

[BFG99] K. P. Bennett, U. Fayyad, and D. Geiger. Density-based indexing for approximate nearest-neighbor queries. In *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*, pages 233–243. ACM, 1999.

[BLEW04] A. Berenzweig, B. Logan, D. P. W. Ellis, and B. P. W. Whitman. A large-scale evaluation of acoustic and subjective music-similarity measures. *Computer Music Journal*, 28(2):63–76, 2004.

[BM01] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pages 245–250, New York, NY, USA, 2001. ACM.

[BMDG05] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749, 2005.

[BP98] J. C. Bezdek and N. R. Pal. Some new indexes of cluster validity. *IEEE transactions on systems, man, and cybernetics*, 28(3):301, 1998.

[BPJ03] C.J.C. Burges, J.C. Platt, and S. Jana. Distortion discriminant analysis for audio fingerprinting. *IEEE Transactions on Speech and Audio Processing*, 11(3):165 – 174, May 2003.

[Bre67] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967.

[BYRN99a] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[BYRN99b] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[Cay08] L. Cayton. Fast nearest neighbor retrieval for bregman divergences. In *Proceedings of the 25th International Conference on Machine Learning. ICML'08*, pages 112–119. ACM, 2008.

[CCKB06] P. Cano, O. Celma, M. Koppenberger, and J. M. Buldú. Topology of music recommendation networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 16, 2006.

[Cel06] O. Celma. Foafing the music: Bridging the semantic gap in music recommendation. In *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 927–934. Springer Berlin / Heidelberg, 2006.

[Cel08] Òscar Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2008.

[CF00] W. W. Cohen and W. Fan. Web-collaborative filtering: Recommending music by crawling the web. *Computer Networks*, 33(1-6):685–698, 2000.

[CHU+08] C.-H. Chen, W. Härdle, A. Unwin, M. A. A. Cox, and T. F. Cox. Multidimensional scaling. In *Handbook of Data Visualization*, Springer Handbooks of Computational Statistics, pages 315–347. Springer Berlin Heidelberg, 2008.

[CHV99] O. Chapelle, P. Haffner, and V. N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999.

[CKGB02] P. Cano, M. Kaltenbrunner, F. Gouyon, and E. Batlle. On the use of fastmap for audio retrieval and browsing. In *Proceedings of the 3rd International Conference of Music Information Retrieval. ISMIR'02*, pages 275–276, 2002.

[CKW05] P. Cano, M. Koppenberger, and N. Wack. An industrial-strength content-based music recommendation system. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 673–673, New York, NY, USA, 2005. ACM.

[CL01] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

[CS06] M. Casey and M. Slaney. Song intersection by approximate nearest neighbor search. In *Proceedings of the 7th International Conference of Music Information Retrieval. ISMIR'06*, pages 144–149, 2006.

[CS07] M. Casey and M. Slaney. Fast recognition of remixed music audio. In *IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP 2007*, volume 4, April 2007.

[CZZM07] R. Cai, C. Zhang, L. Zhang, and W. Y. Ma. Scalable music recommendation by search. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 1065–1074, New York, NY, USA, 2007. ACM.

[DBC09] J.S. Downie, D. Byrd, and T. Crawford. Ten years of ismir: Reflections on challenges and opportunities. In *Proceedings of the 10th International Society for Music Information Retrieval Conference. ISMIR'09*, pages 13–18, 2009.

[DLM+98] George Doddington, Walter Liggett, Alvin Martin, Mark Przybocki, and Douglas Reynolds. Sheep, goats, lambs and wolves a statistical analysis of speaker performance in the nist 1998 speaker recognition evaluation. In *Proceedings of the 5th International Conference on Spoken Language Processing (ICSLP-98)*, 1998.

[Dow03] J. S. Downie. Music information retrieval. *Annual review of information science and technology*, 37(1):295–340, 2003.

[Dow08] J. Stephen Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[ELBMG07] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. *Advances in neural information processing systems*, 20:385–392, 2007.

