UNIVERSITY OF CALIFORNIA, SAN DIEGO

Probabilistic Topic Models for Automatic Harmonic Analysis of Music

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Diane J. Hu

Committee in charge:

Professor Lawrence Saul, Chair
Professor Serge Belongie
Professor Shlomo Dubnov
Professor Charles Elkan
Professor Gert Lanckriet

2012

The dissertation of Diane J. Hu is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2012

DEDICATION

To my mom and dad
who inspired me to work hard....
and go to grad school.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Lawrence Saul, for many things: First, for accepting me as his student (as he had many to choose from) and helping me reach my academic goals; second, for teaching me all things research, writing, and presentation related; and third, for encouraging me and discouraging me from all the right things at the right time. Although he is highly regarded in the research community for his work and brilliance, he is still one of the most hard-working, humble, flexible, and good-natured people I know. It is easy to look to him as an example of a great person and academic.

I would like to thank my "big brothers" Kilian Weinberger and Fei Sha who I greatly admire. They were always fun to be around and never failed to make me laugh (or, at least amused). I thank them for their kindness, humor, wisdom and support. I appreciate all of their honest advice on everything, from research, to careers, to just life in general.

I am grateful to Xiaofeng Ren (Intel Research) and Sudarshan Lamkhede (Yahoo! Labs), who accepted me as an intern for the summers of 2010 and 2011, respectively. Both experiences gave me invaluable hands-on experience for how machine learning was used in the "real-world", as well as the skills needed to impress companies during my post-graduate job-hunting process. I am also grateful for the opportunity to work with Professor Carol Krumhansl, and want to thank her for her time and insight while on sabbatical at UCSD. Her passion for research and love for life was a huge inspiration to me.

I also had many friends in the department who offered their ideas, knowledge, support, and friendship. They made grad school fun and so much more bearable. Some of these people include: Chih-Chieh Cheng, Shibin Parameswaran, Justin Ma, Alvin AuYoung, Daniel Hsu, Laurens van der Maaten, Brian McFee, Kaisen Lin, Elizabeth Bales, Youngmin Cho, Do-kyum Kim, Shankar Shivappa, and Kevin Li.

I also couldn't have made it through these six years without amazing friends from outside of the department as well. They encouraged me, prayed for me, kept me sane and well-adjusted, and always reminded me that there was so much more

to life than the narrow-lenses that I would often look through. I owe a lot to my best bud Hoang Nhan, the Ethnos Community, my mentors Yucan Chiu and Angela Bortone, my small group from Graduate Christian Fellowship, my brother Matt, and my Musketeers (Christine, Cyndi, and Ophelia) from back home.

I am also indebted to my parents, who taught me the virtues of hard work and setting goals from a young age. They never missed an opportunity to help build my academic and artistic interests. When I was young, they bought me my very first sketch book, sat me on a piano, and got me addicted to legos and model trains: my first forays into art, music, and engineering – three subjects that I am still extremely passionate about today.

Finally, I'd like to thank my husband, Robert – my favorite person in the world! I am so thankful for his love and his desire for me to be happy and follow my dreams. He continues to be supportive, even when it sometimes means putting his own dreams on hold. He has greatly enriched my life with unending love, laughter, fun, and encouragement.

Chapter 3, in part, is a reprint of the material as it appears in Proceedings of the International Conference on Music Information Retrieval (ISMIR) 2009. Hu, Diane; Saul, Lawrence K. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in part, is a reprint of the material as it appears in Proceedings of the Neural Information Processing Systems (NIPS) Workshop on Topic Modeling 2009. Hu, Diane; Saul, Lawrence K. The dissertation author was the primary investigator and author of this paper.

VITA

2006	B.S. in Computer Science *cum laude*, University of Washington, Seattle
2009	M.S. in Computer Science, University of California, San Diego
2012	Ph.D. in Computer Science, University of California, San Diego

PUBLICATIONS

D. J. Hu, L. Bo, X. Ren, A joint framework for material and object recognition, *British Machine Vision Conference (BMVC)*, Dundee, Scotland, 2011.

