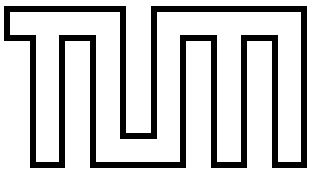
Technische Universität München
Fakultät für Informatik

Master's Thesis in Informatics

Unsupervised Audio Feature Extraction for Music Similarity Estimation

Jan Schlüter



Technische Universität München
Fakultät für Informatik

Master's Thesis in Informatics

Unsupervised Audio Feature Extraction for Music Similarity Estimation

Ein Musikähnlichkeitsmaß auf Basis unüberwacht gelernter Audiomerkmale

Jan Schlüter

Supervisor:
Prof. Dr. Alois Knoll

Advisor:
Dipl.-Inf. Christian Osendorfer

October 8, 2011

There is a theory which states that if ever anybody discovers exactly what the Universe is for and why it is here, it will instantly disappear and be replaced by something even more bizarre and inexplicable. There is another theory which states that this has already happened.

Douglas Adams

Declaration of Originality

I assure the single handed composition of this master's thesis only supported by declared resources.

Eigenständigkeitserklärung

Ich versichere, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Wien, den Unterschrift:

Acknowledgements

Foremost, I would like to thank *Christian Osendorfer* for introducing me to the art of machine learning at the Technical University of Munich, for giving me the opportunity to work on Music Information Retrieval, and for pointing me to mcRBMs before I had even understood RBMs. I am also grateful to *Breno Faria*, with whom I enjoyed studying probabilities and likelihoods, priors and posteriors, classification and regression, kernel tricks and feature spaces, NumPy and SciPy, Büchi and van Emde Boas, top-fermented and bottom-fermented beer.

Furthermore, I wish to thank *Prof. Gerhard Widmer* for calling me to Vienna for a first project while I was in the thick of writing up this thesis, then giving me the time I needed to finish it. I am already looking forward to pursuing further research in the realms of machine learning with music at the OFAI – and even to writing the next thesis!

Finally, I want to thank my parents for supporting me in all my endeavors, even if they drove me farther and farther away from home, and my friends and my brother, who never let me down.

Abstract

Fostered by the constant advancement of digital technologies, both catalogs of music distributors and personal music collections have grown to sizes that call for automated methods to manage them. In this context, music similarity estimation plays an important role: It can be used to recommend music based on examples, to organize a collection into groups, or to generate well-sounding playlists. Content-based methods are especially interesting in that they rely on sound only, not requiring any metadata.

Most existing content-based music similarity estimation methods build on complex hand-crafted feature extractors, which are difficult to engineer. As an alternative, unsupervised machine learning allows to learn features empirically from data. Based on a review of existing approaches to music similarity estimation and related tasks in three other domains of pattern recognition, we design a music similarity estimation system based on unsupervisedly learned audio features.

In particular, we train a recently proposed generative model, the mean-covariance Restricted Boltzmann Machine (mcRBM), on music spectrogram excerpts and employ it for feature extraction. In experiments on three public datasets, it clearly outperforms classic MFCC-based methods, surpasses simple unsupervised feature extraction using basic RBMs or k-Means and comes close to the state-of-the-art. This shows that unsupervised feature extraction poses a viable alternative to engineered features.

Zusammenfassung

Begünstigt durch die stetige Weiterentwicklung digitaler Technologien haben sowohl das Angebot der Musikhändler als auch persönliche Musiksammlungen inzwischen Größen erreicht, die ohne automatische Methoden kaum zu handhaben sind. Musikähnlichkeitsmaße spielen in diesem Zusammenhang eine wichtige Rolle: Sie können dazu verwendet werden, Musik auf Basis von Beispielen zu empfehlen, eine Sammlung in Gruppen aufzuteilen, oder wohlklingende Wiedergabelisten zu generieren. Inhaltsbasierte Methoden sind besonders interessant, da sie ausschließlich auf dem Klang aufbauen und keine Metadaten benötigen.

Die meisten existierenden inhaltsbasierten Musikähnlichkeitsmaße basieren auf komplizierten, von Hand erstellten Merkmalsextraktoren, die schwierig zu entwickeln sind. Eine Alternative bieten unüberwachte maschinelle Lernmethoden, die Merkmale empirisch aus Daten lernen können. Ausgehend von einer Begutachtung existierender Musikähnlichkeitsmaße sowie dazu verwandter Methoden aus drei anderen Gebieten der Mustererkennung entwickeln wir ein Musikähnlichkeitsmaß auf Basis unüberwacht gelernter Audiomerkmale.

Insbesondere trainieren wir ein vor kurzem entwickeltes generatives Modell, die mean-covariance Restricted Boltzmann Machine (mcRBM), auf Auschnitten aus Musikspektrogrammen und setzen es zur Merkmalsextraktion ein. In Experimenten auf drei frei verfügbaren Datensätzen ist es klassischen MFCC-basierten Ansätzen klar überlegen, übertrifft einfachere unüberwachte Merkmalsextraktion mit gewöhnlichen RBMs oder k-Means und kommt nahe an den aktuellen Stand der Technik. Das zeigt, dass unüberwachte Merkmalsextraktion eine praktikable Alternative zu von Hand entwickelten Features darstellt.

Contents

List of Figures

List of Tables

1 Introduction

Nowadays, music gets produced and published at a higher rate than any individual can listen to it: Estimates reach from a yearly "11,000 (nonclassical) major label albums averaging some ten cuts per album" [Vog04, p. 261] up to 97,751 albums released in the United States in 2009, as reported by Nielsen SoundScan[1].

In this chapter, we will explain that this creates a need for automatic music recommendation systems, motivate how content-based music similarity estimation would help in this context and point out further useful applications, briefly reason why unsupervised machine learning methods are appealing for this task and finish with an outline of the main parts of this thesis.

1.1 Content-based music similarity estimation – what for?

1.1.1 Automated music recommendation

According to the estimations given above, the corpus of available music is growing at an extraordinary rate of possibly over five hours of newly released music per hour. As a consequence, any music listener has to rely on filters to provide a selection of music she or he likes. A listener looking for new music traditionally only had limited influence on these filters, such as by choosing a radio station to listen to or a shelve in a music store to look through, and was largely dependant on others' choices, such as the station's editorial's decisions on the playlist, or the selection and sorting of records by staff in the store.

Digital technologies have changed this situation in at least two respects: Digital music distribution channels such as iTunes or Amazon can provide quick access to millions of music pieces with low expenses, so they are less strictly filtered, and, with the abandonment of physical records, they shifted granularity from albums to single tracks, making it even harder for potential customers to make a choice. To fill in this gap of missing filters, automatic music recommendation systems have arisen, attempting to provide highly individually targeted access to the world's music.

[1] `http://www.billboard.biz/bbbiz/content_display/industry/news/e3i4ad94ea6265fac02d4c813c0b6a93ca2`

1.1.2 Established music recommendation systems

An informal study among university students (aged 20 to 25, German or Austrian, experienced Internet users, but not familiar with Music Information Retrieval, Signal Processing or Computer Science) resulted in a list of six Internet platforms that are commonly associated with music recommendation. In the following, we will briefly review how these systems work to show in which respect industry's state-of-the-art still falls short of satisfyingly solving the problem at hands. See [Sca09, pp. 9–15] for an alternative and more extensive review of some of these services.

Amazon suggests albums or songs based on what has been purchased in the same order or by the same customers as items one searched for or bought. This is a form of *collaborative filtering* [HKBR99], which assumes that users who have agreed in the past (in their purchase decisions) will also agree in the future (by purchasing the same items).

Collaborative filtering generally suffers from two related problems: The *cold-start problem* and the *popularity bias*. In this case, the former means that albums that have not yet been purchased by anybody can never be suggested, and the latter describes that for any given item, popular albums are more likely to have been purchased in conjunction with it than unpopular ones, and so have a better chance of being recommended. In consequence, collaborative filtering is incapable of suggesting new music releases. An additional problem specific to Amazon is that users may purchase items for somebody else (e.g., as a present), which will flaw the recommendations generated both for them and for other users of allegedly the same taste.

Spotify, a music streaming service, bases its recommendations on its users' listening behavior, analyzing which artists are often played by the same listeners. While this may potentially result in better suggestions than analyzing sparse data such as record purchases, it is again subject to the cold-start problem and popularity bias. Furthermore, Spotify only recommends related artists, which is rather unspecific.

Genius is a function in Apple iTunes which generates playlists and song recommendations by comparing music libraries, purchase histories and playlists of all its users, possibly integrating external sources of information. Assuming such external information does not play a major role, this system is based on collaborative filtering and will suffer from its problems.

Last.fm combines information obtained from users' listening behavior and user-supplied tags (words or short expressions describing a song or artist). While tags can help making recommendations transparent to users – e.g., a user listening to a love song may be suggested other tracks that have frequently been

tagged as "slow" and "romantic" – they are easily manipulated by malicious users, requiring counter measures [LFZ09], and are affected by the cold-start problem and popularity bias.

Pandora, another music streaming service, recommends songs from its catalog based on expert reviews of tracks with respect to a few hundred genre-specific criteria. This allows very accurate suggestions of songs that sound similar to what a user listens to, including sophisticated explanations for why a song was suggested – e.g., a track may be recommended because it is in "major key", features "acoustic rhythm guitars", "a subtle use of vocal harmony" and exhibits "punk influences" [2]. As it is based on analyzing the musical content rather than collaborative filtering of users' consumption or unrestricted tag descriptions, it is not affected by the popularity bias. However, the expert reviews incur high costs in terms of time and money: An annotation time of 45 minutes per track [Sca09, p. 45] makes it infeasible to extend the catalog at a rate that can keep up with new releases, limiting the selection of music available to users similarly to the cold-start problem.

Shazam has often been named as well. While it analyzes the sound of music tracks, the original Shazam service actually does not provide any recommendations, but only finds the exact same track in its database to provide the user with metadata such as the artist or title of the song. The audio features extracted by the system (constellations of peaks in the spectrogram [Wan03]) are tailored to solving this task under very noisy conditions, but not useful for finding and recommending songs that sound *similar* rather than identical.

1.1.3 Use of content-based music similarity estimation

1.1.3.1 Large-scale recommendation

Despite these recommendation systems, only a small fraction of new albums attract a wider audience: According to Nielsen SoundScan, only 2.1% of 2009's releases sold more than 5,000 copies, and 12 releases (0.01%) sold more than a million. Album sales thus follow what is commonly known as a Power Law (or Pareto) distribution, where very few items sell very well, while the other items remain in the *Long Tail* and hardly sell at all [And06]. This means that either an overwhelming majority of today's music sounds bad, or a lot of good music still never gets found by interested listeners (or a combination of both).

We assume that at least two factors contribute to the latter: Firstly, music recommendation systems are not yet widespread enough to dominate album sale

[2] song description from `http://www.pandora.com/music/song/shins/kissing+lipless`

statistics, and secondly, all common systems except for Pandora are based on collaborative filtering and are thus susceptible to popularity bias, which encourages a Pareto distribution rather than giving users easy access to the Long Tail.

Revealing music pieces hidden in the Long Tail that suit the taste of a user seemingly needs a recommendation system that is not based on collaborative filtering. While Pandora is such a system, it requires costly experts and so can only handle a limited catalog of music. Audio content-based music similarity estimation attempts to automate the process of listening to music pieces and inferring information about their musical style or structure using digital signal processing, statistics and machine learning. Recommendation systems based on these methods are – just like human expert recommendations – not prone to popularity bias [CC08], making them a viable alternative for finding music that would otherwise pass by unnoticed.

Of course, audio content-based methods can also be combined with collaborative filtering to form hybrid systems that leverage the "hive mind" [Kro05] while alleviating the cold-start problem.

1.1.3.2 Small-scale recommendation

Digital technologies have not only changed the commercial distribution of music, but also the collection and listening habits of users. Widespread music compression methods such as Mp3 or Ogg Vorbis foster mass transmission, storage and transportation of music, allowing users to grow large private collections and carry around hundreds of albums in their pockets. Interestingly, a Pareto distribution arises in this scale as well: Analyzing over 5,600 iPod users' listening behaviors, [Lam06] found that of the average 3,542 songs in a collection, about 23% received 80% of plays, while 64% of the songs were not even played a single time. As it is hard to imagine that users dislike more than half of the songs they chose to copy to their portable music player, a reasonable conclusion is that private music collections nowadays are too large for people to manage manually. Content-based music recommendation could help (re-)discovering forgotten music in the Long Tail.

1.1.3.3 Playlist generation

Closely connected to the previous issue, but also applicable on a larger scale is the use of content-based similarity measures for automatic playlist generation. Given a song to start from, choosing perceptually similar songs in a row will yield a playlist with smooth transitions, in contrast to current music player's common *shuffle* function [Log02]. By also defining a target song, a system could generate a playlist by finding a path in the similarity space connecting both given songs [FSGW08]. Another interesting application is creating a circular playlist of all tracks in a collection with a Traveling Salesman Problem solver, allowing for easy

access of its parts, grouped by similarity [PPW05].

1.1.3.4 Scientific advancement

All preceding motivations focus on applications, but working on content-based music similarity is just as interesting from a purely scientific perspective.

As we will elaborate in Chapter 3, many methods employed in music similarity estimation are inspired by or borrowed from other fields of pattern recognition working on speech, images or text documents. In a similar manner, advancements in understanding and comparing music may prove to be useful for speech recognition, object recognition in images or text document retrieval (cf. [Auc06, pp. 51–52] on "Cross-Fertilisation").

Last, but not least, it is not yet fully understood how exactly humans perceive music. Building a working system in a computer, possibly even using biologically-inspired methods, could help building a reasonable model of how the human brain processes music.

1.2 Unsupervised Machine Learning for Feature Extraction

1.2.1 Hand-Crafting vs. Machine Learning

The first step of every audio content-based music similarity estimation system consists in extracting features from the audio signal (cf. Section 2.1). Most such feature extractors have been manually designed for capturing a specific aspect of music its developer deemed important, such as the instrumentation, pitch, key, or rhythm, and then evaluated to find out how well they represented the intended aspect and if it helped in finding similar music pieces.

Cambouropoulus terms this approach *knowledge engineering* [Cam98, p. 33]. The alternative is *empirical induction* from a given set of music pieces, which can be practically achieved by employing machine learning methods. Supervised machine learning attempts to algorithmically find patterns in audio data that are suited best for discriminating between or predicting high-level descriptions, such as the musical key. This way features targeted towards specific goals can be found empirically, ideally without designers needing to reason or guess how to extract them. However, data usually still needs to be properly preprocessed for machine learning algorithms to work, and most importantly, supervised machine learning needs ground truth data (in this example, the musical key for a large number of music pieces) to learn from.

1.2.2 Unsupervised Feature Extraction

It has been conjectured that humans are capable of inferring structure from data without any given ground truth to guide learning. For example, Fiser asserts that "while many processes of human knowledge acquisition are goal-directed and also rely on explicit external error-measures, the first step toward these more cognitive types of memory formations is a dimension-reducing unsupervised learning process" [Fis09]. Specifically focusing on perception learning, Abdallah concludes from a review of related literature that the goals of perception include an "efficient representation of relevant information through a process of redundancy reduction" and "the construction of statistical models of sensory data". He points out that the latter "can in principle be achieved by data-driven unsupervised learning" [Abd02, p. 30] and in fact, unsupervised techniques have been shown to yield feature extractors comparable to those found in most mammals' primary visual cortex [O$^+$96, BS97] and cochlea [Lew00]. Abdallah suggests to apply this to music, assuming "that musical structure is inherent in musical data, and that any mental structures that we use to process music are there in response to the structure of the data." [Abd02, p. 30]

Unsupervised machine learning thus seems to be an interesting alternative for learning feature extractors. Even if Abdallah's assumption should be wrong and our musical understanding partly stems from congenital auditory abilities[3] formed by evolution (which is supervised learning guided by natural selection), we assume in this thesis that some relevant musical structure can be learned in an unsupervised fashion. Compared to supervised methods, a possible disadvantage is that there is little to no control about which features will emerge. But on the plus side, no ground truth data is needed, just lots of music files which are relatively easy to obtain, and designers do not need any domain knowledge, permitting the use on any kind of music. And from a scientific perspective, unsupervisedly trained feature extractors would not only give a model of how humans might perceive music, but even show how they might *learn* to do so.

1.3 Thesis Outline

In the following chapter, we will review a range of proposed content-based music similarity measures from the literature to understand their general architecture and design space, to introduce several common feature extractors and to learn from their successes and failures.

Chapter 3 briefly explores three other domains of pattern recognition and information retrieval, analyzing similarities and differences to music processing and

[3] For example, infants have repeatedly been shown to remember and recognize melodies weeks before their birth [GDBR$^+$11].

highlighting methods potentially useful for music similarity estimation.

Based on our insights from Chapters 2 and 3, we propose a new system for music similarity estimation in Chapter 4. Its main component is a recent method for unsupervised feature extraction, the mean-covariance Restricted Boltzmann Machine, which has already proven useful for processing both images and speech, but has never been applied to music. To achieve a certain level of self-containment, we fully describe this component as well as any other methods used that we cannot assume to be commonly known.

In Chapter 5, after establishing the methodology for evaluating music similarity measures, we perform a range of experiments starting with a baseline system and gradually changing it into our target system, closely evaluating the effects of each step to show how each of our ideas contributes to the final result. We observe a clear improvement over the baseline system. For two datasets of Western popular and classical music, our results are closely behind a hand-crafted state-of-the-art approach, on a third dataset of ballroom music our approach performs moderately well in comparison to the best results reported in literature.

We will finish the thesis with a review of our achievements and an outlook on future work in Chapter 6.

2 Existing Content-based Music Similarity Measures

Although content-based music similarity estimation seems to only slowly find its way into commercial applications (e.g., the first hi-fi system capable of automatically creating playlists based on audio content rather than metadata has been released in 2009), research in this area commenced at least as early as 1997 [Foo97] and has since seen an ever-growing interest, as the increasing number and length of publications of the yearly ISMIR[1] conference show [DBC09]. This provides us with a vast body of literature to learn from.

This chapter will give a very selective review of particularly interesting attempts at building music similarity measures, pointing out their advantages and drawbacks. The first section describes the common architecture of most music similarity estimation systems, the second section exemplarily analyzes a classic approach in great detail to establish an understanding of the different parts, and the remainder of the chapter deals with several successors and alternatives to this method. Even if most of these are hand-crafted rather than machine learned, their study will help us in building our own system.

2.1 General Scheme

The great majority of systems follow the same basic structure depicted in Figure 2.1, which we will further describe below.

Representation of Songs Each music piece, a function of time to amplitude (or to a tuple of amplitudes for multiple channels), is digitally represented as a finite sequence of discrete samples (or tuples of samples) at a fixed sampling rate[2]. Typical sampling rates range from 11 to 44 kHz, and each sample (of each channel) can usually take one of 2^{16} to 2^{48} amplitude values. For convenience, we will disregard the quantization of amplitudes and assume real-valued amplitudes in the interval $[-1, +1]$ instead. Unless noted otherwise, we will additionally restrict ourselves to

[1] International Society for Music Information Retrieval

[2] Symbolic representations of music (such as scores, lyrics or MIDI data) and methods for their comparison are not dealt with in this thesis.

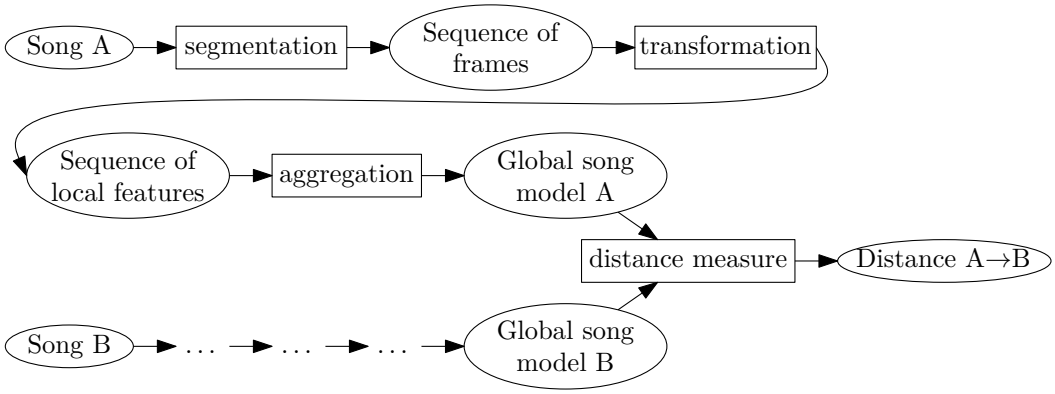


Figure 2.1: The common architecture of most content-based music similarity estimation systems. Ellipses denote data, blocks denote transformations.

single-channel (mono) recordings, so each song can be regarded as a single discrete-time finite-domain real-valued one-dimensional signal.

Local Feature Extraction Rather than handling a whole music piece at once (e.g. a 4-million-dimensional real-valued vector for a 3-minute mono track in 22 kHz), systems first divide a song into smaller, usually fixed-size, possibly overlapping parts often termed *frames*, which are then transformed separately or in small groups of neighboring elements to obtain a sequence of local features.

While this segmentation is necessary for some feature transformations either to reduce the variability of input data and yield useful features or plainly to reach computational feasibility, an even better justification for performing local feature extraction is that humans (whose sense for music we try to capture) also process music in short excerpts – as songs are signals *in time*, nobody can perceive a complete music piece at once[3].

Building a Global Song Model Although a song could be readily described by its sequence of local features, this is impractical: Local features are typically on the order of 10^1 to 10^3 floating point values, at a frame rate of up to 50 Hz, which makes the sequence of features not much smaller or even larger as the original audio signal. However, much of this information is redundant (think of reused sound fragments or long-term repetitions) or not relevant for similarity computations (such as the exact order of all features, or their exact values). It is therefore useful to abstract from

[3] Of course this does not mean that computers could not profit from being able to process a whole song at once, but it shows that this ability is not necessary for understanding music the way humans do.

the actual sequence of local features by building a more compact global model of the song, which captures relevant aspects of the sequence and discards unnecessary details. A simple example would be taking the average of all local features, but more sophisticated statistical models will be presented and discussed in the following sections.

Comparing Global Models To finally estimate the similarity of two music pieces, their global models have to be compared. Depending on the type of the model, many different distance measures are applicable, the use of which always has to be evaluated in conjunction with the chosen type of global model and the local features it is based on. The calculated distance between two global models usually cannot be interpreted by itself, but can subsequently be compared to other distances to permit conclusions of the form "Song A is more similar to Song B than to Song C", "Songs X, Y, Z are this collection's most similar songs to Song A", or even to produce a low-dimensional visualization of a collection of songs representing their respective distances.

2.2 Classic Approach: GMMs of MFCCs

One of the best-known approaches is the one of Aucouturier and Pachet [AP02b]. While it is not the first attempt at modelling music similarity, it is one of the most-cited. Furthermore, it is a very instructive example, allowing us to learn a lot about the parts that make up a music similarity estimation system.

Aucouturier and Pachet position their approach as a method for estimating *timbre similarity*. Timbre can be defined as "that attribute of auditory sensation whereby a listener can judge that two sounds are dissimilar using any criterion other than pitch, loudness and duration" [PD76]. While this excludes the pitch, melody or rhythm, it captures important musical qualities such as the instrumentation and singing, and a timbre component plays a fundamental role in state-of-the-art methods for estimating general music similarity [ADP07].

2.2.1 Local Features: Mel-Frequency Cepstral Coefficients

Originally based on an idea for analyzing time series data for echoes [BHT63], Mel-Frequency Cepstral Coefficients (MFCCs) were recognized to be useful features for speech recognition [DM80] and are now a standard feature in this field [RJ93]. Usage of MFCCs for Music Similarity Estimation has been pioneered by [Foo97], but only [Log00] claims to perform the first systematic evaluation of whether assumptions behind MFCCs actually apply to music (we will review her results in Section 2.2.4.1). Since then, MFCCs have become an established audio feature in the MIR community.

In the following, we will describe the computation steps of MFCCs along with their motivation for speech recognition and an attempt at interpreting what they do to a polyphonic music signal. A formal definition of MFCCs will be given in Section 4.2 (p. 48), more details on their justification can be found in [RJ93].

Segmentation into frames: First, short sequences of samples called *frames* are extracted from the signal, typically on a scale of tens of milliseconds and with 50% overlap. The goal is to work on excerpts that are short enough to be assumed *stationary* signals, i.e., signals for which the statistics do not change over time. Practically, the most important aspect is for the excerpts to have a more or less constant frequency distribution over time, as time information within the excerpt will be lost in a later step.

Hamming windowing: Each frame is multiplied with a Hamming window to minimize spectral leakage of energy into neighboring frequency bins in the next step. This is standard practice in computing a Short-Time Fourier Transform (STFT); for further details please see [Har87].

Discrete Fourier Transform: A DFT is applied to each frame, to obtain the distribution of energy into frequencies (along with their phases). Regarding speech as a convolution of a speaker-dependent glottal excitation signal with an (ideally) speaker-independent vocal tract response (the *source-filter model*), the DFT turns a speech signal into a product of the speaker-dependent part (the voice) and speaker-independent part (the articulated phone) [RS78]. To a certain extent, the DFT might also separate the pitch of a note from the instrument it is played with.

Magnitude spectrum: Phase information is discarded by taking the absolute value of each (complex-valued) frequency bin. While the original assumption of the human ear's general insensitivity to relative phases of components of complex tones [Ohm43] has been proven wrong, the ear is "phase deaf" under some circumstances (see [Moo02] for a review of results), indicating that the magnitude may be more important than the phase.

Mel scaling: Frequencies are mapped to a perceptual pitch scale, the so-called *Mel scale*. Human's perception of pitch has been found to be linear in frequency up to about 1000 Hz, above which pitch perception is logarithmic [SVN37], i.e., multiplications of frequency by a constant factor are perceived as linear pitch increments (for example, octaves in music appear linearly-spaced but are actually doublings of frequency). This reduces the frequency resolution at higher frequencies, which are less relevant for speech recognition. For music, it results in a fixed spacing pattern of harmonics (and linearly spaced Western scale notes), which might be useful for transposition-independent

features. Mel scaling is also used for dimensionality reduction: the frequency bins of the magnitude spectrum (usually between 256 and 1024) are mapped to relatively few mel bands (e.g., 40), justified by the fact that the human ear only distinguishes few so-called *critical bands* as well [FZ07, p. 159].

Log: The log function is applied to the mel magnitudes.[4] This can be motivated from two perspectives: Firstly, loudness perception of humans is logarithmic, and secondly, the logarithm transforms the product of the two parts of speech (the source and the filter, see above) into an addition which is easier to separate apart.

Discrete Cosine Transform: A discrete cosine transform is applied to each frame of mel log magnitudes, of which the highest coefficients are discarded (e.g., keeping the first 20). Again, this can be justified in different ways: Seen as a Fourier Transform for real signals, the DCT finds periodicities in the log-magnitude spectrum (the result is referred to as the signal's *cepstrum*, an anagram of "spectrum" coined in [BHT63]) such as harmonics, which were emphasized by the log function. As these will tend to occur in relatively small offsets in the spectrum, omitting the higher DCT components will discard them. For speech, this tends to remove the speaker-dependent part of the signal [RS78], and at least for monophonic music, it yields a certain level of pitch-independency [JCEJ09]. Keeping only the first DCT components can also be regarded as determining the coarse spectral envelope of the signal. Interestingly, the DCT basis functions also resemble the first PCA components of spectra of audio signals [Log00], so the DCT achieves an approximate decorrelation of the cepstral frames, and dropping the highest components compresses the data.

In summary, MFCCs describe the spectral envelope of short frames of a signal. The algorithm leaves a lot of hyperparameters to be chosen: frame length, frame rate (amount of overlap), minimum and maximum frequency to be considered when mapping to mel bands, number of mel bands, number of DCT coefficients to keep, and whether to keep the first coefficient (the *DC offset* representing the total energy of a frame). Different authors have come to different implementations [SPLS06], and Aucouturier and Pachet [AF04] try to find an optimal setting for their setup in a large set of grid search experiments (one of their results being that the optimal number of coefficients lies between 11 and 29, leading many subsequent authors to use 20 coefficients in reference to [AF04]).

[4] In some early publications, the logarithm was applied before mapping the spectrum to the mel scale [Foo97] [Log00, p.2]. This gives slightly different results, and in our preliminary experiments it had a negative impact on similarity estimations, if any. More recent publications apply the steps in the order presented here [SPLS06, Poh10, Sey10].

2.2.2 Global Model: Gaussian Mixture Model

To represent a song, Aucouturier and Pachet approximate the statistical distribution of its MFCC vectors with a Gaussian Mixture Model (GMM). A GMM is a weighted sum of K Gaussian distributions. Sampling this generative model can be understood as picking one of the Gaussians from a multinomial distribution with probabilities $\boldsymbol{\pi}$, then picking a sample from this Gaussian. The probability density function is

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}), \tag{2.1}$$

where

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})} \tag{2.2}$$

is the d-dimensional multivariate Gaussian probability distribution function with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

The parameters $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\mu_1}, \ldots \boldsymbol{\mu_K}, \boldsymbol{\Sigma_1} \ldots \boldsymbol{\Sigma_K})$, that is, the prior probabilities $\boldsymbol{\pi}$ and the parameters of the Gaussians, are adapted to maximize the model's likelihood under the data of a song via an *Expectation Maximization* algorithm (see [Bis06, Sec. 9.2.2]). This can be seen as modeling the density of the data in the feature space, or as clustering the data, completely discarding the original order of MFCC vectors in the song. The goal is to obtain compact representations of songs to compare against each other.

While not stated in [AP02b] and [AF04], side notes in [AP02a, Sec. 2.4.1] and [Auc06, p. 230] indicate that the Gaussian components were restricted to diagonal covariance matrices, which reduces both computational costs and memory needs. The number of components K was initially chosen to be 3 in [AP02b], but the grid search experiments of [AF04] showed better performance for $K = 50$.

2.2.3 Distance Measure: Kullback-Leibler Divergence

The GMMs of two songs being two probability distributions, an obvious idea is to compare them using the Kullback-Leibler divergence, which measures the degree to which two probability distributions match. For distributions P and Q over a continuous random variable X, it is defined as

$$KL(P\|Q) = \int_{-\infty}^{\infty} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x}, \tag{2.3}$$

where p and q are the respective probability density functions defining P and Q. While this is not a metric distance measure, it is at least nonnegative, fulfills

$KL(P\|Q) = 0 \Leftrightarrow P = Q$ and can be easily symmetrized as

$$KL_2(P,Q) = \frac{1}{2}KL(P\|Q) + \frac{1}{2}KL(Q\|P), \tag{2.4}$$

making it a useful indicator for the similarity of two song models.

Unfortunately, no closed form for the Kullback-Leibler divergence of two GMMs exists. However, we can perform a Monte-Carlo approximation, replacing the integral over $p(\boldsymbol{x})$ by an average over a set of samples X_P from P:

$$\int_{-\infty}^{\infty} p(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} d\boldsymbol{x} \approx \frac{1}{|X_P|} \sum_{\boldsymbol{x} \in X_P} \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \tag{2.5}$$

Substituting (2.5) into (2.3) and the result into (2.4), we obtain an approximation of the symmetric Kullback-Leibler divergence as

$$\begin{aligned} 2 \cdot KL_2(P,Q) &\approx \sum_{\boldsymbol{x} \in X_P} \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} + \sum_{\boldsymbol{x} \in X_Q} \log \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} \\ &= \sum_{\boldsymbol{x} \in X_P} \log p(\boldsymbol{x}) + \sum_{\boldsymbol{x} \in X_Q} \log q(\boldsymbol{x}) - \sum_{\boldsymbol{x} \in X_P} \log q(\boldsymbol{x}) - \sum_{\boldsymbol{x} \in X_Q} \log p(\boldsymbol{x}), \end{aligned} \tag{2.6}$$

where p and q are the two models' probability densities according to Equation 2.1. (The same derivation can be found in [Auc06, pp. 36–38], albeit starting from a wrong definition of the Kullback-Leibler divergence.)

Comparing two GMMs P and Q thus reduces to sampling from both models and computing a number of log likelihoods. The higher the number of samples, the more accurate the approximation of the divergence, at the expense of computation time. In the experiments of [AF04], Aucouturier and Pachet see an improvement in timbre similarity estimations by increasing the sample size up to 1000, above which results do not notably improve. For further experiments, they still chose a sample size of 2000, which they also used in [AP02b].

2.2.4 Shortcomings of this Method

While Aucouturier and Pachet achieve good results in retrieval experiments on a database of 350 music titles manually split into clusters of similar timbre [AF04], there are a number of theoretical and practical issues with their approach. Not questioning the general architecture (Section 2.1), we will again discuss the local feature extraction, global modeling and comparison method separately.

2.2.4.1 MFCCs

The use of MFCCs in the MIR community was promoted by [Log00], who showed that (a) the logarithmic mel scale performs not worse than a linear scale on a speech/music discrimination task and (b) the first few PCA components of frames of music bear some visual resemblance to the first components of a DCT. Both results do not indicate whether MFCCs are suited well for music similarity estimations, and there are several reasons to doubt that this is the case. In particular, many of the justifications for the computational steps are only valid for speech.

Cepstrum: Regarding the original signal as a convolution of an excitation signal with a filter response, the cepstrum (the spectrum of the log magnitude spectrum) helps separating the two parts, as explained above. The underlying *source-filter model* applies to speech and possibly some music instruments, but does not carry over to the case of polyphonic music. Even for single instruments, it has been shown that MFCCs fail to separate pitch from instrument class [Sey10, pp. 102–103]. It is therefore unclear if the cepstrum is useful for modeling timbre, or music in general.

Magnitude spectrum: MFCCs discard the phase spectrum of the frames, so playing a frame backwards will not change the features assigned to it [Auc06, p. 109]. For stationary signals, or sufficiently short frames extracted from music, this reflects the hearing capabilities of humans (e.g., even a sawtooth wave sounds identical when reversed), but for longer frames including a change in pitch or loudness this invariance is unjustifiable (except when asserting that such reversed music frames never occur in real data).

Log magnitudes: Taking the log of the magnitude spectrum mimics the loudness perception of humans. However, it prevents the easy separation of a linear combination of multiple sources (such as the different instruments in a polyphonic recording), which could be a disadvantage for modeling music.

Discrete Cosine Transform for decorrelation: One of the arguments for applying a DCT as the last computation step is that it approximates the Karhunen-Loève Transform (KLT) or PCA for music signals, which optimally decorrelates the data. Logan [Log00] did not systematically verify this, but only visually compared a plot of the first few PCA components estimated from 100 Beatles songs to the first basis functions of the DCT. To the best of our knowledge, later publications did not examine this matter any closer, neither theoretically nor empirically. A possible reason is that MFCCs appear to work well, and at least in early publications, the low computational costs of a DCT in comparison to training and applying a PCA were essential for being able to conduct large-scale experiments at all.

Earlier theoretical results [HP76, SGP+95, BYR06] show that as the frame size tends to ∞, the KLT/PCA indeed approximates the DCT if the covariance between components decays exponentially with their distance (i.e., the covariance matrix is a special form of Toeplitz matrix[5] with entries of the form $C_{ij} = \rho^{|i-j|}$ for some constant ρ). If and how well spectral frames of music tracks satisfy these conditions can only be evaluated empirically, however.

As decorrelation of spectral frames is an important step in our proposed music similarity estimation system as well, we will perform a set of experiments to better understand if the DCT really is a good choice for decorrelation and, more importantly, how this choice affects music similarity estimation performance (Section 5.3, p. 92).

2.2.4.2 GMMs

In representing a song with a GMM approximating the distribution of frame-wise low-level features such as MFCCs, one makes at least two questionable assumptions:

1. Audio frames within a song are *exchangeable*, i.e., their order does not matter, and they are statistically independent (*Bag of Frames*, cf. [ADP07]). The latter assumption is already violated by the mere fact that frames overlap in time, and the former assumption results in any permutation of frames to be assigned the same global representation. Combined with the invariance of MFCCs to the reversal of frames (Section 2.2.4.1), playing a song backwards does not change its global descriptor.

2. The most statistically significant frames of a song are the most relevant [Auc06, p. 1]. A song representation will be influenced the most by frames that occur often in the song and thus need to be modeled well by the GMM to achieve a high likelihood, while exceptional frames will be largely ignored in the model. It is not clear if this reflects which frames human listeners perceive as being relevant to the sound of a music piece – possibly, exceptional frames are just the ones that strike out to a listener and should be modeled better, or statistical significance is not related to perceptual relevance at all.

Some of the approaches discussed in Section 2.3 attempt to weaken these assumptions. Regarding the assumption of exchangeability, authors have tried to include temporal information at the feature extraction stage (by including the first-order and second-order derivatives of frame-wise features, or by calculating local features over blocks of multiple consecutive frames) or in the global models (by modeling statistics over segments of data rather than frame-wise features, or by explicitly

[5] A Toeplitz matrix is a matrix with entries A_{ij} depending only on $i - j$, i.e., a matrix with constant diagonals.

modeling dynamics, e.g., with a Hidden Markov Model). To counter the second assumption, some authors employ methods for finding and modeling only supposedly interesting frames, such as frames around note onsets.

2.2.4.3 KL Divergence

As explained in Section 2.2.3, two GMMs are compared by a Monte-Carlo approximation of their Kullback-Leibler divergence. This poses several problems:

1. Calculating the approximation involves sampling a large set of points from both models, as well as computing the log likelihood for each point under both models. Each such likelihood computation requires evaluating the probability distribution functions of all Gaussian components of the mixture model, which, in the case of full-covariance Gaussians, involves a matrix inversion. Even if some intermediate results are cached and reused, calculating the distances needed to answer a query such as "Which are this collection's most similar songs to Song X?" is computationally very expensive, as we will see in Chapter 5.

2. The KL divergence is not a metric, prohibiting the use of metric-based index structures such as M-Trees [CPZ97] or MVP-Trees [BO99] to speed up nearest neighbor queries.

3. Gaussian-based models compared by KL divergence have been found to be prone of producing so-called *hubs*: songs that are close neighbors to a large fraction of songs in a collection, despite not being perceptually similar to most of these songs. The existence of hubs in music similarity measures has originally been reported in [AF04], and has since been the subject of several publications [Ber06, AP08], also outside of the MIR community [RNI10]. While the hub problem is not limited to Gaussian-based models, some other models seem to be less severely affected [HBC08].

As we will see in the next section, researchers have found ways to alleviate all three problems by simplifying the model, employing specially adapted indexing methods and post-processing the distance space, respectively, or to avoid the problems altogether by using different global models and distance measures.

2.3 Advancements and Alternatives

In this section, we will discuss several other approaches reported in literature, some of which are based on Aucouturier and Pachet's method described above, and some of which are independent or even predate [AP02b]. Our goal is not to give an

extensive review of research on content-based music similarity estimation, but to give an impression of the design space, focusing on ways to tackle the problems discussed in Section 2.2.4. This will help us in designing a system which performs better than and avoids the shortcomings of Aucouturier and Pachet's approach.

Although publications typically describe and evaluate a complete system consisting of a feature extractor, global model and comparison method, the local feature extraction and global modeling stages are typically independent and could be recombined at will. We will therefore consider these parts separately, which also easily allows us to include approaches for feature extraction that have only been used for classification tasks rather than similarity estimation. As many methods are based on the same local features as Aucouturier and Pachet's (i.e., MFCCs), we will start reviewing variations of the global models along with their distance functions, then continue with alternative frame-wise local features (which MFCCs are an example for) and finish with long-term local features.

2.3.1 Global Models and Distance Functions

Clustering Predating [AP02b], Logan and Salomon [LS01] designed a music similarity measure which clusters each song's MFCCs by k-Means, then builds a song model from the mean, covariance and weight of each cluster and compares those models by the *Earth Mover's Distance* (EMD). In their experiments, about half of the songs recommended by the system for a given query were rated as good matches by two users. Conceptually, this method is very similar to the KL Divergence of GMMs, and similarly computationally expensive.

Single Gaussian Mandel and Ellis [ME05] simplified Aucouturier and Pachet's approach by modeling the feature distribution of a song with a single full-covariance Gaussian instead of a diagonal-covariance GMM. This reduces the time needed to fit the model to a song's feature vectors and enables comparison of song models with a closed-form expression of the KL divergence, making the similarity measure several orders of magnitude faster [Pam06, p. 36]. Surprisingly, this simplification does not have a negative impact on the music similarity measure [Pam06, p. 66][LS06][HBC08], making the "Single Gaussian of MFCCs" a popular baseline method to compare other approaches against.

Vector Space As an even simpler global model, a state-of-the-art method by Seyerlehner et al. [SWP10] represents songs just by the median or other percentiles of their local features (which are far more complex than MFCCs, though, see Section 2.3.3). In contrast to Gaussian-based approaches, this yields a vector space representation which can be compared by any vector distance measure such as the Euclidean or Manhattan distance. This is not only much faster than computing

or approximating KL divergences, but enables the use of both metric-based indexes (Section 2.2.4.3) and position-based indexes such as KD-trees [Ben75] for fast nearest-neighbor searches.

Vector Quantization Histograms The first published content-based music similarity measure [Foo97] represents each song by a histogram over its MFCC vectors. For this to work, the multidimensional real-valued local features have to be discretized. One way to do so is to create a global codebook of prototypical features, then replace each feature of a song by the index (the codeword) of the most similar prototype in the codebook. This approach is called *Vector Quantization* (VQ) [LBG80].[6] Afterwards, the sequence of codewords can easily be histogrammed, resulting in a vector space representation. In [Foo97], the codebook is built in a supervised way, partitioning the feature space such that MFCCs of different artists tend to be assigned different codewords. Comparing histograms of codewords for each song by cosine distance (Section 4.6.1.2) yields a similarity measure which is good in finding songs of the same artist.

Later works implemented variations of this approach and compared it to other methods. Aucouturier [Auc06, pp. 206–212] experiments with up to 500-element MFCC codebooks built both by unsupervised K-Means clustering and a supervised method. In timbre similarity estimation, both methods perform significantly worse than Aucouturier and Pachet's GMM-based system, with the supervised codebook outperforming the unsupervisedly built one. [VP05] and [LS06] build MFCC codebooks with a 16×16 Self Organizing Map (SOM). While [VP05] do not compare their approach to other systems, [LS06] report worse performance than the "Single Gaussian of MFCCs" method in k-NN genre classification experiments on three datasets. [HBC08] compute up to 50-element codebooks by k-Means, normalize histograms to unit ℓ_1 norm and treat them as multinomial distributions, compared by KL divergence (cf. Section 4.6.1.3). In contrast to other authors, they include delta features along with the MFCCs (cf. Section 2.3.3), and normalize the data to unit standard deviation in each dimension before clustering. With this approach, they outperform GMM- and Single-Gaussian-based song models in genre-based retrieval tasks on a small dataset (121 songs of 7 genres).

To reduce computational load and ensure that all songs get a fair chance of contributing to the codebook, [SWK08] build the codebook in two steps (referred to as Multi-Level VQ): First the MFCCs of each song are clustered separately, then the cluster centroids of all songs are collected and clustered again. Additionally, the authors restrict clustering to so-called "event vectors" extracted at automati-

[6] This can also be seen as a part of the feature extraction and transformation rather than the global modeling step. However, as it is independent of the type of local feature used, we decided to discuss it along with other global models.

cally estimated note onsets. This measure performs similar to the "Single Gaussian of MFCCs" in k-NN genre classification (3180 songs, 19 genres), but produces far fewer hubs. [MLG+11] also build a codebook in two steps, but using GMMs to find the most significant frames per song and k-Means only for global clustering. MFCCs are combined with a small set of further frame-wise features (Section 2.3.2). For comparison, they also create a codebook by randomly sampling local feature vectors from songs. Interestingly, this random codebook performs best both in similarity estimation (evaluated via k-NN genre classification) and classification. Like [SWK08], the authors of [MLG+11] observe that histogram-based models produce fewer hubs than Gaussian models.

Spectral Histograms Another way to discretize and histogram local features is by binning. [PDW03] calculate a histogram directly on a song's log-frequency log-magnitude spectrogram, with a fixed quantization of 20 frequency bands and 50 loudness levels, accumulated over the song. However, this model performs considerably worse than Aucouturier's GMM [Auc06, p. 206]. [XZ08] use a similar histogram, quantizing frequencies into 12 semitones and separately accumulating over 8 subsequences of the songs. They report their model to perform well in cover detection and genre classification, albeit on much too small datasets to draw any meaningful conclusion (14 and 12 songs, respectively).

HDP As a variation on the GMM and VQ models, [HBC08] train a *Hierarchical Dirichlet Process* (HDP) on all local features (frame-wise MFCCs plus their deltas) of a database. This can be seen as a GMM with indefinitely many Gaussian components, of which only a finite subset is used to explain the training data. A song can then be represented by calculating the posterior distribution over the Gaussian mixture weights given the song's local features, which yields a representation that is conceptually similar to a normalized VQ histogram, but defines a Dirichlet distribution rather than a multinomial. Comparing the Dirichlet distributions of two songs by the KL divergence, the authors obtain a fast similarity measure that outperforms Gaussian-based models and produces fewer hubs (on the small 121-piece genre-retrieval dataset mentioned above).

Dynamic Modeling In order to include information on the time course of a music piece rather than completely discarding the order of local features (cf. Section 2.2.4.2), several authors have tried to employ sequential data models. Most prominently, Hidden Markov Models (HMMs) with GMM observations have been used to model sequences of local features. For similarity estimations, these models can be compared by the Monte-Carlo approximation of the KL divergence described above (Equation 2.6), replacing the unordered sets of samples with sequences. Re-

sults, however, have been disappointing: [Auc06, pp. 202–204] see no improvement in timbre similarity estimation over a static GMM. Similarly, in [SZ05], (genre-wide) HMMs perform worse than a static SVM in genre classification, and a type of recurrent neural networks called Elman networks gives even worse results. Upon closer examination, [FPW05] find that HMMs actually provide a better song model than GMMs in terms of the likelihood, but fail to improve similarity estimations (evaluated via k-NN genre classification). This suggests that HMMs do successfully model dynamic aspects of the songs, but none of these aspects are relevant for discriminating music.

In Section 2.3.3, we will discuss approaches to capture temporal aspects directly at the feature extraction stage instead.

Feature Selection One of the implicit assumptions of all statistical models of local features is that the most statistically significant features are the most important (Section 2.2.4.2). To weaken this assumption, several authors have tried to include only a subset of a song's local features in the global model.[7] Both [SWK08] and [Sca08] restrict the global model to local features centered at likely note onsets, as determined by an onset detection function. While the main motivation in [SWK08] was to reduce computational costs, [Sca08] hoped to achieve some translation invariance. None of the authors compared their method to an implementation without onset filter, but [SWK08] noted that their models tended to only model percussion, which possibly degraded similarity estimates of their system. [SPWS09] instead filter out and discard low-energy frames, having found that they contribute much to the similarity estimate in the "Single Gaussian of MFCCs" approach although the silent parts of songs are arguably not the most perceptually relevant. By discarding between 50% and 70% of the lowest-energy MFCCs, k-NN genre classification accuracy improves from about 19% to 22% (on a dataset of 3180 songs and 19 genres).