[ES03] D.M. Endres and J.E. Schindelin. A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49(7):1858 – 1860, July 2003.

[ESK03] L. Ertöz, M. Steinbach, and V. Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of the third SIAM International Conference on Data Mining*, volume 3, 2003.

[EWBL02] D. P. W. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence. The quest for ground truth in musical artist similarity. In *Proceedings of the 3rd International Symposium on Music Information Retrieval. ISMIR'02*, 2002.

[FA10] A. Frank and A. Asuncion. Uci machine learning repository, 2010. http://archive.ics.uci.edu/ml.

[Far58] P. R. Farnsworth. *The social psychology of music.* Dryden, 1958.

[FG94] B. Feiten and S. Günzel. Automatic indexing of a sound database using self-organizing neural nets. *Computer Music Journal*, 18(3):53–65, 1994.

[FGS10] A. Flexer, M. Gasser, and D. Schnitzer. Limitations of interactive music recommendation based on audio content. In *Proceedings of the 5th Audio Mostly Conference: A Conference on Interaction with Sound*, AM '10, pages 13:1–13:7, New York, NY, USA, 2010. ACM.

[FL95] C. Faloutsos and K. I. Lin. Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, SIGMOD '95, pages 163–174, New York, NY, USA, 1995. ACM.

[FS10] A. Flexer and D. Schnitzer. Effects of album and artist filters in audio similarity computed for very large music databases. *Computer Music Journal*, 34(3):20–28, 2010.

[FSGP10a] A. Flexer, D. Schnitzer, M. Gasser, and T. Pohle. Combining features reduces hubness in audio similarity. In *Proceedings of the 11th International Society for Music Information Retrieval Conference. ISMIR'10*, 2010.

[FSGP10b] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Tim Pohle. Combining features reduces hubness in audio similarity. In *Proceedings of the Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010.

[FSGW08] A. Flexer, D. Schnitzer, M. Gasser, and G. Widmer. Playlist generation using start and end songs. In *Proceedings of the 9th International Conference of Music Information Retrieval. ISMIR'08*, 2008.

[FU01] J. Foote and S. Uchihashi. The beat spectrum: A new approach to rhythm analysis. *IEEE International Conference on Multimedia and Expo*, 2001.

[FWV07] D. Francois, V. Wertz, and M. Verleysen. The concentration of fractional distances. *IEEE Transactions on Knowledge and Data Engineering*, 19:873–886, 2007.

[GB03] S. Gunter and H. Bunke. Validation indices for graph clustering. *Pattern Recognition Letters*, 24(8):1107–1113, 2003.

[GDB08] V. Garcia, E. Debreuve, and M. Barlaud. Fast k nearest neighbor search using gpu. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshop'08*, 2008.

[GDPW04] F. Gouyon, S. Dixon, E. Pampalk, and G. Widmer. Evaluating rhythmic descriptors for musical genre classification. In *Audio Engineering Society Conference: 25th International Conference: Metadata for Audio*, pages 196–204, 2004.

[GFK05] T. Ganchev, N. Fakotakis, and G. Kokkinakis. Comparative evaluation of various mfcc implementations on the speaker verification task. In *Proceedings of 9th International Conference on Speech and Computer, SPECOM'05*, pages 191–194, 2005.

[GFS10] M. Gasser, A. Flexer, and D. Schnitzer. Hubs and orphans-an explorative approach. In *Proceedings of the 7th Sound and Music Computing Conference. SMC'10*, 2010.

[GLA+09] S. J. Green, P. Lamere, J. Alexander, F. Maillet, S. Kirk, J. Holt, J. Bourque, and X. W. Mak. Generating transparent, steerable recommendations from textual descriptions of items. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 281–284, New York, NY, USA, 2009. ACM.

[GNN10] V. Garcia, F. Nielsen, and R. Nock. Hierarchical gaussian mixture model. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing. ICASSP'10*. IEEE, 2010.