D. J. Hu, L.P. van der Maaten, Y. Cho, L. K. Saul, S. Lerner, Latent variable models for predicting file dependencies in large-scale software development, *Neural Information Processing Systems (NIPS)*, Vancouver, Canada, 2010.

D. J. Hu, L. K. Saul, A topic model for audio and symbolic music analysis, *Neural Information Processing Systems (NIPS) Workshop on Topic Modeling*, Whistler, Canada, 2009.

D. J. Hu, L. K. Saul, A probabilistic topic model for unsupervised learning of musical key-profiles, *International Conference on Music Information Retrieval (ISMIR)*, Kobe, Japan, 2009.

C. Cheng, D. J. Hu, L. K. Saul, Nonnegative matrix factorization for real-time musical analysis and sight-reading evaluation, *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Las Vegas, NV, 2008

ABSTRACT OF THE DISSERTATION

Probabilistic Topic Models for Automatic Harmonic Analysis of Music

by

Diane J. Hu

Doctor of Philosophy in Computer Science

University of California, San Diego, 2012

Professor Lawrence Saul, Chair

Though music is an art form with tremendous expressive capacity, the underlying frameworks of most musical compositions are highly structured and organized. As such, musical pieces are commonly studied by analyzing their melodic and harmonic structures. Two important concepts in any such analysis are the *key* and *tonic*. The key of a musical piece identifies a principal set of pitches that the composer uses to build its melodies and harmonies; the tonic is the most stable pitch in this set. Each musical piece is characterized by one overall key. However, the key can be shifted within a piece by a compositional technique known as modulation. Notwithstanding the infinite number of variations possible in music, most pieces can be analyzed in these terms.

In this dissertation, we propose novel methods for modeling a corpus of musical pieces in terms of its harmonic structure. We describe several different models for symbolic and audio musical input, as we find both to be valuable data formats to perform harmonic analysis on. Our models are based on Latent Dirichlet Allocation (LDA), a popular probabilistic model traditionally used for discovering latent semantic topics in large collections of text documents. In our variant of LDA, we model each musical piece as a random mixture of keys. Each key is then modeled as a *key-profile* – a distribution over a set of pitch classes. Using this representation, we show how to accomplish the tasks of key-finding and modulation-tracking, the first steps to any kind of harmonic analysis.

Our approach to such tasks is unlike that of previous studies, which depend on extensive prior music knowledge or supervision by domain experts. Based on a model of unsupervised learning, our approach bypasses the need for manually key-annotated musical pieces, a process that is both expensive and prone to error. As an additional benefit, it can also discover correlations in the data of which the designers of rule-based approaches are unaware. Since we do not rely on extensive prior knowledge, our model can also be applied in a straightforward way to other, non-western genres of music with different tonal systems.

Chapter 1

Introduction

In this thesis, we propose novel methods for automatically extracting harmonic content – global and local key information – from music in the form of both symbolic and raw audio input. All of our methods are based on a model of unsupervised learning, bypassing the need for key-annotated training examples. In this introductory chapter, we first talk broadly about the music information field (1.1) – to which our work belongs. Second, we give a problem overview in which we define the problem, explain the motivation for this work as well as where it fits more broadly in the MIR field, and finally, present unique contributions (1.2). We conclude this chapter with an overview of how this thesis is organized (1.3).

1.1 Music Information Retrieval

Music information retrieval (MIR) is a very broad term that can refer to any component of an automatic system that allow us to understand, extract, or interact with information from music. The ultimate vision for the MIR field is to achieve a certain way of life – one in which we are surrounded by technologies that can understand music the way we humans do, while retaining all of the benefits of modern technology (e.g. fast search, efficient computation, and large memory). Here is a popular quote by Downie [18] – a long-time researcher in the field – describing what this world would look like:

> Imagine a world where you walk up to a computer and sing the song fragment that has been plaguing you since breakfast. The computer accepts your off-key singing, corrects your request, and promptly suggests to you that "Camptown Races" is the cause of your irritation. You confirm the computer's suggestion by listening to one of the many MP3 files it has found. Satisfied, you kindly decline the offer to retrieve all extant versions of the song, including a recently released Italian rap rendition and an orchestral score featuring a bagpipe duet.