Distance Transformations Some attempts have been made to improve a similarity measure by post-processing the calculated distances. [PSS+09] and [SWP10] calculate a full distance matrix between all songs of a collection, then normalize each distance with respect to the mean and standard deviation of distances in the complete row and column (*Distance Space Normalization*). This changes the nearest neighbor ranks, often improving k-NN genre classification results [PSS+09, p. 5], [SWP10, Table 1]. Building on this idea, [SFSW11] model the distribution of neighbor distances to each song as a Gaussian, then interpret the Gaussian's cumulative

[7] Here, "feature selection" refers to filtering the sequence of local features in time, rather than trying to find a combination of different kinds of features that is suited best for a particular task.

density function as a mapping from distances to negated probabilities of being a close neighbor. Combining these neighbor probabilities (*Mutual Proximity*), they obtain an affinity matrix with more symmetric neighborhood relations, which reduces the number of hubs and improves k-NN genre classification. See [SFSW11] for references to further approaches.

With a different goal in mind, Schnitzer et al. [SFW09, SFW11] explore a way to map Gaussian-based song models to low-dimensional vectors such that Euclidean distances in the vector space closely resemble the KL divergences of the original models. This allows to use fast indexing methods even for the non-metric Gaussian-based similarity measures.

Metric Learning [SWW08] experiment with four methods to map song models to low-dimensional vectors such that their Euclidean distances approximate known similarities between music pieces. Binary similarity ground truth for learning and evaluating the mapping was obtained from song metadata ("same album" or "same artist") or by mining the web ("posted in the same music blog"). k-NN classification results on separate test data show that their new global models (the low-dimensional vectors) significantly outperform the original models in classification accuracy.

[MEDL09] propose a different way to learn a metric: They train a classifier on pairs of global models to learn binary similarity ground truth. The global models are single Gaussians of MFCCs and autocorrelation coefficients and two song-level features (all concatenated into a single vector), the classifier is a Multi-Layer Perceptron (MLP) pre-trained as a Stacked Denoising Autoencoder (SdA), and the similarity ground truth is obtained from playlists of commercial radio stations. The classifier successfully learns to distinguish probable from improbable playlist transitions, and its predictions can be used as a measure of song similarity.

Semantic Models Some authors report improved results with a semantic model built on top of a feature-based song model. Such a model can be built by supervisedly training a *tag predictor* on song models, e.g., a set of binary classifiers each of which outputs the probability that a human would label a given song with a particular tag. Ground truth is obtained by asking a sufficiently large group of humans to label a collection of songs (either in a controlled experiment, or in public web services such as last.fm described on p. 2). Each tag usually consists of one or few descriptive words, such as "hard rock", "fast", or "work out", and the set of possible tags is restricted either beforehand or afterwards by regarding a limited-size vocabulary only. The collection of tags for a song carries semantically meaningful descriptions in terms of genre, mood, instrumentation, singing or preferred listening contexts. The idea of semantic models is to learn how humans describe songs in terms of tags, then generate hopefully good descriptions for previously unseen songs

based solely on their audio content. The predicted tag probabilities (or *tag affinities*) for a song form a high-dimensional vector which can be used as a higher-level vector space song model – put differently, the tag predictor maps an audio-based model into a new "tag feature space" in which songs can be compared to each other. [BCTL07] and [Sey10], to name but two examples, report better similarity estimates with such models, compared to purely feature-based models. However, training the tag predictors requires a large collection of high-quality ground truth.

2.3.2 Short-term Local Features

As an alternative or supplement to MFCCs, a wealth of other short-term features has been suggested, each trying to capture a specific aspect of the spectrum on a time scale of one to three frames.

Spectral Shape The spectral centroid, spread, skewness, kurtosis, slope and roll-off give different scalar descriptions of aspects of the spectral envelope, calculated on each mel-spectral frame (see e.g., [Pee04, Sec. 6], [TC02]). Variations of the spectral flatness [Pee04, Sec. 9] and spectral contrast [JLZ+02, ASH09] measure the noisiness or tonality of frames by regarding the difference between valleys and peaks in the spectrum. More music-specific features aim to find the fundamental frequency [DR93] or the notes present in a frame ("chroma" features [SGHS08]).

Dynamics Features such as the spectral flux [Pee04, Sec. 6.2] or delta and double delta (acceleration) features [Fur86] aim to capture some of the dynamics of a signal. While the former is defined directly for spectra, delta features just compute the difference of consecutive frame-wise features (and acceleration features compute the difference of consecutive deltas) and are applicable to any kind of local feature. This simple extension helps a lot in speech recognition [Fur86] and mitigates the exchangeability assumption in Bag-of-Frames models for music (see Section 2.2.4.2).[8]

Learned Features As an alternative to such hand-crafted features, some attempts have been made to learn short-term features, but only in classification tasks.[9] [HWE09] train a *Deep Belief Net* (DBN, see Section 4.4.3 on p. 75) on a set of MFCCs and other short-term spectral features described above, then fine-tune it to classify frame-wise features into instruments, with comparable results to an SVM

[8] Actually, frames plus delta frames are a representation of bigrams, and if there are no duplicate frames in a music piece, a set of bigrams already suffices to reconstruct the complete original sequence.

[9] So learning could just as well be seen as a part of the classifier, not the feature extraction, but it is still an interesting approach to consider.

and MLP classifier. [HE10] instead train a DBN directly on spectral frames (i.e., the high-dimensional frame-wise DFT result), then extensively fine-tune the network as a genre classifier. Removing the DBN's classification layer, they use the network as a frame-wise feature extractor, and train an RBF-kernel SVM to classify the features into the genres they have previously been tuned for. Classifying songs with a winner-takes-all scheme, they show that their tuned local features outperform MFCCs in separating said genres (when mapped to an infinite-dimensional feature space by the RBF kernel). However, it is unclear whether these specialized features would be useful for a general genre-independent music similarity measure.

2.3.3 Long-term Local Features

Several features have been designed to capture more long-term information than the frame-wise spectral energy distribution (and its frame-wise deltas).

ZCR One of the most primitive long-term feature is the Zero-Crossing Rate (ZCR). It is calculated directly in the time domain and gives the number of sign changes between consecutive samples per time unit (i.e., the number of times the signal crosses the zero amplitude line). While fast to compute, it is difficult to interpret – it is dependent on the frequency (higher frequencies have a higher ZCR), but can also be regarded as a measure of noisiness.

Texture Windows Other features build on frame-wise features. For example, a simple way to include more long-term information in local features is to form so-called *Texture Windows*: Frame-wise features such as MFCCs are aggregated over windows of one or a few seconds, and each window is represented by simple statistics such as the mean and variance of its frame-wise features. These statistics form new higher-dimensional long-term local features. [Auc06, pp. 201–202] do not find any improvement in their timbre similarity estimates with this approach. However, [TC02] and [WC04] obtain a significant improvement in genre classification accuracy when compared to directly modeling frame-wise features. As a variation, [WC05] compute statistics over onset-aligned windows rather than fixed-size windows, further improving results. [SZ05], however, neither see a significant improvement nor degradation with beat-aligned windows.

Block-wise Features A more explicit way to locally model dynamics is by processing blocks of consecutive frames, i.e., excerpts of a sequence of short-term features. In contrast to texture windows, the order of short-term features within a window is kept, allowing to infer information about the time envelope of a note, the rhythm, or note transitions. Even a simple concatenation of a couple of short-term features yields long-term local features that can help in classifying music into genres

[SZ05]. More advanced features are obtained from further processing such blocks of short-term features.

Using mel-spectral frames as short-term features (i.e., MFCCs without applying the DCT), blocks of features are equivalent to spectrogram excerpts with log-scaled frequency and magnitude. Many state-of-the-art features are based on this representation. *Fluctuation Patterns* (FPs) [PRM02][Pam06, pp. 36–42] aim to capture rhythmic aspects of a song. Their key idea is to apply an FFT to each frequency band of 3- to 6-second blocks, thus determining regular modulations of loudness over time (separately for each frequency band). Only modulations of 0–10 Hz are regarded, corresponding to 0–600 beats per minute (bpm). Results are weighted according to a perceptual model, attenuating very slow modulations, peaks in the pattern are emphasized with a gradient filter, and the final pattern is smoothed with a Gaussian filter both with respect to modulation frequencies (periodicities) and frequency bands, to enable a more fuzzy comparison of patterns. [SWP10] scale the periodicities of fluctuation patterns logarithmically, such that a change in tempo results in a linear shift along the periodicity axis of the patterns. The filter bank used for the rescaling step also induces a loss of precision, such that the patterns become invariant to minor changes in tempo. [PSS+09] do the same, but apply an onset filter before computing the modulation strengths (resulting in *Onset Patterns*). The onsets are determined on a high-resolution cent scale rather than the usual coarse mel scale, then reduced to a low-resolution critical band representation, aiming to reliably detect note onsets only semitones apart without increasing the dimensionality of the final pattern. Several higher-level features are derived from fluctuation patterns: [Pam06, pp. 57–45] suggests to compute the center of gravity, the focus (whether modulations are concentrated in a few periodicity-frequency bins or widely spread over the pattern) and various ratios of energy in specific ranges of periodicity and frequency. [PSS+09] apply a 2D DCT to the patterns and keep a limited range of *Onset Coefficients* in both dimensions. The suitability of fluctuation patterns for modeling rhythms has been demonstrated on a dataset of ballroom music, in which they significantly outperformed MFCCs in terms of k-NN genre classification [PSS+09].

[SWP10] propose a range of additional features based on log-frequency log-magnitude spectrogram excerpts. The *Spectral Pattern* takes short blocks of 10 frames and sorts the 10 time steps of each frequency band by value. The *Delta Spectral Pattern* does the same on 25-frame blocks of a spectrogram with emphasized onsets (by calculating differences and half-wave rectifying the result). The *Correlation Pattern* measures the correlation between frequencies in 256-frame blocks, and the *Spectral Contrast Pattern* calculates the frame-wise spectral contrast mentioned in Section 2.3.2, then sorts the values of each frequency band in blocks of 40 frames. The calculations, block lengths and the number of frequency bands have been carefully tuned for each of these features.

[PKSW10] apply two hand-designed image filters to a cent-scaled spectrogram which emphasize horizontal and vertical patterns, respectively, i.e., continuous tones and short attacks. Calculating the means over the two filtered spectrograms and dividing the "attack" mean by the sum of the two means yields a scalar feature dubbed the *H2A ratio*. Low values indicate a dominance of harmonic parts, high values indicate strong percussion. This can be calculated as a global feature for sorting songs [PKSW10], or on a local level like other features [PSS+09].

Learned Features Like for short-term features, attempts have been made at learning features rather than hand-crafting them. [Abd02] shows that applying *Independent Component Analysis* (ICA) to short music waveform excerpts yields components similar to Fourier basis functions, and that sparse coding of harpsichord music spectrograms could reveal notes. [HBC09] train a shift-invariant HDP on spectrograms and yield decompositions of songs into musically meaningful components such as drum sounds and vocals, as well as transcriptions of songs in terms of these components. However, such complex song descriptions are not directly usable for similarity estimation. Lee et al. [LLPN09] train convolutional DBNs on spectrograms (compressed on the frequency axis via PCA), yielding 10-frame features that surpass MFCCs in 5-way genre and 4-way artist classification on 3-second song snippets (not stating the classification method used). However, it is unclear whether the features are useful for similarity estimation as well, and it is possible they outperform MFCCs simply because they integrate information over multiple frames. [PKSW06] apply ICA to PCA-compressed mel-sone music spectrogram excerpts and compare the components' activation histograms for music pieces to estimate their musical similarity. However, in 1-NN genre classification this approach proves inferior to Aucouturier and Pachet's method. [PBK09] unsupervisedly extract features from a high-dimensional cortical representation of music pieces using *Locality-Preserving Non-negative Tensor Factorization* (LPNTF). They then train a supervised sparsity-based classifier on these features and surpass state-of-the-art results on two genre classification datasets. However, it is unclear if this is because of the features or because of the classifier.

2.3.4 Combined Features

State-of-the-art systems combine short-term features with long-term features, yielding a similarity measure that regards both timbral and rhythmic aspects. Combining different features has been shown both to improve similarity estimates and to reduce hubness [FSGP10].

Pampalk's *G1C* [Pam06] combines the "Single Gaussian of MFCCs" with Fluctuation Patterns (and two higher-level features derived from the patterns, the *Gravity* and *Bass*). Distances are computed separately with KL divergence and Euclidean

distance, respectively, then linearly combined with hand-tuned weights. This measure won the MIREX 2006 music similarity and retrieval task.[10]

Pohle's *RTBOF*[11] [Poh10, PSS+09], winner of the MIREX 2009 music similarity and retrieval task, augments MFCCs with Spectral Contrast, Harmonicness and Attackness features, modeled as a full-covariance Gaussian for each song and compared by the Jensen-Shannon divergence (see Section 4.6.1.4, p. 83). This timbre component is combined with onset coefficients, which are modeled as a second Gaussian per song and also compared by Jensen-Shannon divergence. The timbre and rhythm distance matrices for a collection are separately normalized with DSN (Section 2.3.1, p. 19) and added up to form the final distance matrix.

Seyerlehner's *BLS* (Block-Level Similarity) approach [Sey10, SWP10] separately computes distance matrices for the Spectral Pattern, Delta Spectral Pattern, Variance Delta Spectral Pattern, Fluctuation Pattern, Correlation Pattern and Spectral Contrast Pattern (each aggregated to a global model via different percentiles, then compared by Manhattan distance), normalizes the distance matrices with DSN, linearly combines them with weights optimized on a genre classification dataset, and renormalizes the final distance matrix again with DSN. On the dataset it has been optimized for, BLS beats RTBOF; on five other datasets it performs similarly or inferior to RTBOF. Combining BLS with RTBOF (again by applying DSN to the separate distance matrices, adding them and applying DSN to the result) yields a similarity measure surpassing both BLS and RTBOF in k-NN genre classification on six datasets [SWP10].

2.4 Conclusions

From the preceding discussion of existing content-based music similarity estimation systems, we can draw several insights useful for building our own system, aiming to compete with the state-of-the-art:

1. First of all, the general architecture of a local feature extraction, global modeling and comparison stage is well-tested and does not call for a change.

2. Machine learning has been employed at all stages, but not often for feature extraction. In particular, *unsupervised* learning methods for feature extraction are almost unexplored in the context of music similarity estimation.

[10] MIREX (Music Information Retrieval Evaluation eXchange) is a yearly contest held in conjunction with the International Society for Music Information Retrieval Conference (ISMIR). The audio similarity and retrieval task evaluates music similarity measures via human listening tests. See `http://www.music-ir.org/mirex/` for more information.

[11] "Rhythm-Timbre Bag of Frames" (RTBOF) is the name given to it in [SWP10]; Pohle termed it the "unified" algorithm.

3. Capturing the temporal evolution of music pieces with sophisticated dynamic models of local features (HMMs, Recurrent Neural Networks) does not seem to work well. Block-wise local features constitute a much better way for modeling dynamics, significantly surpassing simple frame-wise features.

4. The best-performing methods are based on block-wise features defined as elaborate transformations of log-frequency log-magnitude spectrogram excerpts. These features have been hand-crafted over the course of several doctoral dissertations [Pam06, Poh10, Sey10], sometimes optimized to a specific dataset, and include several elusive design decisions and parameter choices. It would be desirable to find more effective, more efficient or better justifiable features. The prevalent basis of log-frequency log-magnitude spectrogram excerpts may be a good starting point for this.

5. State-of-the-art systems combine rhythmic and timbral features, which gives better results than either type of feature alone. To be competitive, any system probably needs to capture both short-term and long-term aspects of songs.

6. Choosing a type of local feature (or a combination of features) and extracting it at a steady rate throughout a song is easy and works reasonably well. Attempts at restricting feature extraction to points of interest in a song, such as detected note onsets, do not yield a consistent improvement.

7. Vector space song models have a range of advantages over Gaussian-based models: Distances are faster to compute, easier to index (for even faster nearest neighbor retrieval) and less prone to hubs.

We will consider these insights in the design of our system in Chapter 4. Before, we will briefly compare the task of music similarity estimation to similar tasks in other domains that have been explored better, in order to learn about methods that could be worthwhile to adopt into our system.

3 Inspiration from other Domains of Pattern Recognition and Information Retrieval

In this chapter, we will take a look at three other fields also dealing with the problems of extracting features, modeling documents or finding similarities: Text Retrieval, Computer Vision and Speech Processing. We will review some of the methods employed and relate the three domains to the music domain to point out arguments for adopting particular methods, to show how this would be possible, but also to give reasons for why this may fail.[1]

This examination is by no means exhaustive, as it is not intended to form the central contribution of our thesis. It may be worthwhile to explore the relations of the domains more in-depth to reveal further methods that could be employed in the context of Music Information Retrieval. Ultimately, however, only empirical investigations can show whether a method works or not.

3.1 Text Retrieval Systems

Firstly, we will regard the field of Text Retrieval. As we will see later, it has already inspired several methods for Computer Vision, although at the first glance, text does not seem to have much in common with images. Here, we will introduce the methods in their original context, then reason how to apply them to music. Before, we will briefly describe the purposes of Text Retrieval systems.

Purposes

One of the main areas in Text Retrieval research considers the task of document retrieval: Given a collection of text documents and a textual query, which documents are relevant to the query? Popular use cases are search engines for a library or the world wide web. The query may consist of a couple of words (search terms) entered by a user, or a complete document a user wants to find similar documents to.

[1] The strategy of borrowing methods from other domains has been described by Aucouturier as a design pattern termed "Cross-Fertilisation" [Auc06, pp. 51–52].

Another important problem is document classification: Given a document and a number of classes, which class does the document belong to? Possible use cases are spam detection in emails (classifying emails as "spam" or "ham") or categorizing scientific articles into topics.

Methods

A big challenge in document retrieval is to make a system fast enough for interactive use – consider a web search engine that has to answer requests for the best matches to a query in a database of up to a trillion documents within seconds. However, here we are mostly interested in how documents can be matched to a query or another document at all, and how documents can be categorized. Both problems require some abstract model of a document, as simply comparing or categorizing documents character by character will not turn up anything useful.

The first system for document classification [Mar61] worked as follows: Each document in a corpus of training items (260 abstracts of scientific articles in informatics that had been manually grouped into 32 categories, with each document assigned to up to 3 categories) was separated into words by trivially splitting the text at spaces and punctuation. The most frequent logical type words (e.g., "the", "of") and nonlogical type words (e.g., "computer", "data") of the corpus were regarded as uninformative and the least frequent words were considered to be too inefficient to use (as they only occurred in few documents). The remaining words were manually reduced to a vocabulary of 90 "clue words" that seemed highly indicative of a category. A corpus of test items (145 scientific abstracts manually grouped into the same set of 32 categories) was processed by testing each document for the occurrence of clue words, then computing the posterior probabilities that a document belongs to a particular class C_j, given its clue words $W_1, \dots W_n$. The posterior probability of a class was computed by multiplying the prior probability of the class with the likelihoods of the class under each occurring clue word, using Bayes' rule: $P(C_j|W_1, \dots W_n) \propto P(C_j) \cdot P(W_1|C_j) \cdot \dots P(W_n|C_j)$, where the priors and likelihoods were estimated from the training data by simple counting. This naively assumes that each word in a document independently contributes to the probability that the document belongs to a particular class, not regarding the order or co-occurrence of words in a document; it is consequently termed a *Naïve Bayes* classifier [Lew98]. Despite these simplifications, for 33 of the 66 items that contained at least two clue words and were manually assigned to only one category, the predicted category was the correct one.

Many of the ideas introduced in [Mar61] are still in use today. Both document classification and retrieval systems mostly follow the same basic structure: A document is separated into words, matched against a fixed vocabulary, and represented by a vector denoting the number of occurrences of each vocabulary word in the

document. This is commonly referred to as a *Bag of Words* (BoW) representation, as it disregards the order of words in a document, imaginable as pouring all the words into a bag. This representation can be used to train a classifier on or for directly matching documents against other documents or textual queries.

Over time, improvements have been proposed on all levels (see [Seb02] for a review):

Stemming: Prior to filtering the words with a fixed vocabulary, the words of a document can be mapped to some nominal form, e.g., from plural to singular, or from conjugated verb forms to infinitive. This *stemming* removes irrelevant variations (in a BoW model, grammatical suffixes do not carry any useful information, but only increase the size of the vocabulary).

Term weighting: Instead of binarily distinguishing important from unimportant words, different *term weighting* schemes have been proposed [SB88], automatically judging the importance of each word of a vocabulary based on their occurrence in documents, then weighting words in query matching or classification. One of the most popular schemes for document retrieval is TF/IDF: The *term frequency* (TF), i.e., the number of times a word occurs in a document in proportion to the document length, is weighted by the *inverse document frequency* (IDF), given as the logarithm of the inverse of the proportion of documents in a corpus containing the term. The intuition is that a word which occurs in almost all the documents is less important (at least in a discriminative sense) than a word occurring in a small fraction of the corpus only.

Topic modeling: The BoW model assumes the occurrences of words in a document to be statistically independent. However, words are in fact often correlated – for example, the word "milk" will often co-occur with "butter", but seldom with "asteroid".[2] More severely, some different words even mean the same (synonymy), such as "buy" and "purchase". To respect this, *Latent Semantic Indexing* (LSI) [DDL+90] maps document count vectors (the BoW representation) from the original word space to a lower-dimensional space of implicit meta-terms, or *topics*, by a linear transformation learned from a corpus. The mapping is learned in an unsupervised way, just regarding the correlations between words, similar to PCA (more concretely, it performs a Singular Value Decomposition on the term-document occurrence matrix).

More recent methods define and learn a *generative model* of documents. *Probabilistic Latent Semantic Indexing* (pLSI) [Hof99] models documents as mixtures of topics (i.e., multinomial distributions), and topics as mixtures of

[2] Despite when speaking of the "milky way"; see the entry about word sense disambiguation.

words. To generate a word of a document, it picks a topic from the document-specific mixture of topics, then picks a word from a topic-specific mixture of words. The word mixtures for each topic are shared globally, and can be related to the word-to-topic mapping of LSI. The topic mixtures for each document are estimated from the data, in conjunction with learning the topics. The topic mixtures then serve as compact document representations.

While this gives a generative model for all documents of a corpus, it cannot generate new documents, because their topic mixture is unknown. *Latent Dirichlet Allocation* (LDA) [BNJ03] solves this by placing a prior on the topic mixture for documents, which is a Dirichlet (the conjugate prior of a multinomial) chosen to be symmetric (with no preference for any topic) and with $\alpha \ll 0$ (favoring sparse mixtures of topics, i.e., only few active topics per document). Additionally, a Dirichlet prior is placed on the word mixture for topics, smoothing the distributions estimated from a training corpus. In this model, a new document is generated by picking a topic mixture from the Dirichlet, then proceeding as for pLSA. The sparseness enforced by the priors leads to more useful topics and consequently better document representations.

As a further extension, the Hierarchical Dirichlet Process (HDP) [TJBB06] infers the number of topics from the data, instead of requiring the number to be given in advance.

As an alternative to modeling documents as mixtures of topics, Salakhutdinov and Hinton [SH09b] model documents with two-layer undirected graphical models called *Restricted Boltzmann Machines* (RBMs, see Section 4.4.1, p. 61). One of the layers models word occurrences in a document with a special architecture termed *Replicated Softmax*, and the second layer consists of latent binary units indicating the topics of the document. The latent units form a *Product of Experts* [Hin02] which multiply and renormalize their predictions on words. This conjunctive coding produces much sharper probability distributions over words than the disjunctive coding of mixture models [HS11], and the latent representations of documents outperform LDA representations in document retrieval [SH09b].

Semantic hashing: Extending the idea of modeling documents with RBMs, Salakhutdinov and Hinton [SH09c, HS11] train a stack of three RBMs on word count vectors to form a deep generative model with a small top layer and fine-tune the resulting four-layer network as a deep autoencoder (in which the former top layer forms an information bottleneck). The small layer now produces binary codes for documents such that similar documents are assigned close code words (in terms of Hamming distance). Using very short codes (e.g., 20 bits), this allows indexing all documents by their code, and quickly retrieving

similar documents to a query by simply accessing all codes in a Hamming ball around the code of the query document, without any search.

Word sense disambiguation: The approaches above assume that each entry in a document's word count vector refers to a specific word with a specific meaning. However, if these words are directly extracted from a text by splitting it at word boundaries, this assumption does not hold: In most languages, there are groups of words called *homographs* that share the same spelling, but have different meanings. For example, "train" can refer to a vehicle or the act of teaching; "bank" can refer to a credit institution, the side of a river, a filter bank, and more. *Word sense disambiguation* techniques try to infer the meaning of a word by its context (see [Nav09] for a survey). Without word sense disambiguation, topic models will always regard the *average* meaning of a word – for words with a seldom used second meaning (such as "bass", which refers to low sound frequencies far more often than to a species of fish), this produces reasonable results in most instances, but for other words, it may well reduce performance.

n-gram models: The topic models discussed above model the words of a document independently. To incorporate at least the immediate local context, words can be modeled in pairs or triplets; more generally, in *n-grams* (n-tuples of consecutive words). [Wal06] extends LDA by a bigram topic model, yielding better generative/predictive performance and better topics, which may be useful in categorization and retrieval tasks.

Adoption

Regarding songs as documents, the basic architecture of text retrieval systems shows a clear resemblance to content-based music similarity estimation systems (Section 2.1, p. 9): Documents are split into short entities, which are first processed separately, then aggregated into a more compact global model (abstracting from the order of local entities) for classification or comparison purposes. However, it is not immediately clear how to reasonably relate music pieces to text documents in order to apply text retrieval methods to music.

Let us consider a very straightforward mapping first: Words, the local entities of documents, could be assumed to be equivalent to frames, the prevalent segmentation of music signals. Stemming may be regarded as the equivalent to vector quantization of frames (or frame-wise features), which also maps slightly different variations of local entities to a common form. However, vector quantization also enforces a fixed-size vocabulary – in text retrieval, filtering for a set of "clue words" constitutes a separate step. Creating a BoW model as a word count vector can be directly adopted to a vector-quantized sequence of audio frames, resulting in a

"Bag of Frames" (BoF) model. This equivalence has already been pointed out by [RHG08], who refer to their vector-quantized frame-wise chroma features as "audio words".

For once accepting this equivalence, we can easily apply text retrieval methods to songs. N-grams would be equivalent to blocks of frames, which are already used in several music similarity measures. Word sense disambiguation could be a more intelligent way to incorporate context information and eliminate ambiguities in the interpretation of single frames. Term weighting can be applied to BoF models exactly as for BoW models, possibly solving the problem of distinguishing important from unimportant frames. In fact, [Foo97] already suggests to weight vector quantized features by their inverse entropy (which would devaluate features that occur nearly uniformly), and [RHG08] use TF/IDF weighting on their "audio words". Topic models could form useful representations of songs, and [HBC08] successfully applied HDPs in a music similarity estimation context, albeit modeling topics as Gaussian distributions of MFCCs rather than multinomials over discrete word-like features. Semantic hashing may be a suitable way to implement fast similarity queries on large databases, but require a large database to train on in the first place.

However, it is doubtful whether audio frames are a proper counterpart to words in text. In contrast to words, frames are very short and always have the same length – a "Bag of Frames" may thus be more equivalent to a "Bag of Syllables" than to a "Bag of Words", and shuffling around syllables in a text would obviously destroy most clues about its meaning or topic. Syntactically, words in text can be found via blanks and punctuation. The equivalent in music may be bar lines, although again this would force all words to be of the same length, and lead to an extraordinarily large vocabulary. Regarding pauses as blanks may be tempting, but pauses are scarce at least in popular music. Semantically, there is no consensus on a definition of words, but "perhaps a word is the smallest unit of language that can be used by itself" [BS81]. With this loose definition, words in music could be notes. However, unlike words, notes in music overlap, suggesting the need for source separation to properly divide a song into words (separating critical bands could be seen as a half-hearted attempt at this). Apart from that, single notes arguably do not carry as much information as single words – occurrence of the word "soccer" probably limits the possible topics of a document much more than a c' played on a piano tells about the genre of a song.

All in all, although we pointed out a way to apply text retrieval methods to music via vector quantization of local features, it is not clear whether this is a good way and what might be a better one. However, the discussion of computer vision systems will show that text retrieval methods have been successfully applied to images in a similar way, encouraging their use on music.

3.2 Computer Vision Systems

We will now examine the field of Computer Vision. We will first describe the problems it aims to solve, comparing them to tasks in Music Information Retrieval, then review the basic methods it employs, and finally see how these could be adopted.

Purposes

Computer Vision research works on methods for extracting information from images. For our purposes, it suffices to regard two broad categories of problems:

Object recognition: Given an image, which objects are present and where? This includes a whole range of more specific tasks:

- Classifying images of single objects (e.g., cars, dogs, airplanes ...)
- Classifying images of scenes (e.g., beach, wood, city ...)
- Character recognition (printed or hand-written)
- Multiple object detection (e.g., finding cars, humans or faces in surveillance images or videos)
- Face recognition (identification of persons from an image of their face)

Image retrieval: Given a collection of images and a query, which images match the query?

- For a textual query, this requires generating a description of each image the query can be matched against (this description does not need to be textual, but it needs to be relatable to the query).
- If the query is an image, this involves some sort of comparison of the query image to the images in the collection.

Object recognition mainly deals with classification problems at different levels of difficulty, from a low number of classes (recognizing scenes or digits) to a high number (face recognition) and from low intra-class variability (printed character recognition) to high variability (object classification). As with any classification task, difficulties arise either in capturing the inter-class differences (e.g., distinguishing cats from dogs, or even different kinds of cats), or in ignoring intra-class variability (e.g., different lighting conditions and orientations in face recognition). Multiple object detection plays a special role in that the number and position of objects is not known beforehand. Image retrieval, on the other hand, could either build on object recognition or, when querying by image, on image similarity estimation.

Relating these problems to Music Information Retrieval and regarding songs as the equivalent to images (we will scrutinize this equivalence further below), we find the following:

- Scene recognition in images is roughly equivalent to genre classification in music: Both deal with documents that have a rich local structure, but can usually be assigned to one of a limited set of global classes. Local structure can be unspecific (clouds, drums) or highly indicative of a class (trees, distorted guitar). In both problems, classes may be blended (trees on a beach, Country Soul), and classes can be further divided into sub-classes.
- Multiple object detection is roughly equivalent to music transcription: Both aim to decompose a complex document into a set of known entities and their locations. Both need to be invariant to certain transformations of these entities (lighting conditions, mastering), but extract others (pose, accentuation), and both need to cope with intra-class variability (car shapes, instrument manufacturers).
- Image retrieval by images is strongly related to music similarity estimation: As stated above, implementing query-by-image retrieval on an image collection requires a comparison of images, which poses the same problems as a comparison of songs, if we regard images and songs to be equivalent.

Methods

Most methods start by constructing a global model for each image, which is subsequently classified or compared against other models. This model should capture aspects relevant to the task, but be invariant to unimportant details (which may include the lighting conditions, scale, or orientation, for example). The global model can be built from global image features or by aggregation of local features. Here we can only give a superficial overview of different features and global descriptors; please see [DJLW08, pp. 17–24] for a more extensive review.

Global features include descriptions of shape (determined from an automatic segmentation of the image), color, or texture, used for scene recognition (the "gist" of a scene [OT06]), single object recognition or texture classification. Local features may either be densely sampled from the image (i.e., on a regular grid) or sparsely, at interest points (e.g., corners). Each local feature describes properties of the image neighborhood, such as the local image gradient. This description can be made invariant to rotations (e.g., by normalizing it with respect to the largest local gradient direction) and scale (by computing the properties not only in the original image, but also in rescaled versions of it); a popular example being the *Scale-invariant feature transform* (SIFT) [Low04]. The global model abstracts from the exact locations and values of local features, e.g., by histogramming.

An approach especially interesting for us has been inspired by Text Retrieval. [ZRZ02] propose a general framework which regards images as documents composed of *keyblocks*, in analogy to text composed of keywords. First, small quadratic patches are extracted from an image on a regular grid, transforming the image into a 2-dimensional array of possibly overlapping blocks. Optionally a low-level feature is computed for each patch, otherwise the raw pixel data is used. Each patch (i.e., the feature vector or pixel data) is replaced by the index of the closest keyblock in a dictionary previously obtained by clustering a training set of patches. The image can now be regarded as a 2-dimensional array of "visual words". Different global models can be built from this document-like representation: A histogram model states the number of occurrences of each code word, analogous to the "Bag of Words" model in Text Retrieval. The relative term frequency (normalized by image size) or inverse image frequency (the equivalent of the inverse document frequency, see p. 33) can be easily derived from this as well as a boolean model which simply binarizes the term frequencies with a given threshold. To include context information, [ZRZ02] also suggest to adopt n-gram models by capturing co-occurrences of different spatial configurations of bi- or tri-blocks. In image retrieval experiments on three databases, their method outperforms two approaches based on global color features.

Similar architectures are still popular both for image retrieval and classification, with several improvements:

Dictionary learning and Soft words: Alternatives have been proposed both to obtaining a dictionary of visual words with k-means, and to computing codewords for local features by hard assignment to the closest keyblock. For example, [JT05] claim to obtain a more informative dictionary with a radius-based clustering method, and [vGGVS08] improve performance in scene and object classification with soft assignments of features to codewords (via Gaussian kernels). In a recent study, Coates et al. [CN11] decouple dictionary learning from image encoding, evaluating different combinations of clustering and sparse coding methods on three object classification datasets. They conclude that a sparse encoding, i.e., image representations each using only a small subset of the dictionary, performs best, almost independently of the chosen codewords.

Pooling: The "Bag of Visual Words" (BoVW) is completely invariant to translations of local features, which may be insufficient to distinguish particular images from each other (e.g., a blue feature in the top half of an image probably is sky, while in the bottom half it may be a lake instead). As a compromise between the global BoVW and the full 2-dimensional array of local features, the global model may be constructed by pooling the local features in a fixed partitioning of the image, such as a 2×2 grid. Depending on the type of local

feature, the pooling operation could be histogramming, computing the mean, minimum or maximum of each section (see [BPL10] for a thorough analysis and references).

Spatial pyramids: To avoid having to find the best pooling resolution for a task, it has been suggested to model images as a *Spatial Pyramid* [LSP06]: the global BoVW histogram is augmented with local histograms at increasingly fine subdivisions of the image. This technique captures the spatial alignment of local features with different levels of accuracy, allowing to model, say, a face as a particular arrangement of two eyes, a nose and a mouth rather than a mere collection of these parts, without restricting the positions too much.

Topic modeling: Just like in text retrieval, generative models have shown promising results for image classification. The topic models pLSI and LDA have been directly adopted from text retrieval and applied as generative models for BoVW vectors in object and scene classification [SRE+05, FFP05].

Generative image models: Additionally, generative models have also been employed to directly model image pixel data, most prominently methods based on *Restricted Boltzmann Machines* (RBMs, see Section 4.4.1, p. 61). RBMs are two-layer neural networks with a visible layer of neurons modeling the input, and a hidden layer of neurons forming a latent representation. After training an RBM unsupervisedly on a set of training data, the latent representation conditioned on an input vector forms a higher-level description of the input. A trained RBM can thus be used as a feature extractor. RBMs can easily be stacked to form a *Deep Belief Net* (DBN) extracting even higher-level features (see Section 4.4.3, p. 75).

Both RBMs and DBNs have successfully been used to model local image patches [Kri09, CLN10] and complete small images (e.g., in digit recognition [HOT06, BLPL07], but also for modeling scaled-down scene images [KH11]), developing feature detectors adapted to the input data. Ranzato et al. recently proposed two extensions of RBMs with even better capabilities for modeling natural images: the mcRBM [RH10] and mPoT [RMH10], producing state-of-the-art results on public object recognition datasets [RH10, RSMH11].

As a compromise between modeling local patches and modeling complete images, *Convolutional RBMs* [DB08, NRM09] employ weight-sharing to model a grid of local patches of an image, then use max-pooling to create a more compact higher-level image description. With a generative formulation of the max-pooling operation, [LGRN09] learn a hierarchy of feature detectors from fine-scaled parts to larger-scale objects.

Learning invariance: Especially in the context of digit recognition, an easy method to make a classifier more robust to intra-class variability is to extend the training set with synthetically transformed training images, keeping the original labels [LBBH98, p. 14]. Transformations may include translations, small rotations, elastic deformations or task-specific manipulations such as varying the line thickness for hand-written digits.

An interesting alternative is to train a model to represent transformations independently from the image content [HKW11], allowing classifiers to use information on, say, exact locations only when needed. Such models are referred to as being transformation *equivariant*.

Semantic hashing: Based on the idea of modeling images with RBMs, [KH11] train a deep DBN on small images and fine-tune it as an autoencoder to produce compact binary codes. These codes can be used as semantic hashes to quickly retrieve similar images from a large database, just as it has be done for documents (p. 34).

Adoption

Again, the architecture of image retrieval systems we focused on is very similar to content-based music similarity estimation systems (Section 2.1, p. 9), if we regard songs as an equivalent to images: Images are divided into smaller parts (or downsampled to a small size), which are processed separately and combined into a global model. And again, to apply any of the methods, we need to find a way to relate songs to images.

A straightforward link is that images are 2-dimensional signals, while music pieces are 1-dimensional signals. General local features could therefore be computed on signal excerpts, the 1-dimensional equivalent to image patches. However, audio signals are very different from images: While image signals are spatially smooth (i.e., neighboring pixels correlate), audio signals rapidly vary spatially. Thus, it is doubtful whether image features apply well to raw audio signals.

Alternatively, the 1-dimensional audio signal can be transformed into a 2-dimensional signal by calculating a spectrogram via a Short-time Fourier Transform (STFT) – this adds a new dimension (frequency) at the cost of time resolution; the proportion of time to frequency resolution being determined by the frame size. As audio signals change frequencies rather slowly, the spectrogram representation is spatially smooth along the time axis, resembling images much better than the raw signals. Local features can then be computed on spectrogram excerpts, i.e., blocks of consecutive frames. Any feature applicable to image patches can be directly employed on spectrogram blocks, and any global image model can be used on these local image-like features.

Several authors already suggested to treat spectrogram excerpts as image patches and adopt image processing methods. [DSN01] extract features from spectrograms with oriented Difference of Gaussian filters at different scales, but do not achieve convincing results on a small three-way genre classification dataset. [Abd02, pp. 114–115] employs sparse coding to short spectrogram excerpts of harpsichord music, yielding features that detect onsets of different notes. The author did not perform music retrieval experiments with these features, though. [CEK05] use Haar-like feature extractors inspired from object detection on spectrograms to discriminate speech from music, and [PKSW10] applies simple horizontal and vertical edge detectors to spectrograms to measure the amount of harmonic and percussive elements. As mentioned in Section 2.3.3 (p. 27), [PKSW06] applies ICA to spectrogram blocks to extract local features, then builds global models by histogramming the component activations and compares histograms to estimate music similarity, performing worse than Aucouturier and Pachet's method (Section 2.2, p. 11). Unlike the other attempts, [YS09] extract blocks that do not cover the full frequency range, but are limited in both time and frequency. In a log-frequency spectrogram, this makes features invariant both to time shifts and transposition. The authors produce a dictionary of blocks by random sampling from spectrograms of instrument sounds, then create representations of unseen spectrograms by computing the distances between the spectrogram and the dictionary at each valid location, min-pooling the results. With a k-NN classifier, they achieve satisfactory results in distinguishing solo instruments.

To the best of our knowledge, neither vector quantization nor generative models have been applied to spectrogram patches of music, except for a small-scale classification experiment with Convolutional Deep Belief Networks ([LLPN09], see Section 2.3.3, p. 27). Also pooling or spatial pyramids have not been employed, although they could provide a good model: In music, global arrangement should be less important than local arrangement (for two songs to be similar, it will be unimportant whether a guitar solo takes place before a bass solo or the other way round). While a BoW of large-scale local features might reflect this, possibly some hierarchy would work better.

It is not clear whether spectrogram patches form a good base for methods designed for image retrieval, though. While images pixels are locally correlated in all directions, spectrograms are only smooth in time; correlations in frequency are influenced by harmonics (also see Section 5.3, p. 92). Moreover, the two dimensions in a spectrogram have two very different meanings: one denotes time, the other denotes frequency. In images, on the other hand, both dimensions are spatial. As a consequence, flipped, rotated or rescaled images are often still interpretable, possibly not even changing their meaning (such as a picture showing a soccer ball, or a pen on a desk seen from above), while a rotated spectrogram would not resemble any usual sound at all. This can also be seen as an advantage: To model music,

we do not need to consider rotation- or scaled-invariant features such as SIFT. A third difference is that many physical objects are opaque, such that overlapping objects in an image occlude each other, and, ideally, relevant objects occlude the background. In spectrograms, overlapping acoustic events add up instead, possibly rendering many object recognition methods inapplicable to music.

3.3 Speech Processing Systems

Finally, we take a brief look at the field of Speech Processing. Again, we will first compare the problems it targets to problems in Music Information Retrieval, describe commonly employed solutions and see what we can learn for our task.

Purposes

Speech processing aims at extracting information from speech signals. We will regard three exemplary problems:

Speech recognition: The main area of research in this field deals with transforming an audio signal of human speech into a textual sequence of words. Often, the task is simplified by limiting the vocabulary or targetting a specific speaker.

Speaker recognition: Another problem is to identify the speaker by her or his voice, useful for identity verification or monitoring.

Source separation: The *Cocktail Party Problem* is the problem of separating out a single voice from a recording of many people speaking simultaneously. This is an easy task for humans, who can follow a conversation in a crowded room (e.g., at a cocktail party).

The first two problems are classification problems: Speaker recognition identifies one of a known set of speakers from a speech signal, speech recognition classifies short excerpts into phones. The third problem is a source separation problem, aiming to reconstruct the components of a mixed signal.

Insights from speaker recognition might be applicable for instrument recognition, artist identification, or to estimate singing voice similarities, and speech recognition could be similar to finding melodies in music (if we relate phones to notes), but both deal with a single voice recording, which is very different to polyphonic music. Solving the cocktail party problem could help to separate a polyphonic music signal into a set of "solo parts", enabling the application of methods for analyzing monophonic music. However, speech of different speakers at a cocktail party is uncorrelated (except that people within a conversation tend not to speak simultaneously and share a common topic), while simultaneously playing instruments in a

song usually respect a common rhythm and their pitches combine into particular harmonies – put differently, when separating and recombining the sources of two cocktail parties, it will still sound like a cocktail party, whereas doing the same for two music pieces will generally yield something unmusical (or at least not resemble any popular music).

Methods

Speech recognition is based on modeling sequences of local features. Among the most popular local features are MFCCs, which we described in Section 2.2.1 (p. 11). A sequence of local features is then mapped to a probable sequence of words in multiple stages: Short sequences of features are mapped to phones (the acoustic equivalent to phonemes), sequences of phones are mapped to words and sequences of words are mapped to sentences. These stages influence each other not only in a bottom-up, but also in a top-down manner: A good language model can predict probable words from the sentence formed so far, which in turn makes it easier to predict the next phone given acoustic features.

The prevalent base for these stages are *Hidden Markov Models* (HMMs). For each phone, a separate HMM of a few states is trained to model typical short sequences of local features [JR91]. These phone-specific HMMs can then be concatenated to form word HMMs, which can again be concatenated to phrase HMMs. The HMM's emission probabilities, i.e., the probability distribution over local acoustic features given a hidden state within a phone HMM, are often chosen to be diagonal-covariance GMMs. These have to capture the acoustic intra-class variability of local features. To reduce this variability, phone-specific HMMs can be formed for each biphone or triphone instead – as phones sound differently depending on the preceding and following phones, including this context eliminates a major source of intra-class variability (at the cost of a much larger number of first stage models).

To be tractable, HMMs make a strong simplifying assumption: given a sequence of hidden states, the observable local features are treated as conditionally independent. To still include temporal dependencies between local features, the features are often augmented with delta and double delta features [Fur86] (simply joining a few consecutive features instead would not help, as the diagonal-covariance GMMs could not model their dependency either). Several alternatives have been suggested to include longer contexts: [Rob94] replaces the phone-level HMMs with *Recurrent Neural Networks* (RNNs) accumulating information from all the past features to estimate the probability of each phone. [Her03] designed a higher-level feature extractor called TRAP-TANDEM which derives information from much longer contexts than MFCCs, after supervised training of its components. [MDH09] trained a DBN on blocks of 11 consecutive MFCCs augmented with delta and double delta coefficients, fine-tuned it to classify phones and surpassed the previous state-of-the-

art on a public phone recognition dataset. [JH11] outperform this method with a DBN trained on pooled features extracted by an RBM from raw signal excerpts. [DRMH10] train an mcRBM on blocks of 15 consecutive mel-spectral frames (i.e., MFCCs before applying the DCT), then train a usual DBN on the mcRBM's latent representations as a phone classifier (not backpropagating through the mcRBM in the supervised fine-tuning phase). As mcRBMs explicitly model the individual covariance structure of each input example, they naturally capture dependencies between frames without the need for delta features. This approach surpasses [JH11], defining the current state-of-the-art in speaker-independent speech recognition.

The problem of speaker recognition is related to, but different from speech recognition. On the surface, both deal with classifying speech recordings. However, at least speaker-independent speech recognition needs to be invariant to differences between speakers, while speaker recognition aims at capturing just those. Curiously, many speaker recognition systems still adopt MFCCs as local features, and GMMs as a "Bag of Features" model [Bei11].

Source separation is relatively easy if there are at least as many sensors as sound sources (see [Mit04]). The problem becomes much more challenging if many sound sources are mixed into a stereo or mono recording (*underdetermined* source separation). The most basic instantiation of the problem (the case of *instantaneous mixing*) assumes that N source signals $\boldsymbol{S}$ are linearly mixed into $M < N$ channels $\boldsymbol{X}$ by a mixture matrix $\boldsymbol{A}$ (which defines the mixing coefficients for each recording channel): $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{S}$. The task is to infer $\boldsymbol{A}$ and $\boldsymbol{S}$ from the observed mixture $\boldsymbol{X}$. To solve it, the space of solutions has to be restricted by making further assumptions, leading to different methods: *Independent Component Analysis* (ICA) assumes the sources to be independent and non-Gaussian, *Sparse Component Analysis* (SCA) assumes the sources to be sparse, and *Nonnegative Matrix Factorization* (NMF) assumes $\boldsymbol{X}$, $\boldsymbol{A}$ and $\boldsymbol{S}$ to be nonnegative. Which method is to be preferred depends on the data. ICA has been used on raw signals in [LLGS99] to separate three human speakers in stereo recordings. SCA is not suited well for application to signals in the time domain, because such signals are not sparse. However, it is possible to preprocess the sources with a linear transformation that results in sparser data, such as a Fourier Transform. Applying SCA to spectrograms, [BZ01] reconstruct the sources from stereo mixes of four voices, five songs or six flutes with some success (especially for the songs, notable cross-talk remains). NMF is usually applied to magnitude or power spectrograms, which are nonnegative by definition [SO06].

Adoption

Just as Text Retrieval, Computer Vision and Music Similarity Estimation systems, speech recognition builds on global models of local features. Speaker recognition does not seem to add anything, but source separation could be useful as an inter-

mediate step in processing music.

Local features used in speech processing can be directly applied to music, as both fields handle audio data. And in fact, MFCCs have been adopted to music similarity estimation from the very beginning [Foo97]. [Sca08] also applied a variation of genre-tuned TRAP-TANDEM features to genre classification, outperforming MFCCs. The prevalent global speech model, the HMM, has also already been employed for modelling music, without satisfying results (see Section 2.3.1, p. 21). However, it has only been attempted to directly model complete songs – learning and concatenating separate HMMs for some musical equivalent to phones and words might improve performance. Generative models for local features have not been adopted for music similarity estimation, though. The mcRBM approach is especially interesting: It applies a particularly powerful generative model from computer vision to audio data, treating spectrogram excerpts as the equivalent to image patches, and it achieves state-of-the-art results in speech recognition.