[HBD07] X. Hu, M. Bay, and J. S. Downie. Creating a simplified music mood classification ground-truth set. In *Proceedings of the 8th International Conference on Music Information Retrieval. ISMIR'07*, 2007.

[HK03] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system with an efficient search strategy. *Journal of New Music Research*, 32(2):211–221, 2003.

[HMM+05] H. Homburg, I. Mierswa, B. Möller, K. Morik, and M. Wurst. A benchmark dataset for audio classification and clustering. In *Proceedings of the 6th International Conference on Music Information Retrieval. ISMIR'05*, pages 528–31, 2005.

[HO07] J.R. Hershey and P.A. Olsen. Approximating the Kullback Leibler Divergence Between Gaussian Mixture Models. In *IEEE International Conference on Acoustics, Speech and Signal Processing. ICASSP 2007*, volume 4, pages IV–317 –IV–320, April 2007.

[HS09] A. Holzapfel and Y. Stylianou. A scale transform based method for rhythmic similarity of music. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 317–320, 2009.

[HUW05] A. Hicklin, B. Ulery, and C. I. Watson. *The myth of goats: How many people have fingerprints that are hard to match?* US Dept. of Commerce, National Institute of Standards and Technology, 2005.

[IFP10] IFPI. *IFPI Digital Music Report 2010 - Music how, when, where you want it*. IFPI, 2010.

[ISO93] ISO/IEC-11172-3. *Part 3: Audio: Information technology – Generic coding of moving pictures and associated audio information*. ISO, Geneva, Switzerland, 1993.

[JHS07] Herve Jegou, Hedi Harzallah, and Cordelia Schmid. A contextual dissimilarity measure for accurate and efficient image search. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[JLZ+02] Dan-Ning Jiang, Lie Lu, Hong-Jiang Zhang, Jian-Hua Tao, and Lian-Hong Cai. Music type classification by spectral contrast feature. In *Proceedings of the IEEE International Conference on Multimedia and Expo. ICME'02*, volume 1, pages 113 – 116, 2002.

[JP73] R. A. Jarvis and E. A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, pages 1025–1034, 1973.

[JSHV10] Herve Jegou, Cordelia Schmid, Hedi Harzallah, and Jakob Verbeek. Accurate image search using the contextual dissimilarity measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1), 2010.

[JTHW06] W. Jin, A. Tung, J. Han, and W. Wang. Ranking outliers using symmetric neighborhood relationship. In *Advances in Knowledge Discovery and Data Mining*, volume 3918 of *Lecture Notes in Computer Science*, pages 577–593. Springer Berlin / Heidelberg, 2006.

[Kas66] M. Kassler. Toward musical information retrieval. *Perspectives of New Music*, 4(2):59–67, 1966.

[KL51] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):pp. 79–86, 1951.

[Koh01] T. Kohonen. *Self-Organizing Maps*. Springer Series in Information Sciences. Springer, third edition, 2001.

[KPSW07] P. Knees, T. Pohle, M. Schedl, and G. Widmer. A music search engine built upon audio-based and web-based similarity measures. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 447–454, New York, NY, USA, 2007. ACM.

[KPW04] P. Knees, E. Pampalk, and G. Widmer. Artist classification with web-based data. In *Proceedings of 5th International Conference on Music Information Retrieval. ISMIR'04*, pages 517–524, 2004.

[KRNI10] Ioannis Karydis, Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Looking through the "glass ceiling": A conceptual framework for the problems of spectral similarity. In *Proceedings of the Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010.

[KSPW07] P. Knees, M. Schedl, T. Pohle, and G. Widmer. Exploring music collections in virtual landscapes. *IEEE Multimedia*, 14(3):46 –54, 2007.

[Lam08] P. Lamere. Social tagging and music information retrieval. *Journal of New Music Research*, 37(2):101–114, 2008.

[LB05] E. Levina and P. J. Bickel. Maximum likelihood estimation of intrinsic dimension. *Advances in Neural Information Processing Systems 17*, 48109:777–784, 2005.