While the current directions of music technology are not the same in 2003, when this quote was issued (most notably: the substantial rise in mobile and internet activity, as well as the increase in music storing capabilities), the scenario described remains relevant as it showcases a potential reality in which computers can understand music. As one can imagine, the road to achieving such a vision is extremely interdisciplinary. MIR broadly encompasses everything from ideas to algorithms to applications, and draws from fields such as musicology, composition, artificial intelligence, machine learning, signal processing, computer systems, psychology, and the cognitive sciences. The upside of such interdisciplinary involvement is the knowledge that its impact reaches far and wide. The downside is that different areas are concerned with different goals for the a similar problem. We will make clear the goals of this thesis later on in section 1.2.2.

1.1.1 Content-Based Music Analysis

The work in this thesis belongs to a sub-area of MIR called *content-based music analysis*. This area seeks to automatically extract *content* from musical pieces. What is musical content? This can be difficult to define. In literature, content is commonly described as the topics or ideas contained in a piece of writing. When applied to music, this can take on a lot of meanings: the mood, story, stylings, influence, melodic structure, compositional techniques, etc. of a song can all be considered as content. Perhaps it is easier to describe what content is *not*. Content is not meta-data information, like the title or composer or year of a song. This information does not tell us anything about what is "inside" the music, or what the music sounds like.

Most methods used to access, search, and organize music collections today are still based on meta-data. When you open your default music browser, you don't really have the ability to search by anything other than the text of the title and composer (and possibly genre) , unless you have spent time adding or obtaining extra tags yourself. Content-based analysis seeks to enhance this experience by automatically generating more useful and meaningful descriptions that are actually related to musical content. As imagined in the quote from the previous section – content-based methods would allow one to query a music database by humming, or automatically create a playlist of upbeat songs. As one can imagine, these content-based descriptions can be used to facilitate more powerful search, organization, and creation of musical pieces.

1.1.2 Music Descriptors

For most content-based analysis, a necessary first step is to describe music input in a way that a computer can understand and process it. The development of these descriptions, aptly called *music descriptors*, have spurred a lot of research in the MIR community – some, for the sake of producing more high-level and intuitive descriptors from raw input; others, as mentioned above, for the development of specific, higher-level MIR applications.

For audio music input, Herrera [29] categorizes descriptors from low to high level. *Low-level* descriptors are closely related to raw audio signal. These descriptors are unintuitive to humans, but are easily inputted to computational systems. Two of the most popular low-level content-based descriptors on which many MIR systems are built upon are the Mel-Frequency Cepstrum Coefficients (MFCC) features [53], and chroma features [24]. For symbolic music, MIDI data provides a complete transcription of the musical score, indicating exact pitches and their durations present in the score; this can be argued to be the most low-level type of descriptor. We note that while it would be useful to have such a transcription for audio input as well, a large amount of literature has shown that the inaccuracies involved in such a transcription usually do more harm than good (as polyphonic transcription is still a challenging task that remains unsolved). Instead, the audio

descriptors described earlier in this section have been shown to be more suitable for most tasks at hand. For this reason, we emphasize that music descriptions do not necessarily have to be transcriptions.

Mid-level descriptors are considered to be those that perform some generalizations about the underlying data. This usually involves some statistics and machine learning techniques, and will have more intuitive meaning for certain users. For example, harmonic content, which this thesis aims to extract automatically, may be considered as a mid-level descriptor – it requires some analysis and transformation of raw input data to draw some more high-level information about the musical content. Even though such annotations are interesting in their own right, they are also useful in higher level applications, as we will discuss in section 1.2.2. Another example of mid-level descriptors may include music tags – words that describe musical content. These tags can range from descriptions about mood (melancholy, upbeat), to stylings (avante garde, classical, triple meter), to influences (african, blues, bop, 80s), just to name a few. While these tags can be obtained through human annotations[1] (making it a border-line *high level* descriptor), much work has also been devoted to automatically extracting these text-descriptions from audio [78, 30].