Several attempts have been made to apply source separation techniques to music, adding domain-specific restrictions on the extracted sources to improve performance: [VK02] assume signals to be harmonic, [Vir07] assumes continuity in time, and [Bur08] incorporates instrument timbre models. Source separation has not been employed for music similarity estimation, but an experiment in genre classification did not yield significant improvements [LLT05].

3.4 Conclusions

While this exploration revealed a lot of methods possibly useful for content-based music similarity estimation, we will restrict this thesis to a single idea and leave others open for our future work. In all three domains, unsupervisedly trained generative models show encouraging results. In particular, the recently proposed mean-covariance Restricted Boltzmann Machine (mcRBM) achieves state-of-the-art performance in object recognition and phone recognition, modeling image patches or spectrogram excerpts, respectively. Given that it has successfully been used to model speech, an application of mcRBMs to music seems straightforward and promising.

4 Proposed New Method

Drawing from ideas and methods mentioned in Chapters 2 and 3, we propose a system for music similarity estimation that is trained on large amounts of unlabeled data to learn features for describing music excerpts, then uses these features to model and compare music pieces. We will first give an overview of the system, highlighting connections to previous chapters, then discuss each stage in detail.

4.1 The Big Picture

Like existing music similarity estimation systems described in Chapter 2, our system extracts local features from music pieces, aggregates them to global models and compares these global models to estimate the similarity of songs. Figure 4.1 details our system's computation steps.

The local feature extraction stage transforms the audio signal of a music piece into a mel-frequency log-amplitude spectrogram and extracts short excerpts from it. The same type of spectrogram is also used for computing MFCCs (see Section 2.2.1, p. 11) and thus well-justified, and processing blocks of multiple consecutive frames has been shown to be an effective way of capturing the temporal evolution of music signals, as we learned in Chapter 2. Specifically, these *mel-spectral blocks* are similar to what Pampalk et al. [PRM02, Pam06], Pohle et al. [PSS+09, Poh10] and Seyerlehner et al. [SWP10, Sey10] base their hand-crafted features on, and what Pohle et al. [PKSW06] and Abdallah [Abd02] use for unsupervised learning on music by ICA.

Instead of directly aggregating these low-level local features into a global model, an abstraction stage priorly creates a higher-level description of each block. For this, the blocks are preprocessed, decorrelating and compressing the data, and then transformed into binary feature vectors by a mean-covariance Restricted Boltzmann Machine (mcRBM) [RH10], optionally followed by one or more layers of Bernoulli-Bernoulli RBMs. Unlike the low-level feature extraction stage, neither preprocessing nor the mcRBM's and RBM's feature transformations are hand-crafted: Instead, they are previously adapted to their task by unsupervised training on a large number of mel-spectral blocks extracted from music pieces. This setup for the feature extraction and abstraction chain closely follows Dahl's application of mcRBMs to speech recognition [DRMH10], but uses longer blocks so it is capable

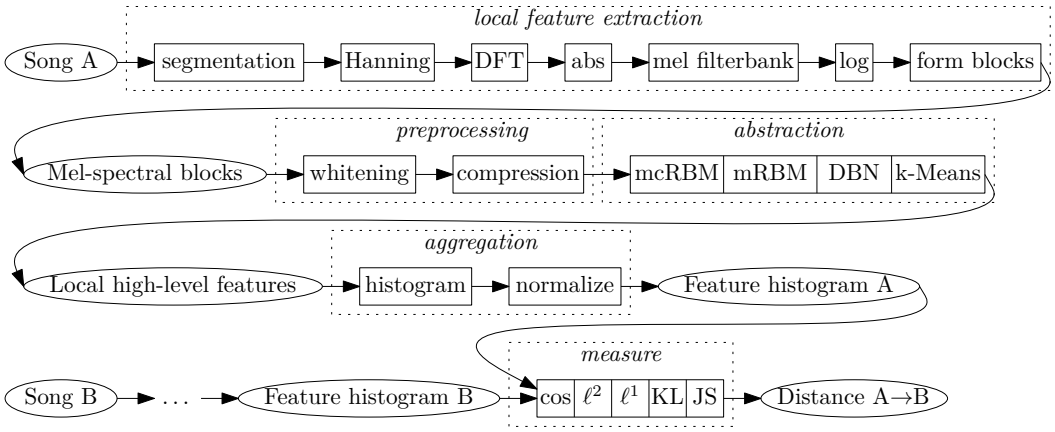


Figure 4.1: Block diagram for the proposed music similarity estimation system. Ellipses denote data, blocks denote transformations, tiled blocks represent alternatives.

of capturing musical features that are on a much larger time scale than phones, such as rhythms.

To assess the benefit of using an mcRBM for feature abstraction over more basic models, we compare it to a cheap drop-in replacement for unsupervised feature learning: k-Means clustering. Despite its simplicity, k-Means has lately been shown to outperform both sparse RBMs and mcRBMs on an image classification task [CLN10], making it an interesting benchmark candidate. It has also been applied to music feature extraction with both positive and negative results, but only in a frame-wise manner (see Section 2.3.1). Here, we apply it to the same blocks the mcRBM is trained on.

The system then builds a global model from the sequence of high-level local features extracted from a music piece. Instead of fitting a Gaussian as in [ME05], [Pam06] or [PSS+09], we compute a histogram of features. Like a Gaussian model, this discards the order of features, but it results in a vector space model similar to the bag-of-words model commonly used for text documents [Lew98], which enables the use of any document retrieval techniques such as term weighting schemes, and permits computationally cheaper distance measures than those used for comparing Gaussian distributions.

4.2 Local Feature Extraction

In the first stage, the system extracts short mel-spectral frames from the audio signal. Both the computation steps and the chosen hyperparameters are adopted

from MFCCs and can be justified as in Section 2.2.1 (p. 11). In the following, we will describe the computation in detail.

First, the signal is segmented into a sequence of (possibly overlapping) short frames. Formally, we obtain the mth frame $\boldsymbol{x}_m$ from a signal $(s_1, s_2, \ldots, s_N)$ as

$$\boldsymbol{x}_m = \begin{bmatrix} s_{((m-1)h+1)} \\ s_{((m-1)h+2)} \\ \vdots \\ s_{((m-1)h+l)} \end{bmatrix}, \quad m \in [1, M], \quad M = \left\lfloor \frac{N-l}{h} \right\rfloor + 1,$$

where l and h are the frame length and hop size in samples, respectively. In the MIR literature, MFCCs have been reported to be calculated on frames as short as 20 ms [RMT+10] and as long as 100 ms [MEDL09], typically with 50% overlap. For our system, we chose a frame length of 64 ms and a frame hop size of 32 ms, which falls into that range, but has not been optimized in our experiments.[1]

Each frame is multiplied with a Hanning window and mapped to the frequency domain using a Discrete Fourier Transform, yielding

$$\boldsymbol{y}_m = \boldsymbol{D} \operatorname{diag}(\boldsymbol{w}) \boldsymbol{x}_m,$$

where $\boldsymbol{w}$ is the Hanning window given as

$$w_i = 0.5 \left(1 - \frac{\cos(2\pi(i-1))}{l-1}\right), \quad i \in [1, l]$$

and $\boldsymbol{D}$ is the l-point DFT given as

$$D_{ij} = e^{-\frac{2\pi\sqrt{-1}(i-1)(j-1)}{l}}, \quad i, j \in [1, l].$$

The phases of these spectral frames are discarded by taking the absolute values, the frequency bins are multiplied with a mel filter bank to mimic humans' cochlear pitch perception, and the log function is applied to the resulting filter activations to imitate humans' perception of loudness. Formally, we have

$$\boldsymbol{z}_m = \log(\boldsymbol{M} |\boldsymbol{y}_m|),$$

where $\log(\cdot)$ and $|\cdot|$ denote the element-wise logarithm and absolute, respectively. The mel filter bank $\boldsymbol{M}$ is a collection of triangular filters given as

$$M_{ij} = \begin{cases} 0 & \text{if } f(j) < p(i-1) \\ \frac{f(j)-p(i-1)}{p(i)-p(i-1)} \cdot \frac{1}{2(p(i+1)-p(i-1))} & \text{if } p(i-1) \le f(j) < p(i) \\ \frac{f(j)-p(i+1)}{p(i)-p(i+1)} \cdot \frac{1}{2(p(i+1)-p(i-1))} & \text{if } p(i) \le f(j) < p(i+1) \\ 0 & \text{if } p(i+1) \le f(j) \end{cases}, \quad i \in [1, d], j \in [1, l],$$

[1] We chose it such that frames were comprised of exactly 1024 samples in our first experiments, because frame lengths of powers of two allow computing the DFT via the Fast Fourier Transform. In these experiments we worked with music sampled at 16 kHz, hence a frame length of 64 ms. For later experiments, we kept this length of 64 ms independently of the sample rate.

where $f(j) = \frac{(j-1)r}{2(l-1)}$ is the frequency of the jth DFT bin at a sampling rate of r in Hz, $d \ll l$ is the number of mel filters, and $p(i-1)$, $p(i)$, $p(i+1)$ are the lowest, peak, and highest frequency of the ith mel filter in Hz, respectively. Adjacent filters overlap such that each filter's peak coincides with the next filter's lower and the previous filter's upper limit, and each filter is scaled to unity energy (see Figure 4.2). The filters are spaced linearly on the mel scale such that the lower limit of the first filter equals a given frequency $p(0) = f_{min}$, and the upper limit of the dth filter equals $p(d+1) = f_{max}$. Computing the remaining peak frequencies is done by mapping $p(0)$ and $p(d+1)$ to the mel scale as

$$P(i) = 1127 \log\left(1 + \frac{p(i)}{700}\right), \quad i \in \{0, d+1\},$$

regularly spacing the other peaks on the mel scale as

$$P(i) = \frac{i}{d+1}\Big(P(d+1) - P(0)\Big), \quad i \in [1, d],$$

then mapping them back to Hertz frequencies as

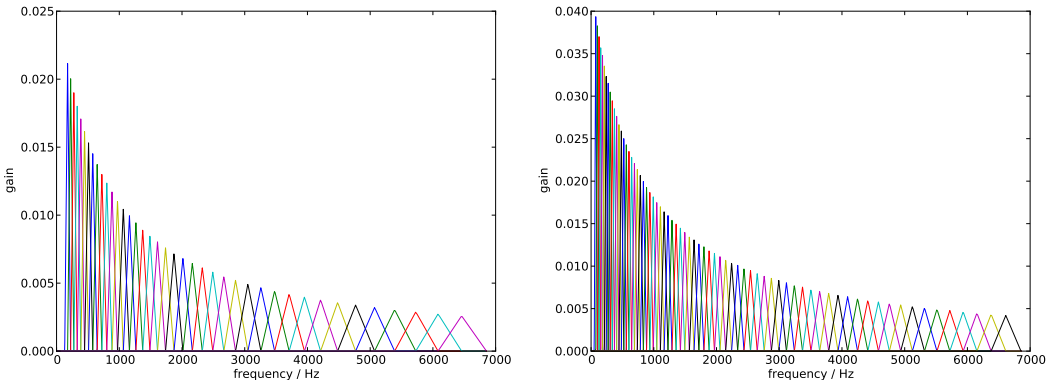
$$p(i) = 700\left(e^{\frac{P(i)}{1127}} - 1\right), \quad i \in [1, d].$$

We use two different mel filter banks in our experiments: $d = 40$ log-spaced triangular filters from $f_{min} = 130\,\text{Hz}$ to $f_{max} = 6854\,\text{Hz}$, and 70 filters from 50 to 6854 Hz (both depicted in Figure 4.2). The first corresponds to the default setting of *yaafe* [MEF+10], an audio feature extraction software targetted at music information retrieval, the latter has been manually tuned in an informal listening test to provide a clear reconstruction of a short music excerpt featuring drums and an acoustic guitar, by increasing the frequency resolution and including deeper bass frequencies. We did not increase the upper frequency limit because at least for MFCCs, this may make the features sensitive to MP3 compression [SPLS06].

The sequence of mel-spectral frames $(\boldsymbol{z}_m)$ is then transformed into a sequence of overlapping blocks $(\boldsymbol{b}_o)$ by joining fixed numbers of consecutive frames, similarly to how we initially formed frames from a sequence of samples:

$$\boldsymbol{b}_o = \Big[\boldsymbol{z}_{((o-1)H+1)} \quad \boldsymbol{z}_{((o-1)H+2)} \quad \cdots \quad \boldsymbol{z}_{((o-1)H+L)}\Big], \quad o \in [1, O],\ O = \left\lfloor \frac{M-L}{H} \right\rfloor + 1,$$

where L and H are the block length and hop size in frames, respectively. [DRMH10] used a block length of 15 frames (a hop size does not apply to their setting), but as their system was targetted at speech recognition, this choice is not necessarily suited well for our task. In our experiments, in addition to a block length of 15 frames, we thus tried larger lengths of 39 and 75 frames to enable the system to capture long-term musical features. While 75 frames (corresponding to about 2.5 s of audio) are



(a) Default mel bank

(b) High-resolution mel bank

Figure 4.2: Two alternative mel filter banks we use for mapping the linear-frequency magnitude spectra onto a mel-frequency scale. Each triangular filter corresponds to a single mel band. The filters are equidistant on the mel scale, resulting in log-spaced peak frequencies, and are each scaled to an energy of unity. The second filter bank obtains a higher frequency resolution and captures deeper bass frequencies.

still shorter than what is used for hand-crafted rhythmic features (e.g., fluctuation patterns [PRM02] are computed on excerpts of 6 s), our computational resources were insufficient for easily processing larger blocks. Additionally, we experimented with shorter blocks of 9 frames, which were the shortest blocks yielding interesting features. We set the hop size to the smallest possible value of 1 frame to prevent any aliasing effects (cf. [CLN10, p. 7]). We did not try larger hop sizes as we do not expect a positive effect on the similarity estimation quality, only on computational efficiency, which is merely a secondary goal.

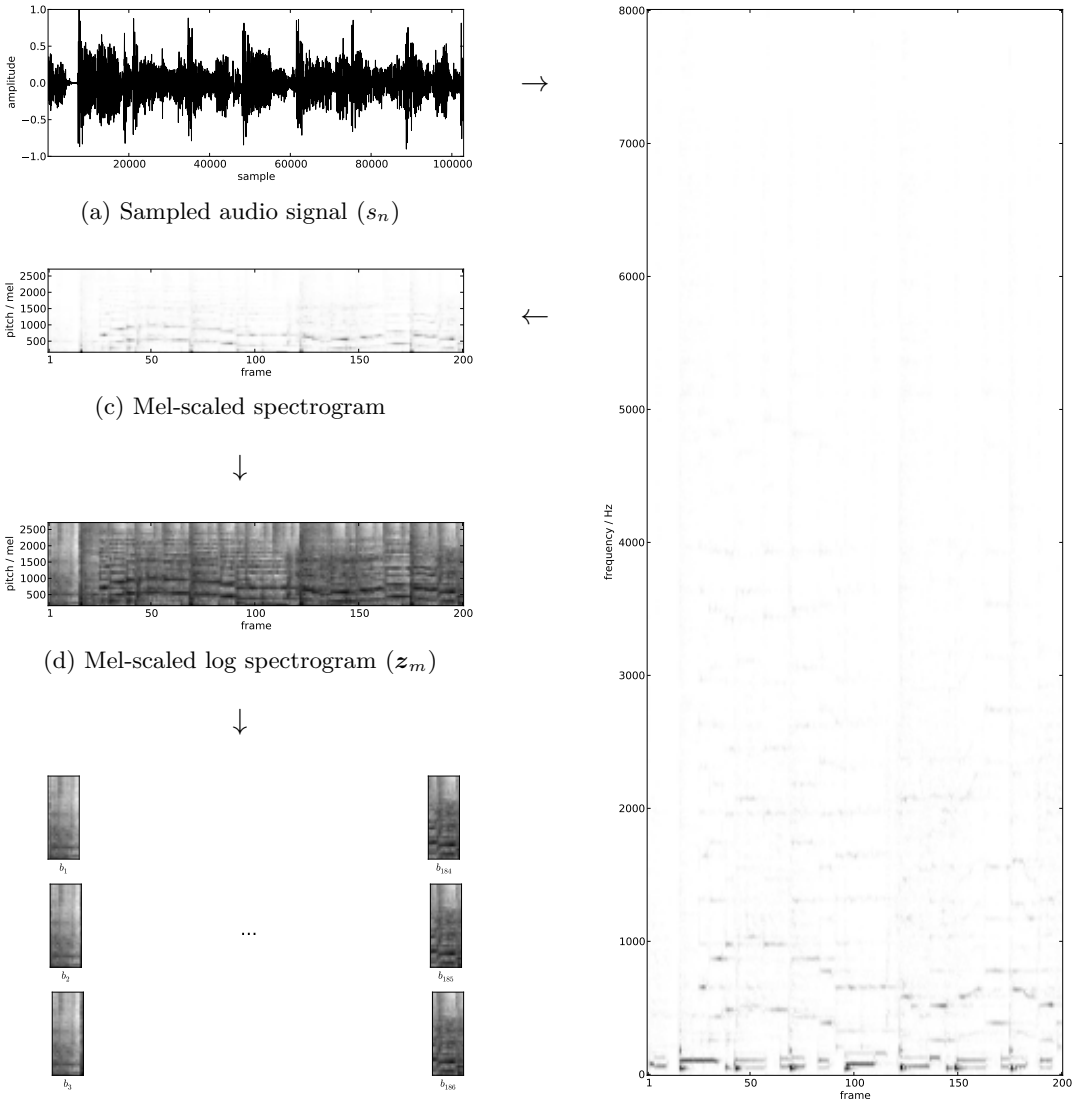
In summary, we obtain the mth mel-spectral frame $\boldsymbol{z}_m$ from a signal $(s_1, s_2, \ldots, s_N)$ as

$$\boldsymbol{z}_m = \log\left(\boldsymbol{M}\left|\boldsymbol{D}\,\mathrm{diag}(\boldsymbol{w})\begin{bmatrix} s_{((m-1)h+1)} \\ s_{((m-1)h+2)} \\ \vdots \\ s_{((m-1)h+l)} \end{bmatrix}\right|\right),$$

then form mel-spectral blocks $\boldsymbol{b}_o$ as

$$\boldsymbol{b}_o = \begin{bmatrix} \boldsymbol{z}_{((o-1)H+1)} & \boldsymbol{z}_{((o-1)H+2)} & \cdots & \boldsymbol{z}_{((o-1)H+L)} \end{bmatrix}.$$

As depicted in Figure 4.3, this can also be interpreted as computing a mel-scaled spectrogram and extracting patches at different positions in time, drawing an analogy to feature extraction schemes from the field of Computer Vision. Note that the



(a) Sampled audio signal (s_n)

(c) Mel-scaled spectrogram

(d) Mel-scaled log spectrogram ($\boldsymbol{z}_m$)

(e) Mel-spectral blocks ($\boldsymbol{b}_o$)

(b) Spectrogram ($\boldsymbol{y}_m$)

Figure 4.3: The local feature extraction process interpreted as computing a spectrogram, mapping it to the mel scale and extracting patches. The audio signal (a), here a 6-seconds guitar solo accompanied by bass and drums, is transformed to a spectrogram (b). The guitar solo becomes visible as two dim curves, the bass as horizontal stripes at the bottom, and percussion as regularly spaced vertical stripes. Using the mel filterbank of Figure 4.2a, frequencies are mapped to mel pitches (c). Note that the harmonics of the guitar are now spaced uniformly, as the mel scale is logarithmic in frequency, and that frequencies below 130 Hz are not included. Applying a log function yields the mel-scaled log spectrogram (d), which emphasizes fine low-volume structures. At each valid position, a mel-spectral block is cut out of this spectrogram (e).

computation steps for $\boldsymbol{z}_m$ are the same steps as performed in computing MFCCs (see Section 2.2.1, p. 11), just omitting the final Discrete Cosine Transform. As we will decorrelate blocks of consecutive frames anyway, an invertible linear transform such as the DCT would not have any effect at this point. Computing the frame-wise DCT and dropping the highest components, as is commonly done for MFCCs as an approximation to PCA compression, would be similarly unhelpful as we are compressing blocks of frames later.[2]

4.3 Preprocessing

The mel-spectral blocks described in the previous section are highly correlated: In addition to correlations inherent in music, the overlapping of mel-spectral frames induces correlations in time, and the overlapping of triangular filters in the mel filter bank induces correlations in frequency. While mcRBMs can deal with correlated data, training converges faster on *whitened* inputs [RKH10], and whitening also significantly improves results with k-Means features (as we will see in Section 5.4.3.3; also cf. [CLN10]).

Apart from that, the blocks are high-dimensional, with up to 2730 (39 frames of 70 mel bands) and 3000 (75 frames of 40 mel bands) dimensions. Given the high correlation between feature dimensions, we can compress the mel-spectral blocks to lower-dimensional vectors without losing much information, thus reducing the computational effort for the feature abstraction stage.

Both whitening and compression can be accomplished with computationally cheap linear transformations learned from the data. In particular, we experimented with the PCA and ZCA transformations, which we will motivate and describe in the following.

4.3.1 Whitening

Whitened data is decorrelated and has zero mean and unit variance in each dimension, so the covariance matrix is the identity matrix. Any data can be whitened, or *sphered*, by an affine transformation adapted to the data.

Assume we have extracted a set of t mel-spectral blocks from music pieces we use as training data. We reshape each block into an Ld-dimensional row vector (L denotes the block length, d the number of mel bands) and stack these vectors into a $t \times Ld$-dimensional matrix $\boldsymbol{X}$ of training data. We will now find an affine transformation for whitening the training data, which we can subsequently apply to any similarly distributed data.

[2] Actually, compressing the frames by DCT in this way before compressing the blocks would even lead to a suboptimal compression scheme in terms of the mean squared error (MSE) of the reconstruction, because the DCT is only an approximation of the optimal PCA transform.

To mean-center the data, we simply subtract the population mean:

$$\boldsymbol{Y} = \boldsymbol{X} - \bar{\boldsymbol{X}}, \quad \text{where } \bar{X}_{ij} = \frac{1}{t}\sum_{n=1}^{t} X_{nj}$$

Additionally, we need to rotate and scale the mean-centered training data $\boldsymbol{Y}$ such that the transformed data $\boldsymbol{Z}$ has a covariance matrix equal to the identity matrix:

$$\frac{1}{t-1}\boldsymbol{Z}^T\boldsymbol{Z} = \boldsymbol{I}$$

Expressing the transformation as a matrix multiplication

$$\boldsymbol{Z} = \boldsymbol{Y}\boldsymbol{W},$$

we thus need to find a matrix $\boldsymbol{W}$ such that

$$\begin{aligned}
\frac{1}{t-1}\boldsymbol{W}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{W} &= \boldsymbol{I} \\
\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{W} &= (t-1)\boldsymbol{W} \\
\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{Y}^T\boldsymbol{Y} &= (t-1)\boldsymbol{I} \\
\boldsymbol{W}\boldsymbol{W}^T &= (t-1)(\boldsymbol{Y}^T\boldsymbol{Y})^{-1}. \qquad (4.1)
\end{aligned}$$

Many such whitening matrices are possible. By imposing additional constraints on $\boldsymbol{W}$, we can derive different whitening transforms holding special properties [BS97].

4.3.1.1 Principal Component Analysis

If we restrict the columns of $\boldsymbol{W}$ to be orthogonal, that is

$$\boldsymbol{W}\boldsymbol{W}^T = \operatorname{diag}(\boldsymbol{s}) \qquad (4.2)$$

for some Ld-dimensional vector $\boldsymbol{s}$, the solution is obtained from the eigendecomposition of the covariance matrix of the training data: Being real and symmetric, the covariance matrix can be decomposed as

$$\frac{1}{t-1}\boldsymbol{Y}^T\boldsymbol{Y} = \boldsymbol{E}\operatorname{diag}(\boldsymbol{\lambda})\boldsymbol{E}^T, \qquad (4.3)$$

where the columns of $\boldsymbol{E}$ form an orthonormal set of eigenvectors and $\boldsymbol{\lambda}$ is the vector of corresponding eigenvalues. Substituting (4.3) into (4.1), we can solve for $\boldsymbol{W}$:

$$\begin{aligned}
\boldsymbol{W}\boldsymbol{W}^T &= \left(\boldsymbol{E}\operatorname{diag}(\boldsymbol{\lambda})\boldsymbol{E}^T\right)^{-1} \\
\boldsymbol{W}\boldsymbol{W}^T &= \boldsymbol{E}\operatorname{diag}(\boldsymbol{\lambda})^{-1}\boldsymbol{E}^T \qquad (4.4) \\
\boldsymbol{W}\boldsymbol{W}^T &= \boldsymbol{E}\left(\operatorname{diag}(\boldsymbol{\lambda})^{-\frac{1}{2}}\operatorname{diag}(\boldsymbol{\lambda})^{-\frac{1}{2}T}\right)\boldsymbol{E}^T \\
\boldsymbol{W}\boldsymbol{W}^T &= \left(\boldsymbol{E}\operatorname{diag}(\boldsymbol{\lambda})^{-\frac{1}{2}}\right)\left(\boldsymbol{E}\operatorname{diag}(\boldsymbol{\lambda})^{-\frac{1}{2}}\right)^T \\
\boldsymbol{W} &= \boldsymbol{E}\operatorname{diag}(\boldsymbol{\lambda})^{-\frac{1}{2}}. \qquad (4.5)
\end{aligned}$$

Using the orthogonality of $\boldsymbol{E}$ we can verify that $\boldsymbol{W}$ fulfills (4.2) with $\boldsymbol{s} = \boldsymbol{\lambda}$. Note that computing $\operatorname{diag}(\boldsymbol{\lambda})^{-\frac{1}{2}}$ just requires to separately take each element of $\boldsymbol{\lambda}$ to the power of $-\frac{1}{2}$.

This solution is related to the *Principal Component Analysis* (PCA): It rotates the (mean-centered) input data to the space of its principal components, decorrelating it, then divides each component by the square root of its variance, whitening the data. To understand the latter, note that the variance of the data $\boldsymbol{Y}$ projected onto a column $\boldsymbol{e}_{:j}$ of $\boldsymbol{E}$ is directly given by the corresponding eigenvalue λ_j in $\boldsymbol{\lambda}$, as we can infer from the eigenvalue equation by using $\boldsymbol{e}_{:j}^T \boldsymbol{e}_{:j} = 1$:

$$\left(\frac{1}{t-1}\boldsymbol{Y}^T\boldsymbol{Y}\right)\boldsymbol{e}_{:j} = \lambda_j \boldsymbol{e}_{:j}$$
$$\frac{1}{t-1}\left(\boldsymbol{e}_{:j}^T\boldsymbol{Y}^T\boldsymbol{Y}\boldsymbol{e}_{:j}\right) = \lambda_j \boldsymbol{e}_{:j}^T\boldsymbol{e}_{:j} = \lambda_j.$$

4.3.1.2 Zero-phase Component Analysis

Constraining $\boldsymbol{W}$ to be symmetric, that is

$$\boldsymbol{W} = \boldsymbol{W}^{\boldsymbol{T}}$$
$$\boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{W}^2, \tag{4.6}$$

we obtain a different solution. Substituting (4.6) into (4.4) and solving for $\boldsymbol{W}$, we get

$$\boldsymbol{W}^2 = \boldsymbol{E}\operatorname{diag}(\boldsymbol{\lambda})^{-1}\boldsymbol{E}^T$$
$$\boldsymbol{W} = \left(\boldsymbol{E}\operatorname{diag}(\boldsymbol{\lambda})^{-1}\boldsymbol{E}^T\right)^{\frac{1}{2}}$$
$$\boldsymbol{W} = \boldsymbol{E}\operatorname{diag}(\boldsymbol{\lambda})^{-\frac{1}{2}}\boldsymbol{E}^T. \tag{4.7}$$

This whitening matrix can be interpreted as rotating the (mean-centered) input data to the space of its principal components, dividing each component by its standard deviation, then rotating the data back to its original space [Kri09]. It is known as the *Zero-phase Component Analysis* [BS97], and we will shortly see why.

4.3.1.3 Discussion

PCA is a well-known data preprocessing step used in many fields of computing. It has been chosen as a preprocessor for mean-covariance RBMs [RH10, DRMH10] and has been shown to strongly relate to k-Means clustering [DH04]. ZCA, on the other hand, has recently become popular for decorrelating image patches especially in unsupervised settings similar to ours: ZCA whitened image data has successfully

been fed to Restricted Boltzmann Machines [CLN10], Deep Belief Nets [Kri09], covariance RBMs [RKH10], Sparse Autoencoders [CN11] and k-Means based vector quantizers [CLN10, CN11]. Literature does not indicate which of the two methods is suited better for our problem, so to make an informed decision on the whitening method to use in our setup, we will set PCA in contrast to ZCA, analyzing their common properties and their differences.

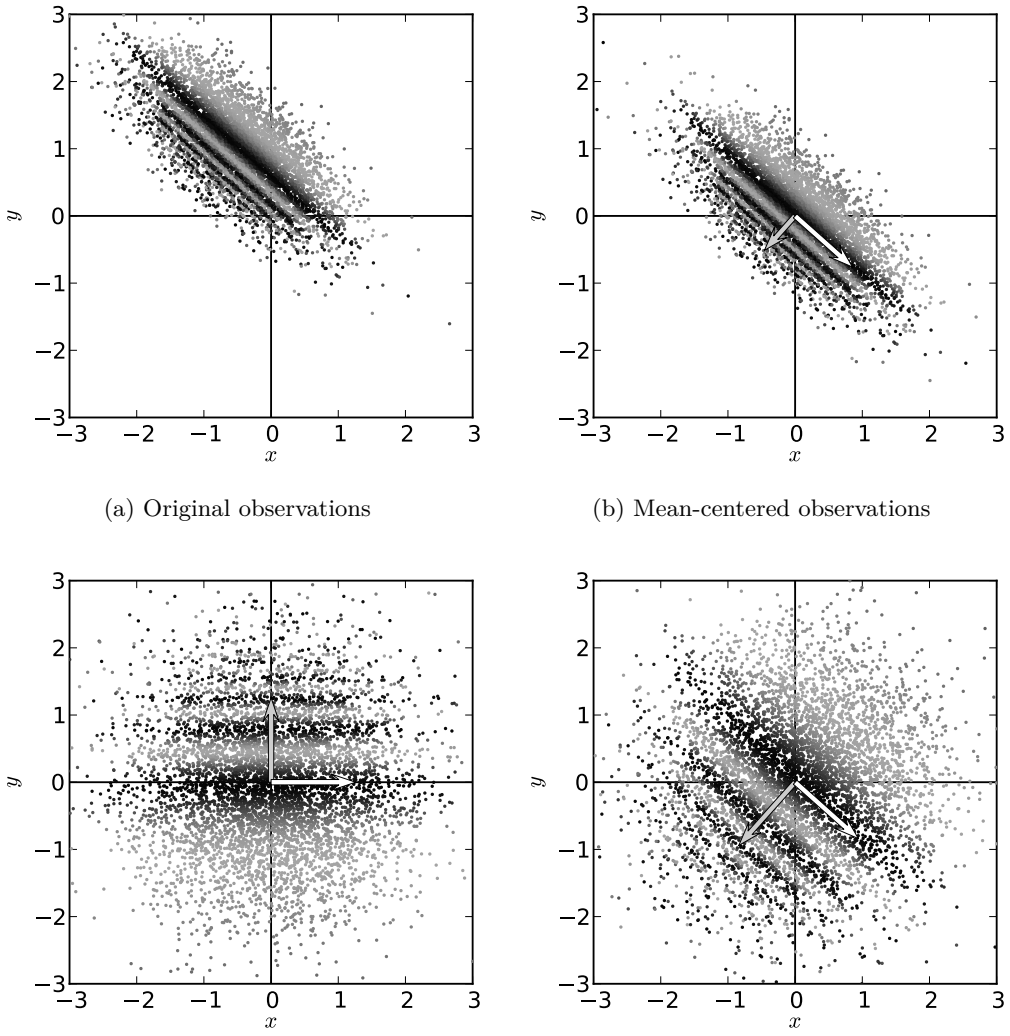
Let us first consider the two-dimensional case to get a better understanding of the transformations employed by PCA and ZCA. Figure 4.4a shows a set of two-dimensional correlated Gaussian[3] observations $\boldsymbol{X}$. By subtracting the mean, we remove first-order statistics from the data, obtaining the set of mean-centered observations $\boldsymbol{Y}$ (Figure 4.4b). Note that any rotated, scaled or reflected set $\boldsymbol{AY}$ remains mean-centered. From the eigendecomposition of the covariance matrix we obtain the two principal components of $\boldsymbol{Y}$, shown as arrows. Each arrow is scaled by the standard deviation of the data in direction of the principal component, i.e., by the square root of the respective eigenvalue. Principal Component Analysis rotates the data such that the principal components become aligned with the coordinate axes, which decorrelates the data. PCA whitening additionally scales the data to unit variance, sphering it (Figure 4.4c). Note that any further rotation or reflection of the data (i.e., multiplication by an orthogonal matrix $\boldsymbol{E}$) retains this spheric property. ZCA employs the specific rotation which reverses the initial rotation of PCA, returning the data to its original orientation (Figure 4.4d). Alternatively, ZCA can be thought of sphering the data by scaling it along its principal components, without rotating it at all.

As was already evident from the formulae, the two transformations are equivalent except for a rotation of the sphered data. However, we can make two interesting observations:

1. The unwhitened data (Figure 4.4b) has a principal component of high variance (white arrow) and another one of lower variance (gray arrow). After whitening, these components become indistinguishable using only second-order statistics (as perfectly illustrated by our Gaussian example, which does not have any higher-order statistical moments, resulting in Figures 4.4c and 4.4d looking virtually the same safe for the gray tones). With PCA whitening, however, they can be trivially distinguished in the new coordinate system, which may facilitate learning in case the principal components capture relevant structure in the data. Additionally, this will prove useful for compression.

2. The ZCA transform keeps the orientation of the data unchanged: If dimension x denoted the value of the first pixel of a gray-tone image, it will still be

[3] Gaussian distributions are of course a special case, as any whitening transformation results in the multivariate standard normal distribution, but it illustrates two key insights without distracting by the shape of the distribution.



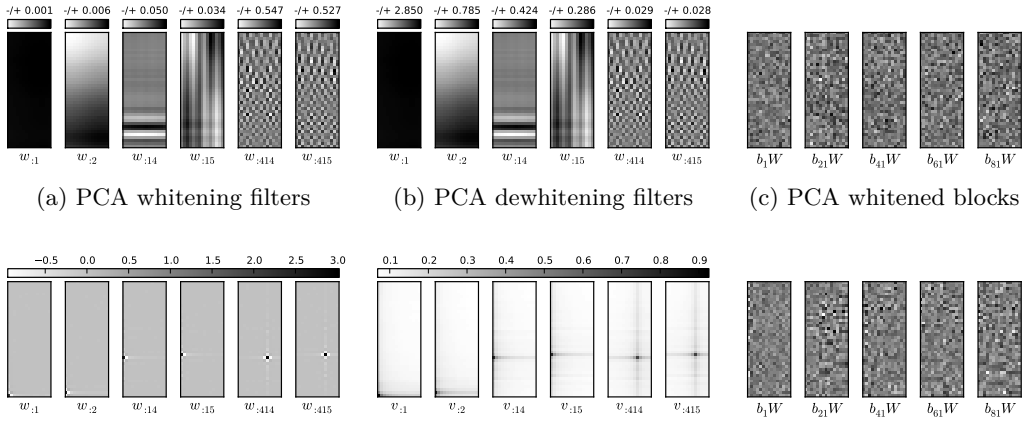
(a) Original observations

(b) Mean-centered observations

(c) PCA whitened observations

(d) ZCA whitened observations

Figure 4.4: Scatter plots contrasting PCA to ZCA whitening. The observations (a), here a set of 2-dimensional data points (shades of gray merely serve as visual cues), are mean-centered (b) and the principal components, denoted by arrows scaled by the standard deviation of the data in the respective direction, are estimated from the samples. Both transformations scale the data to unit variance in direction of the principal components, thereby sphering it. PCA (c) rotates the data to align the principal components with the coordinate axes, allowing to compress data by projection onto a subset of the axes. ZCA (d) keeps the orientation, allowing to interpret each dimension as before.



(a) PCA whitening filters (b) PCA dewhitening filters (c) PCA whitened blocks

(d) ZCA whitening filters (e) ZCA dewhitening filters (f) ZCA whitened blocks

Figure 4.5: PCA and ZCA whitening filters and dewhitening filters (or bases) for mel-spectral blocks, as well as whitened data (cf. Figure 4.3e).

associated with this particular pixel after the whitening step. Not changing the interpretation of dimensions could be useful in case of heterogeneous data such as multimodal signals. As we will see in the following, it also manifests itself in the transformation matrices.

Having formed a basic understanding of the two transformations, let us now see how they apply to our task of decorrelating mel-spectral blocks. Recalling that the transformed data is computed as $\boldsymbol{Z} = \boldsymbol{Y}\boldsymbol{W}$, the columns of $\boldsymbol{W}$ can be regarded as a set of filters applied to the data, in which the activation of a single filter $\boldsymbol{w}_{:j}$ for a data point $\boldsymbol{y}_{i:}$ is given by their dot product $\boldsymbol{y}_{i:}\boldsymbol{w}_{:j}$. Reshaping the filters to $d \times L$-dimensional matrices allows us to directly visualize their effect on mel-spectral blocks.

Figure 4.5 visualizes a subset of the filters learned from a million blocks extracted from music signals. We can immediately see a striking difference between PCA and ZCA whitening: The former yields global filters, the latter yields localized filters of only small support. This matches our second observation above: ZCA whitening keeps each spectrogram bin at its position, decorrelating it from the other bins, while PCA transforms the data to a very different coordinate system. The PCA dewhitening filters (Figure 4.5b) are just scaled versions of the whitening filters (due to the constraint on the columns of $\boldsymbol{W}$ being orthogonal); the ZCA dewhitening filters (Figure 4.5e) reintroduce the correlations removed by the whitening filters. Note how they spread widely both horizontally and vertically, indicating strong correlations in time and frequency, and note the horizontal artifacts especially in

the last two filters shown, resembling harmonics – both is not found in ZCA filters learned on natural images, cf. [BS97, Kri09]. Another interesting observation is that the ZCA filters are almost symmetrical[4] and almost convolutional, which could be utilized for computational optimizations. This symmetricity of the filters lead to the name *Zero-phase* Component Analysis (considering the spectrum of the filters).

Looking at the whitened data, however, seems to counter our insights: Neither the PCA nor the ZCA whitened data (Figures 4.5c and 4.5f, respectively) resemble the original inputs (Figure 4.3d). While this was expected for PCA, it needs further explanation for ZCA. As the whitening step scales each principal component to unit variance, components of small eigenvalues are strongly amplified, and due to the nature of the input data, these components have a high spatial frequency such as the last two filters in Figure 4.5a (compare their scaling to the other filters shown). Furthermore, the filters remove horizontal and vertical correlations. As a result, the visible structure in the spectrogram, such as the guitar and percussion, becomes invisible, while fine structures are magnified and look like noise. This would not happen for smoother input such as the linear-magnitude spectrogram (Figure 4.3c) or most natural images [Kri09].

In summary, our investigation does not uncover a clear advantage of one of the methods over the other. On the contrary: As the only difference is a rotation of the data, at least for k-Means both methods will yield the same results (because k-Means is invariant to orthogonal transformations [DH04, p. 5]). For other feature abstractors, we will try both PCA and ZCA in our experiments. Nevertheless, the insights of this analysis will prove useful in the following section.

4.3.2 Compression

In addition to decorrelating and whitening the data, we want to reduce the dimensionality of the data especially for mel-spectral blocks of high frequency resolution or many frames. The obvious way to compress data is by redundancy reduction, another one is by removing irrelevant information (we include lossy compression in this discussion). As correlations are indicating (linear) redundancies, analyzing correlations in the data is a possible first step to compressing it, and we already do this for whitening the data. In fact, we can combine whitening and compression into a single step.

Examining Figure 4.4b again, Principal Component Analysis finds the direction of highest variance in the data, and the orthogonal direction. Imagine the special case of a dataset having zero variance in the latter direction: the dataset would lie on a one-dimensional linear manifold embedded in a two-dimensional space, and

[4] Note that this is not implied by the symmetricity constraint on $\boldsymbol{W}$ – the constraint means that the first filter determines the first component (here, the bottom left pixel) of all other filters, but it still allows individual filters to be asymmetric.

could be losslessly compressed to one dimension by projection onto the direction of highest variance (i.e., by regarding only the x coordinates of the samples in Figure 4.4c, as all y coordinates would be zero). If a dataset does not have components of zero variance, but variances close to zero, we can imagine that we would not lose much information by discarding these components – and in fact, applying PCA to a dataset and dropping the components of lowest eigenvalues yields a linear transformation to a lower-dimensional space which is optimal in terms of the mean squared error of the reconstruction [Pea01]. Formally, for a given target dimensionality m, a minimum solution to

$$\frac{1}{t}\sum_{i=1}^{t}\|\boldsymbol{y}_{i:}-(\boldsymbol{y}_{i:}\boldsymbol{U})\,\boldsymbol{U}^T\|^2,$$

where t is the size of the training dataset and $\boldsymbol{y}_{i:}$ is the ith mean-centered training data point, is obtained by setting $\boldsymbol{U}$ to the transformation matrix formed from the columns of the PCA whitening matrix that correspond to the m largest eigenvalues.

However, the components of lowest variance are not necessarily irrelevant, that is, the reconstruction error may not always be a meaningful measure for the information lost in compressing the data. To better understand what kind of information would be lost in our case, let us examine Figure 4.5 again. We sorted the filters in order of decreasing eigenvalues, so performing PCA compression in addition to whitening would be accomplished by simply discarding output dimensions starting with the top right pixel in Figure 4.5c. As the corresponding filters have a high spatial frequency (similar to the two rightmost filters of Figure 4.5a), such a compressing scheme would be similar to applying a low-pass filter to the data. This is often regarded as a form of noise filtering, which is, however, only true for data in which noise is associated with high frequencies. In our experiments we will see whether it helps or hurts the performance of our music similarity measure.

Accepting the reconstruction error as a measure of information loss, the number of dimensions to discard can be chosen by defining the acceptable error. This can be trivially accomplished by setting a threshold on the discarded variance, given by the sum of eigenvalues of dropped principal components. In fact, this is exactly equal to the reconstruction error [Bis06, p. 565]:

$$\frac{1}{t}\sum_{i=1}^{t}\|\boldsymbol{y}_{i:}-(\boldsymbol{y}_{i:}\boldsymbol{U})\,\boldsymbol{U}^T\|^2=\sum_{i=m+1}^{L\cdot d}\lambda_i,$$

where $L \cdot d$ is the dimension of the mel-spectral blocks, and $\boldsymbol{\lambda}$ is the vector of eigenvalues in decreasing order. In our experiments, we will set the threshold in relation to the total variance of the input data, a common setting being to retain 99% of the original variance.

Note that this method of compressing the data only applies to PCA whitening. While we can also drop principal components (columns of $\boldsymbol{E}$) in the construction

of the ZCA whitening transform, this would only project the data onto a lower-dimensional linear manifold embedded in the original feature space. For experiments on high-dimensional mel-spectral blocks we will thus restrict ourselves to PCA whitening, additionally using ZCA whitening as a comparative method only when no dimensionality reduction is needed.

4.4 Feature Abstraction

The core of our system is a data-driven feature abstraction stage obtaining useful high-level descriptions of local features. In particular, we train a mean-covariance Restricted Boltzmann machine (mcRBM) unsupervisedly as a generative model on a large unlabeled training set of mel-spectral blocks, then use its latent representations as high-level features describing any input data.

As mcRBMs are a sophisticated type of Restricted Boltzmann Machines (RBMs), we will introduce the basic binary RBM before proceeding to mcRBMs. RBMs can also be trained on the latent representations of another RBM, further abstracting the features. The resulting stack of RBMs forms a Deep Belief Net (DBN), which we will briefly describe as well. For all these models we will present their basic idea, explain the maths (with derivations for the binary RBM in the Appendix) and give several intuitions on how and why they work.

In addition, we experimented with k-Means clustering as an alternative way of generating feature descriptions in an unsupervised manner. We used an algorithm that proceeds data in *mini-batches* to allow clustering large datasets, and combined it with a seeding technique known as *k-means++*, both of which will be described at the end of this section.

4.4.1 Restricted Boltzmann Machines (RBMs)

4.4.1.1 Basics

A Restricted Boltzmann Machine (RBM) [Smo86] is an undirected graphical model (or Markov Random Field) of two types of units: visible units representing observ-

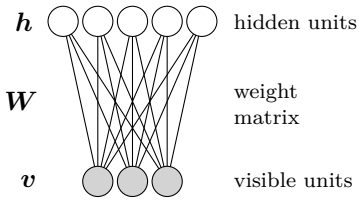


Figure 4.6: A Restricted Boltzmann Machine (RBM)

able data, and hidden units representing latent variables. The model is restricted to a bipartite graph such that the visible units form a layer that is fully connected to a layer of hidden units, with no intra-layer connections (Figure 4.6).

An RBM specifies a joint probability density $p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{\theta})$ of visible and hidden unit states $\boldsymbol{v}$ and $\boldsymbol{h}$, where $\boldsymbol{\theta}$ denotes the model parameters (such as the matrix of connection weights between units, $\boldsymbol{W}$).[5] It is an energy-based model [LCH+06] that expresses the probability density in terms of an *energy function* $E(\boldsymbol{v}, \boldsymbol{h}, \boldsymbol{\theta})$ as a Boltzmann distribution

$$p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{\theta}) = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h},\boldsymbol{\theta})}}{Z(\boldsymbol{\theta})}, \tag{4.8}$$

where

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{u},\boldsymbol{g}} e^{-E(\boldsymbol{u},\boldsymbol{g},\boldsymbol{\theta})} \tag{4.9}$$

is the normalizing *partition function*.[6]

Training an RBM means finding parameters $\boldsymbol{\theta}$ such that the marginal distribution of the visibles, defined by $p(\boldsymbol{v}|\boldsymbol{\theta})$, approximates the observed distribution of training data, which is equivalent to maximizing the likelihood of the model under the training data. Training is unsupervised, that is, it requires only a set of unlabeled data points to learn from.

Once an RBM is trained, the states of the hidden units conditioned on a data vector, distributed according to $p(\boldsymbol{h}|\boldsymbol{v}, \boldsymbol{\theta})$, can be interpreted as features extracted from the data. As there are no intra-layer connections in the model, the hidden units are conditionally independent given a data vector, so their states can be inferred efficiently and in parallel. This is beneficial when using an RBM as a feature extractor.

Different types of RBMs can be defined by choosing the energy function appropriately. In this section, we will only consider the most basic type of RBMs, which has binary (Bernoulli-distributed) visible and hidden units (when conditioned on the respective opposite layer of units). We will first take a deeper look at the mathematics, then give different interpretations for what the model represents and how it is trained. More advanced types of RBMs, including the mean-covariance Restricted Boltzmann Machine, will be introduced in Section 4.4.2.

4.4.1.2 Mathematics

We will now give a more detailed presentation of the mathematics behind Restricted Boltzmann Machines. Most of this thesis can be understood from the higher-

[5] Depending on the type of RBM, $\boldsymbol{v}$ and $\boldsymbol{h}$ can be samples of continuous or discrete random variables, so $p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{\theta})$ may be a probability density function or a probability mass function. To simplify the description, we will use the term "density" throughout.