[LGH08] C. Laurier, J. Grivolla, and P. Herrera. Multimodal Music Mood Classification Using Audio and Lyrics. In *Seventh International Conference on Machine Learning and Applications. ICMLA '08*, pages 688 –693, Dec. 2008.

[Lin91] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[LKM04] B. Logan, A. Kositsky, and P. Moreno. Semantic analysis of song lyrics. In *IEEE International Conference on Multimedia and Expo. ICME '04*, volume 2, pages 827–830 Vol.2, June 2004.

[LLZ03] D. Liu, L. Lu, and H. J. Zhang. Automatic mood detection from acoustic music data. In *Proceedings of the 4th International Symposium on Music Information Retrieval. ISMIR'03*, 2003.

[LO04] Tao Li and M. Ogihara. Content-based music similarity search and emotion detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP '04*, volume 5, May 2004.

[Log00] B. Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the first International Symposium on Music Information Retrieval. ISMIR'00*, 2000.

[Log02] B. Logan. Content-based playlist generation: Exploratory experiments. In *Proceedings of the 3rd International Symposium on Music Information Retrieval. ISMIR'02*, 2002.

[LR05] 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. ISMIR'05*, pages 34–41, 2005.

[LS01] B. Logan and A. Salomon. A music similarity function based on signal analysis. *IEEE International Conference on Multimedia and Expo*, 2001.

[LS06] M. Levy and M. Sandler. Lightweight measures for timbral similarity of musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, AMCMM '06, pages 27–36, New York, NY, USA, 2006. ACM.

[LT07] S. Leitich and M. Topf. Globe of music: Music library visualization using geosom. In *Proceedings of the 8th International Conference on Music Information Retrieval. ISMIR'07*, 2007.

[Lüb05] D. Lübbers. Sonixplorer: Combining visualization and auralization for content-based exploration of music collections. In *Proceedings of the 6th International Conference on Music Information Retrieval. ISMIR'05*, 2005.

[LVADC03] E. L. M. Law, L. Von Ahn, R. B. Dannenberg, and M. Crawford. Tagatune: A game for music and sound annotation. In *Proceedings of the 8th International Conference on Music Information Retrieval. ISMIR'07*, pages 361–364, 2003.

[ME05] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Conference on Music Information Retrieval ISMIR'05*, 2005.

[ME07] M. Mandel and D. P. Ellis. Labrosa's audio music similarity and classification submissions. In *International Symposium on Music Information Retrieval, MIREX Submission 2007*, 2007.

[MF04] C. McKay and I. Fujinaga. Automatic genre classification using large high-level musical feature sets. In *Proceedings of the 5th International Conference on Music Information Retrieval. ISMIR'04*, 2004.

[MMC+05] J. P. G. Mahedero, Á. MartÍnez, P. Cano, M. Koppenberger, and F. Gouyon. Natural language processing of lyrics. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pages 475–478, New York, NY, USA, 2005. ACM.

[MP88] M. Minsky and S. Papert. *Perceptrons*. MIT press, 1988.

[MUNS05] F. Mörchen, A. Ultsch, M. Nöcker, and C. Stamm. Databionic visualization of music collections according to perceptual distance. In *Proceedings of the 6th International Conference on Music Information Retrieval. ISMIR'05*, 2005.

[NBN07] F. Nielsen, J.-D. Boissonnat, and R. Nock. Bregman Voronoi Diagrams: Properties, Algorithms and Applications. Technical Report RR-6154, INRIA, 2007.

[NDW05] N. M. Norowi, S. Doraisamy, and R. Wirza. Factors affecting automatic genre classification: an investigation incorporating non-western musical forms. In *Proceedings of the 6th International Conference on Music Information Retrieval. ISMIR'05*, pages 13–20, 2005.

[NLR05] R. Neumayer, T. Lidy, and A. Rauber. Content-based Organization of Digital Audio Collections. In *Proceedings of the 5th Open Workshop of MUSICNETWORK*, 2005.