High-level descriptors are most far-removed from this thesis. Herrera requires such descriptors to *bridge the semantic gap*; where the semantic gap is defined by Smeulders [72] as being *the lack of coincidence between the information that one can extract from the (sensory) data, and the interpretation that the same data has for a user in a given situation.* As such, high-level descriptors should be relevant to users and require modeling user behavior. Disciplines such as music cognition or psychology are more heavily involved in devising these descriptors.

As mentioned above, the work in this thesis seeks to automatically extract harmonic content of both symbolic and audio music input. Such a descriptor most closely aligns with the content-based, mid-level descriptors discussed above. In the next few sections, we describe the problem in more details, as well as the motivation for such a descriptor, and potential MIR applications.

[1]Last.FM, is a personalized, online radio that allows users to tag songs they hear

1.2 Problem Overview

In the last section, we briefly discussed what MIR is about, and where content-based analysis fits into that picture. In this section, we will focus specifically on the work of this thesis, including a more formal problem definition, motivation behind this work, the goals of this thesis, and potential applications that the proposed descriptors could contribute to.

1.2.1 Problem Definition

Though music is an art form with tremendous expressive capacity, the underlying framework (for the most part) is highly organized and structured. Two important components of this framework is the *key* and *tonic*. The key of a musical piece identifies a principal set of pitches that the composer uses to build its melodies, chords, and harmonies. The tonic is the most stable pitch in this set. Each musical piece is characterized by one overall key, but the key can also be shifted within a piece by a compositional technique known as *modulation*. In this thesis, we seek to extract meaningful harmonic information from musical pieces by automatically predicting the overall key of a musical piece, as well as the local key of shorter music segments (thereby, tracking its modulations). We consider 24 possible keys (12 major, and 12 minor keys), and primarily evaluate our models based on how well our predictions align with ground-truth labels (both, in the form of a single "correct" key).

1.2.2 Motivation

The motivation for extracting harmonic content comes from several different directions. The first motivation is practical: automatically obtaining key annotations for symbolic or audio music is useful for musicians for music analysis. Just as writers exercise literary analysis, musicians must also have a deep understanding of the compositional techniques that are used in certain pieces. Key annotations are the first step to any such analysis, and being able to do so automatically would give aid in the form of music pedagogy, as well as a "second opinion" from a data-

based perspective. Key annotations can also assist human transcribers to perform their tasks more accurately and quickly.

A second source of motivation comes from the MIR applications. As we described above, many applications can benefit greatly from good content-based music descriptors. Though these applications are not in the scope of this thesis, we list previous work that have successfully used harmonic descriptors similar to the ones proposed in this paper (key-annotations, chord transcriptions and modulation-tracking):

1. **Music annotation** – software, such as *The Clam Annotator* [2] uses harmonic descriptors to automatically annotate and generate music meta-data at different levels of abstraction

2. **Song segmentation** uses harmonic descriptors to find where important segments of music begin and end (for example: verse, chorus, and bridge in pop songs, or repeated sections in classical pieces). Harmonic descriptors are more robust than melody-based descriptors, since there may be slight variation on the melody between repeated sections, while the harmonic structure should stay the same [59].

3. **Cover song detection** uses harmonic descriptors to find cover songs in large collections of music. Harmonic descriptors are very suitable for this task as they are robust to changes in instrument, key, and performers [47, 3].

4. **Automatic accompaniment** – software, such as *SongSmith* [71] automatically generates accompanying chords from songs that the user sings and records. The chord generation process uses similar approaches as chord transcription.