[6] In the continuous case, sums over configurations $\boldsymbol{u}$ or $\boldsymbol{g}$ become integrals.

level intuitions in Section 4.4.1.3, but for technical details an understanding of the mathematical background is required.

Model The most basic type of RBMs is guided by the following energy function:

$$\begin{aligned} E_b(\boldsymbol{v}, \boldsymbol{h}, \boldsymbol{\theta}) &= -\sum_{i,j} v_i W_{ij} h_j - \sum_i v_i a_i - \sum_j h_j b_j \\ &= -\boldsymbol{v}^T \boldsymbol{W} \boldsymbol{h} - \boldsymbol{v}^T \boldsymbol{a} - \boldsymbol{h}^T \boldsymbol{b}, \end{aligned} \tag{4.10}$$

where the model parameters $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b})$ comprise the connection weights, visible and hidden bias terms, respectively, and $\boldsymbol{v}$ and $\boldsymbol{h}$ are *binary* states of the visible and hidden units.

The energy function induces a probability density $p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{\theta})$ over joint configurations of $\boldsymbol{v}$ and $\boldsymbol{h}$ according to Equation 4.8. From this density we can derive several other densities discussed in the following.

The marginal probability density of the visibles,

$$p(\boldsymbol{v}|\boldsymbol{\theta}) = \sum_g p(\boldsymbol{v}, \boldsymbol{g}|\boldsymbol{\theta}) = \frac{\sum_g e^{-E(\boldsymbol{v}, \boldsymbol{g}, \boldsymbol{\theta})}}{Z(\boldsymbol{\theta})}, \tag{4.11}$$

is the density of observable data as estimated by the model; i.e., it defines the distribution to sample from if using the RBM as a generative model.

Alternatively, the generative model may be defined via an implicit prior over the hiddens and the conditional probability of the visibles given the hiddens (see Section 4.4.3). The implicit prior is the marginal density of the hiddens:

$$p(\boldsymbol{h}|\boldsymbol{\theta}) = \sum_u p(\boldsymbol{u}, \boldsymbol{h}|\boldsymbol{\theta}) = \frac{\sum_u e^{-E(\boldsymbol{u}, \boldsymbol{h}, \boldsymbol{\theta})}}{Z(\boldsymbol{\theta})} \tag{4.12}$$

The conditional probabilities depend on the chosen energy function. For the function defined in Equation 4.10, we obtain the following probability distributions (see Appendix A for the derivation):

$$P(h_k = 1|\boldsymbol{v}, \boldsymbol{\theta}) = \sigma\Big(b_k + \sum_i v_i W_{ik}\Big) \tag{4.13}$$

$$P(v_k = 1|\boldsymbol{h}, \boldsymbol{\theta}) = \sigma\Big(a_k + \sum_j W_{kj} h_j\Big) \tag{4.14}$$

Here, σ denotes the logistic sigmoid function

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Equations 4.13 and 4.14 define the probability that a given unit takes its "on" state. This probability is referred to as the *activation* of a unit. The first equation is needed when using an RBM as a feature extractor, and both conditional probabilities are also needed during training.

Training The goal of training an RBM is to find parameters $\boldsymbol{\theta}$ such that it is likely to generate training data, and unlikely to generate vectors that do not resemble the training data. More formally, this objective can be defined in two ways:

- Minimizing the Kullback-Leibler divergence between the observed distribution of the training data and the model distribution $p(\boldsymbol{v}|\boldsymbol{\theta})$, or
- Maximizing the likelihood of the model under the training data.

Both objectives are actually equivalent [vdM09, p. 3]; here, we focus on the maximum likelihood formulation. Given a set T of training data and assuming the samples are i.i.d., the likelihood is given as

$$\prod_{\boldsymbol{v}\in T} p(\boldsymbol{v}|\boldsymbol{\theta}), \tag{4.15}$$

from which we can derive the gradient of the log likelihood w.r.t. a single weight W_{ij} (see Appendix A for the derivation), averaged over the training samples:

$$\frac{\partial}{\partial W_{ij}} \frac{1}{|T|} \sum_{\boldsymbol{v}\in T} \log p(\boldsymbol{v}|\boldsymbol{\theta}) = \frac{1}{|T|} \sum_{\boldsymbol{v}\in T} \Big(v_i \langle h_j \rangle_{p(\boldsymbol{h}|\boldsymbol{v},\boldsymbol{\theta})} \Big) - \langle v_i h_j \rangle_{p(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})} \tag{4.16}$$

$$= \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \tag{4.17}$$

Here, $\langle \cdot \rangle_{\dots}$ denotes expectation with respect to the distribution defined by the probability density given in the subscript. The first term is the correlation of the unit states v_i and h_j when the visible units are clamped to the training data and the hidden units are sampled from $p(\boldsymbol{h}|\boldsymbol{v},\boldsymbol{\theta})$, and the second term is the correlation when all unit states are freely sampled from the model density $p(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})$. We can derive similar gradients for the bias terms:

$$\frac{\partial}{\partial a_i} \frac{1}{|T|} \sum_{\boldsymbol{v}\in T} \log p(\boldsymbol{v}|\boldsymbol{\theta}) = \langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}} \tag{4.18}$$

$$\frac{\partial}{\partial b_j} \frac{1}{|T|} \sum_{\boldsymbol{v}\in T} \log p(\boldsymbol{v}|\boldsymbol{\theta}) = \langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}} \tag{4.19}$$

As the likelihood function is highly nonconvex, we cannot just set the gradients to zero and solve for the parameters. Instead, we start with random parameters and perform gradient ascent according to the update rules

$$\begin{aligned} W_{ij} &\leftarrow W_{ij} + \eta_w \Big(\langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}} \Big) \\ a_i &\leftarrow a_i + \eta_a \Big(\langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{model}} \Big) \\ b_j &\leftarrow b_j + \eta_b \Big(\langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{model}} \Big), \end{aligned}$$

where η_w, η_a and η_b are the learning rates for the different parameters.

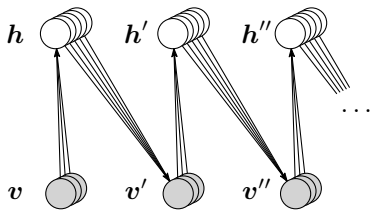


Figure 4.7: Performing blocked Gibbs sampling to sample from the model distribution: Starting from a random visible vector $\boldsymbol{v}$, all the hidden states and visible states are sampled in turn until convergence. For clarity, incoming weights are drawn for a single unit per layer only.

However, there is still a caveat. We can easily compute the first expectation, the *positive phase*, from the first term in Equation 4.16 using Equation 4.13, that is, we compute the *activation* of the hidden units at each training data vector, multiply with the respective visible states and take the average.[7] But computing the second expectation, the *negative phase*, is intractable for reasonably sized models, as it involves evaluating the energy function at all possible configurations of unit states. Instead we employ a Monte-Carlo simulation, that is, we draw a limited number of samples from the underlying distribution and average over these samples to estimate the expectation.

Obtaining an unbiased sample of the distribution defined by $p(\boldsymbol{v}, \boldsymbol{h}|\boldsymbol{\theta})$ can be achieved by a Markov chain Monte Carlo (MCMC) algorithm known as *Gibbs sampling*. It makes use of the fact that while the joint distribution of $\boldsymbol{v}$ and $\boldsymbol{h}$ is hard to compute or sample from, the *conditional* distributions defined by $p(\boldsymbol{h}|\boldsymbol{v}, \boldsymbol{\theta})$ and $p(\boldsymbol{v}|\boldsymbol{h}, \boldsymbol{\theta})$ are easy to sample according to Equations 4.13 and 4.14. Instead of sampling $\boldsymbol{v}$ and $\boldsymbol{h}$ simultaneously, Gibbs sampling starts from a random visible vector $\boldsymbol{v}$ and then samples $\boldsymbol{h}$ and $\boldsymbol{v}$ in turns. Note that due to the bipartite nature of the model, we can sample all hidden unit states in parallel, then sample all visible states in parallel (blocked Gibbs sampling). This process is depicted in Figure 4.7. Each round of updates (sampling $\boldsymbol{h}$, then sampling $\boldsymbol{v}$) constitutes a transition in a Markov chain starting at the random visible vector. When running this chain to convergence, we are sampling from the joint distribution of $\boldsymbol{v}$ and $\boldsymbol{h}$, obtaining the so-called *negative particles* needed for the negative phase.

While tractable, running a Gibbs sampler to convergence is still computationally expensive. Hinton [Hin02] proposed a much faster training method for RBMs that does not follow the maximum likelihood gradient, but approximates the gradient of another objective called *Contrastive Divergence*. Instead of starting the Gibbs

[7] In a sense, this is just a sampled expectation, because the training examples are only samples from the underlying data distribution (which is usually unknown for real-world problems). However, it is an exact expectation with respect to the observed distribution of training data.

sampler at a random visible vector, it starts at a training example – this way, the Markov chain starts closer to its equilibrium distribution (after we started training the model) [BD09]. In addition, it does not wait for the Gibbs sampler to converge, but only runs it for k steps (CD-k). In practice, even $k = 1$ works well (CD-1). Effectively, this replaces $\langle \cdot \rangle_{\text{model}}$ in Equations 4.17–4.19 by $\langle \cdot \rangle_{\text{reconstruction}}$. A related idea, *Persistent Contrastive Divergence* [Tie08], also runs the Gibbs sampler for k steps only to obtain negative particles, but instead of restarting it after each update of the model parameters $\boldsymbol{\theta}$, the sampler continues to run with these new parameters. We will explain the intuitions behind CD and PCD in Section 4.4.1.3.

In practice, the Gibbs sampler for CD and PCD uses the unit activations instead of sampling their binary states, to reduce sampling noise in the learning signal. Only the first sample of the hidden units driven by the training data must be binary, in order to create an information bottleneck for the reconstruction [Hin10]. Implementations also rely on standard machine learning practices: Instead of calculating the gradient averaged over the whole set of training data, it is calculated on mini-batches of around 100 samples (this can be seen as a compromise between online learning and batch learning), the connection weights $\boldsymbol{W}$ are regularized, and the update rules for each parameter θ_m are augmented with a momentum term

$$\theta_m^{(t)} \leftarrow \theta_m^{(t-1)} + \Delta\theta_m^{(t-1)} + \alpha\Big(\theta_m^{(t-1)} - \theta_m^{(t-2)}\Big),$$

where $\Delta\theta_m^{(t)}$ is the CD or PCD learning signal for parameter θ_m of the model $\boldsymbol{\theta}^{(t)}$. Details will be given in Chapter 5.

4.4.1.3 Interpretations

Model The RBM is a generative model for data vectors – it learns an estimate of the underlying probability density of a set of training data, and it allows to draw new samples from this distribution. Intuitively, if a model can generate data, it has captured all its structure (as opposed to a discriminatively trained model, which only captures the parts that are relevant for a given classification task).

What makes the model useful for us is its set of hidden units. The weights represent pairwise correlations between visible units and hidden units, such that an activation pattern of visible units corresponds to a likely activation pattern of hidden units and vice versa. When used to generate data, the hidden units' connection weights (columns of $\boldsymbol{W}$) form basis functions (or templates) that are weighted and summed in the log domain, i.e., the hidden units' predictions for the binary state of the visible units are effectively *multiplied* to form the overall prediction of the model (a *Product of Experts* [Hin02] as opposed to a Mixture Model). As the model has been trained to generate data from a given distribution well, as many configurations of hidden units as possible will correspond to probable

data vectors,[8] so the basis functions will capture typical configuration patterns of sets of visible units, i.e., typical features (rather than just determining the value of a single visible unit).

When used as a feature extractor, the network can be seen as a feed-forward network with logistic units, in which the input weights to each hidden unit (again, columns of $\boldsymbol{W}$) form a filter applied to the data vectors, and the filter activations are squashed by a logistic sigmoid function (binary sampling of the hidden states is not used in this case). Again, the filters will react to typical features found in the training data, so the hidden unit activations indicate the presence of such features, forming a possibly useful higher-level description of a data vector.

Training To maximize the likelihood of the model under the training data (i.e., the probability that the model generated the training data), the model parameters $\boldsymbol{\theta}$ have to be adjusted such that training examples are assigned a high probability and other data points in the input space are assigned a low probability. Most importantly, we have to lower the probability of data that is assigned a very high probability by the model despite not resembling the training data. Equivalently, to minimize the Kullback-Leibler divergence between the observed distribution of training data and the learned distribution of the model, we have to shift probability mass from the data the model "believes in" (i.e., the data it generates) to the training examples. As can be seen from Equation 4.8, this is equivalent to lowering the energy for observed data, and raising the energy for "fantasized" data.

Seen abstractly, training by maximum-likelihood achieves this in the following way: We tell the model that the training data is correct (i.e., we lower the energy for training data), then ask the model to generate some own examples and tell the model it is wrong (i.e., we raise the energy for generated data), without even looking at these examples. When the model starts generating good examples only, the two learning signals (lowered and raised energy) cancel out, and training converges.

Contrastive Divergence (CD) works a bit differently. Observing that we can generate data from the model by repeatedly sampling the hidden and visible units until convergence (Gibbs sampling, Figure 4.7), and our goal is to make the model generate data resembling the training examples, CD tries to adapt the transitions in the sampling chain such that it is already in its equilibrium distribution when starting from training data. By initializing the Gibbs sampler at a training vector and performing k sampling steps (CD-k) or even just a single step (CD-1), we can observe in which direction it tends to deviate and adapt the weights to stop it from

[8] Actually, this is not always desired, as this may lead to very unspecific features. Forcing only small sets of active hidden units to represent data vectors can be achieved by specifying a *sparsity target* during training [NH09], encouraging each hidden unit to describe a large set of visible units.

deviating. Adapting the weights is equivalent to raising the energy for the data the Gibbs sampler produced after k steps (or one step).

A potential problem is that CD only changes the energy surface close to the training data. As illustrated in Figure 4.8b, it forms ravines around the training data [RBL08, p. 2], but does not explore modes far away from the training data, possibly leaving data points assigned a high probability despite being dissimilar to all training examples. Persistent Contrastive Divergence (PCD) avoids this by not resetting the Gibbs sampler to the training data after each update, i.e., the "fantasy particles" generated by the sampler can wander across the whole energy surface, finding modes far away from the training data (Figure 4.8c). Although PCD also runs very few sampling steps per update only, the particles can quickly explore a large area of the data space: With each update, the energy surface is raised at the data points they settled for, so the next sampling step will push them away.[9] PCD can thus build a better density model than CD. However, CD ensures that the k-step or one-step reconstructions of training points are accurate, i.e., the hidden unit activations convey all information needed to reconstruct training examples. This may be more important for feature extraction than learning a good density estimation for the data [Hin10, p. 15]. In our experiments, we will thus try both variants.

4.4.2 Mean-Covariance Restricted Boltzmann Machines (mcRBMs)

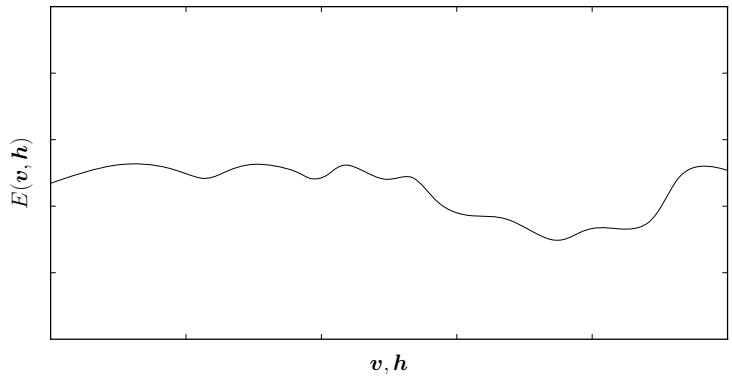
4.4.2.1 Basics

In the preceding section, we considered a basic type of RBM with binary visible and hidden states. To model real-valued data, we could map each input dimension to the range from 0 to 1 and treat the remapped values as probabilities of the visible units taking their "on" state [BLPL07, p. 4]. A better way is changing the energy function to allow the model to directly generate the kind of data it is trained on.

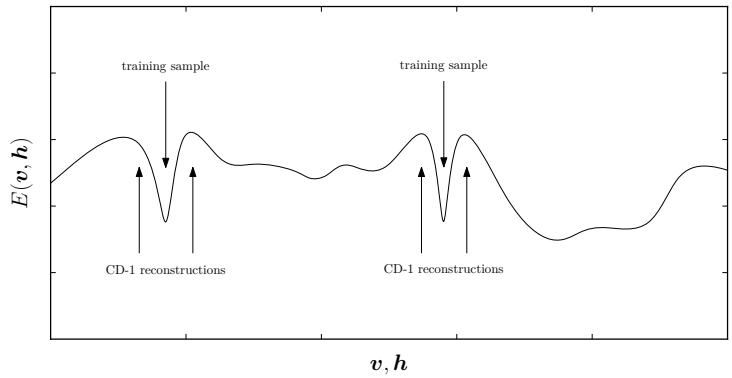
A popular variant is the Gaussian-Bernoulli RBM, which changes the energy function to replace the logistic visible units by linear units with independent Gaussian noise, while the hidden units remain binary. As it is difficult to learn the noise variance for each unit [Kri09], it is often fixed to unity [Sal09, Hin10], so $p(\boldsymbol{v}|\boldsymbol{h}, \boldsymbol{\theta})$ becomes Gaussian with identity covariance. As only the means of these Gaussians are dependent on the hidden unit states, we will refer to this model as an *mRBM* (Figure 4.9a).

While an mRBM can model real-valued inputs, independent Gaussian noise does not yield a good generative model for most real-world data. To incorporate pairwise

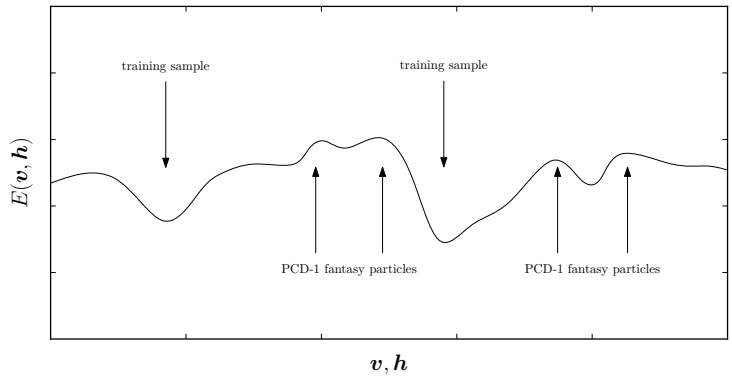
[9] In practice, PCD needs a smaller learning rate than CD so the particles can keep up with the ever-changing model and are not kicked around chaotically. Learning is not slower than for CD, though, as the negative particles are further away from the positive particles (the training examples), inducing larger differences in the statistics used for learning.



(a) Energy surface of an untrained model

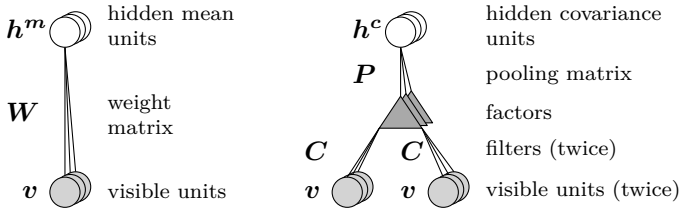


(b) Energy surface after a CD training step



(c) Energy surface after a PCD training step

Figure 4.8: Starting from an untrained model (a), Contrastive Divergence (b) lowers the energy of training data and raises the energy of reconstructions by the model, which are close to the training samples. Persistent Contrastive Divergence (c) finds and raises low-energy regions possibly far from the training data.



(a) mRBM (b) cRBM

Figure 4.9: Diagrams of the two parts of a mean-covariance RBM.

dependencies of visible units gated by hidden units, a third-order RBM can be defined, with weights $W_{i,j,k}$ connecting a hidden unit h_k to a *pair* of visible units v_i, v_j. This results in an impractically large weight tensor $\boldsymbol{W}$, but by factorizing and tying these weights [MH10, RKH10], parameters can be reduced to a filter matrix $\boldsymbol{C}$ connecting the input twice to a set of *factors* and a pooling matrix $\boldsymbol{P}$ mapping factors to hidden units (Figure 4.9b). In this model, termed *cRBM*, visible units are conditionally Gaussian given the hiddens, with a covariance matrix depending on the hidden unit states. However, it can only generate Gaussian inputs of zero mean.

For general Gaussian-distributed inputs, the mRBM and cRBM can be combined by simply adding their respective energy functions. The resulting *mcRBM* [RH10] has two types of hidden units: mean units $\boldsymbol{h^m}$, and covariance units $\boldsymbol{h^c}$. The conditional distribution of the visible units $p(\boldsymbol{v}|\boldsymbol{h^m}, \boldsymbol{h^c}, \boldsymbol{\theta})$ becomes the product of the Gaussian distributions of the mRBM and cRBM, which is a Gaussian with mean and covariance both gated by the hidden units.

Just as for binary RBMs, we will first give a more elaborate explanation of the mathematics, then analyze the model on a higher level.

4.4.2.2 Mathematics

Before considering the mcRBM as a whole, we will investigate its two parts: the mRBM and cRBM.

mRBM The Gaussian-Bernoulli RBM is guided by the energy function

$$E(\boldsymbol{v}, \boldsymbol{h^m}, \boldsymbol{\theta}) = -\sum_{i,j} \frac{v_i}{\sigma_i} W_{ij} h_j^m - \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_j h_j^m b_j,$$

where $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\sigma})$ includes a standard deviation for each visible unit's Gaussian noise. Setting all standard deviations to unity, we can drop these model pa-

rameters and obtain the simpler energy function of the *mRBM*:

$$\begin{aligned} E_m(\boldsymbol{v}, \boldsymbol{h^m}, \boldsymbol{\theta}) &= -\sum_{i,j} v_i W_{ij} h_j^m - \sum_i \frac{1}{2}(v_i - a_i)^2 - \sum_j h_j^m b_j \\ &= -\boldsymbol{v}^T \boldsymbol{W} \boldsymbol{h^m} - \frac{1}{2}(\boldsymbol{v} - \boldsymbol{a})^T (\boldsymbol{v} - \boldsymbol{a}) - \boldsymbol{b}^T \boldsymbol{h^m} \end{aligned} \tag{4.20}$$

From the joint probability density of visible and hidden states (Equation 4.8), we can derive the conditional probability density of the hidden units as in Equation 4.13, and the conditional probability density of the visibles as

$$p(\boldsymbol{v}|\boldsymbol{h^m}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{v}|\boldsymbol{a} + \boldsymbol{W}\boldsymbol{h^m}, \boldsymbol{I}), \tag{4.21}$$

where $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the multivariate Gaussian probability density function (Equation 2.2, p. 14) with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ (see [Kri09, p. 13] for the derivation). An mRBM can be trained the same way as a bRBM, sampling the visible units according to (4.21) instead of (4.14). In practice, just as for bRBMs, noise in the learning signal is reduced by using the *expected* values of the visible units (here, the means of the Gaussians) instead of sampling. The learning rate is set orders of magnitude smaller than for bRBMs [Hin10, p. 12].

cRBM As explained above, the cRBM is a *third-order* Boltzmann Machine which models pair-wise dependencies of visible units gated by hidden units. This requires an energy function which defines potentials over sets of three variables (two visible units and one hidden unit). The simplest such function,

$$E(\boldsymbol{v}, \boldsymbol{h}, \boldsymbol{\theta}) = -\sum_{i,j,k} v_i v_j h_k W_{ijk}, \tag{4.22}$$

does not scale well to high-dimensional models, with the number of connection weights being quadratic in the number of visible units. Memisevic and Hinton [MH10] propose to replace the large weight tensor by a sum of three-way outer products termed *factors*:

$$W_{ijk} = \sum_f B_{if} C_{jf} P_{kf}$$

This factorization makes use of the fact that for real-world data, the pair-wise dependencies of input components often have a regular structure which can be approximated with fewer parameters than a full W_{ijk} tensor. As $\boldsymbol{B}$ and $\boldsymbol{C}$ are both connected to the same set of visible units, Ranzato et al. [RKH10] further reduce the number of parameters by enforcing $\boldsymbol{B} = \boldsymbol{C}$, resulting in

$$W_{ijk} = \sum_f C_{if} C_{jf} P_{kf}. \tag{4.23}$$

Thus, each factor is connected twice to the visible units via a set of filters $\boldsymbol{C}$ and once to the hidden units via a pooling matrix $\boldsymbol{P}$ (see Figure 4.9b). Substituting (4.23) into (4.22) and adding hidden unit bias terms $\boldsymbol{c}$, the energy function becomes

$$\begin{aligned} E_c(\boldsymbol{v}, \boldsymbol{h^c}, \boldsymbol{\theta}) &= -\sum_f \Big(\sum_i v_i C_{if}\Big)\Big(\sum_j v_j C_{jf}\Big)\Big(\sum_k h_k^c P_{kf}\Big) - \sum_k h_k^c c_k \\ &= -(\boldsymbol{v}^T\boldsymbol{C})^2\boldsymbol{P}\boldsymbol{h^c} - \boldsymbol{c}^T\boldsymbol{h^c}, \end{aligned} \tag{4.24}$$

yielding conditionally binary hidden units with activation probabilities

$$p(h_k^c = 1|\boldsymbol{v}, \boldsymbol{\theta}) = \sigma\Big(\boldsymbol{c} + \big((\boldsymbol{v}^T\boldsymbol{C})^2\boldsymbol{P}\big)^T\Big) \tag{4.25}$$

and conditionally Gaussian visible units

$$p(\boldsymbol{v}|\boldsymbol{h^c}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{v}|0, \boldsymbol{\Sigma}), \tag{4.26}$$

where

$$\boldsymbol{\Sigma} = \Big(\boldsymbol{C}\,\mathrm{diag}(\boldsymbol{P}\boldsymbol{h^c})\boldsymbol{C}^T\Big)^{-1}. \tag{4.27}$$

In principle, training could be performed as for the bRBM and mRBM discussed above. However, we cannot do blocked Gibbs sampling of the visibles, because they are no longer independent given the hidden unit states (i.e., $p(\boldsymbol{v}|\boldsymbol{h}, \boldsymbol{\theta})$ no longer factorizes). We could sample the visibles from their Gaussian distribution given in (4.26), but this requires a large matrix inversion in (4.27), which would render the training procedure impractically slow. A more efficient way of sampling negative particles is offered by the Hybrid Monte Carlo algorithm (HMC). It simulates particles rolling on the surface of the *free energy* function

$$F(\boldsymbol{v}, \boldsymbol{\theta}) = -\log \sum_{\boldsymbol{g}} e^{-E(\boldsymbol{v}, \boldsymbol{g}, \boldsymbol{\theta})},$$

which only depends on the visible units and the model parameters, obtained by summing over (or integrating out) the corresponding hidden unit states. Note how the free energy is connected to the likelihood of the model:

$$p(\boldsymbol{v}|\boldsymbol{\theta}) = \sum_{\boldsymbol{g}} p(\boldsymbol{v}, \boldsymbol{g}|\boldsymbol{\theta}) = \frac{\sum_{\boldsymbol{g}} e^{-E(\boldsymbol{v}, \boldsymbol{g}, \boldsymbol{\theta})}}{\sum_{\boldsymbol{u}, \boldsymbol{g}} e^{-E(\boldsymbol{u}, \boldsymbol{g}, \boldsymbol{\theta})}} = \frac{e^{-F(\boldsymbol{v}, \boldsymbol{\theta})}}{\sum_{\boldsymbol{u}} e^{-F(\boldsymbol{u})}}$$

Clearly, visible states of lower free energy are more probable under the model. Instead of performing alternate blocked Gibbs sampling steps of the hiddens and visibles, initialized at the data points (Contrastive Divergence) or the previous negative samples (Persistent Contrastive Divergence), HMC starts from the same initialization and lets the samples roll towards local minima of the free energy function. Just like Gibbs sampling, this yields samples that are closer to the model distribution than the starting points, which is enough for training to work. For more details on HMC and a closed form of the free energy function for cRBMs, please see [RKH10].

mcRBM Adding the energy functions of the mRBM and cRBM yields the energy function of the mcRBM,

$$E_{mc}(\boldsymbol{v}, \boldsymbol{h^m}, \boldsymbol{h^c}, \boldsymbol{\theta}) = E_m(\boldsymbol{v}, \boldsymbol{h^m}, \boldsymbol{\theta^m}) + E_c(\boldsymbol{v}, \boldsymbol{h^c}, \boldsymbol{\theta^c}), \tag{4.28}$$

where $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{C}, \boldsymbol{P}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ are the parameters of the mcRBM, combined from the mRBM parameters $\boldsymbol{\theta^m} = (\boldsymbol{W}, \boldsymbol{a}, \boldsymbol{b})$ and the cRBM parameters $\boldsymbol{\theta^c} = (\boldsymbol{C}, \boldsymbol{P}, \boldsymbol{c})$. The conditional probability densities of the two types of hidden units – the *mean units* $\boldsymbol{h^m}$ and the *covariance units* $\boldsymbol{h^c}$ – are the same as for the separate mRBM and cRBM models, given in Equations 4.13 and 4.25, respectively. The conditional density of the visible units is the product of the two models' Gaussian distributions, which is again a Gaussian:

$$p(\boldsymbol{v}|\boldsymbol{h^m}, \boldsymbol{h^c}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{v}|\boldsymbol{\Sigma}(\boldsymbol{a} + \boldsymbol{W}\boldsymbol{h^m}), \boldsymbol{\Sigma}) \tag{4.29}$$

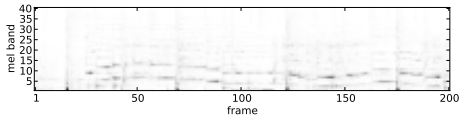
$\boldsymbol{\Sigma}$ is the covariance matrix of the cRBM defined as in Equation 4.27. Note that the mean of this Gaussian depends both on the state of the hidden mean units and covariance units.

Like cRBMs, mcRBMs can be trained efficiently using CD or PCD, generating negative particles with HMC on the free energy. In practice, there are a few subtleties to be considered: It has proven useful to make the covariance part of the model invariant to the scale of the input, by normalizing the training vectors and the filters in $\boldsymbol{C}$ to unit ℓ_2 norm. For the filters, this constraint is enforced during training, for the inputs, it is included in the energy function E_c. This normalization results in a non-Gaussian conditional density of visible units $p(\boldsymbol{v}|\boldsymbol{h^m}, \boldsymbol{h^c}, \boldsymbol{\theta})$, but does not adversely affect inference or training via HMC [RH10]. In addition, the pooling matrix $\boldsymbol{P}$ is constrained to be negative, as positive entries could result in extremely low free energies for extreme visible states [RKH10, p. 3].[10] For more robust training, $\boldsymbol{P}$ is initialized to be sparse (e.g., to the negative identity matrix if the number of factors equals the number of hidden covariance units, or to a topographical mapping), optionally enforced to remain sparse (by applying a fixed mask that resets most entries to zero after each update), only updated after the filters $\boldsymbol{C}$ had enough time to converge and constrained to columns of unit ℓ_1 norm [RH10].

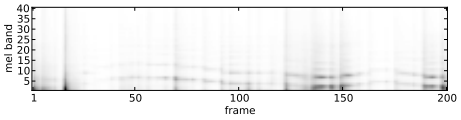
4.4.2.3 Interpretations

The mcRBM is a powerful generative model. With its binary hidden mean and covariance units, it can choose one of exponentially many mean vectors and one of exponentially many covariance matrices to explain a single data point. These

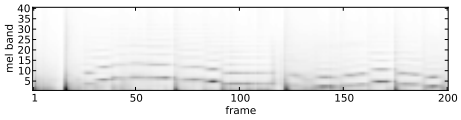
[10] With the ℓ_2 normalization of the inputs this is actually not true, but we follow [RH10] in keeping this constraint.



(a) original data



(b) mean unit reconstruction



(c) mean and covariance unit reconstruction

Figure 4.10: Reconstruction of a music excerpt with (b) mean units and (c) both mean and covariance units of an mcRBM. The covariance units smoothen the mean unit's prediction along the time axis and create harmonics, except if the covariance unit states indicate a violation of these smoothness assumptions (e.g., for the four percussion hits).

exponentially many possible mean vectors and covariance matrices are combined from the templates and filters of $\boldsymbol{W}$ and $\boldsymbol{C}$,[11] which are trained to be useful for modeling the data.

As for the bRBM, the columns of $\boldsymbol{W}$ form templates that are linearly combined to represent a data point (but without squashing the result with a sigmoid function to yield binary activation probabilities, as the mcRBM has linear real-valued instead of logistic binary visible units). The columns of $\boldsymbol{C}$, however, encode common *smoothness constraints* seen in the training data. An activation pattern of hidden covariance units describes the sets of constraints that are satisfied (active hidden units, creating negative energy) and violated (inactive hidden units, not contributing to the energy of a configuration). For natural images, for example, the default assumption could be that neighboring pixels are correlated, and a possible exception captured by a filter of $\boldsymbol{C}$ could be a vertical edge occurring in the image. The pooling matrix $\boldsymbol{P}$ maps each hidden covariance unit to a group of similar filters in $\boldsymbol{C}$, achieving a certain level of invariance against transformations of the data (for example, the model could learn to ignore small translations).

When generating data, the hidden mean units provide an approximation of a data point as a combination of templates in $\boldsymbol{W}$, and the hidden covariance units represent knowledge about pair-wise correlations of visible units (the set of satisfied smoothness constraints). This knowledge is expressed in a covariance matrix which is multiplied with the approximation provided by the mean units, smoothing

[11] This is important – if the possible mean vectors and covariance matrices would not share any information, the model would be too expressive and prone to overfitting.

the visible unit states in a way the model learned from training data. Figure 4.10 visualizes this for a music excerpt. For a trained mcRBM, we inferred the hidden unit activations for each 15-frame mel-spectral block of the excerpt, then reconstructed each block by calculating the visible units' conditional distribution given the hidden unit states, and averaged the (overlapping) block-wise reconstructions to obtain a full reconstruction of the excerpt. In Figure 4.10b we show the reconstruction of the model's mRBM part, ignoring the hidden covariance units. While the guitar and percussion are visible (cf. Figure 4.3 on p. 52), the spectrogram has a lot of gaps. By including the covariance units (Figure 4.10c), the noisy mRBM reconstruction is smoothed along the time axis, and for the guitar, harmonics are emphasized or even newly introduced. Note that this smoothing is not the same for each block, but depends on the content: For example, smoothing in time does not occur for percussion and note changes. In summary, the covariance units allow the mcRBM to generate plausible data even with inaccurate or incomplete representations from the mean units, by incorporating knowledge of pair-wise dependencies between visible units. For a similar demonstration with natural images, please see [RH10, Fig. 1].

When used as a feature extractor, the hidden mean unit states are inferred exactly the same as in a bRBM: The templates (columns of $\boldsymbol{W}$) act as filters detecting the presence of common visible unit configuration patterns, and the filter activations are squashed by a logistic sigmoid function. The filters of the factors (columns of $\boldsymbol{C}$) are treated differently: First their output is nonlinearly rectified by squaring it, then activations of similar filters are pooled by the matrix $\boldsymbol{P}$ and mapped to the logistic hidden covariance units. This strongly resembles the simple/complex cell architecture found in the primary visual cortex of mammals, and when trained on natural images, the filters indeed develop to edge detectors similar to these simple cells [RKH10] (in our experiments, we will investigate what kinds of filters emerge when training the model on music data). The fact that an mcRBM can directly detect and model important features such as occluding edges independently of the image content on either side of the edge makes it a possibly better choice for feature extraction than an mRBM.

4.4.3 Deep Belief Nets (DBNs)

All types of RBMs discussed above have binary hidden units, yielding binary latent representations of given input data. The most basic type of RBM, the bRBM described in Section 4.4.1.2, also has binary visible units, usable to model binary data. This allows us to train a bRBM on the latent representations of another RBM, yielding a three-layer model with binary hidden variables. By recursively applying this principle, we can build a stack of RBMs referred to as a *Deep Belief Net* (DBN) [HOT06, HS06, BLPL07]. Figure 4.11 visualizes the resulting model.

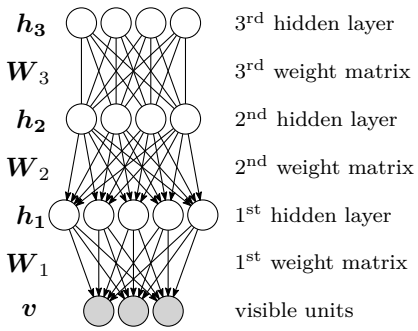


Figure 4.11: A Deep Belief Network (DBN) composed of three RBMs.

Seen as a feature extractor, a DBN built in this way is just a Multi-Layer Perceptron (MLP) with logistic hidden units. The first hidden layer already gives an abstract representation of input data presented to the visible layer. Treating all these representations for a dataset as new input data, the next-level RBM learns even more abstract representations – "features of features" – capturing higher-order correlations in the data. This process is repeated to learn more and more abstract concepts.

When regarded as a generative model, the top two layers of a DBN form a Restricted Boltzmann Machine with undirected connections, and the remaining layers are connected by directed top-down connections (with the lowest layer being the layer of visible units) [HOT06]. To see why this is useful at all, let us consider how the DBN is constructed. The first RBM is trained to approximate the observed distribution of input data by the marginal probability density over the visible units, $p(\boldsymbol{v}|\boldsymbol{\theta})$. We can also express this density (the generative model) via a prior over $\boldsymbol{h}$ and a conditional probability:

$$p(\boldsymbol{v}|\boldsymbol{\theta}) = \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{\theta}) p(\boldsymbol{v}|\boldsymbol{h}, \boldsymbol{\theta})$$

The prior over the hidden units, $p(\boldsymbol{h}|\boldsymbol{\theta})$ is implicitly defined by the model parameters of the RBM (see Equation 4.12, p. 63). We can improve the generative model by keeping $p(\boldsymbol{v}|\boldsymbol{h}, \boldsymbol{\theta})$ and replacing $p(\boldsymbol{h}|\boldsymbol{\theta})$ by a better prior: the conditional distribution of the hidden units given the training data. We can obtain samples from this distribution by simply passing the training data through the RBM (i.e., by inferring the hidden states $\boldsymbol{h}$ for training samples $\boldsymbol{v}$ according to $p(\boldsymbol{h}|\boldsymbol{v}, \boldsymbol{\theta})$) and then use a second RBM to learn a generative model of the distribution [OH08, p. 2]. As we discarded the first RBM's $p(\boldsymbol{h}|\boldsymbol{\theta})$, only keeping $p(\boldsymbol{v}|\boldsymbol{h}, \boldsymbol{\theta})$, the newly formed generative model consists of an RBM in the top two hidden layers and directed top-down connections to the visible layer. This process can be repeated to learn better and better priors of priors. It can be shown that each added layer increases

a lower bound on the likelihood of the model, provided that the number of hidden units does not decrease, the layers are initialized in a particular way and training of each RBM follows the maximum likelihood gradient [HOT06, BLPL07]. In practice, DBNs achieve good results even if all of these requirements are violated.

Once trained, a DBN can be *fine-tuned* to a specific task. In a supervised setting, rather than treating the deepest-layer activations as features and training a classifier on them, we can treat the DBN as an MLP, add an additional classification layer and train the resulting network with backpropagation. In such a setup, building the DBN from RBMs can be regarded as a greedy layer-by-layer *pre-training*, which is far superior to random initialization of a deep network [EBC+10]. For unsupervised tasks, a DBN can be fine-tuned with an up-down-up algorithm [HOT06] or stacked in reverse onto itself to form a Deep Autoencoder, the weights of which can be refined with backpropagation to minimize the reconstruction error. Alternatively, RBMs can be stacked in a slightly different way to form a Deep Boltzmann Machine (DBM) rather than a DBN, for which another unsupervised learning algorithm exists [SH09a, SL10].

In this thesis, we do not perform any fine-tuning of DBNs. Due to the lack of ground truth, we cannot easily provide a supervised learning signal for our similarity estimation tasks, and whether unsupervised fine-tuning of the representations would help in estimating music similarity has to be determined in future research.

4.4.4 Mini-batch k-Means++

A simple alternative for abstraction is clustering, which we will use in the experiments as a base line for comparison to mcRBMs. To cope with our large datasets, we use two special extensions of Lloyd's original version of the k-Means clustering algorithm: Firstly, we initialize the cluster centroids with a technique called *k-means++*, and secondly, we refine the clustering in *mini-batches*. Below we will describe both methods, preceded by a brief outline of the k-Means algorithm.

4.4.4.1 k-Means

k-Means is an algorithm for unsupervisedly clustering a dataset $X = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots \boldsymbol{x_N})$ into a given number K of clusters. Each cluster is represented by a *cluster centroid* $\boldsymbol{c_k} \in \mathbb{R}^d$, and each data point $\boldsymbol{x_n} \in \mathbb{R}^d$ is assigned to the cluster of the closest centroid (in terms of Euclidean distance):

$$\mathrm{cl}(\boldsymbol{x}) = \arg\min_k \left\| \boldsymbol{c_k} - \boldsymbol{x} \right\|_2 \tag{4.30}$$

The optimization problem to solve is to find a clustering $C = (\boldsymbol{c_1}, \boldsymbol{c_2}, \dots \boldsymbol{c_K})$ that minimizes the mean squared distance between the data points and their associated

cluster centroids:

$$J(C) = \frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{c}_{\mathrm{cl}(\boldsymbol{x_i})} - \boldsymbol{x_i} \right\|_2^2 \tag{4.31}$$

Once we have a clustering, we can replace each data point $\boldsymbol{x}$ by its cluster index $\mathrm{cl}(\boldsymbol{x})$ to obtain a more abstract, albeit very simple description of it. This is known as *Vector Quantization* (VQ). The optimal clustering $C^* = \arg\min_C J(C)$ minimizes the mean squared error of the reconstruction of a vector quantized dataset.

Finding the optimal clustering is generally hard, as the objective function $J(C)$ is nonconvex and nondifferentiable. Lloyd's k-Means algorithm [Llo82] attempts to find a good solution by starting from an initial set of centroids C, then refining the clustering by iteratively alternating between the following two steps:

E-Step: For each data point, find the closest centroid according to Equation 4.30. This establishes an association of data points to clusters:

$$\mathrm{mb}(k) = \{\boldsymbol{x} \mid \mathrm{cl}(\boldsymbol{x}) = k\} \tag{4.32}$$

M-Step: Replace each centroid with the mean of all data points associated to it:

$$\boldsymbol{c_k} \leftarrow \frac{1}{|\mathrm{mb}(k)|} \sum_{\boldsymbol{x} \in \mathrm{mb}(k)} \boldsymbol{x} \tag{4.33}$$

This *Expectation Maximization* algorithm is guaranteed to converge [BB95], but not necessarily to the global optimum. Instead, it will often get stuck in a local minimum, deterministically dependent on the initial set of centroids C.

4.4.4.2 k-Means++

As stated above, the result of Lloyd's k-Means algorithm depends on the initialization of cluster centroids. One of the simplest initialization methods is to pick the initial set of centroids C as a random sample of K data points. Arthur and Vassilvitskii [AV07] propose a more advanced seeding technique and show that their initial clustering is expected to be only $O(\log k)$ times worse than the optimal solution C^*. They call their initialization method in combination with Lloyd's algorithm *k-Means++*. Here, we will refer to the seeding technique as *k-Means++ initialization*. It proceeds as follows:

1. Pick a random data point as the first centroid $\boldsymbol{c_1}$.
2. Sample the next cluster centroid $\boldsymbol{c_k}$ from a probability distribution over the dataset defined by
$$p(\boldsymbol{x_n}) \propto D(\boldsymbol{x_n})^2, \tag{4.34}$$

where

$$D(\boldsymbol{x}) = \min_{i \in [1,k-1]} \|\boldsymbol{c_i} - \boldsymbol{x}\|_2 \tag{4.35}$$

is the distance of $\boldsymbol{x_n}$ to the closest centroid picked so far.

3. Repeat step 2 until we have sampled K centroids.

Intuitively, the sampling step favors data points that currently contribute much to the objective function (4.31) and thus will probably reduce $J(C)$ a lot when added to the set of centroids. Sampling outperforms a greedy choice of the data point maximizing (4.35), which is sensitive to outliers that are far from the centroids merely because they are far from all other data points, not because they are in a yet uncovered cluster. For the theoretical proof of $O(\log k)$-competitiveness, please see [AV07].

In our experiments, we use an extension of the k-Means++ initialization suggested in [AV07, p. 8] which improves results in practice, although the theoretical guarantees do not apply any longer: Instead of sampling a single centroid in step 2, we sample $2 + \lfloor \log k \rfloor$ candidates and greedily select the one reducing the objective function (4.31) the most.

4.4.4.3 mini-batch k-Means

Lloyd's classic batch learning for refining the initial choice of cluster centroids is slow when handling millions of data points. Sculley [Scu10] proposes an adaption of Lloyd's algorithm to work with *mini-batches* (small random subsets of samples), converging orders of magnitude faster to only slightly worse solutions. Starting from an initialization of centroids C and given a mini-batch size b, the algorithm works as follows:

1. Initialize an update count v_k for each cluster c_k to zero.
2. Randomly pick a mini-batch M of b data points from X.
3. Find and store the index of the closest centroid $\mathrm{cl}(\boldsymbol{x})$ for each data point $\boldsymbol{x} \in M$.
4. For each data point $\boldsymbol{x} \in M$ perform the following steps:
   a) Retrieve the index of the previously stored closest centroid k.
   b) Increment the update count v_k of this centroid by one.
   c) Move the centroid closer to the data point:

$$\boldsymbol{c_k} \leftarrow \Big(1 - \frac{1}{v_k}\Big)\boldsymbol{c_k} + \frac{1}{v_k}\boldsymbol{x}$$

5. Return to step 2 (unless a stopping criterion is reached, such as a maximum number of iterations).

Step 3 is important because the cluster associations might change during the following phase of updates, and we want to make sure each centroid accumulates the updates from all data points assigned to it in a chosen mini-batch. If we calculated the closest centroid on-the-fly in step 4a, the algorithm would reduce to the online stochastic gradient descent method of [BB95], which only finds low quality solutions due to stochastic noise [Scu10].

Details on the chosen hyperparameters for our experiments (the mini-batch size and number of iterations) will be given in Chapter 5.

4.5 Global Modeling

To create a global song descriptor from the sequence of local feature descriptions, we calculate a histogram of the descriptions. This discards the order of features similarly to Bag-of-Words approaches successfully used in text processing [Lew98]. Unlike the Gaussian models popular in Music Similarity Estimation (Chapter 2), histograms does not assume a Gaussian distribution of the local feature descriptions and are much faster to compare (see the next section). While a histogram may not be the best way for modeling the distribution of local feature descriptions, we purposely chose the simplest model we could think of in order to minimize handcrafting.

k-Means For k-Means, histogramming is straightforward: We just count the number of occurrences of each cluster index in a song's sequence of feature descriptions, yielding a K-dimensional histogram.

RBMs For (mc-)RBMs, we consider two options:

1. Binarizing the hidden activations by thresholding[12] and treating each combination of active units as a separate feature to count, creating a histogram over all combinations, or

2. interpreting hidden units as independent soft feature detectors and separately adding up each unit's real-valued activations over the whole song.

For a top hidden layer of H units, the first option creates a histogram of 2^H bins (one for each binary activation pattern) and is thus only applicable to models with a small top layer. The second option creates H bins (one for each unit); the resulting histogram is equal to the mean over all latent descriptions $\boldsymbol{h}$ up to a factor.