[NN09] F. Nielsen and R. Nock. Clustering multivariate normal distributions. *Emerging Trends in Visual Computing*, pages 164–174, 2009.

[OMKS08] N. Ono, K. Miyamoto, H. Kameoka, and S. Sagayama. A real-time equalizer of harmonic and percussive components in music signals. In *Proceedings of the 9th International Conference of Music Information Retrieval. ISMIR'09*, 2008.

[PA04] F. Pachet and J. J. Aucouturier. Improving timbre similarity: How high is the sky? *Journal of negative results in speech and audio sciences*, 1(1), 2004.

[Pam03] E. Pampalk. Islands of music: Analysis, organization, and visualization of music archives. *Journal of the Austrian Society for Artificial Intelligence*, 22(4):20–23, 2003.

[Pam04] E. Pampalk. A matlab toolbox to compute music similarity from audio. In *Proceedings of the 5th International Conference on Music Information Retrieval. ISMIR'04*, pages 254–257, 2004.

[Pam06] E. Pampalk. *Computational models of music similarity and their application in music information retrieval*. PhD thesis, Vienna University of Technology, 2006.

[PBK08] I. Panagakis, E. Benetos, and C. Kotropoulos. Music genre classification: A multilinear approach. In *Proceedings of the Ninth International Conference on Music Information Retrieval. ISMIR'08*, 2008.

[PC00] F. Pachet and D. Cazaly. A taxonomy of musical genres. In *Proceedings of the 6th Conference on Content-Based Multimedia Information Access. RIAO'00*, pages 1238–1245, 2000.

[PDW03] Elias Pampalk, Simon Dixon, and Gerhard Widmer. On the Evaluation of Perceptual Similarity Measures for Music. In *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)*, pages 7–12, 2003.

[PFW05] 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. ISMIR'05*, 2005.

[PHH04] E. Pampalk, P. Hlavac, and P. Herrera. Hierarchical organization and visualization of drum sample libraries. In *Proceedings of the 7th International Conference of Digital Audio Effects. DAFx-04*, pages 378–383, 2004.

[PHVG02] P. Pérez, C. Hue, J. Vermaak, and M. Gangnet. Color-based probabilistic tracking. In *Computer Vision – ECCV 2002*, volume 2350 of *Lecture Notes in Computer Science*, pages 661–675. Springer Berlin / Heidelberg, 2002.

[PKS+07] T. Pohle, P. Knees, M. Schedl, E. Pampalk, and G. Widmer. Reinventing the wheel: A novel approach to music player interfaces. *IEEE Transactions on Multimedia*, 9(3):567–575, 2007.

[PKSW06] T. Pohle, P. Knees, M. Schedl, and G. Widmer. Automatically adapting the structure of audio similarity spaces. In *Proceedings of the 1st Workshop on Learning the Semantics of Audio Signals. LSAS'06*, 2006.

[Poh10] T. Pohle. *Automatic Characterization of Music for Intuitive Retrieval.* PhD thesis, Johannes Kepler University Linz, 2010.

[PPW05] 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. DAFx-05*, 2005.

[PRM02a] E. Pampalk, A. Rauber, and D. Merkl. Content-based organization and visualization of music archives. In *Proceedings of the tenth ACM international conference on Multimedia*, MULTIMEDIA '02, pages 570–579, New York, NY, USA, 2002. ACM.

[PRM02b] E. Pampalk, A. Rauber, and D. Merkl. Using Smoothed Data Histograms for Cluster Visualization in Self-Organizing Maps. In José Dorronsoro, editor, *Artificial Neural Networks – ICANN 2002*, volume 2415 of *Lecture Notes in Computer Science*, pages 81–81. Springer Berlin / Heidelberg, 2002.

[PSS+09] T. Pohle, D. Schnitzer, M. Schedl, P. Knees, and G. Widmer. On rhythm and general music similarity. In *Proceedings of the 10th International Conference on Music Information Retrieval. ISMIR'09*, 2009.