5. **DJ Mixing** uses harmonic descriptors to automatically detect the key of songs (and then adjusts it) in order to mix songs with close tonalities. Some examples including *Harmonic Mixing*[2] and *Mixmeister*[3]

[2]http://www.harmonic-mixing.com
[3]http://www.mixmeister.com

6. **Modeling Musical Tension** – the building and resolving of musical tension is closely related to whether or not the tonalities present are harmonious, or conflicting. Prior work [17] has shown that harmonic descriptors are useful to automatically predicting areas of rising and falling tension in classical music.

A third source of motivation is aimed specifically at the part of our work that uses symbolic input. While symbolic (MIDI) input has not received as much attention as audio input, it is still important for two reasons: (1) it allows more exact analysis when musical scores are available in this format, and (2) many audio-based models use audio synthesized from MIDI to easily obtain large training data for their models; hence they are in need of methods that perform accurate key and chord estimation on MIDI input.

Lastly, we find motivation for the purposes of pure research. In our work, we have introduced a new application for a popular machine learning model that was traditionally used primarily to discover semantic topics in text corpora. It is interesting to understand (1) the relationship of semantic topics and words vs. harmonic content and notes, (2) how our very different unsupervised models perform in comparison to supervised models of harmonic content (3) how important prior music knowledge is in harmonic modeling, (4) how important note order is to extracting harmonic content, and (5) how closely statistical methods can imitate the way humans perceive tonality. These are all interesting topics that we have been able to address as a result of our work.

1.2.3 Contributions

This thesis proposes a novel method for global and local key-finding, using both symbolic and audio input. The main contributions are found in chapter 3 and 4, where three novel, probabilistic, LDA-based models are presented. Appendix C also contains the derivations for variational inference algorithms proposed in our audio LDA-based models. To summarize, the novelty of our model includes the following:

1. For audio input, our model uses *unsupervised learning* with very little prior music knowledge. Most approaches seen previously that obtain high performance use supervised frameworks where key and chord-annotated data is necessary for training, or use hand-chosen, musically inspired parameters instead of estimating them from data.

2. For audio input, our model is one of the few that does not rely on a time-series representation, and instead, models songs as a *bag-of-notes*. (The majority of supervised models in the past are based on hidden Markov models, used to estimate key and/or chord transition information.)

3. For symbolic and audio input, our model requires very little prior music knowledge, while most others require hand-tuned, tonal-system-dependent weights.

While these novel features make our model easier to use (i.e. requires no training data) and more generalizable (i.e. requires no hand-tuned key-profiles), they also pose more challenges to the key and chord estimation problem by providing less prior information. While our audio model does not obtain higher key-finding accuracy than the state-of-the-art, our symbolic models were able to achieve higher accuracy than previously published and frequently used models.

We also note that even though key-finding accuracy can be a good way to evaluate models on a large-scale, the generative nature of our model does more than just key prediction. More precisely, our approach models the music collection in terms of its harmonic structure, and assigns key proportions to every song, as well as every segment within the song. This type of modeling is arguably more realistic than assigning a single key, as composers would agree that the harmonic make-up of most musical compositions is not black and white.

1.3 Thesis Organization

The following is a summary of how the remainder of this thesis is organized:

Chapter 2: Background and Previous Work gives scientific background to the acoustics of music, as well as basic building blocks of musical compositions. In particular, we review important concepts such as frequency, pitch, tonality, and harmony. In the second part of this chapter, we review previously published work in the area of chroma feature computation, as well as key and chord-finding systems for both symbolic and audio input.

Chapter 3: Harmonic Analysis for Symbolic Input introduces the LDA-MIDI model, an unsupervised and generative framework for modeling musical pieces and musical segments as a random mixture of musical keys. We also introduce an extension to this model, called CTM-MIDI, which models key-correlations. We evaluate both models on key-finding performance, as well as modulation-tracking.

Chapter 4: Harmonic Analysis for Audio Input introduces the LDA-AUDIO model, which is the model from the previous chapter, adapted for audio input. Probabilistic inference is more challenging in this model, and thus, a full derivation is provided in Appendix C. We use similar evaluations tasks on the LDA-AUDIO model by observing key and chord-estimation performance.