[12] Of course we could also sample the binary states according to their Bernoulli probabilities, but this would introduce unnecessary noise and make the descriptor indeterministic.

Normalization To account for songs of different lengths, all histograms are normalized to unit ℓ_1 norm before comparing them.

4.6 Song Comparison

Having created global descriptors for a set of songs, we can estimate the similarity of songs by comparing their descriptors. In our experiments, we tried a range of different distance measures and additionally employ a technique called *Distance Space Normalization* to optimize the similarity space.

4.6.1 Distance Measures

The feature histograms can be compared using any vector distance measure. Alternatively, as the histograms are normalized, they can be interpreted as discrete (multinomial) probability distributions, for which further comparison methods exist. We experiment with the cosine, euclidean (ℓ_2) and manhattan (ℓ_1) distance as well as the symmetrized Kullback-Leibler (KL) and Jensen-Shannon (JS) divergence, which we will list and briefly describe below.

4.6.1.1 Euclidean and Manhattan Distance

The Euclidean distance between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$,

$$\|\boldsymbol{x}-\boldsymbol{y}\|_2 = \sqrt{(\boldsymbol{x}-\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y})} = \sqrt{\sum_i (x_i-y_i)^2}, \tag{4.36}$$

and the Manhattan distance,

$$\|\boldsymbol{x}-\boldsymbol{y}\|_1 = \sum_i |x_i-y_i|, \tag{4.37}$$

are two special cases of the Minkowski distance (p-norm),

$$\|\boldsymbol{x}-\boldsymbol{y}\|_p = \left(\sum_i |x_i-y_i|^p\right)^{\frac{1}{p}}, \tag{4.38}$$

a standard vector distance measure.

4.6.1.2 Cosine Distance

The cosine distance of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as

$$1 - \frac{\boldsymbol{x}^T\boldsymbol{y}}{\|\boldsymbol{x}\|_2 \cdot \|\boldsymbol{y}\|_2}. \tag{4.39}$$

Geometrically, this is the cosine of the angle between the vectors, subtracted from 1, so it ranges from 0 for collinear vectors to 1 for orthogonal ones. It is invariant to vector lengths and a common choice for comparing document word-count vectors (which are equivalent to our feature histograms for k-Means features, if we consider the cluster centroids to represent "music words" and the vector quantization to resemble stemming).

4.6.1.3 Symmetrized Kullback-Leibler Divergence

The Kullback-Leibler divergence [KL51] between two probability distributions P and Q over a discrete random variable X is defined as

$$KL(P\|Q) = \sum_{\boldsymbol{x}} P(\boldsymbol{x}) \log \frac{P(\boldsymbol{x})}{Q(\boldsymbol{x})} = \sum_{\boldsymbol{x}} P(\boldsymbol{x}) \left(\log P(\boldsymbol{x}) - \log Q(\boldsymbol{x})\right). \tag{4.40}$$

It is always non-negative. One interpretation is that the divergence calculates the expected increase in code length for transmitting a sample $\boldsymbol{x} \sim P(X)$ when encoded with an optimal code for Q instead of the optimal code for P. By itself, the KL divergence is not symmetric, but it can easily be symmetrized by adding two KL divergences (repeated from Equation 2.4, p. 15):

$$KL_2(P,Q) = \frac{1}{2}KL(P\|Q) + \frac{1}{2}KL(Q\|P)$$

This measure can directly be applied to our ℓ_1 normalized feature histograms.

There are two caveats, however:

1. The KL divergence is not a metric, because it does not satisfy the triangle inequality.[13] This hampers the usage of metric-based indexing structures to speed up the retrieval of similar music pieces to a query (see Section 2.2.4.3), which is one of the most relevant uses of music similarity measures (see Section 1.1.3). However, this thesis focuses on effectivity, not efficiency, so computation speed is not an important issue.

2. The divergence (4.40) is undefined if distribution Q assigns zero probability to an event that is not impossible under P, or equivalently: if there is a histogram bin that is empty in histogram Q but non-empty in histogram P. This problem can be eliminated by adding a small constant ϵ to each histogram bin and renormalizing, at the disadvantage of introducing a new hyperparameter that may affect the results.

[13] For example, three Bernoulli distributions P, Q, R with success probabilities 0.1, 0.5, 0.8 give $KL_2(P,R) \not\leq KL_2(P,Q) + KL_2(Q,R)$.

4.6.1.4 Jensen-Shannon Divergence

The Jensen-Shannon divergence [Lin91] eliminates the problems of the KL divergence, adding some computational overhead. It is defined as

$$JS(P\|Q) = KL(P\|M) + KL(Q\|M), \tag{4.41}$$

where M is the averaged distribution

$$M = \frac{1}{2}(P + Q).$$

It is symmetric, avoids the problem of undefined KL divergences for zero probability events, and can be transformed into a proper metric satisfying the triangle inequality by taking the square root of the divergences [ES03]. Furthermore, it has an upper bound of 1 [Lin91].

4.6.2 Distance Space Normalization

For better comparability to the state-of-the-art methods of [PSS+09] and [SWP10], we optionally perform what Seyerlehner et al. termed Distance Space Normalization (DSN): After computing the full distance matrix, each entry is normalized with respect to the mean and standard deviation of its row and column. That is, given an $N \times N$ matrix of distances $\boldsymbol{D}$, we compute a normalized distance matrix $\boldsymbol{D'}$ with entries

$$d'_{ij} = \frac{d_{ij} - m(\boldsymbol{D}, i, j)}{s(\boldsymbol{D}, i, j)}, \tag{4.42}$$

where

$$m(\boldsymbol{D}, i, j) = \frac{1}{2N}\left(\sum_n D_{in} + \sum_n D_{nj}\right)$$

is the mean of row i and column j and

$$s(\boldsymbol{D}, i, j) = \sqrt{\frac{1}{2N-1}\left(\sum_n \Big(D_{in} - m(\boldsymbol{D}, i, j)\Big)^2 + \sum_n \Big(D_{nj} - m(\boldsymbol{D}, i, j)\Big)^2\right)}$$

is the respective standard deviation.

Originally developed for combining different distance measures by adding their normalized distances, it has been found that the changes in nearest neighbor ranks induced by the normalization have a positive effect on similarity estimations [PSS+09, p. 5], [SWP10, Table 1]. This effect is investigated in more detail in forthcoming work of Schnitzer et al. [SFSW11]. Here we apply it merely for comparisons to the state-of-the-art approaches of Pohle et al. and Seyerlehner et al., which both employ DSN in their methods.

5 Experimental Evaluation

How well a system performs in music similarity estimation can only be determined empirically. The first section of this chapter will show possible ways to evaluate a music similarity measure, as well as public datasets to evaluate on. Section 5.2 describes the evaluation methods and datasets we have chosen for our experiments. The first set of experiments, Section 5.3, explores the decorrelation properties of the DCT, confirming our choice of using PCA as a preprocessing step for mel spectra instead. Section 5.4 constitutes the main set of experiments: Starting from a baseline system (the "Single Gaussian of MFCCs", p. 19), we gradually change parts until we reach the system proposed in the previous chapter. With this approach, we do not only find our system to outperform the baseline, but also see how much each step contributes to the final performance. The closing section of this chapter gives a final comparison of the base line system, state-of-the-art systems and the best instantiations of different stages of our own system, summarizing our findings.

5.1 Evaluating Music Similarity Measures

Under a good music similarity measure, pairs of songs assigned a low distance should be perceptually similar. Such pairs can be found by considering the k nearest neighbors to a query song in a sufficiently large collection. Ideally, we would obtain a lot of low-distance pairs using our music similarity estimation system and ask human test subjects to judge their perceptual similarity. This would give an impression of how well our system performs, and repeating this for a range of different parameterizations would allow us to find the optimal instantiation of our system. Further experiments with other approaches would finally enable a comparison to the state-of-the-art.

However, such extensive listening tests would exceed our resources. Even the largest-scale comparative listening test in the MIR community, the MIREX competition,[1] only evaluates 10–15 systems per year, which is still below the number of variations of our system considered here. Furthermore, listening tests are not freely repeatable, which would make it impossible for other authors to compare our results to theirs.

[1] `http://www.music-ir.org/mirex/`

As an alternative, several proxies to perceived similarity have been proposed, allowing an easy and objective evaluation of different methods. Some authors have also published datasets of songs along with such "proxy" ground truth to enable the comparison of systems by their evaluation results on common test sets.

5.1.1 The Quest for Ground Truth

In order to be able to evaluate music similarity measures without conducting listening tests, several sources of similarity ground truth have been explored.

Many are based on partitioning a collection into sections, establishing a binary similarity relation: Songs of the same section are assumed to be similar, songs of different sections are not. Two objective ways to do so are forming groups of songs by the album they first appeared in, or by the artist who interpreted them [Foo97, LS01, BP03, WL07, SWW08]. The partitioning can also be defined manually, requiring somebody to cluster the collection by a subjective criterion. For example, Aucouturier et al. [AF04] divide a test collection of 350 songs into groups of similar timbre. Another popular choice is to partition a collection by genre [LS01, AP02b, FPW05, PFW05, LS06, PKSW06, SWK08, HBC08, SPWS09, PSS+09, SWP10, MLG+11]. Although genre attributions are not objective, as there is not even consensus about the taxonomy [PC00], music distributors commonly use musical genres to organize their collections, so genre annotations are easy to obtain. Once a partitioning has been defined, music similarity measures can be easily evaluated: For multiple query songs picked from the collection (often simply for all possible queries), the k nearest neighbors in the collection are obtained with the system under evaluation, representing the songs the measure defines to be most similar to the query. A score for the system is computed as (a) the fraction of songs among the neighbors that are in the same section as the query song, averaged over all queries, or (b) the number of times that the majority of neighbors are in the same section as the query song, divided by the number of queries. Method (a) can be interpreted as a measure from information retrieval: Regarding songs in the same section as the query as relevant to the query, others as nonrelevant, it computes the precision of the system for the k top-ranked results. Method (b) is equivalent to regarding each section as a class, and computing the system's accuracy in classifying the queries with a k-nearest neighbor classifier. We refer to these evaluation metrics as the "precision at k" and "k-NN classification accuracy", respectively (cf. Section 5.2).

Instead of forming a strict partitioning of the collection, some sources of information naturally lead to possibly overlapping sets of similar songs. The fraction of sets in which two songs co-occur can then be interpreted as their similarity, and computing this for all pairs results in a similarity matrix for the collection. [SWW08] crawled music blogs and treated songs posted in the same blog as a set of similar

songs. However, they did not account for overlapping sets (or there simply were none), resulting in a binary "same blog" similarity matrix used exactly like the album, artist or genre partitionings explained in the previous paragraph. [BLEW03] applied the technique to user-generated playlists mined from a hobbyist web site[2] and to user collections crawled from a peer-to-peer file sharing network, treating each playlist or each collection as a set of similar songs. However, as they were only interested in artist similarity, song co-occurrences were mapped down to artist co-occurrences. To eliminate the effects of popularity bias, the authors normalized the similarity matrix by dividing each fraction of co-occurrences of two artists by the fractions of occurrences of either artist alone. They evaluate a system against this ground truth matrix by computing an agreement score between the lists of k nearest neighbors produced by both, averaged over multiple queries, weighting agreement between the top ranks higher than between low ranks.

[MEDL09] use radio station playlists as a source of ground truth. In contrast to [BLEW03], they do not treat playlists as unordered collections, but consider playlist transitions as an indication of similarity. For each query song, they construct a list of all songs that directly followed it in any playlist, ordered by the occurrence count of each such transition among the playlists. The top k songs of this list are compared with the k nearest neighbors given by a music similarity measure by simply computing the number of songs occurring in both lists divided by k. Averaging this over all possible queries results in a total score for the measure.

[GKS07] and [ELBMG08] propose to use social tags or user listening habits of last.fm (see p. 2) to infer similarity ground truth. Although they only deal with artist similarities and are mainly interested in evaluating the performance of an auto-tagger, it might be possible to leverage the data of last.fm also as song-level ground truth for evaluating music similarity measures.

Several authors have collected song-level descriptions in online "games with a purpose". In these games, groups of users are presented with music clips to tag with words or to rate by some criterion, and score points for blindly agreeing with each other [ME07, KSE08, BOTL09], or for correctly asserting whether they have been presented the same song [LvA09]. This results in reliable descriptions confirmed by multiple human judgements which could be used to construct similarity ground truth.

One of the games, "Tag-a-tune" [LvA09], also has a bonus round which directly asks users for similarity judgements: presented with triplets of songs, users have to select the clip which is most dissimilar to the other two. A music similarity measure could be evaluated by assessing how well it reproduces the users' choice for each triplet, possibly weighted by how strongly the human voters agreed. However, only few such judgements have been collected (see Section 5.1.2).

[2] `http://www.artofthemix.org`

5.1.2 Available Datasets

Regardless of the type, ground truth is only useful for evaluating a measure when paired with audio data the measure can be computed on. Unfortunately, most songs (at least those which are popular enough to find reliable ground truth for) are under copyright, making it difficult to share datasets between researchers. Some authors of datasets circumvent copyright infringement by only sharing audio features (such as MFCCs) rather than the original audio data, but this prevents evaluating systems such as ours that are designed to extract their own features. The best way to obtain a large dataset might be to create one ourselves, but this would exceed our resources and make it difficult to compare our system to others. We will thus restrict ourselves to using existing datasets.

In the following, we review a number of public datasets, in order to form a well-founded decision on the evaluation method and data to use for our experiments.

RWC: Contains 215 tracks of 4 categories (Popular, Royalty-free, Classical, Jazz). Full-length audio can be ordered on CDs, as well as MIDI scores and lyrics. To avoid any copyright issues, all tracks have been recorded or written exclusively for this dataset. [GHNO02], `http://staff.aist.go.jp/m.goto/RWC-MDB/`

GTZAN: Contains 1,000 tracks of 10 genres (100 per genre). Includes 30-second audio excerpts of different recording quality, and no artist information. [TC02], `http://marsyas.info/download/data_sets`

USPOP2002: Contains 8,752 tracks with artist, album and title metadata. Audio data is not provided, but MFCCs can be ordered on DVDs. Includes a lot of artist-level ground truth: genre labels, "style" tags and expert similarity judgements from AllMusic.com, playlist and collection co-occurrence counts crawled as described in Section 5.1.1, and artist similarity judgements obtained in an online survey. [BLEW03], `http://labrosa.ee.columbia.edu/projects/musicsim/uspop2002.html`

ISMIR2004: Contains 1,458 tracks of 6 genres, with partially available artist information. Full-length audio files are provided. The genre distribution is highly imbalanced, with the largest genre (classical) spanning almost half of the collection. `http://ismir2004.ismir.net/genre_contest/index.htm`, `http://mtg.upf.es/ismir2004/contest/`

Ballroom: Contains 698 tracks of 8 different ballroom dances (Cha-cha, Jive, Quickstep, Rumba, Samba, Tango, Viennese Waltz, Waltz). Includes 30-second audio snippets of different quality, and no artist information. [GDPW04], `http://mtg.upf.es/ismir2004/contest/rhythmContest/`

Homburg: Contains 1,886 tracks of 9 genres, with artist information. Only a random 10-seconds excerpt of each song is included. Additionally provides 24 ways of how users would organize these songs in a personal collection (*user taxonomies*). [HMM+05], `http://www-ai.cs.uni-dortmund.de/audio.html`

Coidach: Contains 26,420 tracks of 55 genres, with carefully verified artist, album and title information. Audio data is not freely available, but custom audio features are supposed to be calculable on demand by the dataset maintainers. [MMF06], `http://jmir.sourceforge.net/Codaich.html`

CAL500: Contains 500 tracks of popular western music, labeled by paid students with 135 tags of 6 categories (genre, emotion, instruments, vocal style, song characteristics, usage). [TBTL07], `http://cosmal.ucsd.edu/cal/projects/AnnRet/`

Magnatagatune: Contains 5,306 tracks from 230 artists in 446 albums. Each track is split into 30-second clips, resulting in 25,863 clips in total. Includes full audio content and labels for each clip out of a vocabulary of 190 tags. Additionally provides dissimilarity votes for 1019 different clips in 533 distinct comparison triples (as described in Section 5.1.1). [LvA09], `http://tagatune.org/Magnatagatune.html`

1517-Artists: Contains 3,180 tracks of 19 genres. Includes full-length audio data and artist information. In addition, provides precalculated similarity matrices for several similarity measures. [SWK08], `http://www.seyerlehner.info`

Unique: Contains 3,115 tracks of 14 genres. Includes 30-second audio excerpts, and no two songs are by the same artist. Again, a number of precalculated similarity matrices are provided. [SWP10], `http://www.seyerlehner.info`

Million Song: Contains 1,000,000 tracks with artist, album and title metadata. Audio data is not provided, but "MFCC-like features" are included and audio samples for many tracks can be downloaded from 7digital.com. [BMEWL11], `http://labrosa.ee.columbia.edu/millionsong/`

5.2 Chosen Evaluation Approaches

Our choice of evaluation methods is limited by available data. While we could, for instance, crawl radio station playlists from the web, hardly any of the corresponding audio data will be freely available. Furthermore, as we would like to compare our system to state-of-the-art approaches, we will have to restrict ourselves to datasets and evaluation methods for which state-of-the-art results have been published – the

alternative of reimplementing another author's system for the purpose of evaluating it is out of scope for this thesis.

The best-performing systems to date seem to be Pohle's RTBOF [Poh10, PSS+09] and Seyerlehner's BLS [Sey10, SWP10], as well as their combination CMB[3] (see Section 2.3.4). These systems have been evaluated on several of the genre-labeled datasets described above, either in terms of k-NN genre classification accuracy or precision at k, and in addition to respective results across different choices of k [SWP10][Sey10, pp. 141–142 and p. 147], full similarity matrices are available for Homburg, Ballroom, GTZAN, ISMIR2004, 1517-Artists and Unique online at `http://www.seyerlehner.info`.

From these selection of datasets, we decide to use 1517-Artists, as it is one of the largest and features a uniform genre distribution; Homburg, because it is older and would allow a comparison to earlier published results; and Ballroom, since it is widely used to evaluate rhythm features and enables us to assess if and how well our system captures rhythmic aspects of songs. On GTZAN and ISMIR2004, both state-of-the-art systems and the basic "Single Gaussian of MFCCs" already reach k-NN genre classification accuracies between 80 and 90%, not leaving much room for improvement, and the lack of artist information prevents usage of an artist filter (see below). Unique might be interesting as an additional confirmation of results on 1517-Artists and Homburg, but would probably not add much to our findings. In addition, we also performed precursory experiments using the similarity judgements provided with Magnatagatune. However, data proved to be too scarce to produce any conclusive results, so we restrict ourselves to evaluating the precision at k and k-NN genre classification accuracy on 1517-Artists, Homburg and Ballroom.

While this seems to be a very simplistic way of evaluating a music similarity measure, crudely approximating similarities by a binary "same-genre" relation, genre precision at k has actually been shown to highly correlate with human judgements in listening tests [Poh10, p. 28 and 26][4] [Sey10, p. 52 and 57][5], and both precision at k and the related k-NN genre classification accuracy are established methods for evaluating music similarity measures [LS01, AP02b, FPW05, PFW05, LS06, PKSW06, SWK08, HBC08, SPWS09, PSS+09, SWP10, MLG+11]. Note that we are not actually interested in achieving a high precision or accuracy on a particular dataset. Unlike in classification, we use genre labels merely as binary similarity ground truth and do not train on them, as this would tie the measure to a specific genre taxonomy and dataset.

Some methods (especially those based on cepstral features) have been shown to pick up on peculiarities in recording or mastering processes, leading to the artist and

[3] We refer to CMB in [SWP10], which is CMB3 in [Sey10].
[4] Pohle refers to it as "(probabilistic) genre classification accuracy".
[5] Seyerlehner terms it "k-Neighbor Accuracy".

album effect [Sca09, FS10]: Songs of the same album or same artist are assigned a low distance independently of their perceptual similarity. As all songs of an artist tend to be attributed to the same genre, such a method would achieve overly optimistic results in k-NN genre classification [Fle07]. To ensure that we only evaluate the capability of finding similar songs, not of identifying songs of the same artist or album, we use an *artist filter* throughout our evaluation: when computing the nearest neighbors to a query under a similarity measure, songs by the same artist as the query are ignored.

Formally, the evaluation proceeds as follows, for all three datasets. Let $S = \{s_1, s_2, \dots s_N\}$ denote the set of songs of a dataset, let $G = \{g_1, g_2, \dots g_M\}$ denote the genres, and let $c : S \to G$ be a function mapping each song to its genre. Furthermore, let $A = \{a_1, a_2, \dots a_K\}$ denote the artists, and let $b : S \to A$ map each song to its artist (if no artist information is available, let $A := S$ and $b(s) := s$, so each song is mapped to a unique pseudo-artist). Following Knuth [Knu92], we also introduce the following notation:

$$[\mathcal{A}] = \begin{cases} 1 & \text{if } \mathcal{A} \text{ is true} \\ 0 & \text{if } \mathcal{A} \text{ is false} \end{cases}$$

Now, let $m : S^2 \to \mathbb{R}$ be a dissimilarity measure to evaluate, and let $n_m : S \times \mathbb{N} \to 2^S$ be defined as the set of artist-filtered nearest neighbors to a song with respect to the measure m:

$$n_m(s, 0) := \emptyset$$
$$n_m(s, k) := n_m(s, k-1) \cup \Big\{ \arg\min_{s' \in (S \setminus s)} \Big(m(s, s') + \infty \cdot [s' \in n_m(s, k-1) \vee b(s') = b(s)] \Big) \Big\},$$

with ties in arg min broken arbitrarily, but deterministically. As a shorthand notation, let $x_m(s, k, g)$ denote the number of (artist-filtered) k nearest neighbors of song s under measure m that are attributed to the genre g:

$$x_m(s, k, g) := \sum_{s' \in n_m(s,k)} [c(s') = g]$$

We then define the precision at k for a measure m as

$$\frac{1}{|S|} \sum_{s \in S} \frac{x_m(s, k, c(s))}{k}, \tag{5.1}$$

and the (leave-one-out) k-NN genre classification accuracy as

$$\frac{1}{|S|} \sum_{s \in S} \Big[c(s) = \arg\max_g \Big(x_m(s, k, g) \Big) \Big], \tag{5.2}$$

with ties in arg max also broken arbitrarily, but deterministically.

The precision at k gives the average fraction of similar (in the sense of "same-genre") neighbors, the k-NN classification accuracy evaluates how often the most prominent genre is the "correct" one. We will concentrate on the first evaluation metric as it does not only rely on the most prominent genre in a set of neighbors (which may be close to a tie more often than not for 8–19 genres among 1–20 neighbors, resulting in noisy scores), but determines the number of songs of the genre we are currently interested in (i.e., the genre of the query). The classification accuracies will only serve as a complementary measure to validate the precision results (e.g., a precision of 40% could still result in 0% classification accuracy if the remaining 60% of neighbors are always from a single "wrong" genre, indicating some systematic error).

To gain insight into the kind of mistakes made by the systems, we will also visualize the genre confusion matrices for some measures, showing which kinds of query songs tend to be associated with songs of which foreign genres (see Section 5.5.2).

Finally we will compare the *hubness* between systems. Loosely speaking, the hubness can be defined as the amount of hubs produces by each system, which are songs that are irrelevantly close to a large number of other songs (see Section 2.2.4.3, p. 18). As a simplification, commonly the hubness is determined by just counting the number of songs that occur among the k nearest neighbors of abnormally many songs in a collection at all, boldly assuming that most of these occurrences will be perceptually irrelevant, or acknowledging the fact that even relevant songs are detrimental to a recommender or playlist generator if they occur too often. We will describe the exact way of computing the hubness in Section 5.5.3.

Having established the datasets and evaluation methods to use, we can now proceed to the experiments.

5.3 Results on Decorrelating Musical Spectra

In this section's short set of experiments we will question the common assumption that the discrete cosine transform[6], the final step in computing Mel-Frequency Cepstral Coefficients (MFCCs), is a good choice for decorrelating musical spectra. As we already noted in Section 2.2.1, MFCCs are a prevalent feature for timbre-based music similarity estimation despite having been designed for processing speech, and some authors cite [Log00] as a justification for applying MFCCs to music [Bur03]. However, in this paper, Logan does not evaluate MFCCs on a music similarity estimation task at all. She only observes that (a) using mel-scaled spectra instead of linear spectra does not hurt performance in a music/speech discrimination test, and (b) the principal components of mel-spectral log magnitude frames extracted

[6] Throughout this thesis, of the eight possible types of discrete cosine transforms [BYR06], we refer to DCT-II.

from music look similar to the basis functions of the discrete cosine transform (and judging from Logan's plots, even the latter is questionable for all but the first 6 components, which is far below the number of cepstral coefficients normally used).

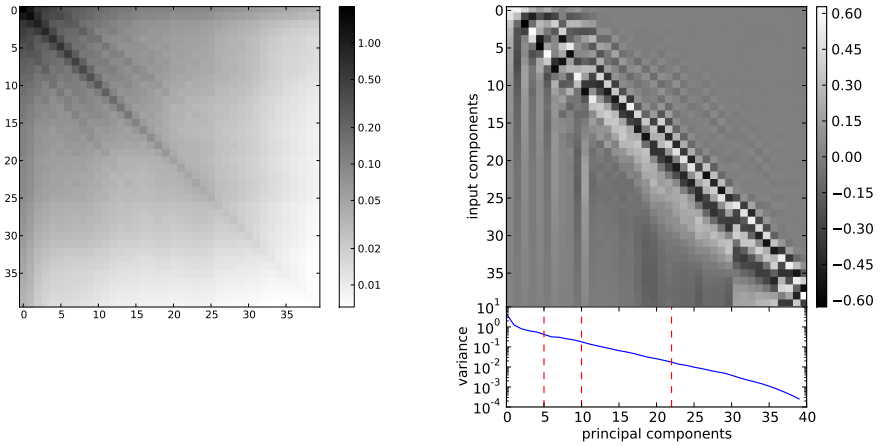
To better understand the similarities and differences between employing the DCT and optimally decorrelating the features using PCA, we will first perform a visual comparison not only of the two sets of basis functions, but also of the covariance matrices yielding these functions as their principal components. Secondly, we will evaluate if and how the differences affect a common music similarity measure. The results corroborate our choice of PCA for decorrelation.

5.3.1 Visual Comparison

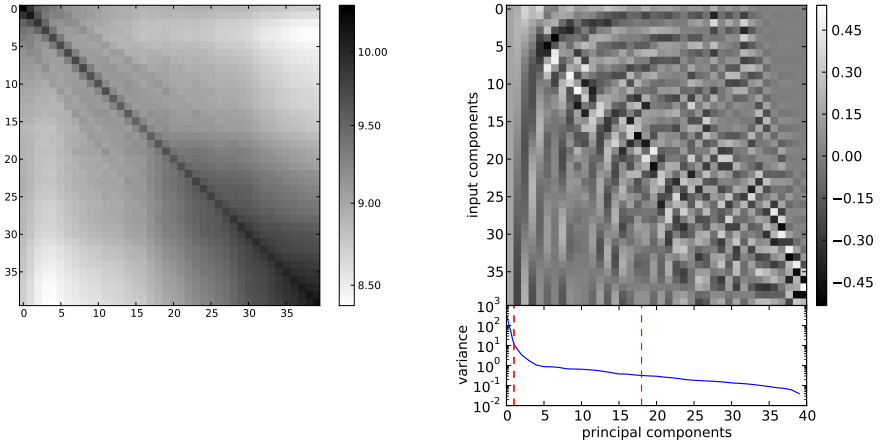
Logan [Log00] visually compared the principal components of spectral frames for 100 Beatles songs to the DCT basis functions. We will do the same for the larger dataset 1517-Artists, and additionally compare the observed covariance matrix of the spectral frames to a Toeplitz matrix. The principal components of Toeplitz matrices of the form $A_{ij} = \rho^{|i-j|}$ have been shown to approximate the DCT basis functions as the matrix size tends to infinity [HP76, SGP+95, BYR06], but are already quite close even for an 8x8 matrix [ANR74]. Comparing the covariance matrices will allow us to see better in which respect musical spectra differ from the kind of data that could be decorrelated with a DCT.

The first two rows of Figure 5.1 show the covariances (left column) and principal components (right column) of a million mel-spectral frames randomly sampled from the 1517-Artists dataset. For the first row (Figure 5.1a), we used a linear scaling of mel magnitudes, i.e., the direct output of the mel filter bank (cf. Figure 4.3c, p. 52), the second row depicts log-scaled magnitudes, i.e., the representation MFCCs are computed from by applying a frame-wise DCT (cf. Figure 4.3d, p. 52). The third row of the Figure, 5.1c, shows the Toeplitz matrix $A_{ij} = 0.9^{|i-j|}$ and its principal components.

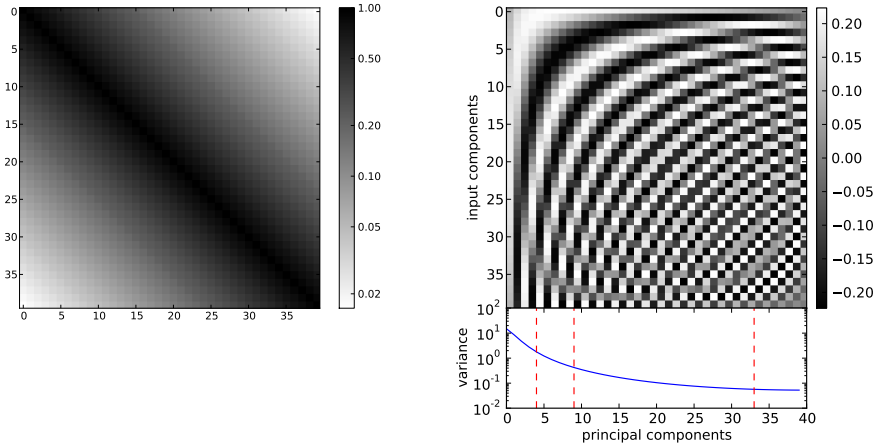
Interpreting the Toeplitz matrix as a covariance matrix, it describes spectra in which all frequency bins are correlated with their neighboring bins, with exponential decay in proportion to their distance in mel space. Its princical components, the columns of the image in the right panel of Figure 5.1c, almost perfectly resemble the basis functions of the 40-dimensional DCT (only the first column is a bit off: in contrast to the first DCT basis function, which is constant, it has a slight oscillation in it). The plot below the image of principal components shows the variance captured by each component. We can clearly see that this variance decreases with increasing spatial frequency of the component (number of oscillations in the column of the image) – this is the common assumption made when compressing data with a DCT by omitting the highest components (as is done for MFCCs). Thus, for spatially correlated data of the form described above, a DCT is suited well for both



(a) Covariance matrix and principal components of mel-spectral frames



(b) Covariance matrix and principal components of log mel-spectral frames



(c) Toeplitz matrix and its principal components

Figure 5.1: Covariance matrices and principal components for mel-spectral frames with linear and log magnitudes in comparison to a Toeplitz matrix and its principal components. The plots below the principal components show the variance of the data projected on them, the dashed vertical lines denote the cut-off points for 75%, 90% and 99% of variance.

decorrelation and lossy compression.

However, mel-spectral frames of music do not seem to be of this form. While each dimension is correlated with its two direct neighbors (see the left panels of Figures 5.1a and 5.1b) – which is evident as each mel filter overlaps with two neighbors (Figure 4.2, p. 51) –, generally no regularly decaying correlation with farther neighbors can be observed. Instead, there are some faint diagonals visible in both covariance matrices, indicating another kind of regular correlation. A tempting idea is to interpret these as harmonic overtones, but as the mel scale is logarithmic in frequency, overtones would appear in parallel to the main diagonal; the observed diagonals seem to point to a common origin of 0 Hz instead (which is not part of these covariance matrices as the mel filters used here start at 130 Hz, see p. 50).

Another deviation from the form of a Toeplitz matrix is the dependency on frequency. For the linear-magnitude mel-spectral frames (Figure 5.1a), low frequencies (the top rows and left columns) weakly covary with all other frequencies, while higher frequencies do not. This is partly due to the low variance of higher frequencies (values on the main diagonal decrease when moving to the bottom right), which seem to carry less energy in music. The lack of covariance causes the principal components to be spatially local, in strong contrast to DCT basis functions (compare the right panels of Figures 5.1a and 5.1c).

With log-scaled magnitudes, the difference in variance between low and high frequencies is reduced. Still, the covariances are strongly dependent on the frequency: The highest frequencies now covary with each other (visible as a dark field in the bottom right corner of the left panel of Figure 5.1b), while low frequencies still mostly correlate with their direct neighbors only. Possibly, the highest frequencies are only reached with percussion, which has a broad frequency spectrum and thus activates most high frequency mel filters simultaneously, or the log-scaling emphasizes high-frequency low-amplitude noise in songs. The corresponding principal components (right panel of Figure 5.1b) are less spatially local, and show some signs of increasing spatial frequency with decreasing variance, similar to the principal components of the Toeplitz matrix. Another interesting effect of log-scaling the magnitudes is the seemingly better compaction of energy in the first components: While for linear-scaled magnitudes, 5, 10 and 22 components were necessary to capture 75%, 90% and 99% of the variance, respectively, the first principal component of log-magnitude mel spectra alone captures 90%, and 18 components suffice to capture 99% (shown as vertical bars in the plots of the right panels of Figures 5.1a and 5.1b). This property of energy compaction is important for lossy compression by dropping high components, and frequently attributed to the DCT as well.[7]

[7] Intuitively, the first principal component of log-magnitude spectra is of course not as informative as the first 10 components of linear-magnitude spectra, but it explains the same amount of variance in the respective data. This shows that variance is not always a good indicator of informativeness, so energy compaction does not necessarily result in useful compression.

In conclusion, the principal components of mel-spectral frames are very different to the basis functions of a DCT (right panels of Figure 5.1). This is because mel-spectral frames differ from the kind of spatial correlation that could be optimally decorrelated with a DCT in several respects: (a) Covariances mainly occur between directly adjacent mel filters, energy is not smoothly distributed over the spectrum of each frame, (b) covariances are dependent on the frequency, (c) albeit weak, there are systematic interactions between frequencies which are not close neighbors (visible as diagonal artifacts in covariance matrices). While the principal components of log-magnitude mel spectra are closer to DCT basis functions than the ones of linear-magnitude spectra, they still lack the DCT's spatial globality. In contrast to Logan [Log00], we are skeptical about whether a DCT is a good approximation of the PCA for musical spectra.

5.3.2 Effects on Music Similarity Estimation

While the previous section's visual inspection established that the PCA of mel-spectral frames is noticeably different from the DCT used in the computation of MFCCs, we are only interested in whether this difference has an effect on music similarity estimation. We will thus evaluate a simple music similarity measure, the "Single Gaussian of MFCCs" proposed by Mandel and Ellis [ME05] (see Section 2.3.1, p. 19), on 1517-Artists and experiment with variations of the low-level features (i.e., replacing the DCT with an optimally decorrelating transformation) and of the global model (i.e., restricting the model's ability to handle correlations in the features).

Recapitulating Mandel and Ellis' method, the MFCC feature extraction chain computes the mel spectrum, log-scales the magnitudes, applies a frame-wise DCT and keeps only the first components. A song is then modeled by fitting a single full-covariance Gaussian to its set of MFCCs, and songs are compared by the symmetric Kullback-Leibler divergence (cf. [LSMW05]):

$$KL_2\Big(\mathcal{N}(\boldsymbol{\mu_0}, \boldsymbol{\Sigma_0}) \| \mathcal{N}(\boldsymbol{\mu_1}, \boldsymbol{\Sigma_1})\Big) = \frac{1}{4}\Big(\operatorname{trace}(\boldsymbol{\Sigma_1}^{-1}\boldsymbol{\Sigma_0} + \boldsymbol{\Sigma_0}^{-1}\boldsymbol{\Sigma_1}) + (\boldsymbol{\mu_1} - \boldsymbol{\mu_0})^T(\boldsymbol{\Sigma_0}^{-1} + \boldsymbol{\Sigma_1}^{-1})(\boldsymbol{\mu_1} - \boldsymbol{\mu_0}) - 2N\Big) \quad (5.3)$$

Table 5.1 lists the genre precision at k (with k ranging from 1 to 20) for several variations of the local features in this method, with the last row showing the original MFCCs. Specifically, we change the MFCC feature extraction chain by transforming the magnitudes with PCA or ZCA whitening instead of the DCT. As we want to validate the choice of low-level representation for our own proposed system, we also assess the effect of omitting the log scaling of magnitudes. We make three interesting observations:

local feature variations			precision at			
magn. scaling	transformation	dim.	1	5	10	20
linear	none	40	18.0	15.1	14.1	12.9
	PCA	40	18.0	15.1	14.1	12.9
		20	17.0	15.0	13.9	12.8
log	none	40	**21.1**	**17.9**	**16.5**	15.1
	PCA	40	**21.1**	**17.9**	**16.5**	15.1
		20	20.6	17.8	16.4	**15.2**
	ZCA	40	**21.1**	**17.9**	**16.5**	15.1
	random	40	**21.1**	**17.9**	**16.5**	15.1
	DCT	20	20.0	17.5	16.1	14.9

Table 5.1: Precision at k on 1517-Artists for several variations of mel-spectral features (two scalings of magnitudes, different linear transformations with or without dimensionality reduction) modeled by a full-covariance Gaussian and compared with symmetric KL divergence.

1. Log-scaled magnitudes consistently outperform linear magnitudes.

2. When modeling songs with full-covariance Gaussians and comparing those with the symmetric KL divergence, the similarity measure is invariant to invertible linear transformations of the local features. That is, PCA, ZCA and DCT perform exactly like unprocessed features if no dimension reduction is employed. Even a random transformation does not change the results, provided that it is invertible.

3. When performing dimensionality reduction from 40 to 20 dimensions (a typical setting for MFCCs [SWK09]), PCA is slightly superior to the DCT (see sixth row and last row of Table 5.1). However, uncompressed mel spectra perform better.

We repeat the experiment, this time restricting the Gaussian models to diagonal covariance (Table 5.2). Song models are still compared by symmetric KL divergence. With this change, the model cannot capture correlations between local feature dimensions any longer, so linear transformations of the features change the results. Specifically, we observe the following:

1. The results are considerably worse than for full-covariance Gaussians – but here, we are only interested in comparing the feature variations to one another.

2. Log-scaled magnitudes still outperform linear magnitudes, except for raw (untransformed) features.

local feature variations			precision at			
magn. scaling	transformation	dim.	1	5	10	20
linear	none	40	15.2	13.7	13.0	12.3
	PCA	40	14.9	12.4	11.6	10.8
		20	14.0	13.0	12.0	11.2
log	none	40	13.1	11.0	10.4	9.7
	PCA	40	18.5	15.9	**15.0**	**13.9**
		20	17.8	15.6	14.5	13.5
	ZCA	40	**18.9**	**16.2**	14.8	13.5
	random	40	9.8	8.0	7.9	7.8
	DCT	20	17.6	15.4	14.4	13.4

Table 5.2: Repetition of the previous experiment (Table 5.1), restricting Gaussians to diagonal covariance, still comparing with symmetric KL divergence.

3. PCA and ZCA whitening of log-magnitude frames perform best; for linear magnitudes, the PCA transformation hurts performance, though. As to be expected, the random transformation gives very poor performance (but still above the random baseline of $1/19 \approx 5\%$).

4. Again, PCA-compressed frames slightly outperform MFCCs (last row), but uncompressed whitened frames perform better.

Further simplifying the model, we restrict the Gaussians to be isotropic with unit variance (i.e., the covariance matrix is the identity matrix), compared with symmetric KL divergence. With this restriction, the model just captures the mean of a song's frames, and the KL divergence degenerates to a quarter of the squared Euclidean distance between the mean vectors. In our nearest-neighbor-based evaluation, this is equivalent to directly comparing the means by Euclidean distance (squaring does not change the neighbor ranks). While this is a blatantly simple model, k-Means builds on the same operations (calculating the mean and comparing by Euclidean distance), so results with different feature variations could be interesting for later experiments with models based on k-Means. From Table 5.3, we see:

1. The results are again much worse than for less restricted Gaussian models.

2. Again, log-scaled magnitudes are superior to linear-magnitude spectra.

3. PCA- and ZCA-whitened log-magnitude frames perform best (with identical results[8]).

[8] This affirms what we already learned in Section 4.3.1.3, p. 59: ZCA is a non-scaling rotation of

local feature variations			precision at			
magn. scaling	transformation	dim.	1	5	10	20
linear	none	40	12.0	10.6	10.1	9.8
	PCA	40	14.3	12.7	12.1	11.4
		20	12.5	11.8	11.3	10.9
log	none	40	13.1	12.6	11.8	10.9
	PCA	40	**15.7**	**13.9**	**12.8**	**11.6**
		20	13.4	12.8	12.2	11.3
	ZCA	40	**15.7**	**13.9**	**12.8**	**11.6**
	random	40	10.2	9.5	9.2	8.6
	DCT	20	13.6	12.2	11.6	10.8

Table 5.3: Repetition of the previous experiments, modeling songs just by the mean of their features, compared with Euclidean distance.

4. With dimensionality reduction, PCA again outperforms MFCCs, with a larger margin than before.[9] Again, uncompressed features perform better.

Summarizing the results, we draw a range of conclusions regarding the applicability of MFCCs to music similarity estimation.

Reducing the dimensionality of mel-spectral frames hurts performance for all three models, so it seems that some relevant information is lost when dropping higher cepstral coefficients or principal components. This is contrary to Aucouturier et al. [AF04], who found dimensionality reduction to be helpful – however, they evaluated via R-Precision on an (unpublished) small timbre-clustered dataset, used diagonal-covariance GMMs for global modeling and probably another (unspecified) mel filter bank, and each aspect alone could explain the difference in our findings. We will have to perform further experiments with different models to understand the influence of dimensionality reduction.

When performing dimensionality reduction, PCA gives slightly better results than the DCT used for computing MFCCs. So if training data is available beforehand, PCA-whitened mel-spectral frames are to be preferred over MFCCs.

Optimal decorrelation and whitening (by PCA or ZCA) help, unless the model is capable of capturing (co-)variances itself. For the classical "Single Gaussian

PCA whitening, so the distance between the means of two sets of data points is the same under both transforms. For the same reason, it does not make a difference for k-Means.

[9] Note that we perform PCA *whitening*, so the improvement over MFCCs may be due to better decorrelation for compression, due to whitening, or both. In the previous two experiments, the models were invariant to whitening (i.e., component-wise linear scaling), and PCA outperformed MFCCs by a smaller margin. This indicates that both optimal decorrelation and whitening can be helpful, depending on the model.

of MFCCs" it does not make a difference – a "Single Gaussian of unprocessed mel-spectral frames" performs exactly the same. For a mean-only model, both decorrelation and whitening are important, and this may also apply to k-Means.

Finally, log-scaled magnitudes were found to generally outperform linearly-scaled magnitudes, corroborating [LR05].

5.4 Stepwise Evaluation of the New System

In the previous section, we performed a limited evaluation of some variations of log-frequency, log-magnitude spectrograms, which form the base of the system proposed in Chapter 4. Here we will evaluate the complete system on all three datasets.

Section 5.4.1 introduces the random baseline as well as results with the "Single Gaussian of MFCCs" and three state-of-the-art methods for 1517-Artists, Homburg and Ballroom, allowing us to value the performance of our system. Instead of evaluating our complete system right away, we will start from the "Single Gaussian of MFCCs" and gradually change it into the mcRBM-based system we proposed. Specifically, we will replace the single Gaussian with a Gaussian Mixture Model in Section 5.4.2 to see the effect of a more powerful (multimode) global model. In Section 5.4.3, we model the songs as histograms of vector-quantized spectral frames instead, then exchange the simple k-Means based vector quantizer with RBMs and mcRBMs as unsupervised feature abstractors in Section 5.4.4. The remaining sections extend the frame-wise local features to blocks of frames (i.e., spectrogram excerpts), again with Gaussian-based (Section 5.4.5), k-Means-based (Section 5.4.6) and RBM-based (Section 5.4.7) song models, the latter constituting the final system. At each stage, we explore the parameter space to find the best possible instantiation, so we can assess the influence of each change at its full potential. Section 5.4.8 applies Distance Space Normalization (DSN) to our best models to make the results comparable to the state-of-the-art, and Section 5.4.9 finally explores how well our method performs when evaluated on a different dataset than trained for.

5.4.1 Lower bound, Baseline and State-of-the-art

To be able to value the performance of our system on a dataset, we establish some results to compare it to:

Lower bound: As a first step, for each of the datasets we compute the random baseline using a randomized dissimilarity measure. This serves as a lower bound that any meaningful measure must surpass.

Baseline: Secondly, we will use the classic "Single Gaussian of MFCCs" of Mandel and Ellis [ME05] as a baseline method. We use 20 cepstral coefficients

dataset	method	precision at 1	5	10	20
1517-Artists	random	4.6	5.1	5.1	5.1
	G1 of MFCCs	20.0	17.5	16.1	14.9
	Pohle RTBOF	30.0	27.5	25.5	23.6
	Seyerlehner BLS	31.3	28.4	26.5	24.4
	Seyerlehner CMB	**33.8**	**30.5**	**28.4**	**26.1**
Homburg	random	15.2	15.1	15.5	15.5
	G1 of MFCCs	44.2	41.7	40.3	38.6
	Pohle RTBOF	49.1	47.5	46.2	44.7
	Seyerlehner BLS	48.8	46.7	45.3	44.1
	Seyerlehner CMB	**51.5**	**49.5**	**47.5**	**46.4**
Ballroom	random	12.5	12.7	13.1	12.9
	G1 of MFCCs	48.4	41.4	37.4	33.5
	Pohle RTBOF	**90.3**	**82.2**	**77.5**	**70.3**
	Seyerlehner BLS	81.0	73.3	67.7	60.8
	Seyerlehner CMB	88.1	81.8	76.7	69.7

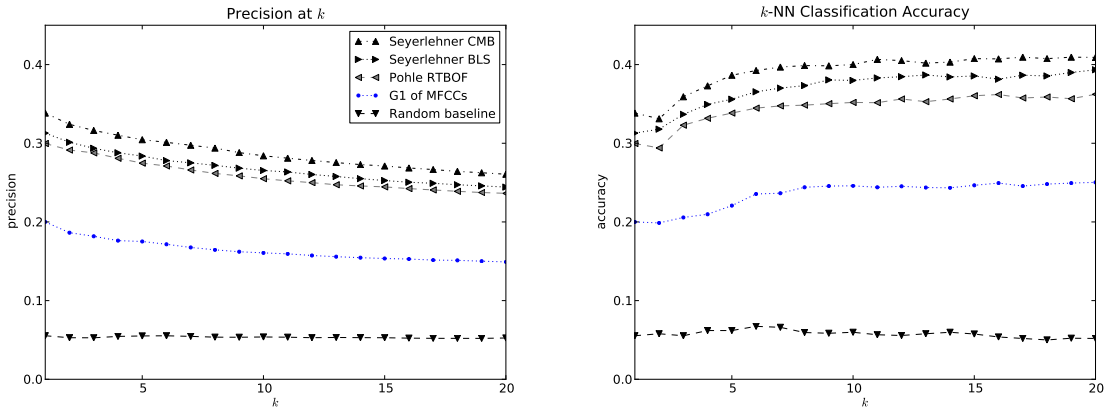
Table 5.4: Lower bound, "Single Gaussian of MFCCs" and state-of-the-art results on all three evaluation datasets

(including the zeroth coefficient) of 40 mel bands (see p. 50 for the exact filter bank), and a single full-covariance Gaussian per song compared by the symmetric KL divergence (Equation 5.3, p. 96). Methods falling short of this baseline of a simple feature and model do not have any practical significance.