[RAPB05] P. Roy, J. J. Aucouturier, F. Pachet, and A. Beurive. Exploiting the tradeoff between precision and cpu-time to speed up nearest neighbor search. In *Proceedings of the 6th International Conference on Music Information Retrieval. ISMIR'05*, 2005.

[RF01] A. Rauber and M. Frühwirth. Automatically analyzing and organizing music archives. In *Research and Advanced Technology for Digital Libraries*, volume 2163 of *Lecture Notes in Computer Science*, pages 402–414. Springer Berlin / Heidelberg, 2001.

[RNI10] M. Radovanović, A. Nanopoulos, and M. Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *The Journal of Machine Learning Research*, 11:2487–2531, 2010.

[Rus80] J. A. Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6), 1980.

[Sal97] Steven Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1:317–328, 1997.

[Sch07] D. Schnitzer. Mirage – high-performance music similarity computation and automatic playlist generation. Master's thesis, Vienna University of Technology, 2007.

[Sch10] Markus Schedl. On the Use of Microblogging Posts for Similarity Estimation and Artist Labeling. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, Utrecht, the Netherlands, August 2010.

[SDI06] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice (Neural Information Processing)*. MIT press, 2006.

[SFSW11] D. Schnitzer, A. Flexer, M. Scheld, and G. Widmer. Using Mutual Proximity to Inprove Content-Based Audio Similarity. In *Proceedings of the 12th International Conference on Music Information Retrieval. ISMIR'11*, 2011.

[SFW09] D. Schnitzer, A. Flexer, and G. Widmer. A Filter-and-Refine Indexing Method for Fast Similarity Search in Millions of Music Tracks. In *Proceedings of the 9th International Conference on Music Information Retrieval. ISMIR'09*, pages 537–542, 2009.

[SFW10] D. Schnitzer, A. Flexer, and G. Widmer. A fast audio similarity retrieval method for millions of music tracks. *Multimedia Tools and Applications*, pages 1–18, 2010.

[SFWG10] D. Schnitzer, A. Flexer, G. Widmer, and M. Gasser. Islands of Gaussians: The Self Organizing Map and Gaussian Music Similarity Features. In *Proceedings of the 11th International Conference on Music Information Retrieval. ISMIR'10*, 2010.

[SJKCK08] C. N. Silla Jr, A. L. Koerich, P. Catholic, and C. A. A. Kaestner. The latin music database. In *Proceedings of the 9th International Conference of Music Information Retrieval. ISMIR'08*, page 451, 2008.

[SK09] M. Schedl and P. Knees. Context-based music similarity estimation. In *Proceedings of the 3rd International Workshop on Learning Semantics of Audio Signals. LSAS'09*, 2009.

[Sla93] M. Slaney. Auditory toolbox. Technical report, Apple Computer Company: Apple Technical Report, 1993.

[SPKW06] M. Schedl, T. Pohle, P. Knees, and G. Widmer. Assigning and visualizing music genres by web-based co-occurrence analysis. In *Proceedings of the 7th International Conference on Music Information Retrieval. ISMIR'06*, 2006.

[SPKW07] D. Schnitzer, T. Pohle, P. Knees, and G. Widmer. One-touch access to music on mobile devices. In *Proceedings of the 6th international conference on Mobile and ubiquitous multimedia. MUM'07*, pages 103–109. ACM, 2007.

[Ste36] S. S. Stevens. A scale for the measurement of a psychological magnitude: loudness. *Psychological Review*, 43(5):405, 1936.

[SW09] Y. Shavitt and U. Weinsberg. Song Clustering Using Peer-to-Peer Co-occurrences. *International Symposium on Multimedia*, pages 471–476, 2009.

[SWK08] K. Seyerlehner, G. Widmer, and P. Knees. Frame level audio similarity-a codebook approach. In *Proceedings of the 11th International Conference on Digital Audio Effects. DAFx-08*, 2008.