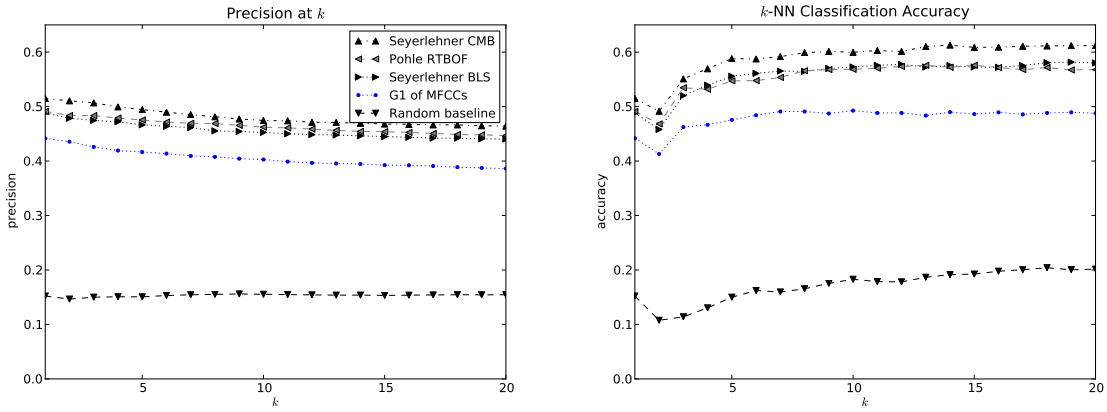
State-of-the-art: As surpassing the baseline of Mandel and Ellis does not show much, we also compare our results to three state-of-the-art measures described in Section 2.3.4 (p. 27): Pohle's RTBOF, Seyerlehner's BLS and the combination of both measures, Seyerlehner's CMB. Results for these datasets are reported in [SWP10] and [Sey10, pp. 141–142 and p. 147]. To get more accurate numbers, however, we download the distance matrices from `http://www.seyerlehner.info` and evaluate with these.

Figure 5.2 shows the precision at k and k-NN classification accuracies for all 5 measures on all three datasets, with k ranging from 1 to 20. Additionally, Table 5.4 lists the precisions in numbers to facilitate the comparison to results in the next sections. As described in Section 5.2 (p. 89), results on 1517-Artists and Homburg are artist-filtered, results on Ballroom are not (due to missing artist information).

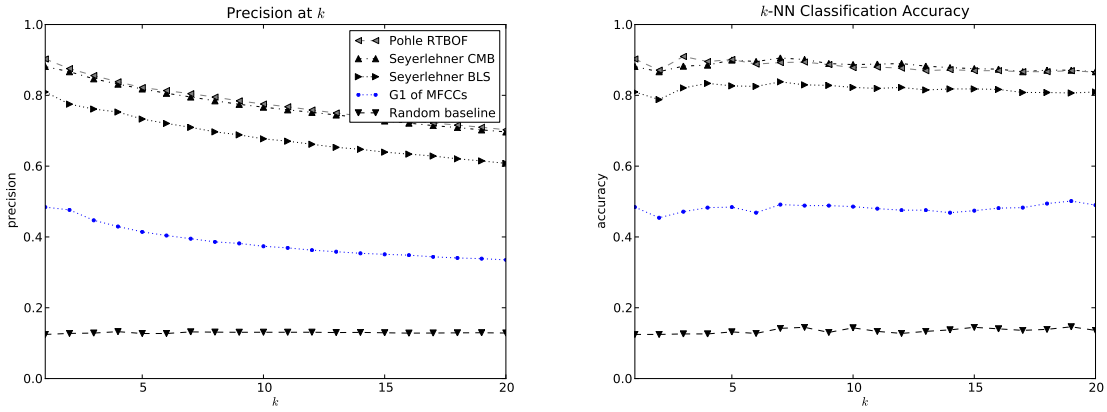
We can see that the "Single Gaussian of MFCCs" (here "G1 of MFCCs") significantly surpasses the random baseline on the datasets, but is again significantly



(a) Results on 1517-Artists



(b) Results on Homburg



(c) Results on Ballroom

Figure 5.2: Lower bound, "Single Gaussian of MFCCs" and state-of-the-art results on all three evaluation datasets

outperformed by the state-of-the-art on 1517-Artists and Ballroom. On Homburg, the margin between the non-random measures is less distinct. BLS was optimized for 1517-Artists [SWP10] and slightly outperforms RTBOF on this dataset, RTBOF was optimized on Ballroom [PSS+09] where it significantly surpasses BLS, and both measures perform similarly on Homburg, with a slight advantage of RTBOF over BLS. Combining the two measures (CMB) gives a significant boost on 1517-Artists, a minor improvement on Homburg and a slight degradation on Ballroom.

For $k = 1$, precision at k and k-NN classification accuracy are identical by definition (see Equations 5.1 and 5.2, p. 91). With increasing k, the precision tends to decrease, while the classification accuracy tends to increase.[10] The ranking of the five systems is the same under both evaluation measures, but the precision at k is more consistent over different choices of k. For some of the future experiments, we will thus only report the precision at 10 if it helps the presentation of results.

5.4.2 Frame-wise Gaussian Mixture Models

As a first extension, we replace the song-wise Single Gaussian model with a Gaussian Mixture Model (GMM), as has been used by Aucouturier et al. ([AF04], see Section 2.2.2, p. 14). This allows us to see whether merely allowing the song model to capture more complex distributions of local features (in particular, multimode distributions) improves the similarity measure. We compute the distance between GMMs using a Monte-Carlo approximation of the symmetric Kullback-Leibler divergence with 2,000 samples of each model (see Section 2.2.3, p. 14).

Table 5.5 shows our results. We evaluated mixture models of 20 diagonal-covariance Gaussians on MFCCs and PCA-whitened log-magnitude mel-spectral frames with or without dimensionality reduction. In all cases, the GMMs could not outperform full-covariance single Gaussians, falling behind by up to 3.5 percent points of precision at 1. Curiously, even a full-covariance GMM of 20 components does not surpass a single Gaussian model on MFCCs.

Note that the GMMs incur considerable computational costs: While computing the 5,054,610 pairwise dissimilarities between all 3,180 songs of 1517-Artists takes 3 minutes for the Single Gaussian model, the Monte-Carlo estimation of the Kullback-Leibler divergence increases the computation time to 3,911 minutes or 2.7 days for the diagonal-covariance GMM, and 20,904 minutes or 14.5 days for the full-covariance GMM of MFCCs.[11]

[10] Note that for Homburg, even thscale=.58e random similarities gain classification accuracy with increasing k: This is because Homburg has an uneven distribution of genres, and with increasing k it becomes more and more probable that the most prominent genre of a random set of songs equals the most prominent genre in the dataset. The other datasets have a uniform distribution of genres, so both precision and classification accuracy are close to $\frac{1}{|G|}$ independent of k (where $|G|$ is the number of genres).

[11] Measured on a 3 GHz machine using unoptimized algorithms of `http://scikit-learn.sf.net`.

local features	mixture model no.	and type of comp.	precision at 1	5	10	20
MFCCs, 20-dim.	1	full-cov. Gauss.	20.0	17.5	16.1	14.9
	20	diag-cov. Gauss.	19.7	16.9	15.6	14.5
	20	full-cov. Gauss.	20.0	16.6	15.6	14.3
log-magn. mel-spectral frames, PCA, 20-dim.	1	full-cov. Gauss.	20.6	17.8	16.4	15.2
	20	diag-cov. Gauss.	19.6	17.0	15.9	14.5
log-magn. mel-spectral frames, PCA, 40-dim.	1	full-cov. Gauss.	**21.1**	**17.9**	**16.5**	**15.1**
	20	diag-cov. Gauss.	17.6	15.4	14.4	13.2

Table 5.5: Gaussian Mixture Models of 1 or 20 full-covariance or diagonal-covariance Gaussians, evaluated on MFCCs or PCA-whitened mel-spectral frames of 1517-Artists.

5.4.3 Frame-wise k-Means Histograms

Instead of directly working with low-level local features, we now employ k-Means as a simple unsupervised feature abstractor. In particular, we create a codebook using mini-batch k-Means++ (Section 4.4.4, p. 77) on mel-spectral frames, then represent songs by replacing all their local features by the index of the respective closest codebook entry and calculating a histogram over these indices (Section 4.5, p. 80). To be invariant to the song lengths, histograms are normalized to unit ℓ_1 norm.

Both learning and applying the dictionary were performed on GPU using cudamat [Mni09]. We compute the k-Means++ initialization on a random sample of 1,000,000 local features from the respective collection we evaluate on (as in [SWK08]), then refine the clustering with mini-batches of 1024 local features each randomly sampled from the whole collection.

In the following, we perform multiple sets of similarity estimation experiments exploring different aspects of the parameter space. Afterwards we will analyze the best performing codebook qualitatively.

5.4.3.1 Distance Measures

As described in Section 4.6.1 (p. 81), there are several possible distance measures for comparing the normalized song-wise histograms. In a first set of experiments, we evaluate and compare five measures: The euclidean (ℓ_1) and manhattan (ℓ_2) distance, cosine distance, Kullback-Leibler (KL) and Jensen-Shannon (JS) divergence. For more reliable results, we evaluate these measures on two different codebooks of 64 and 128 entries. Assuming that the distance measure is independent from the exact type of local feature, we perform these experiments on unprocessed linear-

distance measure	codebook size 64				codebook size 128			
	precision at				precision at			
	1	5	10	20	1	5	10	20
manhattan (ℓ_1)	13.7	12.8	12.0	11.3	14.9	13.7	12.7	12.0
euclidean (ℓ_2)	12.1	11.5	11.1	10.6	14.6	12.7	11.9	11.2
cos distance	12.4	11.7	11.1	10.7	14.0	12.2	11.7	11.2
KL divergence	**14.7**	**13.9**	**12.9**	**11.9**	15.1	14.2	13.4	12.3
JS divergence	14.6	13.8	12.8	11.8	**16.2**	**14.6**	**13.6**	**12.4**

Table 5.6: Comparison of different distance measures for k-Means histogram models using two different codebook sizes on linear-magnitude mel-spectral frames of 1517-Artists.

magnitude mel-spectral frames.

Table 5.6 shows that KL and JS divergence perform best, followed by manhattan, euclidean and cosine distance (in this order). Compared to KL divergence, JS divergence has two advantages: It does not have any hyperparameters (for the KL divergence, a small constant ϵ has to be added to the histograms to avoid zeros), and it can be turned into a proper distance metric by taking the square root (see Section 4.6.1.4, p. 83). The manhattan distance, the best performing of the vector space distance measures we tried, is also metric and a lot faster to compute than a divergence (the full distance matrix for 1517-Artists is computed in 1 second, while it takes about 2 minutes for the JS divergence). We will thus continue our k-Means evaluation using both the JS divergence and manhattan distance.

5.4.3.2 Model Size

Table 5.6 already indicates that performance improves with larger codebooks. In a next set of experiments, we will quantify the effect of the codebook size. We evaluate codebooks from 32 up to 1024 entries, again using linear-magnitude mel-spectral frames as local features.

From Table 5.7 we see that each doubling of codebook size gives an improvement between 0.1 and 0.8 percent points in precision at 10. We decided to stop at a size of 1024 entries to keep the models comparable in dimensionality to the mcRBM-based systems we will evaluate further below, but it would be interesting to see how much this type of system would benefit from even larger codebooks. Computationally, these codebooks do not pose any problem: With 1024 entries, computing the full distance matrix for 1517-Artists takes 6 seconds using manhattan distance, and 7 minutes using JS divergence. For further experiments, we will only use codebooks of 512 and 1024 enries.

code-book size	manhattan distance precision at 1	5	10	20	JS divergence precision at 1	5	10	20
32	13.6	11.7	11.3	10.8	14.8	13.1	12.2	11.4
64	13.7	12.8	12.0	11.3	14.6	13.8	12.8	11.8
128	14.9	13.7	12.7	12.0	16.2	14.6	13.6	12.4
256	15.7	13.7	13.1	12.1	16.2	14.9	13.7	12.5
512	15.2	14.2	13.5	12.3	16.6	15.0	14.0	12.8
1024	**16.3**	**14.8**	**13.8**	**12.6**	**17.3**	**15.3**	**14.2**	**13.0**

Table 5.7: Comparison of different codebook sizes for k-Means histogram models on linear-magnitude mel-spectral frames of 1517-Artists.

5.4.3.3 Preprocessing Variants

Having determined the best distance measures and codebook sizes, we now evaluate several variants of preprocessing the mel-spectral frames. Specifically, we will experiment with a simple normalization of the standard deviation of each dimension – a common preprocessing step for k-Means – and with PCA whitening (with and without dimensionality reduction). Additionally, we compare results for linear-scaled magnitudes with log-scaled magnitudes. We will not employ ZCA whitening because it would give the same results as PCA whitening (see Section 4.3.1.3, p. 59), safe for differences caused by the random k-Means++ initialization. To obtain more reliable results, we evaluate each feature variant with two codebook sizes (512 and 1024 entries) and two distance measures (manhattan distance and JS divergence).

Our results in Table 5.8 indicate that scaling each dimension to unit standard deviation already gives a considerable improvement of results, consistent across both codebook sizes and distance measures (second row of the table). For linear-magnitude frames, whitening tends to decrease performance. Dimensionality reduction helps, but does not generally perform as well as the simple normalization of standard deviation (compare rows 4 and 2). Log-scaled magnitudes significantly outperform linear-magnitude frames, and also surpass any of the Gaussian-based frame-wise models we evaluated. Here, full-dimensional PCA-whitened frames perform best, followed by compressed PCA-whitened frames and "std-normalized" frames.

5.4.3.4 Qualitative Analysis

Hoping to gain a better understanding of why the system works, we analyze the codebook of the best performing instantiation.

Figure 5.3 shows a random excerpt of a codebook learned on PCA-whitened log-

local feature variations			codebook size 512				codebook size 1024			
			ℓ_1		JS		ℓ_1		JS	
magn. scaling	linear transf.	feat. dim.	prec. at 1	prec. at 10	prec. at 1	prec. at 10	prec. at 1	prec. at 10	prec. at 1	prec. at 10
linear	none	40	15.2	13.5	16.6	14.0	16.3	13.8	17.3	14.2
	std	40	17.1	14.6	18.8	15.0	17.7	14.9	18.3	15.5
	PCA	40	14.0	12.0	15.9	13.3	14.1	12.3	16.5	13.4
		20	16.7	13.8	18.9	14.9	17.6	14.2	18.9	15.1
log	std	40	21.1	16.6	21.5	17.1	22.0	16.9	22.5	17.5
	PCA	40	**22.4**	**18.9**	**23.6**	**19.6**	**23.8**	**19.4**	**25.1**	**20.0**
		20	22.1	18.8	22.8	18.9	22.7	19.2	22.8	19.5

Table 5.8: Comparison of different feature preprocessing transformations for mel-spectral frames of 1517-Artists across two differently sized codebooks (512 and 1024 entries) and two distance measures (manhattan distance and JS divergence).

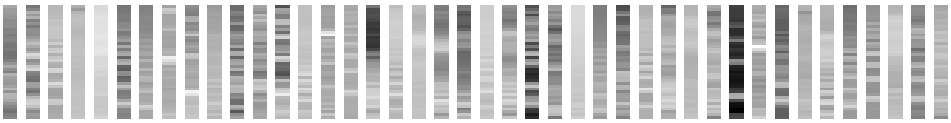


Figure 5.3: Random excerpt of a codebook learned by k-Means on 1517-Artists. Each block depicts a single mel-spectral frame, with mel frequency increasing from bottom to top, and high and low (log-scaled) magnitudes denoted by bright and dark gray tones, respectively.

method	1517-Artists			Homburg			Ballroom		
	precision at			precision at			precision at		
	1	5	10	1	5	10	1	5	10
G1 of MFCCs	20.0	17.5	16.1	44.2	41.7	40.3	48.4	41.4	37.4
Seyerlehner ML-VQ	22.8	20.0							
Seyerlehner CMB	33.8	30.5	28.4	51.5	49.5	47.5	88.1	81.8	76.7
k-Means 1024, JS	25.1	21.3	20.0	46.0	42.9	41.3	46.4	39.2	35.8

Table 5.9: Results for the best frame-wise k-Means histogram model on all three datasets compared to three other approaches.

magnitude mel-spectral frames of 1517-Artists. For the visualization, the codewords have been unwhitened, so they are directly interpretable as spectral frames. As we can see, few codewords focus on a single mel filter only, most encompass multiple non-adjacent frequencies instead, which could be harmonic overtones or chords. Resynthesizing the sound ("audialization") reveals that most centroids are very noisy, but some resemble string or wind instruments or vowels, albeit without any dynamics (as single frames cannot capture any time information). A model of a song might thus capture the proportion of noisy to harmonic frames, as well as the amount of prominent singing, but only in a pitch-dependent manner.

5.4.3.5 Conclusion

In our experiments, we determined the best performing preprocessing of local features for k-Means clustering as well as the best performing distance measure for comparing VQ-based feature histograms to estimate song similarities on 1517-Artists. We also observed that results improve with larger VQ codebooks, but to keep song models compact, we did not attempt to find the optimum codebook size. We will now validate our parameter choices on Homburg and Ballroom, as well as briefly compare our results to other approaches.

Table 5.9 shows the results for our best VQ-based model (a 1024-entry codebook of full-dimensional PCA-whitened log-magnitude spectral frames) on all three datasets. On 1517-Artists and Homburg, the VQ-based model notably outperforms the "Single Gaussian of MFCCs" (and also any of the other Gaussian-based models we evaluated). Interestingly, it also beats the multi-level VQ approach of [SWK08]: On 1517-Artists, they report precisions at 1 and 5 of 22.83% and 19.96%, respectively. They created a codebook from 321-dimensional log-frequency log-magnitude spectral frames in two stages (first for each song, then for the collection), but only extracted frames at likely note onset positions and did not standardize or whiten the features, all of which may be a reason for worse performance. On Ballroom, we

excluded 4 seconds of fade-out of each song as well as 28 songs that have been sampled at only 11 kHz from training, as these atypical frames adversely affected the PCA transform and codebook (resulting in about 1 percent point worse precision at 1). Still the "Single Gaussian of MFCCs" performs better than our VQ-based model, possibly indicating that VQ is not suited for all kinds of music. Given that Ballroom is a dataset of ballroom dance music, the significance of an evaluation of features that are inherently incapable of capturing any rhythm may be limited, however.

5.4.4 Frame-wise (mc)RBM Histograms

For the next set of experiments, we replace the VQ-based feature abstractor with an mRBM or mcRBM. We train a model on a random set of a million mel-spectral frames extracted from 1517-Artists (the same set used for k-Means), then use it to generate high-level feature descriptions by clamping a mel-spectral frame to the visible units and inferring the hidden unit activations according to $p(\boldsymbol{h}|\boldsymbol{v}, \boldsymbol{\theta})$ (see Section 4.4, p. 61). A global model for a song is computed by summing up the hidden state activation vectors across all frames of the song, resulting in a histogram in which each bin represents the cumulative activation of a hidden unit over the song (see Section 4.5, p. 80). As in previous experiments, histograms are normalized to unit ℓ_1 norm to account for differences in song length.

We use a Gaussian-Bernoulli RBM (referred to as "mRBM", see Section 4.4.2.2, p. 70) with 580 or 1024 hidden units, trained on GPU using cudamat [Mni09] for 100 epochs with CD-1, a mini-batch size of 128 samples, a slowly annealed learning rate starting at 0.075, a momentum of 0.9 (halved for the first 20 epochs) and ℓ_1 weight decay of 0.001 (0 for the first 15 epochs). These choices roughly follow [Hin10], but results were very robust to changes of the training hyperparameters. The size of the hidden layer was chosen to fit the number of mcRBM hidden units and k-Means centroids, respectively.

For the mcRBM, we chose an architecture of 1296 factors, mapped topographically to 324 hidden covariance units, and 256 mean units. We defer the explanation of the choice of architecture to Section 5.4.7, in which we train mcRBMs on blocks of frames.[12] Training was also performed on GPU using cudamat, based on the code accompanying [RH10], keeping the suggested hyperparameter settings: 50 training epochs with PCD-1, mini-batches of 128 samples, learning rate and weight decay as for the mRBM described above, HMC with 20 leapfrog steps and a target rejection rate of 10%. For further details regarding the training, please see Section 4.4.2.2 (p. 73).

[12] The reason is that we actually performed extensive block-wise mcRBM experiments first, and only flipped the order of presentation in this chapter.

model	ℓ_1 distance				JS divergence			
	precision at				precision at			
	1	5	10	20	1	5	10	20
mRBM 580	14.2	12.9	12.3	11.7	14.2	12.8	12.3	11.7
mRBM 1024	14.5	13.5	12.7	12.0	14.0	13.3	12.7	12.0
mcRBM 580	**20.5**	**17.7**	**16.5**	**14.9**	**19.6**	**16.8**	**15.4**	**14.0**

Table 5.10: Results for (mc)RBM histogram models on PCA-whitened log-magnitude mel-spectral frames of 1517-Artists.

As for k-Means, we will first evaluate the system in terms of precision at k, then perform a qualitative analysis of the features learned by the model.

5.4.4.1 Quantitative Analysis

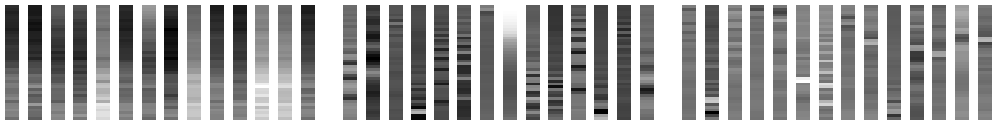
We train the models on the best performing features of the previous experiments: PCA-whitened log-magnitude mel-spectral frames.[13] Table 5.10 shows that the mcRBM performs significantly better than the mRBMs. Specifically, the mcRBM slightly surpasses the "Single Gaussian of MFCCs" (e.g., Table 5.9), while the mRBMs fall 5 to 6 percent points behind. However, both cannot compete with the k-Means histogram models. Interestingly, for this type of histogram, the JS divergence performs worse than manhattan distance – for VQ-based models, it is the other way round.

5.4.4.2 Qualitative Analysis

To visualize the filters learned by the models to infer latent representations with or to generate data from, we unwhiten the incoming weights of mRBM hidden units, mcRBM factors and mcRBM hidden mean units. The unwhitened incoming weights of a unit can be interpreted as the kind of mel-spectral frame the unit would respond to the most (more precisely: the direction in the feature space it is aligned with), where values of zero denote invariance of the filter to the respective part of the spectrum.

The mRBM weights (Figure 5.4a) all seem to concentrate on lower frequencies, but without any clear patterns within this range, and their audialization sounds like low-pass filtered white noise. The mcRBM's covariance filters, or factor weights, mostly react on complex patterns of simultaneously active mel filters, while some (e.g., the 8th in Figure 5.4b) simply capture correlations of a range of neighboring frequencies. Figure 5.4c shows some of the mean unit weights of the same mcRBM.

[13] In Section 5.4.7, we will evaluate (mc)RBMs on other variations of local features.



(a) mRBM weights (b) mcRBM cov. filters (c) mcRBM mean weights

Figure 5.4: Randomly selected examples of mRBM weights (a), mcRBM factor weights (b) and mcRBM mean unit weights (c) learned on mel-spectral frames of 1517-Artists. Each block depicts a single frame, with mel frequency increasing from bottom to top, and high and low magnitudes denoted by bright and dark gray tones, respectively. To enhance contrast for the visualization, tones have been scaled separately for each frame to either assign black to the lowest negative or white to the highest positive value, leaving middle gray assigned to zero.

Most of them seem to just focus on two adjacent frequencies. Audializations of both types of mcRBM filters are still rather noisy (as all frequencies are active in most filters), although some pitch preferences can be heard. In contrast to k-Means centroids, none of the sounds resemble anything heard in music.

In summary, the filters are difficult to interpret both from their visualization and their audialization – this will change when local features include time information.

5.4.5 Block-wise Single Gaussian Models

In the next sets of experiments, we will evaluate the effect of including temporal information in local features. First, we return to Gaussian-based models, just replacing the mel-spectral frames with mel-spectral blocks. Specifically, instead of representing a song as a collection of its frames, we form overlapping blocks of consecutive frames at each valid position in the song (see Figure 4.3e, p. 52) and treat these blocks as higher-dimensional local features to be modeled. As GMMs did not show any improvement over single-component Gaussian models and were expensive to compute, we will only consider single Gaussians.

In particular, we experiment with blocks of 9, 15, and 39 log-magnitude mel-spectral frames, using diagonal- or full-covariance Gaussian models. At a frame hop size of 32 ms, these blocks represent up to 1.2 s of music. For full-covariance models, all blocks were PCA-compressed to 99% of their original variance (retaining 67, 109, and 310 dimensions for 9-, 15-, and 39-frame blocks, respectively). Even then, Gaussian models for the largest block size have 48,515 dimensions, which far exceeds the model sizes of any of the similarity measures considered in this thesis. For the shortest blocks and for diagonal-covariance Gaussians on all block lengths, we also run experiments with uncompressed PCA-whitened features to see how

block length	preprocessed feature dim.	covariance of model	model dim.	precision at 1	5	10	20
1	40	diagonal	80	18.5	15.9	15.0	13.9
		full	3,320	21.1	17.9	16.5	15.1
	20	diagonal	40	17.8	15.6	14.5	13.5
		full	860	20.6	17.8	16.4	15.2
9	360	diagonal	720	22.9	20.0	18.3	16.7
		full	65,340	21.3	18.9	17.5	16.2
	67	diagonal	134	22.2	18.9	17.5	16.0
		full	2,345	21.5	19.1	17.8	16.4
15	600	diagonal	1,200	22.9	20.2	18.6	17.0
	109	diagonal	218	22.9	19.4	18.2	16.8
		full	6,104	22.1	19.1	17.8	16.4
39	1560	diagonal	3,120	**23.4**	20.4	18.6	17.0
	310	diagonal	620	23.2	**20.6**	**19.2**	**17.5**
		full	48,515	21.0	18.4	17.4	16.0

Table 5.11: Results for single Gaussian models on log-magnitude mel-spectral blocks of 1517-Artists. The first column lists the length of blocks in frames, the second lists the feature dimensionality after preprocessing with PCA whitening and optional compression to 99% of the original variance. The third and fourth column list the type of Gaussian and the resulting model dimensionality.

dimensionality reduction affects performance.

Table 5.11 shows the results for several combinations of block lengths, PCA compression and Gaussian models. For comparison, we included results for frame-wise features (block length 1). We can see that performance generally improves with increasing block length. Interestingly, diagonal-covariance models outperform full-covariance models for mel-spectral blocks, while it is the other way round for single frames. PCA compression reduces precision for short blocks, but improves precision for 39-frame blocks. The best performing instantiation, a diagonal-covariance Gaussian on PCA-compressed mel-spectral blocks of 39 frames, is still superseded by our best frame-wise k-Means histogram model, though. Extrapolating the results, we expect that further increasing the block length will not push the limit much further. We will see that other types of models are suited much better for capturing temporal information.

block length	preprocessed feature dim.	codebook size 512				codebook size 1024			
		ℓ_1		JS		ℓ_1		JS	
		prec. at		prec. at		prec. at		prec. at	
		1	10	1	10	1	10	1	10
1	40	22.4	18.9	23.6	19.6	23.8	19.4	25.1	20.0
	20	22.1	18.8	22.8	18.9	22.7	19.2	22.8	19.5
9	360					18.2	15.9	19.0	16.5
	67	25.8	**22.0**	**27.0**	**22.5**	**26.5**	**22.0**	**27.3**	**22.4**
15	109	**25.9**	21.3	25.6	22.2	25.0	21.1	25.9	21.6
39	310	15.9	14.0	17.2	12.4	19.2	16.3	20.4	16.7
	67					24.4	20.1	23.7	21.0

Table 5.12: Results for k-Means histogram models on log-magnitude mel-spectral blocks of 1517-Artists. For block lengths of 1 and 9 frames we list results with both full-dimensional and compressed blocks, for larger blocks only compressed features were evaluated.

5.4.6 Block-wise k-Means Histograms

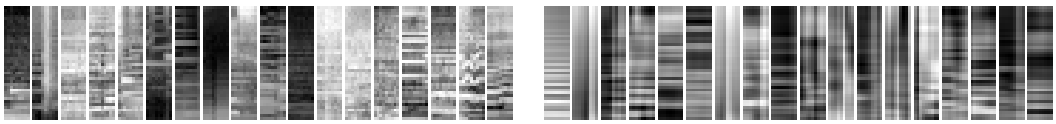
So far, VQ-based methods performed best in our evaluation. The next set of experiments will assess if performance improves with mel-spectral blocks, as it did for Gaussian-based models. Following Sections 5.4.3 and 5.4.4, we will first evaluate precision at k, then perform a qualitative analysis of the codebooks.

5.4.6.1 Quantitative Analysis

Again, we experiment with log-magnitude mel-spectral blocks of 9, 15, and 39 frames, whitened by PCA and optionally compressed to 99% of the original variance. To reduce computational costs, we train the codebook on a million blocks randomly sampled across the songs of a dataset rather than all blocks of a collection. Other than that, we proceed as for frame-wise k-Means: Each song is transformed into a sequence of maximally overlapping mel-spectral blocks, which are replaced by codewords and histogrammed into a global model. The results (Table 5.12) show that blocks of 9 frames perform best, followed by 15-frame blocks, while 39-frame blocks perform significantly worse than single frames. Interestingly, PCA compression hurts single-frame models, but helps for blocks: an experiment with uncompressed 9-frame blocks gave very poor results (third row of the table). To see whether performance is dependent on the block length or merely on the dimensionality of local features, we performed an additional experiment compressing 39-frame blocks to as few dimensions as 9-frame blocks (last row of the table). While this notably improved results, blocks of 9 frames still perform best.

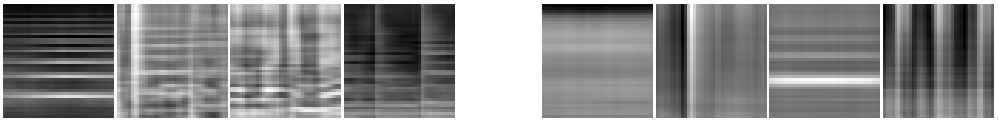
method	1517-Artists			Homburg			Ballroom		
	precision at			precision at			precision at		
	1	5	10	1	5	10	1	5	10
G1 of MFCCs	20.0	17.5	16.1	44.2	41.7	40.3	48.4	41.4	37.4
Seyerlehner CMB	33.8	30.5	28.4	51.5	49.5	47.5	88.1	81.8	76.7
frame-wise k-Means	25.1	21.3	20.0	46.0	42.9	41.3	46.4	39.2	35.8
block-wise k-Means	27.3	24.2	22.4	44.9	44.5	43.5	54.0	45.9	42.1

Table 5.13: Results for the best block-wise k-Means histogram model on all three datasets compared to three other approaches.



(a) 9 frames (288 ms), uncompressed

(b) 9 frames, compressed to 67 dimensions



(c) 39 frames (1248 ms), cpr. to 310 dim.

(d) 39 frames, compressed to 67 dim.

Figure 5.5: Exemplary unwhitened codebook entries learned by k-Means on mel-spectral blocks of different lengths and different PCA compressions. Each block depicts a spectrogram excerpt: Time increases from left to right, mel frequency from bottom to top, and bright and dark gray tones denote high and low (log-scaled) magnitudes, respectively.

Evaluating the best-performing instantiation (9-frame blocks whitened and compressed to 67 dimensions in a 1024-entry codebook compared by JS divergence) on Ballroom, we find that precisions have improved by up to 7 percent points over frame-wise k-Means histograms (Table 5.13). On Homburg, precisions improved by about 2 percent points, but not for the closest neighbor. Block-wise k-Means histograms outperform the "Single Gaussian of MFCCs" on all three datasets, but still fall short of the state-of-the-art.

5.4.6.2 Qualitative Analysis

We now take a look at the codebook entries to better understand the system. Specifically, we search for an explanation as to why dimensionality reduction improves performance. As in Section 5.4.3.4, we unwhiten the codewords, making

them interpretable as spectrogram excerpts.

Figure 5.5 shows randomly picked codewords from four different codebooks learned on 1517-Artists. Comparing Figures 5.5a and 5.5b, two codebooks of 9-frame blocks, we see that the uncompressed version features much sharper visual details than the codebook learned on PCA-compressed frames. Seemingly, "softer" codewords are suited better for capturing the sound of a music piece (see Table 5.12). When resynthesized, the difference in detail can also be heard: most codewords are short (288 ms) excerpts of singing or instruments, but only in uncompressed frames it is possible to understand syllables.

Codewords learned from weakly compressed longer blocks are still very detailed (Figure 5.5c). Their audialization sounds like noisy music excerpts; it is possible to understand full words, recognize rhythms or parts of melodies (without identifying the instrument). Given that the codewords are linear combinations of actual music excerpts, this is not surprising, but it indicates that many cluster centroids are dominated by a single training example. With stronger compression (Figure 5.5d), codewords look more abstract, with clear horizontal or vertical structure. Many are just noisy, some are rhythmic, capture two or three notes or sound like whispering voices.

All in all, it seems that not performing any dimensionality reduction leads to many highly detailed codewords that are almost directly copied from a song and do not form useful generic templates for modeling music. PCA compression reduces the variability of data and seems to help forming more abstract codewords. For single frames, this was not a problem – a spectral frame only describes a stationary distribution of frequencies, which is not as specific to a particular song as a second-long excerpt, and a higher level of detail seemingly helped to better distinguish spectral aspects relevant to perceived music similarity.

5.4.7 Block-wise (mc)RBM Histograms

For frame-wise features, mcRBMs were only marginally better than the "Single Gaussian of MFCCs". We will now evaluate if and how their ability to find and model dependencies between input dimensions helps to capture the dynamics of short music excerpts, and whether this improves music similarity estimation.

5.4.7.1 mRBMs

As in Section 5.4.4, we begin with standard Gaussian-Bernoulli RBMs (mRBMs) of 580 or 1024 hidden units. Instead of training on single frames, we train on a set of a million log-magnitude mel-spectral blocks (centered on the frames we used for frame-wise RBMs) whitened with PCA and compressed to 99% of their original variance. Song-wise accumulations of hidden unit activations are compared

block length	preprocessed feature dim.	hidden units	precision at 1	5	10	20
1	40	580	14.2	12.9	12.3	11.7
		1024	14.5	**13.5**	12.7	**12.0**
9	67	580	**15.2**	**13.5**	12.6	11.8
		1024	14.9	13.4	12.8	11.9
15	109	1024	14.8	13.4	12.5	11.8
39	310	1024	14.3	**13.5**	**12.9**	11.8

Table 5.14: Results for mRBM histogram models on log-magnitude mel-spectral blocks of 1517-Artists. Blocks were PCA-whitened and compressed to 99% of their original variance, single frames (block length 1) were whitened only.

with ℓ_1 distance, as JS divergence produced worse results in our frame-wise RBM experiments.

Table 5.14 shows that the performance of the model is largely independent of the block length – apparently, the model fails to incorporate any useful temporal information and performs just as bad as for single frames (well above the random baseline, but much worse than the "Single Gaussian of MFCCs").

5.4.7.2 mcRBM Architectures

We now proceed to experimenting with mcRBMs. A first critical choice concerns the number of factors, hidden covariance and hidden mean units to use, and the related question of how to map factors to hidden covariance units. For orientation, we will review other authors' decisions on this.

Ranzato et al. [RH10] train mcRBMs of 256 factors, 256 hidden covariance and 100 hidden mean units (from now on, we will use a shorthand notation of the form f256-c256-m100) on 768-dimensional ZCA-whitened 16×16 color image patches to qualitatively analyze the filters, and an architecture of f1024-c1024-m576 on the same data to demonstrate the quality of images generated by the model. For both architectures, the authors used a one-to-one mapping of factors to covariance units (i.e., the pooling matrix P was initialized to the negative identity matrix) which was allowed to change freely during training. In another experiment, Ranzato et al. train an f576-c144-m81 model on 192-dimensional 8×8 color image patches. Obviously, a one-to-one mapping of factors to covariance units is not possible in this case. Instead, they initialize the pooling matrix to a 2-dimensional topography: Arranging the factors and covariance units on a 24×24 and 12×12 grid, respectively, each covariance unit is mapped to a block of 3×3 factors, with adjacent block centers separated by 2 units (cf. Figure 5.6). This encourages neighboring factors to learn

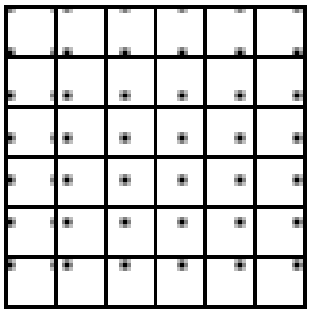


Figure 5.6: Initialization of an mcRBM's pooling matrix to a 2-dimensional topography mapping $6^2 = 36$ hidden covariance units to $12^2 = 144$ factors. Each block shows the outgoing weights of a covariance unit to the factors arranged on a 12×12 grid, with black denoting negative weights, and white denoting zero. 9 factors are associated with each unit, partly shared between neighboring units.

similar filters and makes covariance units invariant to slight variations in the input.

Dahl et a. [DRMH10] train an f1024-c1024-m512 architecture on 15-frame mel-spectral blocks from speech which were PCA-whitened and compressed to 384 dimensions. They initialize the pooling matrix to the negative identity, and only allow updates to the main diagonal and its two neighbor diagonals, corresponding to a 1-dimensional topography.

Both authors do not explain their choice of architecture any further, so we will just start with a similarly sized model and try some variations to understand how the architecture influences performance. We begin with a f324-c324-m256 model trained on log-magnitude mel-spectral blocks of 9 frames, PCA-whitened and compressed to 67 dimensions (explaining 99% of the original data variance). As for mRBMs, song-wise hidden representation histograms are compared with ℓ_1 distance only.

Table 5.15 (row 1) shows encouraging results: In contrast to frame-wise mcRBM models, the first block-wise mcRBM model we tried performs quite close to k-Means histogram models.

Reducing the number of covariance units and factors in favor of mean units (f90-c90-m490, row 2) or adding more factors and covariance units (f1296-c1296-m256, row 3) reduces performance, so we assume that the proportion of covariance to mean units is appropriate in the first model.

When we increase the number of factors and keep the number of covariance units constant (f1296-c324-m256, rows 4–6), we cannot use one-to-one pooling. Simply initializing the pooling matrix to a 2-dimensional topography as in [RH10] (cf. Figure 5.6) makes training instable: In preliminary experiments, we often found that as soon as we allow the model to update the pooling matrix (after 10 epochs,

	architecture				training	precision at			
	fact.	cov.	mn.	pooling	epochs	1	5	10	20
1)	324	324	256	1:1, free	50	26.3	23.0	21.1	19.5
2)	90	90	490	1:1, free	50	25.5	22.7	21.0	19.4
3)	1296	1296	256	1:1, free	50	24.9	21.7	20.7	19.0
4)	1296	324	256	2d, free	50	25.1	22.6	21.0	19.3
5)	1296	324	256	2d, masked	50	**26.6**	23.7	**21.9**	20.1
6)	1296	324	256	1d, masked	50	26.3	23.6	**21.9**	**20.2**
7)	1296	1296	256	1:1, free	50	25.1	21.8	20.7	19.1
8)	1296	324	256	2d, masked	50	26.5	**23.8**	21.8	**20.2**
9)	1296	1296	256	1:1, free	10	24.2	21.6	20.0	18.6
10)	1296	324	256	2d, masked	10	25.5	22.9	21.4	19.7
11)	2500	625	512	2d, masked	50	23.9	21.4	20.1	18.5

Table 5.15: Results for different mcRBM architectures and training variants on 9-frame log-magnitude mel-spectral blocks of 1517-Artists. Blocks were PCA-whitened and compressed to 99% of their original variance.

to let the factors stabilize first), weights change chaotically and eventually *NaNs* (a special code for undefined numbers in floating point arithmetics) occur. We therefore experimented with masking the updates: Entries of the pooling matrix that are zero in the initial topographic mapping are reset to zero after each weight update (subsequently, the columns are renormalized to unit ℓ_2 norm as explained in Section 4.4.2.2, p. 73). This improves precision over the first model by about half a percent point (rows 1 and 5), and also performs better than unconstrained pooling matrix updates (rows 4 and 5). As an alternative to the 2-dimensional topographic mapping, we initialize and constrain the pooling matrix to a diagonal line, effectively creating a 1-dimensional topography. Performance is similar to the 2-dimensional topographic pooling (rows 5 and 6).

To make sure the difference in results stems from the architecture and not from different random initializations, we repeat the experiment for two of the models (f1296-c1296-m256 and f1296-c324-m256, rows 7 and 8).[14] Precisions change by at most 0.2 percent points, so results are fairly consistent (compare rows 3 to 7, and 5 to 8).

Early stopping of training after 10 epochs results in worse models (rows 9 and 10), so for future experiments, we will continue training our mcRBMs for 50 epochs.[15]

[14] As training an mcRBM takes between 6 and 24 hours, depending on the model size and input feature dimensionality, we cannot simply do this for all our models to quantify the influence of random initialization.

[15] It would be interesting to know the exact development of precision at k with the number of

local feature variations						
magn. scaling	linear transf.	feat. dim.	precision at 1	5	10	20
log	PCA	360	**26.6**	23.0	21.2	19.5
		67	**26.6**	**23.7**	**21.9**	**20.1**
		31	22.6	20.1	19.0	17.6
	ZCA	360	26.5	22.9	21.2	19.5
		67*	22.0	19.4	18.0	16.7
linear	PCA	164	20.1	16.8	15.6	14.3

*embedded in 360 dimensions, see text

Table 5.16: Comparison of different feature preprocessings for f1296-c324-m256 mcRBMs trained on 9-frame mel-spectral blocks of 1517-Artists.

A larger version of the best performing model (f2500-c625-m512, row 11) does not perform well, possibly it is oversized for only 67 dimensions of input. For future experiments, we will thus focus on the architecture of 1296 factors, 324 hidden covariance and 256 hidden mean units, with a 2-dimensional topographic pooling matrix, constrained to remain topographic during training (f1296-c324-m256, rows 5 and 8).

5.4.7.3 Preprocessing Variations

As for block-wise k-Means models, we perform a set of experiments to assess the effect of different preprocessing settings on similarity estimation. Specifically, we train the best performing model of the previous section (f1296-c324-m256) on log-magnitude mel-spectral blocks of 9 frames that have been preprocessed with PCA whitening and compression to 100%, 99% or 98% of original data variance (360, 67 or 31 dimensions) or with ZCA whitening (including a version that drops higher principal components, resulting in features that lie on a 67-dimensional linear manifold in the 360-dimensional feature space; see the end of Section 4.3.2, p. 60). In addition, we report results for an mcRBM trained on linear-magnitude blocks. We also tried training mcRBMs on unwhitened blocks, but the factors remained noisy throughout training and the mean units all developed identical weights – possibly, different training hyperparameters would allow the model to learn better, but this is not worth the effort.

Table 5.16 shows that uncompressed blocks lead to a reduction in precision by about 0.5 percent points for $k > 1$ (rows 1 and 2). Strong compression noticeably

training epochs, but we did not have the resources to evaluate this, as it would keep a GPU busy with feature extraction for several days.

model	mel bands	block length	precision at 1	5	10	20
f1296-c324-m256	40	1	20.5	17.7	16.5	14.9
		9	26.6	23.7	21.9	20.1
		15	28.1	24.7	23.0	21.2
		39	29.0	25.4	24.0	21.9
		75	27.5	24.7	23.6	21.7
	70	9	27.8	24.3	22.9	20.8
		15	28.6	25.0	23.6	21.5
		39	**29.5**	**25.5**	23.6	21.7
f2500-c625-m512	40	9	23.9	21.4	20.1	18.5
		15	27.3	24.1	22.4	20.7
		39	28.7	25.4	**24.3**	**22.2**
		75	27.0	25.1	23.4	21.6

Table 5.17: Comparison of different block lengths and frequency resolutions for mcRBM histogram models on log-magnitude mel-spectral blocks of 1517-Artists. Blocks were PCA-whitened and compressed to 99% of their original variance, single frames (block length 1) were whitened only.

hurts performance (row 3), too, so we assume compression to 99% of variance is a reasonably good choice. ZCA whitening performs the same as PCA whitening (rows 1 and 4), but does not allow dimensionality reduction, so we will not consider it in future experiments. As for the other methods evaluated in this chapter, linear-scaled magnitudes perform considerably worse than log-scaled magnitudes.

5.4.7.4 Block Size Variations

As mcRBMs are specifically designed to capture high-order dependencies between input dimensions, we hope them to excel at blocks of longer time scales or higher frequency resolution. We thus train the best performing architecture (f1296-c324-m256) on blocks of 15, 39 and 75 frames, as well as on blocks of 70 instead of 40 mel bands (cf. Figure 4.2, p. 51). Following our findings from Table 5.16, we preprocess these blocks with PCA whitening and compression to 99% of variance. To account for the higher complexity of input data, we will also experiment with the larger architecture f2500-c625-m512.

Our results are listed in Table 5.17. Starting from our previous model of 9-frame blocks (row 2), increasing the block length noticeably improves performance (rows 3–5). The best precision is obtained for blocks of 39 frames; both 15-frame and 75-frame blocks perform worse. Increasing the frequency resolution further improves

block length	first-layer model	additional layers 2.	3.	4.
	f1296-c324-m256 →	580 →	580 →	580
9	21.9	22.3	22.2	21.8
	f1296-c324-m256 →	2048 →	1024 →	580
9	21.9	22.4	22.3	21.4
39	24.0	23.7	23.7	23.9
	mRBM 1024 →	2048 →	1024 →	580
9	12.8	13.0	12.2	12.6

Table 5.18: Comparison of different DBN histogram models on log-magnitude mel-spectral blocks of 1517-Artists. Up to three layers of differently sized bRBMs are stacked on top of an mcRBM or mRBM; each layer's representation is evaluated separately in terms of precision at 10.

performance for blocks of 9 and 15 frames (rows 6 and 7). For blocks of 39 frames (row 8), precision is increased among close neighbors only and decreased for $k > 5$. The larger mcRBM model is inferior to the small architecture except on blocks of 39 frames, on which it surpasses the small model by about 0.5 percent points in precision for $k \geq 5$.

In summary, all variants of models on 39-frame blocks perform similarly well, with deviations depending on the size of the neighborhood used for computing the precision at k.

5.4.7.5 DBNs

The song-wise histogram used in mcRBM-based models has a possible disadvantage: By separately adding up each hidden unit's activations, it cannot capture which combinations of units were active for blocks in a song. An RBM could be trained on an mcRBM's latent representations to learn and represent typical combinations of active units. Recursively applying this principle, a stack of RBMs should be able to capture high-order dependencies in the data.

In fact, Dahl et al. [DRMH10] report a considerable improvement in phone recognition rates by stacking Bernoulli-Bernoulli RBMs (bRBMs) on top of an mcRBM. They were able to reduce the phone recognition error from about 23% with a single added bRBM to 20.5% for a Deep Belief Net (DBN, see Section 4.4.3, p. 75) of 8 hidden layers. In the following, we evaluate whether a DBN also helps to learn better features for music similarity estimation.