[SWP10] K. Seyerlehner, G. Widmer, and T. Pohle. Fusing block-level features for music similarity estimation. In *Proceedings of the 13th International Conference on Digital Audio Effects. DAFx-10*, 2010.

[TC99] G. Tzanetakis and P. Cook. Marsyas: a framework for audio analysis. *Organised sound*, 4:169–175, December 1999.

[TC02] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293 – 302, July 2002.

[TEC01] G. Tzanetakis, G. Essl, and P. Cook. Audio analysis using the discrete wavelet transform. In *Proceedings of the International Conference in Acoustics and Music Theory Applications*, 2001.

[Vel02] R. Veldhuis. The centroid of the symmetrical kullback-leibler distance. *IEEE Signal Processing Letters*, 9(3):96 –99, March 2002.

[WBKW96] E. Wold, T. Blum, D. Keislar, and J. Wheaten. Content-based classification, search, and retrieval of audio. *Multimedia, IEEE*, 3(3):27–36, 1996.

[WL02] B. Whitman and S. Lawrence. Inferring descriptions and similarity for music from community metadata. In *Proceedings of the 2002 International Computer Music Conference. ICMC'02*, pages 591–598, 2002.

[WT06] Y. Wu and M. Takatsuka. Spherical self-organizing map using efficient indexed geodesic data structure. *Neural Networks*, 19:900–910, 2006.

[WWSZ05] J. T. L. Wang, X. Wang, D. Shasha, and K. Zhang. Metricmap: an embedding technique for processing distance-based queries in metric spaces. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(5):973–987, 2005.

[XDDD98] W. Xu, J. Duchateau, K. Demuynck, and I. Dologlou. A new approach to merging gaussian densities in large vocabulary continuous speech recognition. In *IEEE Benelux Signal Processing Symposium. SPS'98*, pages 231–234, 1998.

[Yia93] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, SODA '93, pages 311–321, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[ZP05] Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems*, volume 17, pages 1601–1608. MIT Press, Cambridge, MA, 2005.

Curriculum Vitae of the Author

Dipl. Ing. Dominik Schnitzer
Born on July 18, 1981 in Vienna, Austria.

Current Position

- Research Assistent
  Austrian Research Institute for Artificial Intelligence (OFAI)
  Freyung 6/6, A-1010 Wien, Austria
  Phone: (+43-1) - 5336112 - 21
  WWW: http://www.ofai.at/~dominik.schnitzer/
  Email: dominik.schnitzer@ofai.at

- Reserarch Assistent
  Department of Computational Perception (CP)
  Johannes Kepler University, Linz
  Altenbergerstrasse 69, 4040 Linz, Austria

Main areas of research

- Music information retrieval, large scale data processing and search, machine learning, audio signal processing

Academic career

- May 2007 - Present: Research Assistent at the Department of Computational Perception, Johannes Kepler University Linz
- October 2006 - Present: Research Assistent at the Intelligent Music Processing Group at the OFAI
- M.S in Computer Sciences - Intelligent Systems, Technical University of Vienna, March 2007
- B.S in Software and Information Engineering, Technical University of Vienna, February 2005

Academic Prizes and Recognition

- 2009: 2nd place in the international Capture The Flag (iCTF) Security Contest of the University of Santa Barbara, CA.
- 2007: Best Student Paper Award at the ACM International Conference on Mobile and Ubiquitous Multimedia (MUM'07)

Publications

Patents

Schnitzer D., Method and a system for identifying similar audio tracks, US Patent Application 12/458,230, July 2009.

Journal Articles

Schnitzer D., Flexer A., Widmer G.: A Fast Audio Similarity Retrieval Method for Millions of Music Tracks, Multimedia Tools and Applications, in press, published online December 2010.

Flexer A., Schnitzer D.: Effects of Album and Artist Filters in Audio Similarity Computed for Very Large Music Databases, Computer Music Journal, Volume 34, Number 3, pp. 20-28, 2010.

Knees, P. and Pohle, T. and Schedl, M. and Schnitzer, D. and Seyerlehner, K.: A document-centered approach to a natural language music search engine, Lecture Notes in Computer Science, Springer, pp. 627-631, 2008.

Papers in Refereed Conference Proceedings

Schnitzer D., Flexer A., Schedl M., Widmer G.: Using Mutual Proximity to Improve Content-Based Audio Similarity, Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR'11), Miami, FL, 2011 (to appear).

Schnitzer D., Flexer A., Widmer G., Gasser M.: Islands of Gaussians: The Self Organizing Map and Gaussian Music Similarity Features, Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR'10), Utrect, NL, 2010.

Schnitzer D., Flexer A., Widmer G.: A Filter-and-Refine Indexing Method for Fast Similarity Search in Millions of Music Tracks, in Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR'09), Kobe, Japan, 2009.

Schnitzer D., Pohle T., Knees P., Widmer G.: One-touch access to music on mobile devices, Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia (MUM'07), ACM New York, NY, USA, 2007.

Flexer A., Schnitzer D., Gasser M., Pohle T.: Combining features reduces hubness in audio similarity, Proceedings of the Eleventh International Society for Music Information Retrieval Conference (ISMIR'10), Utrecht, NL, 2010.

Flexer A., Gasser M., Schnitzer D.: Limitations of interactive music recommendation based on audio content, Proceedings of the 5th Audio Mostly Conference: A Conference on Interaction with Sound, pp. 96-102, 2010.

Schedl, M., Seyerlehner, K., Schnitzer, D., Widmer, G., and Schiketanz, C.: Three Web-based Heuristics to Determine a Person's or Institution's Country of Origin. In Proceedings of the 33rd Annual ACM SIGIR Conference (SIGIR 2010), Posters and Demos, Geneva, Switzerland, 2010.

Gasser M., Flexer A., Schnitzer D.: Hubs and Orphans - an Explorative Approach, Proceedings of the 7th Sound and Music Computing Conference (SMC'10), Barcelona, Spain, 2010.

P. Knees, T. Pohle, M. Schedl, D. Schnitzer, K. Seyerlehner, and G. Widmer.: Augmenting Text-Based Music Retrieval with Audio Similarity, Proceedings of the tenth International Society for Music Information Retrieval Conference (ISMIR09), Kobe, Japan, 2009.

Seyerlehner, K. and Pohle, T. and Widmer, G. and Schnitzer, D.: Informed Selection of Frames for Music Similarity Computation, Proceedings of the 12th International Conference on Digital Audio Effects (DAFx'09), Como, Italy, 2009.

T. Pohle, D. Schnitzer, M. Schedl, P. Knees, and G. Widmer: On Rhythm and General Music Similarity Proceedings of the 10th International Society for Music Information Retrieval

Conference (ISMIR'09), Kobe, Japan, 2009.

K. Seyerlehner, P. Knees, D. Schnitzer, and G. Widmer: Browsing Music Recommendation Networks Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR'09), Kobe, Japan, 2009.

Flexer A., Schnitzer D.: Album and Artist Effects for Audio Similarity at the Scale of the Web, in Proceedings of the 6th Sound and Music Computing Conference (SMC'09), Porto, Portugal, 2009.

P. Knees, T. Pohle, M. Schedl, D. Schnitzer, and K. Seyerlehner: A Document-centered Approach to a Natural Language Music Search Engine, Proceedings of the 30th European Conference on Information Retrieval (ECIR08), Glasgow, Scotland, UK, March 30-April 3, 2008.

Flexer A., Schnitzer D., Gasser M., Widmer G.: Playlist Generation using Start and End Songs, in Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR'08), Philiadelphia, USA, 2008.

Seyerlehner K., Widmer G., Schnitzer D.: From Rhythm Patterns to Perceived Tempo, Proceedings of the eighth International Conference on Music Information Retrieval (ISMIR'07), Vienna, Austria, 2007.