First, we start from the f1296-c324-m256 architecture on 9-frame blocks. We compute the mcRBM's latent representation for each of the one million blocks

	original DBN	additional top layer							
		combination-wise histogrammed							unit-wise h.
		6	7	8	9	10	11	12	512
prec. at 10	22.4	19.2	19.8	20.4	20.9	21.1	21.7	22.1	22.1

Table 5.19: Comparison of unit-wise and combination-wise histogramming of latent representations of 9-frame spectral blocks. On a f1296-c324-m256→529 DBN, we train bRBMs of 6 to 12 hidden units for combination-wise song histograms, and a bRBM of 512 hidden units for unit-wise histograms. We report the precision at 10 using these histograms as song models compared by ℓ_1 distance.

used for training and treat these representations as a new dataset. We train a bRBM of 580 hidden units on this data and stack it on top of the mcRBM, forming a DBN of two hidden layers (the mcRBM and bRBM layer). We extend this DBN with another bRBM trained on the first bRBM's latent representations, and finally a third bRBM trained on the representations of the second one, all of the same size. Each hidden layer of the final DBN produces abstract 580-dimensional representations of spectral blocks clamped to its visible units. In the top part of Table 5.18, we compare the different representations to the original mcRBM feature descriptions in terms of precision at 10: The first two additional layers provide a marginal improvement over the first layer, the top layer does not.

In the next set of experiments, we increase the size of the additional layers: The first bRBM is set to 2048 hidden units, the second and third to 1024 and 580 units, respectively. We evaluate this setup on the f1296-c324-m256 architecture on 9- and 39-frame blocks, as well as on an mRBM of 1024 hidden units on 9-frame blocks. In all three cases, the additional layers still only result in minor positive or negative changes of precision at 10 (Table 5.18).

As a more direct way to capture combinations of active units (which was one of the motivations for using DBNs, cf. p. 121), we change the method of building a global song model: Instead of summing up the hidden unit activations for all the blocks of a song, we binarize the activations and count how often each possible combination of active units occurs (see Section 4.5, p. 80). Of course, this is only possible with a very small hidden layer. We stack bRBMs with 6 to 12 hidden units on a DBN, resulting in song histograms over $2^6 = 64$ to $2^{12} = 4096$ combinations of active units. For comparison, we also evaluate with a 512-unit top layer histogrammed the original way. Table 5.19 shows that while the precision improves with the number of hidden units, combination-wise histogramming does not perform as well as unit-wise histogramming. Histograms over 512 hidden units perform 1.1 percent points better than histograms over 512 combinations of 9 hidden units, and it takes 12

hidden units (4096 combinations) to close this gap in precision. k-Means codewords provide much better descriptions of local features than these binary RBM codes (see Table 5.12, 113).

In summary, we do not find a notable or consistent improvement in similarity estimations with Deep Belief Nets, but no degradation either. Possibly, additional layers do not actually learn more useful representations, but only increase the network's discriminative power when subsequently fine-tuned to a classification task as in [DRMH10, HE10].

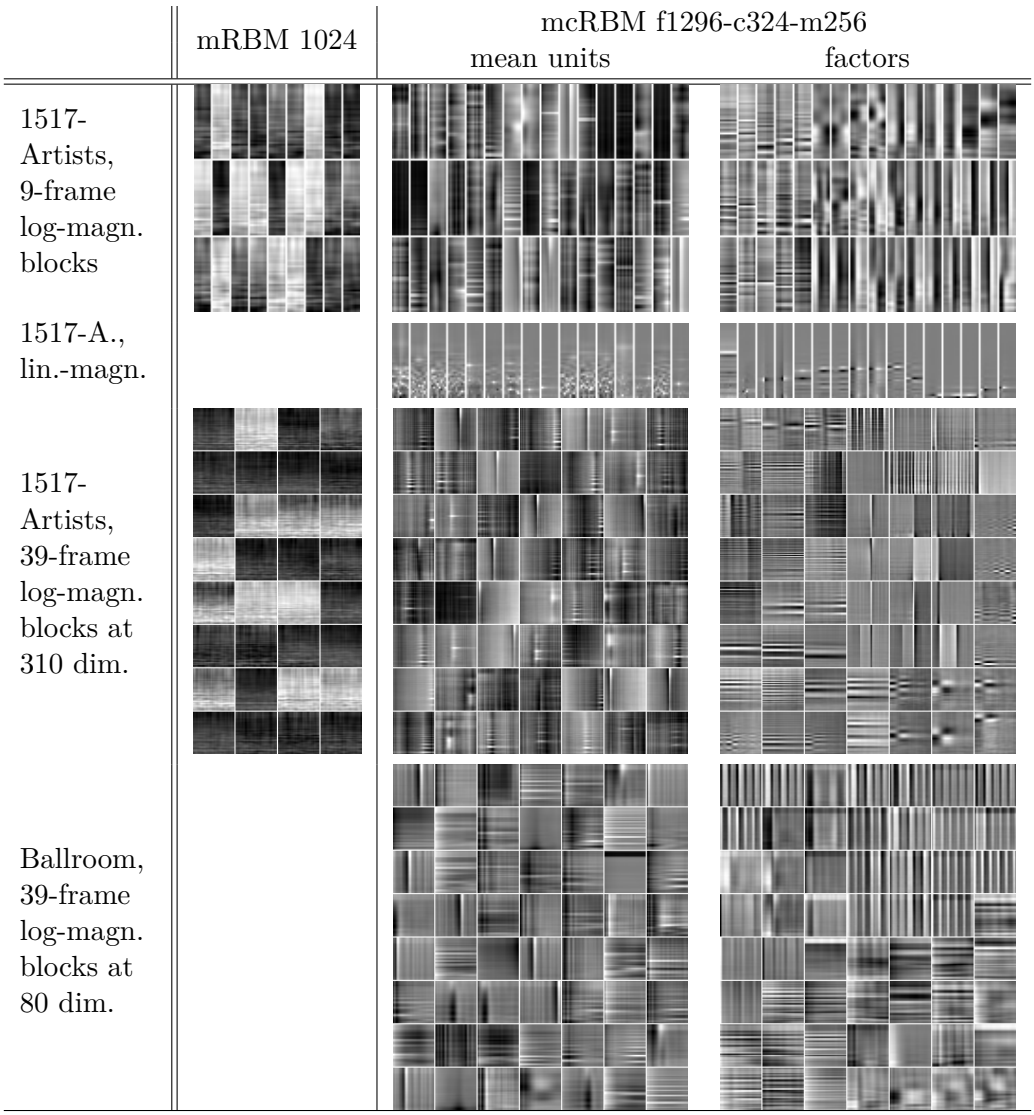
5.4.7.6 Qualitative Analysis

In Figure 5.7a, we visualize filters learned by mRBMs and mcRBMs on 9-frame and 39-frame PCA-compressed mel-spectral blocks of 1517-Artists and Ballroom. As in Section 5.4.4.2, we unwhiten the incoming weights of hidden units for mRBMs, and the incoming weights of mean units and factors for mcRBMs, allowing to interpret them as spectrogram excerpts.

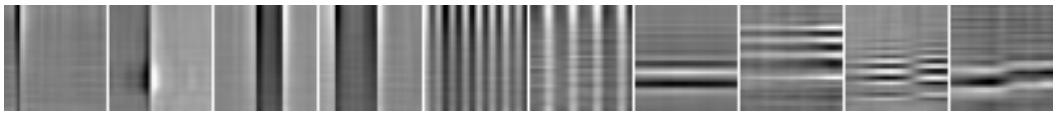
mRBMs (first column) fail to learn clearly-structured features, explaining their poor performance in comparison to k-Means or mcRBMs. This corroborates Krizhevsky's results [Kri09], who only succeeded in training this model on very small image patches of 8×8 pixels (our mel-spectral blocks are 40×9 or 40×39 "pixels").

mcRBMs develop highly interesting filters, however, which are very different from the Gabor-like filters learned on natural images (cf. [RH10]). Before we analyze the filters in detail, let us recapitulate how the mean unit weights and factor weights can be interpreted (cf. Section 4.4.2.3, p. 75). The mean filters can be understood as templates that are linearly combined to represent a data point, or as filters that are multiplied element-wise with a mel-spectral block, then summed (i.e., the filter activation corresponds to the dot product of the filter and a data point). The factor weights encode smoothness constraints in generating a data point: "pixels" that are both bright or both dark are correlated, pixels of opposite brightness are anti-correlated, and mid-gray pixels are uncorrelated. When used as a filter, the activation corresponds to the *squared* dot product of the filter and a data point. For example, a filter with vertical positive and negative ripples (e.g., the fourth and fifth of Figure 5.7b) does not react to excerpts of steady volume, and is most active for regular changes in loudness matching the periodicity of the ripples.

mcRBM filters learned on linear-magnitude blocks (Figure 5.7a, second row) are very local and noisy, but filters on log-magnitude blocks seem to capture musical structure. Mean unit weights learned on 9-frame blocks (first row) mostly show uninterpretable patterns, with a few exceptions clearly localized in frequency. Factors instead often have a clear horizontal or vertical structure, detecting frequency patterns (global in time) or changes in loudness (local in time), respectively. Note how neighboring factors develop similar filters due to the 2-dimensional topographic



(a) Excerpts of the weight matrices of mRBMs and mcRBMs on 1517-Artists and Ballroom



(b) Recurrent schemes in mcRBM factor weights on 1517-Artists (39-frame log-magnitude blocks)

Figure 5.7: Exemplary unwhitened filters learned by mcRBMs on mel-spectral blocks of different lengths and datasets. Each block depicts a spectrogram excerpt of 9 or 39 frames: Time increases from left to right, mel frequency from bottom to top, and bright and dark gray tones denote high and low magnitudes, respectively.

method	1517-Artists			Homburg		Ballroom	
	precision at			precision at		precision at	
	1	5	10	1	10	1	10
G1 of MFCCs	20.0	17.5	16.1	44.2	40.3	48.4	37.4
Seyerlehner CMB	**33.8**	**30.5**	**28.4**	**51.5**	**47.5**	**88.1**	**76.7**
k-Means 1024, 40×9*	27.3	24.2	22.4	44.9	43.5	54.0	42.1
f1296-c324-m256, 40×39	**29.0**	**25.4**	24.0	45.2	44.0	**73.9**	**62.8**
f1296-c324-m256, 70×15	28.6	25.0	23.6	**48.4**	44.0	60.3	50.0
f2500-c625-m512, 40×39	28.7	**25.0**	**24.2**	47.5	**44.2**	73.4	61.0

*size of input blocks in mel bands × block length

Table 5.20: Results for the best block-wise mcRBM histogram models on all three datasets compared to three other approaches.

pooling matrix (we display a 16×3 excerpt of the 1296 factors arranged on a 36×36 grid). On 39-frame blocks, filters seem much easier to interpret. Mean unit weights often look like notes with several harmonics, slightly localized in time (on 1517-Artists) or global (on Ballroom). Some mean unit weights also develop as a single sharply localized impulse over all frequencies (with a slight preference of high frequencies), which could represent percussion. These filters occur separately for all possible positions in the 39-frame block. Factors develop very different patterns, many of which regularly occur in all of the mcRBMs we have studied.

Figure 5.7b shows two examples each for such recurring schemes in the factors of an mcRBM trained on 39-frame blocks of 1517-Artists. In reading order, these filters can be interpreted as note onsets (impulse localized in time), fixed-length notes (an onset followed by an offset with steady activation in between), repetitive percussion (regular vertical ripples), chords (irregular horizontal ripples) and note transitions (horizontal ripples shifting to other frequencies).

On Homburg, results were very similar to 1517-Artists. However, on Ballroom we noted a higher dominance of repetitive percussion filters and fewer note onset or harmonic filters with the best performing preprocessing (blocks compressed to 80 dimensions; see Section 5.4.7.7).

5.4.7.7 Conclusion

We found that mcRBMs gain a lot of performance by incorporating more time information (up to 39 frames) or increasing the frequency resolution of mel-spectral blocks. Seemingly, their ability to model dependencies in time or frequency pays off when applied to sufficiently rich input features. To validate our findings, we evaluate mcRBMs on Homburg on Ballroom, comparing our results to other methods.

Table 5.20 shows our results. On all three datasets, mcRBMs surpass block-wise k-Means histogram models, but still fall behind the state-of-the-art. Note that long blocks perform notably better than short blocks on Ballroom, which is reasonable assuming that ballroom dances are mostly defined via rhythmic aspects rather than short-term features.

On Ballroom, we made several further interesting observations. We first trained our model on the complete clips, which include about 4 seconds of fadeout each, reaching 72.4% and 57.6% precisions at 1 and 10, respectively (for the smaller mcRBM on blocks of 39 frames). Looking at the PCA components and mcRBM filters, we noted that most of them were highly dominated by the fadeouts, possibly wasting most of the model's generative power on an aspect that is not at all relevant to music similarity. When we removed all blocks containing fadeouts from the training set, the precisions *dropped* to 62.6% and 50.1%, respectively. Investigating the issue, we noted that our default choice of PCA compression to 99% of variance resulted in 80-dimensional features when including fadeouts, and 901 dimensions when excluding fadeouts. Forcibly setting the compression to 80 dimensions again, results improved to 74.0% and 62.8% as reported in the table – an improvement by about 12 percent points just from changing the PCA compression strength. This shows that preprocessing the features is still critical with mcRBMs, and a more thorough analysis of the effects of preprocessing may give rise to better results.

5.4.8 Distance Space Normalization

The state-of-the-art approaches of Pohle (RTBOF, [PSS+09]) and Seyerlehner (BLS, [SWP10]) combine similarity estimations obtained with multiple features to include both rhythmic and timbral information. As different features yield distance measures on very different scales, sometimes orders of magnitude apart, they normalize distances obtained with different features before combining them to a total similarity estimate. Specifically, they employ a technique termed "Distance Space Normalization" (DSN, see Section 4.6.2, p. 83). As DSN alone has been shown to improve music similarity estimates, Seyerlehner performs DSN not only on the distances of different features, but also on their combination, resulting in an additional improvement of k-NN genre classification accuracy and precision at k.

For a fair comparison to these state-of-the-art approaches, we apply DSN to the distance matrices of our best methods as well as on Pohle's RTBOF (Seyerlehner's BLS and CMB – the combination of RTBOF with BLS – are already normalized). Table 5.21 shows that DSN consistently improves precision at k by 1 to 3 percent points except for k-Means histogram models (possibly, the JS divergence already results in mostly symmetrical neighborhood relations, so DSN cannot improve the distance space any further; cf. [SFSW11]). We defer a detailed comparison of our normalized models to the state-of-the-art to Section 5.5.1.

method	DSN	1517-Artists precision at 1	1517-Artists precision at 10	Homburg precision at 1	Homburg precision at 10	Ballroom precision at 1	Ballroom precision at 10
G1 of MFCCs	–	20.0	16.1	44.2	40.3	48.4	37.4
	√	21.7	18.1	46.5	42.6	51.7	41.5
Pohle RTBOF	–	30.0	25.5	49.1	46.2	90.3	77.5
	√	30.7	26.8	50.0	46.1	**90.4**	**79.0**
Seyerlehner BLS	√	31.3	26.5	48.8	45.3	81.0	67.7
Seyerlehner CMB	√	**33.8**	**28.4**	**51.5**	**47.5**	88.1	76.7
k-Means 1024, 40×1*	–	25.1	20.0	46.0	41.3	46.4	35.8
	√	25.0	21.4	44.1	41.2	49.1	39.7
k-Means 1024, 40×9	–	27.3	22.4	44.9	43.5	54.0	42.1
	√	28.5	23.5	44.1	42.8	54.0	45.3
f1296-c324-m256, 40×39	–	29.0	24.0	45.2	44.0	73.9	62.8
	√	30.1	24.8	48.0	44.7	**77.2**	**65.1**
f1296-c324-m256, 70×15	–	28.6	23.6	48.4	44.0	60.3	50.0
	√	29.9	25.0	48.7	**45.5**	65.3	53.1
f2500-c625-m512, 40×39	–	28.7	24.2	47.5	44.2	73.4	61.0
	√	**30.2**	**25.1**	**49.6**	**45.5**	75.2	63.4

*size of input blocks in mel bands × block length

Table 5.21: Results for the best models on all three datasets with and without Distance Space Normalization (DSN), compared to the state-of-the-art.

mcRBM trained on	precision at k, evaluated on 1517-Artists 1	1517-Artists 10	Homburg 1	Homburg 10	Ballroom 1	Ballroom 10
1517-Artists	**29.0**	**24.0**	**46.6**	43.5	66.3	51.0
Homburg	26.3	22.0	45.2	**44.0**	59.3	47.3
Ballroom	25.4	20.4	44.9	39.9	**73.9**	**62.8**

Table 5.22: Cross-dataset generalization of the f1296-c324-m256 mcRBM on blocks of 39 frames

On a side note, it is interesting to see how DSN improves RTBOF. As CMB is a combination of Seyerlehner's BLS with normalized RTBOF, part of CMB's advantage only stems from normalizing RTBOF and not from adding new information to the similarity measure.

5.4.9 Cross-Dataset Generalization

Up to now, we have only reported results for models trained unsupervisedly (i.e., not taking into account any meta-information such as genre labels) on the very same dataset they are evaluated on. For most practical applications, it is infeasible to always train a feature extractor on the songs we want to compute similarities for.

Ideally, we would evaluate generalization on a train/test split of each dataset, such that we train a model unsupervisedly on a subset of the collection only, then evaluate on the remaining songs. However, when selecting a sufficiently large and diverse training set, the remaining test set would be too small for reliable genre-based evaluations. The alternative would be n-fold cross-validation: Splitting a collection into n same-sized parts, we could train a feature extractor on $n-1$ parts and evaluate on the remaining fold, then repeat for all n possible assignments of folds for training and testing and average the results. However, we would need to train n models instead of one, which is infeasible for the number of mcRBM models we are evaluating (each taking between 6 and 12 hours to train on an nVidia GTX 480 graphics card).

Instead, to investigate how well a learned feature set generalizes, we evaluate it on the datasets it has not been trained on. This evaluates a much stronger type of generalization than cross-validation could – we do not only investigate if learned features generalize across songs, but also if they generalize across collections of completely different genre taxonomies.

Table 5.22 shows that the feature extractors of 1517-Artists apply well to Homburg, but not vice versa, possibly because Homburg contains less training data. Extractors of Ballroom do not perform well on the other datasets and vice versa.

This indicates that feature extractors trained on a particular collection are only applicable to collections of music similar to it. The Ballroom dataset contains a fairly different set of musical styles and possibly requires features the model could not learn from 1517-Artists or Homburg. Indeed, in Section 5.4.7.6 we noted that models trained on Ballroom seemed to develop more rhythmic feature detectors.

This can be seen as a disadvantage against the state-of-the-art, which performs well on all three datasets with a single set of features. However, both Pohle's RTBOF and Seyerlehner's BLS are engineered towards Western music, and the adaptability of our method to data could be an advantage when applied to music of different cultures. Furthermore, both state-of-the-art methods perform best on the datasets they have been optimized for: RTBOF on Ballroom, and BLS on 1517-Artists. Future experiments will show whether training our method on a larger and more diverse collection yields more general features competitive to the state-of-the-art.

5.5 Final Comparison

Complementing the evaluation of systems in terms of precision at k, we will now perform a deeper comparison of the best performing instantiations of each stage to the state-of-the-art. Specifically, we evaluate similarity measures in terms of k-NN genre classification accuracy in Section 5.5.1, analyze the plausibility of estimated low cross-genre similarities in Section 5.5.2, and examine if and how methods differ in producing hubs in Section 5.5.3.

5.5.1 Retrieval Precision and Classification Accuracy

Figure 5.8 shows the precision at k and k-NN classification accuracies for the "Single Gaussian of MFCCs", frame-wise and block-wise k-Means histogram models, the f2500-c625-m512 mcRBM (termed *mcRBM 1137* for its total number of hidden units) and the three state-of-the-art approaches BLS, RTBOF and CMB. For 1517-Artists (Figure 5.8a), we also show the performance of the 20-component diagonal GMM of MFCCs and an mRBM. Additionally, Table 5.23 lists the precisions in numbers for easier comparison.

In Figures 5.8a and 5.8b we see that each step from the baseline system (the "Single Gaussian of MFCCs") to our final system improves performance on 1517-Artists and Homburg: Histograms of unsupervisedly learned frame-wise features perform better than Gaussian models of MFCCs, block-wise features outperform frame-wise features, and mcRBMs surpass k-Means. Only mRBMs fall behind the baseline system (demonstrating the importance of the mcRBMs covariance model) and will not be considered any further. More importantly than surpassing the baseline, our best model, the f2500-c625-m512 on blocks of 39 frames, is remarkably

method	model dim.	1517-Artists precision at 1	1517-Artists precision at 10	Homburg precision at 1	Homburg precision at 10	Ballroom precision at 1	Ballroom precision at 10
Random baseline		4.6	5.1	15.2	15.5	12.5	13.1
G1 of MFCCs	230	20.0	16.1	44.2	40.3	48.4	37.4
Pohle RTBOF	1,331	30.0	25.5	49.1	46.2	**90.3**	**77.5**
Seyerlehner BLS	9,448	31.3	26.5	48.8	45.3	81.0	67.7
Seyerlehner CMB	10,779	**33.8**	**28.4**	**51.5**	**47.5**	88.1	76.7
k-Means 1024, 40×9*	1,024	27.3	22.4	44.9	43.5	54.0	42.1
f1296-c324-m256, 40×39	580	29.0	24.0	45.2	44.0	73.9	62.8
f1296-c324-m256, 40×39, DSN	580	30.1	24.8	48.0	44.7	**77.2**	**65.1**
f2500-c625-m512, 40×39, DSN	1,137	**30.2**	**25.1**	**49.6**	**45.5**	75.2	63.4

*size of input blocks in mel bands × block length

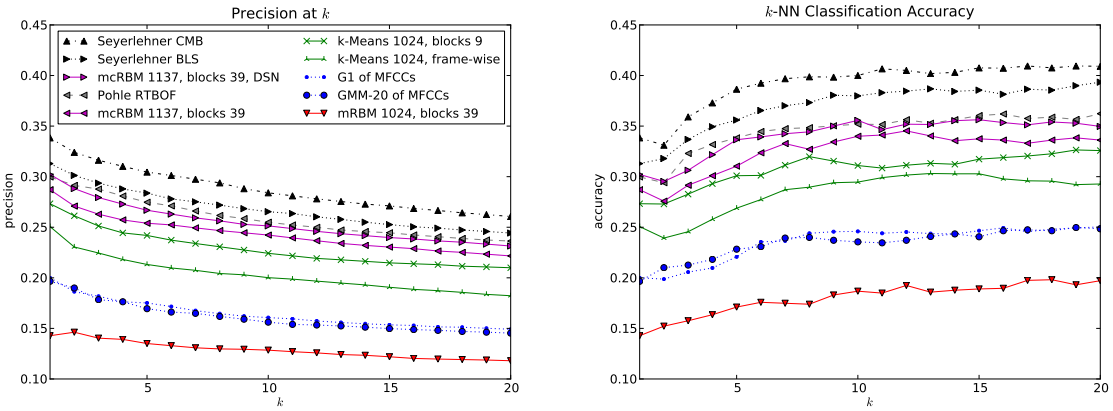
Table 5.23: Our best models compared to the baseline and state-of-the-art on all three evaluation datasets.

close to the state-of-the art on 1517-Artists and Homburg: When using Distance Space Normalization (DSN) as employed by the three state-of-the-art methods, it is 0.4 to 0.7 percent points behind RTBOF, about 1 percent point behind BLS on 1517-Artists, but also 0.8 percent points above BLS on Homburg, and 2 to 3.5 percent points behind CMB on both datasets (Table 5.23).

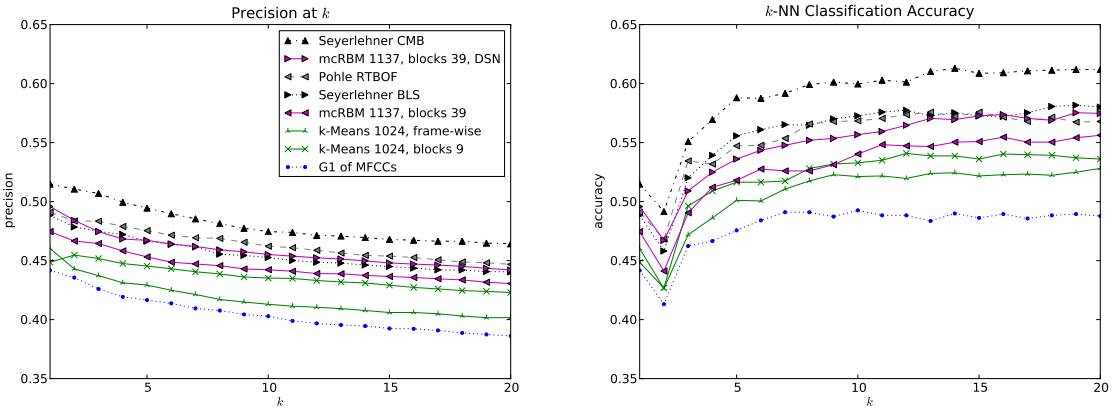
On the Ballroom dataset, results are different. Firstly, frame-wise k-Means features perform worse than the baseline system (Figure 5.8c). As ballroom dances are characterized by their rhythm, evaluation results for frame-wise features are of limited significance, though. Furthermore, the f1296-c324-m256 architecture performs better than the larger mcRBM: Possibly, the larger model is oversized for features of only 80 dimensions. Compared to the state-of-the-art, there is a large gap of 3 to 13 percent points. This suggests that our system is not yet optimally suited for modeling rhythms, although it does learn some useful long-term features if given a chance to: On Ballroom, short blocks perform considerably worse than long blocks.

5.5.2 Genre Confusion

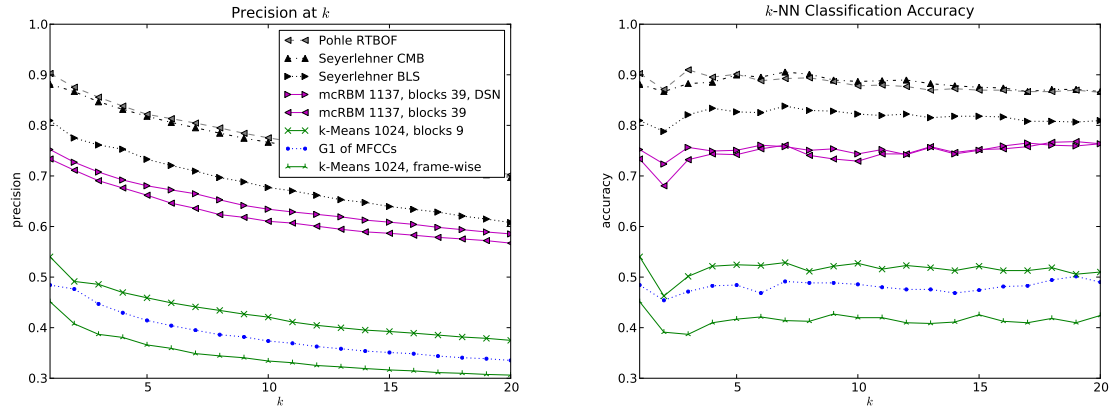
For a better understanding of how our similarity measure falls short of the state-of-the-art, we analyze the k-NN genre classification confusion matrices for the "Single Gaussian of MFCCs", the large mcRBM and Seyerlehner's CMB on 1517-Artists and Ballroom. Although we are not actually interested in genre classification rates, the confusion matrices reveal whether songs of a particular genre are assigned low



(a) Results on 1517-Artists



(b) Results on Homburg



(c) Results on Ballroom

Figure 5.8: Our best models compared to the "Single Gaussian of MFCCs" and state-of-the-art results on all three evaluation datasets. The random baseline has been omitted to improve visibility of differences between methods; it is at 5%, 15%, and 13% for 1517-Artists, Homburg and Ballroom, respectively (cf. Figure 5.2 or Table 5.23).

distances to songs of a perceptually unrelated genre or whether "mistakes" are plausible (or even desirable).

Figure 5.9 depicts the matrices. We immediately see that the Gaussian-based approach produces more misclassifications, as was clear from the classification rates (Figure 5.8). Many of these misclassifications are severe: For example, some Hip-Hop songs are classified as Electronic, some Classical pieces are classified as Folk and Viennese Waltz is often classified as Tango (more often than as Waltz or Viennese Waltz). A misclassification means that, for example, some Hip-Hop songs have more Electronic songs than Hip-Hop songs among their nearest neighbors, which does not seem sensible.

The mcRBM performs significantly better. On 1517-Artists, the most common confusions are between Alternative/Punk and Rock/Pop, Classical and New Age, Folk and Country, Hip-Hop and Reggae, R&B and Reggae. While we assume Alternative and Rock to significantly overlap, explaining confusions, the acceptability of other "mistakes" would have to be assessed in listening tests targetting the songs in question. Children's, Easy Listening, Religious, Soundtracks and World are confused with most other genres. This is plausible: World music and Easy Listening lack substantial intra-genre similarity, and the other three genres are only defined by their lyrics or context, not by musical qualities. On Ballroom, prominent confusions are between Viennese Waltz and Waltz, Waltz and Rumba, Rumba and Quickstep. Waltz and Rumba both frequently feature violins accompanied by slow, soft drums, only differing in their rhythm. Depending on the application, their confusion may thus even be desired. The other confusions are hardly acceptable, though: Waltz and Rumba are much slower than Viennese Waltz or Quickstep.

Comparing the mcRBM to Seyerlehner's CMB, there are surprisingly few qualitative differences on 1517-Artists: While CMB is slightly better in recognizing Blues, R&B, Hip-Hop and Reggae, most mistakes are the same as for mcRBMs. Possibly, some confusions are inherent in the dataset; a closer examination requires listening tests. For Ballroom, there is a noticeable decrease of misclassifications in CMB compared to the mcRBM. Most notably, CMB is better in distinguishing Waltz and Viennese Waltz, and fewer songs are misclassified as Quickstep or Rumba. This indicates that a key advantage of CMB over mcRBMs may be its improved ability to capture a song's tempo.

5.5.3 Hubness

Another quality indicator for similarity measures is its *hubness*: the amount and severity of "hubs" [AP08] or "popular nearest neighbors" [RNI10] appearing in a collection. Originally, Aucouturier et al. [AP08] defined hubs for music similarity measures as songs that occur among the nearest neighbors of extraordinarily many other songs despite not sharing any perceptual similarity to those songs. As a

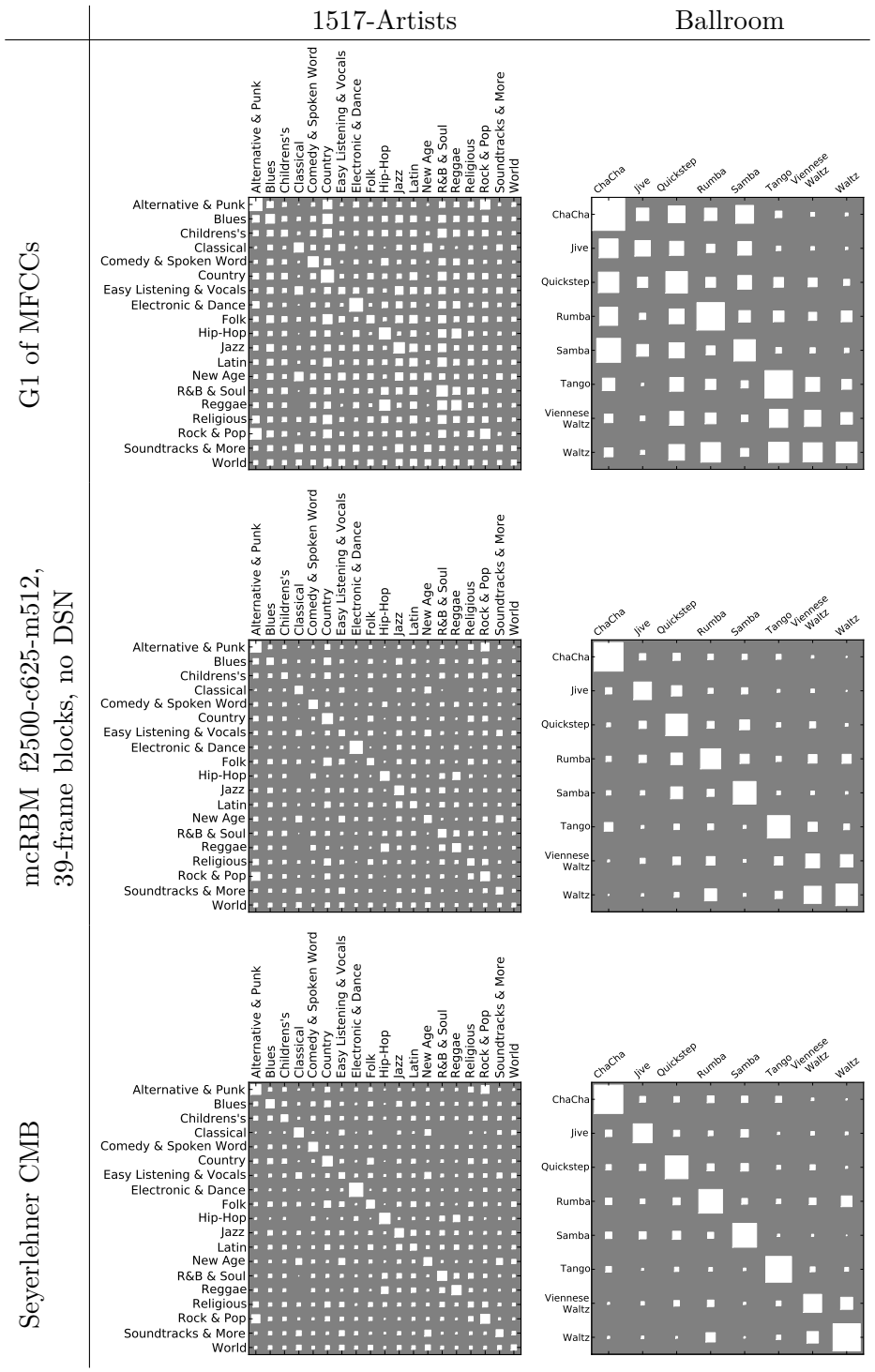


Figure 5.9: Confusion matrices for the baseline, our method and the state-of-the-art on 1517-Artists and Ballroom. The area of each white square is proportional to the number of songs that are labeled with the row's genre and have been classified as the column's genre in 15-NN classification.

simplification, perceptual similarity is often ignored, reducing the definition of hubs to overly popular nearest neighbors.

With this definition, hubness can be evaluated by analyzing the distribution of k-occurrences. The k-occurrence of a song s is defined as the number of times it occurs among the k nearest neighbors of all the other songs in a collection under a dissimilarity measure m. Following Section 5.2 (p. 91), this can be formalized as

$$o_m(s,k) = \sum_{s' \in S} \big[s \in n_m(s',k) \big]. \tag{5.4}$$

The left panel of Figure 5.10a shows the k-occurrences for all the 3,180 songs of the 1517-Artists dataset under a randomized dissimilarity measure, for $k = 20$. While it allows us to see the range of k-occurrences (left axis), it is difficult to assess how many songs are popular nearest neighbors. One way to visualize this would be sorting the songs by their k-occurrence, another is calculating a histogram of how many songs of each k-occurrence value can be found, i.e., calculating the observed distribution of k-occurrences in the dataset. The right panel of Figure 5.10a shows such a histogram for the same randomized measure. We see that it is roughly following a normal distribution with a mean of 20: Many songs occur among the 20 nearest neighbors of 20 other songs, some occur in as few as 7 or as much as 36 nearest-neighbor lists. As a randomized similarity measure can be expected to produce a rather uniform (albeit asymmetrical) neighborhood graph of few hubs, we will use its k-occurrence distribution as a reference to compare other methods against.

To complement the qualitative comparison of distributions, we calculate the *skewness* of each distribution. Following [RNI10, p. 6], we define the skewness, or asymmetry, of a distribution of k-occurrences as

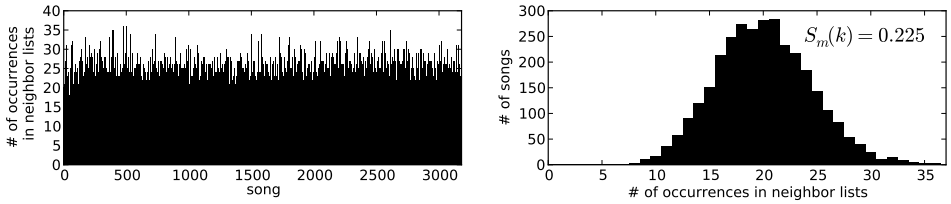
$$S_m(k) = \frac{\Big\langle \big(o_m(s,k) - \mu\big)^3 \Big\rangle_{s \in S}}{\sigma^3}, \tag{5.5}$$

where $\langle \cdot \rangle_{s \in S}$ denotes the average over all songs s and

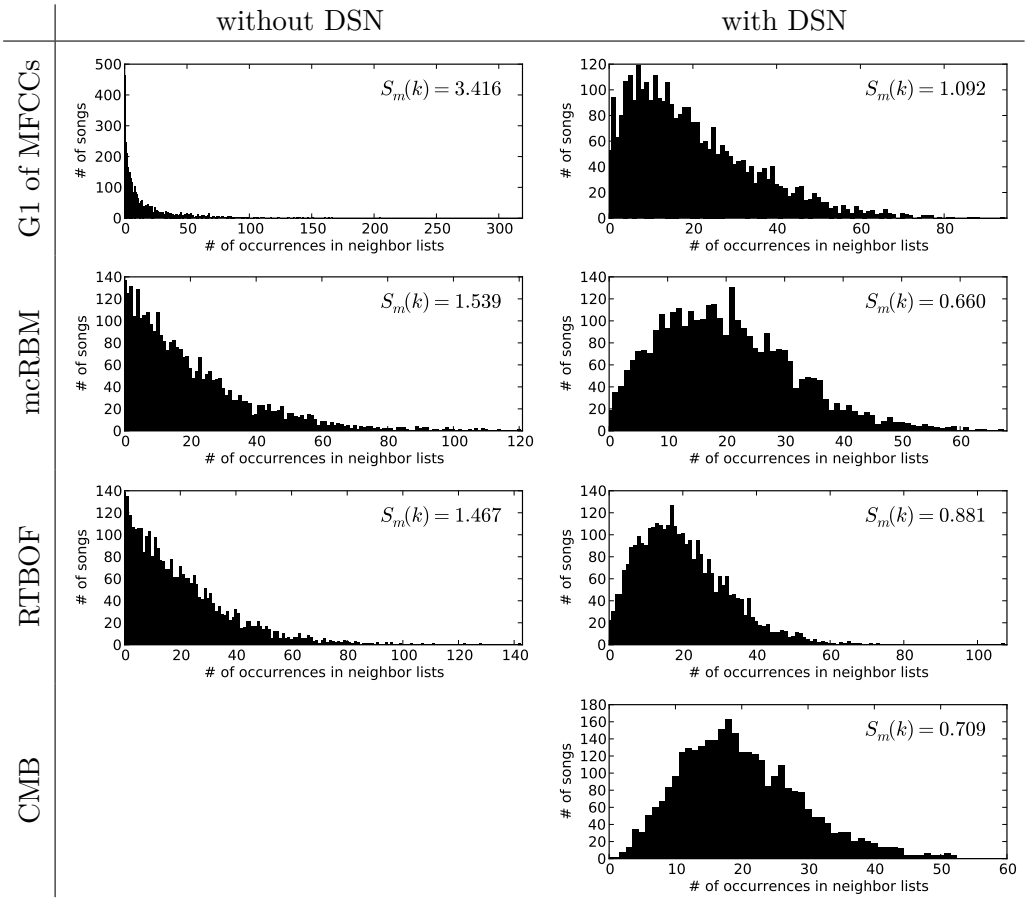
$$\mu = \big\langle o_m(s,k) \big\rangle_{s \in S}$$
$$\sigma = \sqrt{\Big\langle \big(o_m(s,k) - \mu\big)^2 \Big\rangle_{s \in S}}$$

denote the mean and standard deviation of k-occurrences, respectively. A skewness of zero corresponds to a symmetrical distribution around the mean, a positive value indicates a skewness to the right, i.e., to songs of high k-occurrences.

Figure 5.10b shows the k-occurrence histogram and skewness for four similarity measures with and without Distance Space Normalization (DSN). The top left



(a) k-occurrences and corresponding histogram for randomized song similarities



(b) k-occurrence histograms for several similarity measures with and without DSN

Figure 5.10: Analysis of k-occurrences for two baselines, our method and the state-of-the-art on 1517-Artists, for $k = 20$. (a) k-occurrence for each of the 3,180 songs (left) and a histogram over the k-occurrences (right), including the skewness S_{N_k} of the observed distribution, all calculated on randomized song similarities. (b) k-occurrence histograms and skewness for the "Single Gaussian of MFCCs", the f2500-c635-m512 mcRBM on 39-frame blocks, RTBOF and CMB, with and without Distance Space Normalization (DSN).

depicts the histogram for the unnormalized "Single Gaussian of MFCCs". The qualitative difference to the randomized measure is obvious: Only around 50 songs occur among the 20 nearest neighbors of 20 songs (compared to over 250 for the randomized measure), some songs have k-occurrences up to over 300 (strong hubs), and over 400 songs even do not occur in any nearest-neighbor list at all (so-called *anti-hubs* or *orphans*). The skewness is an order of magnitude larger than the randomized measure's. Clearly, a music recommendation system based on the "Single Gaussian of MFCCs" would be suboptimal, suggesting a few songs again and again, while missing a good portion of the collection completely.

In the second row, first column, the histogram for our mcRBM-based similarity measure is displayed. While still far from a normal distribution, both hubs and orphans are less severe than for the Gaussian-based measure: The largest hub has an k-occurrence of 120, and there are less than 140 orphans. Furthermore, the skewness is only about half as large.

The third row, first column, shows the histogram for Pohle's RTBOF. The distance measure of RTBOF is a sum of several distances separately normalized by DSN. Compared to our measure, it produces larger hubs of k-occurrences up to 142, but still has a slightly lower skewness as there are fewer hubs between k-occurrences of 80 and 120.

When employing DSN (second column), all distributions considerably improve in terms of skewness, number of hubs and number of orphans, corroborating results of Schnitzer et al. for a similar normalization of distances [SFSW11]. Qualitatively, our mcRBM-based method is most similar to RTBOF (except that RTBOF produces an additional hub of an k-occurrence of 100). The "Single Gaussian of MFCCs" produces almost three times as many orphans, more hubs and has a higher skewness. Of the measures evaluated, the histogram for CMB (which employs DSN) most closely resembles the randomized measure's distribution: The histogram is smooth, produces 3 orphans only, has a peak of over 160 songs at a sensible k-occurrence of 18, and only few hubs above an k-occurrence of 50. Still, our measure has a slightly lower skewness.

5.5.4 Summary

We have shown that our proposed similarity measure comes close to the state-of-the-art in several respects. Furthermore, by examining different prestages to our final system, we have assessed the influence of each of its components.

A comprehensive evaluation of the music similarity estimation quality, using genre precision at k as a proxy (see Section 5.2, p. 89), shows that each step from a baseline system, the classic "Single Gaussian of MFCCs" proposed by Mandel and Ellis [ME05], towards our final system, which is based on descriptions of short music excerpts provided by an mcRBM, improves the quality of the similarity measure. In

particular, unsupervisedly learned features on PCA-whitened mel-spectral frames outperform MFCCs, block-wise features surpass frame-wise features, and mcRBMs perform better than mRBMs or k-Means. DBNs did not improve results any further. Notably, the advantage of mcRBMs over k-Means becomes apparent only when applying it to sufficiently long music excerpts: k-Means performs best for short excerpts of 0.3 s, while mcRBMs profit from longer mel-spectral blocks up to 1.2 s (Tables 5.12 and 5.17). Especially on the Ballroom dataset, mcRBMs strongly outperform k-Means due to their better ability of capturing temporal structure (Figure 5.8).

Compared to the state-of-the-art, our best mcRBM-based models are almost at par to Pohle's RTBOF [PSS+09] on the 1517-Artists and Homburg dataset, very close to Seyerlehner's BLS [SWP10] on Homburg and in its vicinity on 1517-Artists, and 2 to 3.5 percent points behind the two methods' combination CMB [SWP10]. Note that we achieve these results with 580-dimensional or 1,137-dimensional vector space song models, while BLS uses 9,448 dimensions and RTBOF uses Gaussian-based models of 1,331 dimensions (Table 5.23).

On the Ballroom dataset, however, there is a larger gap of 3 to 13 percent points to the state-of-the-art (mainly to RTBOF, which has been optimized for this dataset). This indicates that our method is not suited well for capturing rhythms, and an examination of the genre confusion matrices revealed that the main weakness appears to be a lack of sensitivity to tempo (Section 5.5.2).

Another possible problem with our approach is that we mostly trained the feature abstractor on the same collection we evaluate on. As we ignore genre information during training, our system cannot overfit to the genre retrieval or classification task we evaluate with. The mcRBM will, however, adapt its filters to the kind of music it is trained on. In Section 5.4.9, we show that we can still take a model from one dataset and apply it to another collection of a different genre taxonomy. Except in the case of transferring an mcRBM from 1517-Artists to Homburg, this degrades performance, though, most severely when exchanging models with mcRBMs trained on Ballroom. This shows how our features adapt to data, which could be a desirable effect or a disadvantage, depending on the application.

In terms of producing hubs, our mcRBM-based method seems to be at least as suited for music recommendation as Pohle's RTBOF, slightly inferior to Seyerlehner's CMB, and much better than the "Single Gaussian of MFCCs" (Section 5.5.3).

All in all, these are encouraging results for unsupervised feature extraction, considering that RTBOF and BLS (and consequently CMB, the combination of RTBOF and BLS) are based on several carefully tuned hand-crafted features developed over the course of several doctoral dissertations [Pam06, Poh10, Sey10].

6 Conclusion

To conclude this thesis, we firstly review what we achieved, then give an outlook on future work.

6.1 Recapitulation

We have developed, presented and evaluated a system for content-based music similarity estimation.

Following a detailed review of existing approaches, we have set up a list of guidelines for designing our own system (Chapter 2). Specifically, we decided to follow the established architecture of a local feature extractor transforming an audio signal into a sequence of features, a global model aggregating these features into a representation of a song, and a distance measure comparing such song models. Noting that the local feature extraction stage of most systems in literature is hand-crafted, we decided to employ machine learning for obtaining better performing or better justifiable features. In particular, we were interested in unsupervised machine learning, as ground truth on music similarity to train with is hard to find, and unsupervised learning has proven useful to learn abstract representations of data in many other tasks.

A survey of related problems in three other domains, namely text retrieval, computer vision and speech processing, revealed several ideas that could be applicable to content-based music similarity estimation (Chapter 3). Most importantly, it showed that generative models of data excel in different tasks of all domains, and that a recently proposed such model, the mean-covariance Restricted Boltzmann Machine, learned features enabling state-of-the-art results in both object recognition and speech recognition. Although originally developed to model images, it has successfully been used on excerpts of audio spectrograms, and it seemed promising to apply it to music as well.

We therefore proposed a system for content-based music similarity estimation using an mcRBM to unsupervisedly learn local features on log-frequency log-magnitude spectrogram excerpts (Chapter 4). These spectrograms have been hand-crafted in the Speech Recognition community, but we only aimed to replace the years of hand-crafting in Music Information Retrieval that build on the same type of spectrograms, engineered over the course of several doctoral dissertations: Fluctuation

Patterns [Pam06], their tempo-independent variant [Poh10], and six types of block-level features [Sey10].

Our comprehensive evaluation showed that our system does not surpass the state-of-the-art, but performs remarkably close to it without incorporating any musical knowledge by hand, and with lower-dimensional song models (Chapter 5). Furthermore, we showed that each component of our final system contributes significantly to its performance: mcRBMs provide better local feature descriptions than either mRBMs or k-Means, our system profits from using longer spectrogram excerpts instead of single spectral frames, and vector space song models outperform Gaussian-based models. Except when it comes to modeling rhythms, simple k-Means clustering for feature extraction turns out to reach good performance as well, if data is appropriately preprocessed. In modeling rhythms, mcRBMs significantly surpass k-Means, but still fall behind the state-of-the-art more clearly than in modeling timbre and short-term temporal structure, as demonstrated by results on a dataset of Ballroom music. For a more extensive summary of our evaluation, please see Section 5.5.4.

On a side note, we showed that the Discrete Cosine Transform (DCT) used in the computation of Mel-Frequency Cepstral Coefficients (MFCCs), the prevalent feature for modeling timbre in music, does not approximate an optimal decorrelation of mel filter outputs too well, and is inferior to mel-spectral frames processed with Principal Component Analysis (PCA) in music similarity estimation (Section 5.3).

6.2 Future Work

To improve on these initial results, we will explore several extensions to our method.

First of all, our current approach of forming a global model by summing the activations of hidden units for each mel-spectral block of a song completely discards information about which features were active in combination. For example, an mcRBM trained on 1-second spectrogram excerpts develops single note onset detectors at multiple positions within the excerpt (Section 5.4.4.2, p. 110), but in the global song model, no information about patterns of simultaneous onsets is retained. By training deep architectures, with higher layers integrating information over longer contexts (such as convolutional RBMs, p. 40), we hope to improve modeling of temporal structure such as rhythms or melody.

In a similar manner, we could hierarchically integrate information in the frequency domain: Limiting and shifting the spectrogram excerpts not only in time, but also in frequency, features would not need to be learned separately for different pitches, but could be reused (this has also been suggested in [HBC09, p. 3]). Depending on how the features extracted at different frequencies are aggregated, features could even become pitch-independent, possibly capturing timbre more ac-

curately.

Inspired by state-of-the-art methods, we will evaluate whether our system can benefit from combining features learned on different time scales, analogous to existing systems combining timbral and rhythmic components. In addition, we will see how our system improves when combined with existing hand-crafted features or similarity measures, similar to [SWP10] – while this partly defeats the purpose of employing unsupervised learning, it could provide further insight into which relevant musical aspects are currently missed by our features, or it could result in an improvement over the state-of-the-art by combining empirical induction and knowledge engineering.

We will also further investigate the adaptability of our method to data: By training on a larger and more diverse dataset, we hope to obtain more general or more competitive features (that are not specifically targetted to ballroom music, for example), and by evaluating our method on collections of non-Western music, we hope to show that our approach can be superior to engineered state-of-the-art features mainly developed for Western popular music.

In addition to employing unsupervised learning for local feature extraction, we will leverage larger datasets to also unsupervisedly learn the global song models. Several of the methods discussed in Chapter 3 might prove beneficial for this.

In parallel, we will investigate if k-means can be further improved by increasing the size of the dictionary or using soft cluster assignments, and if recent powerful generative deep models (e.g., [NCKN11]) can provide better representations of music excerpts. More generally, we think that separating learning of feature sets from using them for encoding of music data, similar to [CN11], is an exciting approach for future research.

Appendix A

Derivations for Restricted Boltzmann Machines

To achieve a higher level of self-containment for this thesis, in this appendix we show how to derive two important quantities for binary Restricted Boltzmann Machines from their energy function (Equation 4.10, p. 63) and the joint probability density of visible and hidden states defined via this energy function (Equation 4.8, p. 62). In particular, we derive the conditional probability of a hidden unit being active given a visible state vector, and the gradient of the model's log likelihood with respect to the connection weights.

A.1 Conditional probabilities

We will derive the probability of a hidden unit h_k being active conditioned on the states of the visible units $\boldsymbol{v}$. For conciseness, we omit the model parameters $\boldsymbol{\theta}$ as a parameter to the energy function and probability distribution functions as we discuss a single static model only. Some of the steps are marked with a number above the equals sign and explained below the following equations.

$$
\begin{aligned}
P(h_k = 1|\boldsymbol{v}) &\overset{1}{=} \frac{P(\boldsymbol{v}, h_k = 1)}{P(\boldsymbol{v})} \\
&= \frac{\sum_{\{\boldsymbol{g}|g_k=1\}} p(\boldsymbol{v}, \boldsymbol{g})}{\sum_{\boldsymbol{g}} p(\boldsymbol{v}, \boldsymbol{g})} \\
&\overset{2}{=} \frac{\sum_{\{\boldsymbol{g}|g_k=1\}} e^{-E(\boldsymbol{v},\boldsymbol{g})}}{\sum_{\boldsymbol{g}} e^{-E(\boldsymbol{v},\boldsymbol{g})}} \\
&\overset{3}{=} \frac{\sum_{\{\boldsymbol{g}|g_k=1\}} e^{-E(\boldsymbol{v},\boldsymbol{g})}}{\sum_{\{\boldsymbol{g}|g_k=1\}} e^{-E(\boldsymbol{v},\boldsymbol{g})} + \sum_{\{\boldsymbol{g}|g_k=0\}} e^{-E(\boldsymbol{v},\boldsymbol{g})}} \\
&= \frac{1}{1 + \frac{\sum_{\{\boldsymbol{g}|g_k=0\}} e^{-E(\boldsymbol{v},\boldsymbol{g})}}{\sum_{\{\boldsymbol{g}|g_k=1\}} e^{-E(\boldsymbol{v},\boldsymbol{g})}}}
\end{aligned}
$$

$$
\begin{aligned}
&\overset{4}{=} \frac{1}{1 + \frac{\sum_{\{\boldsymbol{g}|g_k=0\}} e^{\sum_{i,j} v_i W_{ij} g_j + \sum_i v_i a_i + \sum_j g_j b_j}}{\sum_{\{\boldsymbol{g}|g_k=1\}} e^{\sum_{i,j} v_i W_{ij} g_j + \sum_i v_i a_i + \sum_j g_j b_j}}} \\
&\overset{5}{=} \frac{1}{1 + \frac{\sum_{\{\boldsymbol{g}|g_k=0\}} e^{\sum_i \sum_{j\neq k} v_i W_{ij} g_j + \sum_i v_i a_i + \sum_{j\neq k} g_j b_j}}{\sum_{\{\boldsymbol{g}|g_k=1\}} e^{\sum_i \sum_{j\neq k} v_i W_{ij} g_j + \sum_i v_i a_i + \sum_{j\neq k} g_j b_j}} \cdot \frac{e^{\sum_i v_i W_{ik} \cdot 0 + 0 \cdot b_k}}{e^{\sum_i v_i W_{ik} \cdot 1 + 1 \cdot b_k}}} \\
&\overset{6}{=} \frac{1}{1 + 1 \cdot \frac{1}{e^{\sum_i v_i W_{ik} + b_k}}} \\
&= \frac{1}{1 + e^{-(b_k + \sum_i v_i W_{ik})}} \\
&= \sigma\Big(b_k + \sum_i v_i W_{ik}\Big)
\end{aligned}
$$

Explanation of steps marked with a number:

1. We start from the definition of conditional probability.
2. We substitute the definition of the joint probability density function for an RBM (Equation 4.8, p. 62).
3. We split the sum over all configurations $\boldsymbol{g}$ into two sums over configurations with an active and inactive hidden unit g_k, respectively.
4. We substitute the energy function of the bRBM (Equation 4.10, p. 63).
5. From the sums over j, we move out the terms for $j = k$, for which we substitue the known values of $g_j = g_k$.
6. As the terms depending on g_k have been moved out of the two long exponentials, the sums over $\{\boldsymbol{g}|g_k = 0\}$ and $\{\boldsymbol{g}|g_k = 1\}$ are equal so their quotient evaluates to 1.

For the Bernoulli-Bernoulli RBM regarded here, the conditional probabilities of the visible units can be similarly derived to

$$P(v_k = 1|\boldsymbol{h}) = \sigma\Big(a_k + \sum_j W_{kj} h_j\Big).$$

A.2 Likelihood Gradient

In order to train an RBM by maximum likelihood, we need to derive the gradient of the likelihood function with respect to each model parameter. Instead of optimizing

for the likelihood (Equation 4.15, p. 64), we can optimize for the log likelihood, which is equivalent (because the log function is a stricly increasing function), but simplifies the derivation. Here we exemplarily show how to derive the gradient of the log likelihood of the binary RBM with respect to a connection weight W_{ij}. Again, some of the steps are numbered and explained further below.

$$
\begin{aligned}
&\frac{\partial}{\partial W_{ij}} \sum_{\boldsymbol{v} \in T} \log p(\boldsymbol{v}|\boldsymbol{\theta}) \\
\overset{1}{=} &\frac{\partial}{\partial W_{ij}} \sum_{\boldsymbol{v} \in T} \log \left(\frac{\sum_{\boldsymbol{g}} e^{-E(\boldsymbol{v},\boldsymbol{g},\boldsymbol{\theta})}}{Z(\boldsymbol{\theta})} \right) \\
= &\frac{\partial}{\partial W_{ij}} \sum_{\boldsymbol{v} \in T} \left(\log \sum_{\boldsymbol{g}} e^{-E(\boldsymbol{v},\boldsymbol{g},\boldsymbol{\theta})} - \log Z(\boldsymbol{\theta}) \right) \\
= &\sum_{\boldsymbol{v} \in T} \left(\frac{\partial}{\partial W_{ij}} \log \sum_{\boldsymbol{g}} e^{-E(\boldsymbol{v},\boldsymbol{g},\boldsymbol{\theta})} - \frac{\partial}{\partial W_{ij}} \log Z(\boldsymbol{\theta}) \right) \\
\overset{2}{=} &\sum_{\boldsymbol{v} \in T} \left(\frac{\frac{\partial}{\partial W_{ij}} \sum_{\boldsymbol{g}} e^{-E(\boldsymbol{v},\boldsymbol{g},\boldsymbol{\theta})}}{\sum_{\boldsymbol{g}} e^{-E(\boldsymbol{v},\boldsymbol{g},\boldsymbol{\theta})}} - \frac{\frac{\partial}{\partial W_{ij}} Z(\boldsymbol{\theta})}{Z(\boldsymbol{\theta})} \right) \\
\overset{3}{=} &\sum_{\boldsymbol{v} \in T} \left(\frac{\sum_{\boldsymbol{g}} \left(e^{-E(\boldsymbol{v},\boldsymbol{g},\boldsymbol{\theta})} \cdot \frac{\partial}{\partial W_{ij}} (-E(\boldsymbol{v},\boldsymbol{g},\boldsymbol{\theta})) \right)}{\sum_{\boldsymbol{g}} e^{-E(\boldsymbol{v},\boldsymbol{g},\boldsymbol{\theta})}} - \frac{\sum_{\boldsymbol{u},\boldsymbol{g}} \left(e^{-E(\boldsymbol{u},\boldsymbol{g},\boldsymbol{\theta})} \cdot \frac{\partial}{\partial W_{ij}} (-E(\boldsymbol{u},\boldsymbol{g},\boldsymbol{\theta})) \right)}{Z(\boldsymbol{\theta})} \right) \\
\overset{4}{=} &\sum_{\boldsymbol{v} \in T} \left(\frac{\sum_{\boldsymbol{g}} \left(e^{-E(\boldsymbol{v},\boldsymbol{g},\boldsymbol{\theta})} \cdot v_i g_j \right)}{\sum_{\boldsymbol{g}} e^{-E(\boldsymbol{v},\boldsymbol{g},\boldsymbol{\theta})}} - \frac{\sum_{\boldsymbol{u},\boldsymbol{g}} \left(e^{-E(\boldsymbol{u},\boldsymbol{g},\boldsymbol{\theta})} \cdot u_i g_j \right)}{Z(\boldsymbol{\theta})} \right) \\
= &\sum_{\boldsymbol{v} \in T} \left(\sum_{\boldsymbol{g}} \left(\frac{e^{-E(\boldsymbol{v},\boldsymbol{g},\boldsymbol{\theta})}}{\sum_{\boldsymbol{g}} e^{-E(\boldsymbol{v},\boldsymbol{g},\boldsymbol{\theta})}} \cdot v_i g_j \right) - \sum_{\boldsymbol{u},\boldsymbol{g}} \left(\frac{e^{-E(\boldsymbol{u},\boldsymbol{g},\boldsymbol{\theta})}}{Z(\boldsymbol{\theta})} \cdot u_i g_j \right) \right) \\
= &\sum_{\boldsymbol{v} \in T} \left(\sum_{\boldsymbol{g}} p(\boldsymbol{g}|\boldsymbol{v},\boldsymbol{\theta}) \cdot v_i g_j - \sum_{\boldsymbol{u},\boldsymbol{g}} p(\boldsymbol{u},\boldsymbol{g}|\boldsymbol{\theta}) \cdot u_i g_j \right) \\
= &\sum_{\boldsymbol{v} \in T} \left(v_i \langle h_j \rangle_{p(\boldsymbol{h}|\boldsymbol{v},\boldsymbol{\theta})} \right) - |T| \left(\langle v_i h_j \rangle_{p(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{\theta})} \right)
\end{aligned}
$$

Explanation of steps marked with a number:

1. We substitute the definition of $p(\boldsymbol{v}|\boldsymbol{\theta})$ (Equation 4.11, p. 63).
2. We derive the log functions, applying the chain rule of differentiation.
3. We substitute the definition of the partition function $Z(\boldsymbol{\theta})$ (Equation 4.9, p. 62), then derive the exponential functions, again applying the chain rule.
4. We substitute and derive the bRBM energy function (Equation 4.10, p. 63).

Bibliography

[Abd02] Samer A. Abdallah. *Towards Music Perception by Redundancy Reduction and Unsupervised Learning in Probabilistic Models*. PhD thesis, King's College London, London, UK, 2002.

[ADP07] J.-J Aucouturier, B. Defreville, and F. Pachet. The bag-of-frame approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music. *Journal of the Acoustical Society of America*, 122(2):881–891, 2007.

[AF04] J.-J. Aucouturier and Pachet F. Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

[And06] Christian Anderson. *The Long Tail: Why the Future of Business is Selling Less of More*. Hyperion, 2006.

[ANR74] N. Ahmed, T. Natarajan, and K.R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, Jan 1974.

[AP02a] Jean-Julien Aucouturier and François Pachet. Finding songs that sound the same. In *Proceedings of the 1st IEEE Benelux Workshop on Model based Processing and Coding of Audio (MPCA-2002)*, Leuven, Belgium, Nov 2002.

[AP02b] Jean-Julien Aucouturier and François Pachet. Music similarity measures: What's the use? In *Proceedings of the 3rd International Symposium on Music Information Retrieval (ISMIR 2002)*, pages 157–163, Paris, France, Oct 2002.

[AP08] J.-J. Aucouturier and F. Pachet. A scale-free distribution of false positives for a large class of audio similarity measures. *Pattern Recognition*, 41(1):272–284, 2008.

[ASH09] V. Akkermans, J. Serrà, and P. Herrera. Shape-based spectral contrast descriptor. In *Proceedings of the 6th Sound and Music Computing Conference (SMC 2009)*, pages 143–148, Porto, Portugal., 2009.

[Auc06] Jean-Julien Aucouturier. *Dix Expériences sur la Modélisation du Timbre Polyphonique*. PhD thesis, University of Paris 6, Paris, France, May 2006.

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: The Advantages of Careful Seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'07)*, pages 1027–1035, New Orleans, Lousiana, 2007.

[BB95] Léon Bottou and Yoshua Bengio. Convergence Properties of the K-Means Algorithms. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen, editors, *Advances in Neural Information Processing Systems 7 (NIPS 1994)*, pages 585–592. MIT Press, Cambridge, MA, 1995.

[BCTL07] Luke Barrington, Antoni Chan, Douglas Turnbull, and Gert Lanckriet. Audio Information Retrieval using Semantic Similarity. In *Proceedings of the 2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'07)*, volume II, pages 725–728, Honolulu, HI, 2007.

[BD09] Yoshua Bengio and Olivier Delalleau. Justifying and Generalizing Contrastive Divergence. *Neural Computation*, 21:1601–1621, June 2009.

[Bei11] Homayoon Beigi. Speaker recognition. In Jucheng Yang, editor, *Biometrics*. InTech, 2011.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18:509–517, September 1975.

[Ber06] Adam Berenzweig. *Anchors and hubs in audio-based music similarity*. PhD thesis, Columbia University, NY, USA, Oct 2006.

[BHT63] B. Bogert, M. Healy, and J. Tukey. The quefrency alanysis of time series for echoes: cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking. In M. Rosenblatt, editor, *Proceedings of the Symposium on Time Series Analysis*, pages 209–243, New York, 1963. Wiley.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 6 edition, 2006.

[BLEW03] Adam Berenzweig, Beth Logan, Daniel P.W. Ellis, and Brian Whitman. A Large-Scale Evaluation of Acoustic and Subjective Music Similarity Measures. In *4th International Society of Music Information Retrieval Conference (ISMIR 2003)*, Baltimore, Maryland, USA, 2003.

[BLPL07] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, pages 153–160. MIT Press, Cambridge, MA, 2007.

[BMEWL11] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011, to appear)*, 2011.

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[BO99] Tolga Bozkaya and Meral Ozsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Trans. Database Syst.*, 24:361–404, Sept 1999.

[BOTL09] Luke Barrington, Damien O'Malley, Douglas Turnbull, and Gert Lanckriet. User-Centered Design of a Social Game to Tag Music. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP '09)*, pages 7–10, Paris, France, 2009.

[BP03] Stephan Baumann and Tim Pohle. A comparison of music similarity measures for a p2p application. In *Proceedings of the 6th International Conference on Digital Audio Effects(DAFx-03)*, 2003.

[BPL10] Y-Lan Boureau, Jean Ponce, and Yann LeCun. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)*, pages 111–118, Haifa, Israel, 2010.

[BS81] Dwight Le Merton Bolinger and Donald A. Sears. *Aspects of Language*. Harcourt Brace Jovanovich, New York, NY, USA, 1981.

[BS97] A.J. Bell and T.J. Sejnowski. The "Independent Components" of Natural Scenes are Edge Filters. *Vision research*, 37(23):3327–3338, 1997.

[Bur03] Juan José Burred. An objective approach to content-based audio signal classification. Master's thesis, Technische Universität Berlin, Germany, 2003.

[Bur08] Juan José Burred. *From Sparse Models to Timbre Learning: New Methods for Musical Source Separation.* PhD thesis, Technical University of Berlin, Berlin, Germany, Sept 2008.

[BYR06] V. Britanak, P.C. Yip, and K.R. Rao. *Discrete Cosine and Sine Transforms: General Properties, Fast Algorithms and Integer Approximations.* Academic Press, 2006.

[BZ01] Pau Bofill and Michael Zibulevsky. Underdetermined Blind Source Separation Using Sparse Representations. *Signal Processing*, 81(11):2353–2362, 2001.

[Cam98] Emilios Camboropoulos. *Towards a General Computational Theory of Musical Structure.* PhD thesis, University of Edinburgh, Edinburgh, UK, 1998.

[CC08] O. Celma and P. Cano. From Hits to Niches? or How Popular Artists Can Bias Music Recommendation and Discovery. In *2nd Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition (ACM KDD)*, Las Vegas, USA, 2008.

[CEK05] Norman Casagrande, Douglas Eck, and Balázs Kégl. Frame-Level Speech/Music Discrimination using AdaBoost. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, pages 345–350, 2005.

[CLN10] Adam Coates, Honglak Lee, and Andrew Y. Ng. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2010.

[CN11] A. Coates and A. Ng. The Importance of Encoding Versus Training with Sparse Coding and Vector Quantization. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*, 2011.

[CPZ97] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB '97)*, pages 426–435, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[DB08] Guillaume Desjardins and Yoshua Bengio. Empirical Evaluation of Convolutional RBMs for Vision. Technical Report 1327, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, 2008.

[DBC09] J. Stephen Downie, Donald Byrd, and Tim Crawford. Ten Years of ISMIR: Reflections on Challenges and Opportunities. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 13–18, Kobe, Japan, 2009.

[DDL+90] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[DH04] Chris H. Q. Ding and Xiaofeng He. K-means Clustering via Principal Component Analysis. In *Proceedings of the 21st International Conference on Machine Learning (ICML 2004)*, Banff, Alberta, Canada, 2004.

[DJLW08] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Comput. Surv.*, 40:5:1–5:60, May 2008.

[DM80] Steven B. Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, Aug 1980.

[DR93] Boris Doval and Xavier Robet. Fundamental frequency estimation and tracking using maximum likelihood harmonic matching and hmms. In *Proceedings of the 1993 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'93)*, pages 221–224, Washington, DC, USA, 1993.

[DRMH10] George E. Dahl, Marc'Aurelio Ranzato, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Phone recognition with the mean-covariance restricted Boltzmann machine. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23 (NIPS 2010)*, pages 469–477. 2010.

[DSN01] Hrishikesh Deshpande, Rohit Singh, and Unjung Nam. Classification of Music Signals in the Visual Domain. In *Proceedings of the 4th International Conference on Digital Audio Effects (DAFx-01)*, Limerick, Ireland, 2001.

[EBC+10] Dumitru Erhan, Yoshua Bengio, Aaron C. Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why Does Unsupervised Pre-training Help Deep Learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

[ELBMG08] Douglas Eck, Paul Lamere, Thierry Bertin-Mahieux, and Stephen Green. Automatic Generation of Social Tags for Music Recommendation. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, pages 385–392. MIT Press, Cambridge, MA, 2008.

[ES03] Dominik M. Endres and Johannes E. Schindelin. A New Metric for Probability Distributions. *IEEE Transactions on Information Theory*, 49(7):1858–1860, 2003.

[FFP05] Li Fei-Fei and Pietro Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 524–531, 2005.

[Fis09] J. Fiser. The other kind of perceptual learning. *Learning & Perception*, 1(1):69–87, 2009.

[Fle07] Arthur Flexer. A Closer Look on Artist Filters for Musical Genre Classification. In *Proceedings of the 8th International Society of Music Information Retrieval Conference (ISMIR 2007)*, pages 341–344, Vienna, Austria, 2007.

[Foo97] Jonathan T. Foote. Content-based retrieval of music and audio. In Jay C. C. Kuo, Shih F. Chang, and Venkat N. Gudivada, editors, *Multimedia Storage and Archiving Systems II (Proceedings SPIE)*, volume 3229, pages 138–147, 1997.

[FPW05] Arthur Flexer, Elias Pampalk, and Gerhard Widmer. Hidden markov models for spectral similarity of songs. In *Proceedings of the 8th International Conference on Digital Audio Effects (DAFx-05)*, Madrid, Spain, 2005.

[FS10] Arthur Flexer and Dominik Schnitzer. Effects of album and artist filters in audio similarity computed for very large music databases. *Computer Music Journal*, 34(3):20–28, 2010.

[FSGP10] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Tim Pohle. Combining Features Reduces Hubness in Audio Similarity. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 171–176, Utrecht, Netherlands, 2010.

[FSGW08] Arthur Flexer, Dominik Schnitzer, Martin Gasser, and Gerhard Widmer. Playlist Generation Using Start and End Songs. In *Proceedings of the International Symposium on Music Information Retrieval*, pages 173–178, 2008.

[Fur86] S. Furui. Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(1):52–59, 1986.

[FZ07] Hugo Fastl and Eberhard Zwicker. *Psychoacoustics: Facts and Models*. Springer-Verlag Berlin Heidelberg, 2007.

[GDBR+11] Carolyn Granier-Deferre, Sophie Bassereau, Aurélie Ribeiro, Anne-Yvonne Jacquet, and Anthony J. DeCasper. A Melodic Contour Repeatedly Experienced by Human Near-Term Fetuses Elicits a Profound Cardiac Reaction One Month after Birth. *PLoS ONE*, 6(2):e17304, Feb 2011.

[GDPW04] F. Gouyon, S. Dixon, E. Pampalk, and G. Widmer. Evaluating Rhythmic Descriptors for Musical Genre Classification. In *Proceedings of the 25th International Audio Engineering Society (AES) Conference*, London, UK, 2004.

[GHNO02] Masataka Goto, Hiroki Hashiguchi, Takuichi Nishimura, and Ryuichi Oka. RWC Music Database: Popular, Classical, and Jazz Music Databases. In *Proceedings of the 3rd International Society for Music Information Retrieval Conference (ISMIR 2002)*, pages 287–288, 2002.

[GKS07] Gijs Geleijnse, Peter Knees, and Markus Schedl. The Quest for Ground Truth in Musical Artist Tagging in the Social Web Era. In *Proceedings of the 8th International Society of Music Information Retrieval Conference (ISMIR 2007)*, Vienna, Austria, 2007.

[Har87] Frederic J. Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1987.

[HBC08] Matthew Hoffman, David M. Blei, and Perry R. Cook. Content-Based Musical Similarity Computation using the Hierarchical Dirichlet Process. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, pages 349–354, 2008.

[HBC09] M. Hoffman, D. Blei, and P.R. Cook. Finding Latent Sources in Recorded Music With a Shift-Invariant HDP. In *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09)*, Como, Italy, 2009.

[HE10] Philippe Hamel and Douglas Eck. Learning Features from Music Audio with Deep Belief Networks. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 339–344, 2010.

[Her03] Hynek Hermanksi. TRAP-TANDEM: data-driven extraction of temporal features from speech. In *Proceedings of the 2003 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU'03)*, pages 255–260, 2003.

[Hin02] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[Hin10] Geoffrey Hinton. A Practical Guide to Training Restricted Boltzmann Machines. Technical Report UTML TR 2010-003, Dep. of Computer Science, University of Toronto, 2010.

[HKBR99] J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl. An Algorithmic Framework for Performing Collaborative Filtering. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*, pages 230–237. ACM, 1999.

[HKW11] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. Transforming Auto-Encoders. In *Proceedings of the 21st International Conference on Artificial Neural Networks (ICANN 2011)*, volume 6791 of *Lecture Notes in Computer Science*, pages 44–51, Espoo, Finland, 2011. Springer.

[HMM+05] Helge Homburg, Ingo Mierswa, Bülent Möller, Katharina Morik, and Michael Wurst. A Benchmark Dataset for Audio Classification and Clustering. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, pages 528–531, London, UK, 2005.

[Hof99] Thomas Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99)*, pages 50–57, 1999.

[HOT06] G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[HP76] M. Hamidi and J. Pearl. Comparison of the cosine and Fourier transforms of Markov-1 signals. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 24:428–429, 1976.

[HS06] G. Hinton and R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006.

[HS11] Geoffrey Hinton and Ruslan Salakhutdinov. Discovering Binary Codes for Documents by Learning Deep Generative Models. *Topics in Cognitive Science*, 3(1):74–91, 2011.

[HWE09] Philippe Hamel, Sean Wood, and Douglas Eck. Automatic identification of instrument classes in polyphonic and poly-instrument audio. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 399–404, 2009.

[JCEJ09] Jesper Højvang Jensen, Mads Græsbøll Christensen, Daniel P.W. Ellis, and Søren Holdt Jensen. Quantitative Analysis of a Common Audio Similarity Measure. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):693–703, 2009.

[JH11] Navdeep Jaitly and Geoffrey Hinton. Learning a better Representation of Speech Sound Waves using Restricted Boltzmann Machines. In *Proceedings of the 36th International Conference on Acoustics, Speech and Signal Processing (ICASSP 2011)*, 2011.

[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 2002 IEEE International Conference on Multimedia and Expo (ICME '02)*, pages 113–116, 2002.

[JR91] B.H. Juang and L.R. Rabiner. Hidden Markov Models for Speech Recognition. *Technometrics*, 33(3), 1991.

[JT05] Frederic Jurie and Bill Triggs. Creating Efficient Codebooks for Visual Recognition. In *Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV'05)*, pages 604–610, 2005.

[KH11] Alex Krizhevsky and Geoffrey E. Hinton. Using Very Deep Autoencoders for Content-Based Image Retrieval. In *Proceedings of the 19th European Symposium on Artificial Neural Networks (ESANN 2011)*, Bruges, Belgium, 2011.

[KL51] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[Knu92] Donald E. Knuth. Two notes on notation. *The American Mathematical Monthly*, 99(5):403–422, 1992.

[Kri09] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Dept. of Comp. Science, Univ. of Toronto, 2009.

[Kro05] Ellyssa Kroski. The Hive Mind: Folksonomies and User-Based Tagging. `http://infotangle.blogsome.com/2005/12/07/the-hive-mind-folksonomies-and-user-based-tagging/`, Dec 2005. Retrieved Aug 31, 2011.

[KSE08] Youngmoo E. Kim, Erik Schmidt, and Lloyd Emelle. MoodSwings: A Collaborative Game for Music Mood Label Collection. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, pages 231–236, 2008.

[Lam06] Paul Lamere. What's on your iPod? `http://blogs.oracle.com/plamere/entry/what_s_on_your_ipod/`, May 2006. Retrieved Aug 31, 2011.

[LBBH98] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[LBG80] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

[LCH+06] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*. MIT Press, 2006.

[Lew98] David Lewis. Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval. In Claire Nédellec and Céline Rouveirol, editors, *Machine Learning: ECML-98*, volume 1398 of *Lecture Notes in Computer Science*, pages 4–15. Springer Berlin / Heidelberg, 1998.

[Lew00] M. S. Lewicki. Learning efficient codes of natural sounds yields cochlear filter properties. In *International Conference on Neural Information Processing*, 2000.

[LFZ09] Kaipeng Liu, Binxing Fang, and Yu Zhang. Detecting Tag Spam in Social Tagging Systems with Collaborative Knowledge. In *Proceedings of the Sixth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 09)*, volume 7, pages 427–431, 2009.

[LGRN09] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*, pages 609–616, Montreal, Quebec, Canada, 2009.

[Lin91] Jianhua Lin. Divergence Measures Based on the Shannon Entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[LLGS99] Te-Won Lee, Michael S. Lewicki, Mark Girolami, and Terrence J. Sejnowski. Blind Source Separation of More Sources Than Mixtures Using Overcomplete Representations. *IEEE Signal Processing Letters*, 6(4), 1999.

[Llo82] Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[LLPN09] Honglak Lee, Yan Largman, Peter Pham, and Andrew Y. Ng. Unsupervised Feature Learning for Audio Classification using Convolutional Deep Belief Networks. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 1096–1104. 2009.

[LLT05] Aristomenis S. Lampropoulos, Paraskevi S. Lampropoulou, and George A. Tsihrintzis. Musical Genre Classification Enhanced by Improved Source Separation Technique. In *Proceedings of the 6th International Society of Music Information Retrieval Conference (ISMIR 2005)*, pages 576–581, 2005.

[Log00] Beth Logan. Mel Frequency Cepstral Coefficients for Music Modeling. In *Proceedings of the 1st International Symposium on Music Information Retrieval (ISMIR 2000)*, Plymouth, Massachusetts, USA, 2000.

[Log02] B. Logan. Content-based Playlist Generation: Exploratory Experiments. In *Proceedings of the 3rd International Symposium on Music Information Retrieval (ISMIR 2002)*, 2002.

[Low04] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[LR05] Thomas Lidy and Andreas Rauber. Evaluation of Feature Extractors and Psycho-Acoustic Transformations for Music Genre Classification. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, 2005.

[LS01] Beth Logan and Ariel Salomon. A music similarity function based on signal analysis. In *Proceedings of the 2001 IEEE International Conference on Multimedia and Expo (ICME 2001)*, 2001.

[LS06] Mark Levy and Mark 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.

[LSMW05] Xiao-Bing Li, Frank K. Soong, Tor André Myrvoll, and Ren-Hua Wang. Optimal Clustering and Non-Uniform Allocation of Gaussian Kernels in Scalar Dimension for HMM Compression. In *Proceedings of the 2005 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'05)*, pages 669–672, 2005.

[LSP06] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pages 2169–2178, 2006.

[LvA09] Edith Law and Luis von Ahn. Input-Agreement: A New Mechanism for Collecting Data Using Human Computation Games. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems (CHI 2009)*, pages 1197–1206, Boston, MA, USA, 2009.

[Mar61] M.E. Maron. Automatic indexing: an experimental inquiry. *Journal of the Association for Computing Machinery*, 8(3):404–417, 1961.

[MDH09] Abdel-rahman Mohamed, George Dahl, and Geoffrey Hinton. Deep Belief Networks for Phone Recognition. In *NIPS 22 Workshop on Deep Learning for Speech Recognition*, 2009.

[ME05] M. Mandel and D. Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, pages 594–599, 2005.

[ME07] Michael I. Mandel and Daniel P.W. Ellis. A web-based game for collecting music metadata. In *Proceedings of the 8th International Society for Music Information Retrieval Conference (ISMIR 2007)*, pages 365–366, 2007.

[MEDL09] F. Maillet, D. Eck, G. Desjardins, and P. Lamere. Steerable Playlist Generation by Learning Song Similarity from Radio Station Playlists. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[MEF+10] B. Mathieu, S. Essid, T. Fillon, J. Prado, and G. Richard. Yaafe, an easy to use and efficient audio feature extraction software. In *Proceedings of the 11th International Society of Music Information Retrieval Conference (ISMIR 2010)*, pages 441–445, 2010.

[MH10] Roland Memisevic and Geoffrey E. Hinton. Learning to Represent Spatial Transformations with Factored Higher-Order Boltzmann Machines. *Neural Computation*, 22:1473–1492, 2010.

[Mit04] Nikolaos Mitianoudis. *Audio Source Separation using Independent Component Analysis*. PhD thesis, Queen Mary, University of London, 2004.

[MLG+11] G. Marques, T. Langlois, F. Gouyon, M. Lopes, and M. Sordo. Short-term feature space and music genre classification. *Journal of New Music Research*, 40:127–137, 2011.

[MMF06] C. McKay, D. McEnnis, and I. Fujinaga. A large publicly accessible prototype audio database for music research. In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, pages 160–163, 2006.

[Mni09] Volodymyr Mnih. CUDAMat: A CUDA-based matrix class for Python. Technical Report UTML TR 2009-04, Department of Computer Science, University of Toronto, 2009.

[Moo02] Brian C.J. Moore. Interference effects and phase sensitivity in hearing. *Philosophical Transactions*, 360(1794):833–58, 2002.

[Nav09] Roberto Navigli. Word Sense Disambiguation: A Survey. *ACM Computing Surveys*, 41:10:1–10:69, Feb 2009.

[NCKN11] Jiquan Ngiam, Zhenghao Chen, Pangwei Koh, and Andrew Y. Ng. Learning Deep Energy Models. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*, 2011.

[NH09] Vinod Nair and Geoffrey Hinton. 3d object recognition with deep belief nets. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 1339–1347. 2009.

[NRM09] Mohammad Norouzi, Mani Ranjbar, and Greg Mori. Stacks of Convolutional Restricted Boltzmann Machines for Shift-Invariant Feature Learning. In *Proceedings of the 22th IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*, pages 2735–2742, Miami, FL, USA, 2009.

[O+96] B.A. Olshausen et al. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[OH08] Simon Osindero and Geoffrey Hinton. Modeling image patches with a directed hierarchy of markov random fields. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, pages 1121–1128. MIT Press, Cambridge, MA, 2008.

[Ohm43] Georg Simon Ohm. Über die Definition des Tones, nebst daran geknüpfter Theorie der Sirene und ähnlicher tonbildender Vorrichtungen. In J.C. Poggendorff, editor, *Annalen der Physik und Chemie*, volume 59, pages 513–565, Leipzig, 1843.

[OT06] Aude Oliva and Antonio Torralba. Building the Gist of a Scene: The Role of Global Image Features in Recognition. *Progress in Brain Research*, 155:23–26, 2006.

[Pam06] Elias Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval*. PhD thesis, Vienna University of Technology, Vienna, Austria, March 2006.

[PBK09] Ioannis Panagakis, Emmanouil Benetos, and Constantine Kotropoulos. Music Genre Classification - A Multilinear Approach. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[PC00] François Pachet and Daniel Cazaly. A taxonomy of musical genres. In *Proceedings of the 6th Content-Based Multimedia Information Access Conference (RIAO 2000)*, 2000.

[PD76] R.L. Pratt and P.E. Doak. A subjective rating scale for timbre. *Journal of Sound and Vibration*, 45(3):317–328, 1976.

[PDW03] E. Pampalk, S. Dixon, and G. Widmer. On the evaluation of perceptual similarity measures for music. In *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)*, 2003.

[Pea01] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 6(2):559–572, 1901.

[Pee04] G. Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. Technical report, IRCAM, 2004.

[PFW05] Elias Pampalk, Arthur Flexer, and Gerhard Widmer. Improvements of Audio-based Music Similarity and Genre Classification. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, London, UK, 2005.

[PKSW06] T. Pohle, P. Knees, M. Schedl, and G. Widmer. Independent Component Analysis for Music Similarity Computation. In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, pages 228–233, 2006.

[PKSW10] T. Pohle, P. Knees, K. Seyerlehner, and G. Widmer. A High-Level Audio Feature For Music Retrieval and Sorting. In *Proceedings of the 13th International Conference on Digital Audio Effects (DAFx-10)*, Graz, Austria, 2010.

[Poh10] Tim Pohle. *Automatic Characterization of Music for Intuitive Retrieval*. PhD thesis, Johannes Kepler University, Linz, Austria, Jan 2010.

[PPW05] Tim Pohle, Elias Pampalk, and Gerhard Widmer. Generating Similarity-based Playlists Using Traveling Salesman Algorithms. In *Proceedings of the 8th International Conference on Digital Audio Effects (DAFx-05)*, Madrid, Spain, 2005.

[PRM02] Elias Pampalk, Andreas Rauber, and Dieter Merkl. Content-based Organization and Visualization of Music Archives. In *Proceedings of the 10th ACM International Conference on Multimedia*, pages 570–579, 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 Internationel Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 525–530, 2009.

[RBL08] Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse Feature Learning for Deep Belief Networks. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS 2007)*, pages 1185–1192, Cambridge, MA, 2008. MIT Press.

[RH10] M. Ranzato and G. Hinton. Modeling Pixel Means and Covariances Using Factorized Third-Order Boltzmann Machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10)*, pages 2551–2558, 2010.

[RHG08] Matthew Riley, Eric Heinen, and Joydeep Ghosh. A text retrieval approach to content-based audio retrieval. In *Proceedings of the 7th International Society of Music Information Retrieval Conference (ISMIR 2008)*, pages 295–300, 2008.

[RJ93] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Upper Saddle River, NJ, USA, 1993.

[RKH10] Marc'Aurelio Ranzato, Alex Krizhevsky, and Geoffrey E. Hinton. Factored 3-Way Restricted Boltzmann Machines For Modeling Natural Images. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[RMH10] Marc'Aurelio Ranzato, Volodymyr Mnih, and Geoffrey Hinton. Generating more realistic images using gated MRF's. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23 (NIPS 2010)*, pages 2002–2010. 2010.

[RMT+10] Halfdan Rump, Shigeki Miyabe, Emiru Tsunoo, Nobukata Ono, and Shigeki Sagama. Autoregressive MFCC Models for Genre Classification Improved by Harmonic-Percussion Separation. In *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010.

[RNI10] Milos Radovanovic, Alexandros Nanopoulos, and Mirjana Ivanovic. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11:2487–2531, 2010.

[Rob94] Tony Robinson. An Application of Recurrent Nets to Phone Probability Estimation. *IEEE Transactions on Neural Networks*, 5(2):298–305, 1994.

[RS78] Lawrence Rabiner and Ronald Schafer. *Digital processing of speech signals*. Prentice Hall, Englewood Cliffs, NJ, USA, 1978.

[RSMH11] Marc'Aurelio Ranzato, Joshua Susskind, Volodymyr Mnih, and Geoffrey E. Hinton. On deep generative models with applications to recognition. In *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR'11)*, pages 2857–2864, Colorado Springs, CO, USA, 2011.

[Sal09] Ruslan Salakhutdinov. *Learning Deep Generative Models*. PhD thesis, Dept. of Comp. Science, Univ. of Toronto, 2009.

[SB88] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[Sca08] Nicolas Scaringella. Timbre and rhythmic trap-tandem features for music information retrieval. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, pages 626–632, 2008.

[Sca09] Nicolas Scaringella. *On the Design of Audio Features Robust to the Album-Effect for Music Information Retrieval*. PhD thesis, École Polytechnique Fédérale de Lausanne, Suisse, 2009.

[Scu10] D. Sculley. Web-Scale K-Means Clustering. In *Proceedings of the 19th International Conference on World Wide Web (WWW'10)*, pages 1177–1178, New York, NY, USA, 2010. ACM.

[Seb02] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[Sey10] Klaus Seyerlehner. *Content-Based Music Recommender Systems: Beyond simple Frame-Level Audio Similarity*. PhD thesis, Johannes Kepler University, Linz, Austria, Dec 2010.

[SFSW11] D. Schnitzer, A. Flexer, M. Scheld, and G. Widmer. Using Mutual Proximity to Improve Content-Based Audio Similarity. In *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011, to appear)*, 2011.

[SFW09] Dominik Schnitzer, Arthur Flexer, and Gerhard Widmer. A filter-and-refine indexing method for fast similarity search in millions of music tracks. In *Proceedings of the 10th International Society of Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[SFW11] Dominik Schnitzer, Arthur Flexer, and Gerhard Widmer. A fast audio similarity retrieval method for millions of music tracks. *Multimedia Tools and Applications (to appear)*, 2011.

[SGHS08] J. Serrà, E. Gómez, P. Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech and Language Processing*, 16:1138–1151, 2008.

[SGP+95] V. Sanchez, P. Garcia, A.M. Peinado, J.C. Segura, and A.J. Rubio. Diagonalizing properties of the discrete cosine transforms. *IEEE Transactions on Signal Processing*, 43:2631–2641, 1995.

[SH09a] R. Salakhutdinov and G. Hinton. Deep Boltzmann Machines. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, pages 448–455, 2009.

[SH09b] Ruslan Salakhutdinov and Geoffrey Hinton. Replicated softmax: an undirected topic model. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22 (NIPS 2009)*, pages 1607–1614. 2009.

[SH09c] Ruslan Salakhutdinov and Geoffrey Hinton. Semantic Hashing. *International Journal of Approximative Reasoning*, 50:969–978, Jul 2009.

[SL10] Ruslan Salakhutdinov and Hugo Larochelle. Efficient Learning of Deep Boltzmann Machines. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, pages 693–700, 2010.

[Smo86] P. Smolensky. *Information Processing in Dynamical Systems: Foundations of Harmony Theory*, pages 194–281. MIT Press, Cambridge, MA, USA, 1986.

[SO06] Mikkel N. Schmidt and Rasmus K. Olsson. Single-Channel Speech Separation using Sparse Non-Negative Matrix Factorization. In *Proceedings of the 9th International Conference on Spoken Language Processing (Interspeech 2006)*, 2006.

[SPLS06] Sigurdur Sigurdsson, Kaare Brandt Petersen, and Tue Lehn-Schiøler. Mel Frequency Cepstral Coefficients: An Evaluation of Robustness of MP3 Encoded Music. In *Proceedings of the 7th International Society for Music Information Retrieval Conference (ISMIR 2006)*, 2006.

[SPWS09] Klaus Seyerlehner, Tim Pohle, Gerhard Widmer, and Dominik Schnitzer. Informed selection of frames for music similarity computation. In *Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09)*, Como, Italy, 2009.

[SRE+05] Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. Discovering Objects and their Localization in Images. In *Proceedings of the 10th IEEE International Conference on Computer Vision (ICCV'05)*, volume 1, pages 370–377, 2005.

[SVN37] S. S. Stevens, J. Volkmann, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America*, 8(3):185–190, 1937.

[SWK08] Klaus Seyerlehner, Gerhard Widmer, and Peter Knees. Frame-level Audio Similarity – A Codebook Approach. In *Proceedings of the 11th International Conference on Digital Audio Effects (DAFx-08)*, Espoo, Finland, 2008.

[SWK09] Erik M. Schmidt, Kris West, and Youngmoo E. Kim. Efficient acoustic feature extraction for music information retrieval using programmable gate arrays. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[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)*, Graz, Austria, 2010.

[SWW08] M. Slaney, K. Weinberger, and W. White. Learning a Metric for Music Similarity. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, pages 313–318, 2008.

[SZ05] Nicolas Scaringella and Giorgio Zoia. On the modeling of time information for automatic genre recognition systems in audio signals. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, pages 666–671, 2005.

[TBTL07] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Towards musical query-by-semantic description using the CAL500 data set. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*, pages 439–446, 2007.

[TC02] George Tzanetakis and Perry R. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[Tie08] Tijmen Tieleman. Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 1064–1071. ACM New York, NY, USA, 2008.

[TJBB06] Yee Whye Teh, Michael I. Jordan, Matthew J. Beal, and David M. Blei. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1566–1581, Dec 2006.

[vdM09] Laurens van der Maaten. Learning a parametric embedding by preserving local structure. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 384–391, 2009.

[vGGVS08] Jan van Gemert, Jan-Mark Geusebroek, Cor J. Veenman, and Arnold W. M. Smeulders. Kernel codebooks for scene categorization. In *Proceedings of the 10th European Conference on Computer Vision (ECCV 2008)*, volume 5304 of *Lecture Notes in Computer Science*, pages 696–709. Springer, 2008.

[Vir07] Tuomas Virtanen. Monaural Sound Source Separation by Nonnegative Matrix Factorization with Temporal Continuity and Sparseness Criteria. *IEEE Transactions on Audio, Speech, and Language Processing*, pages 1066–1074, 2007.

[VK02] Tuomas Virtanen and Anssi Klapuri. Separation of Harmonic Sounds Using Linear Models for the Overtone Series. In *Proceedings of the 2002 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'02)*, volume II, pages 1757–1760, Orlando, FL, USA, 2002.

[Vog04] Harold L. Vogel. *Entertainment Industry Economics: A Guide for Financial Analysis*. Cambridge University Press, 6 edition, 2004.

[VP05] Fabio Vignoli and Steffen Pauws. A music retrieval system based on user driven similarity and its evaluation. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, pages 272–279, 2005.

[Wal06] Hanna M. Wallach. Topic modeling: Beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, pages 977–984. ACM, 2006.

[Wan03] Avery Li-Chun Wang. An Industrial-Strength Audio Search Algorithm. In *4th International Society of Music Information Retrieval Conference (ISMIR 2003)*, Baltimore, Maryland, USA, 2003.

[WC04] Kristopher West and Stephen Cox. Features and classifiers for the automatic classification of musical audio signals. In *Proceedings of the 5th International Society for Music Information Retrieval Conference (ISMIR 2004)*, 2004.

[WC05] Kris West and Stephen Cox. Finding an optimal segmentation for audio genre classification. In *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, pages 680–685, 2005.

[WL07] Kris West and Paul Lamere. A Model-Based Approach to Constructing Music Similarity Functions. *EURASIP Journal on Advances in Signal Processing*, 2007(1):149–158, 2007.

[XZ08] Linxing Xiao and Jie Zhou. Using chroma histogram to measure the perceptual similarity of music. In *Proceedings of the 2008 IEEE International Conference on Multimedia and Expo (ICME 2008)*, pages 1317–1320, 2008.

[YS09] Guoshen Yu and Jean-Jacques Slotine. Audio Classification from Time-Frequency Texture. In *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'09)*, pages 1677–1680, 2009.

[ZRZ02] Lei Zhu, Al Bing Rao, and Aldong Zhang. Theory of Keyblock-based Image Retrieval. *ACM Transactions on Information Systems*, 20(2):224–257, Apr 2002.