Chapter 5: Conclusion is a summary of the work described in the thesis, as well as an overall analysis of how the novel features of our model impact performs, in comparison with previously published models. We draw conclusions from this analysis and point to directions for future work.

Chapter 2

Background & Previous Work

2.1 Overview

The goal of this chapter is to provide an introduction to music theory that will allow the reader to understand the musical components of this thesis, and second, to describe how this information can be digitized and understood by computers in a process called *feature extraction*. There are many ways to talk about music, so we specifically guide this discussion in a way that relates to tonality (the focus of this thesis) as much as possible. Understanding tonality requires musical comprehension on several different levels, so we spend the first section discussing basic musical components (pitch, rhythm, melodies, chords) before delving into more high-level musical structures, such as tonality, musical keys, and scales. In the final sections, we also present related work in the area of tonality induction and key-finding, as well as applications and systems that have utilized automatic key and chord estimation algorithms.

2.2 Musical Building Blocks

In this section, we talk about basic musical components. These can be thought of as "musical building blocks" in the sense that all of these components are used in combinations and patterns to create a piece of musical work. We group these components into three parts. First, we talk in depth about pitch and its

physical/scientific counterpart, frequency. Second, we briefly mention some other important basic dimensions of music making, such as rhythm, note durations, tempo, and loudness. Third, we introduce melodies, harmony, and chords.

2.2.1 Pitch and frequency

Music begins at the physical level: it is a sound wave propagating through the air, causing variations in air pressure against our eardrums. The subsequent neural processing and interpretation of these vibrations produce the experience that we call *sound*. In particular, most sounds that people recognize as musical are dominated by periodic or regular vibrations. These periodic vibrations can be quantified in terms of *frequency* – the number of times that a cycle is repeated per second. Its perceptual counterpart is *pitch*. Whereas frequency refers to a scientific/objective measurement, pitch is a subjective quality used to express the musical experience in terms of highness or lowness: larger frequencies produces sounds that humans interpret as "higher sounding" pitch (think: female opera singer); smaller frequencies correspond to "lower sound" pitches (think: bass guitar). For a concrete idea of how frequency related to pitch, we note that in western music, 440 Hz is the standard reference frequency that maps to the pitch labeled A_4 (or "middle A" on the piano). The remaining sections will discuss different aspects of this relationship that will hopefully provide a fuller picture of how pitch and frequency are connected.

Understanding frequency: humans vs. computers

Why is frequency the distinguishing factor for how humans perceive different sounds? Put simply, different regions of the human ear are activated by sounds of difference frequencies. Its remarkable design involves membranes, miniature bones, and specialized hairs of different lengths that respond and vibrate with greater or lesser amount according to how much the sound contains its resonant frequency. Essentially, the human ear functions as an organic filter bank: it identifies and separates out all of the frequencies that are present in a sound wave (along with how dominant each frequency is) and sends this information in the form of electrical

signals to the neurons for further processing.

The computer equivalent of separating out frequencies in a sound wave is called a *Fourier transform*. Just as the ear perceives the sound, computers can record a sound wave as a digital audio signal. In its raw format, the audio signal is said to be in the *time domain* as it is not much more than series of integers, indicating amplitude as a function of time. When a Fourier transform is applied to a segment of audio, it transforms the audio signal into the *frequency domain*, which expresses the contribution (i.e. loudness) of each frequency to the overall sound. Figure 2.1 shows an example of the two different domain representations. Plot (a) shows the original signal in the time-domain, while plot (b) shows the signal in the frequency-domain, after applying a Fourier transform.

A third way of analyzing audio signals is by using a *spectrogram*. Not only does a spectrogram express the loudness of each frequency, but it is also capable of depicting both as a function of time. As such, the spectrogram works in the *time-frequency domain*. There are many variations on the spectrogram; most commonly, they are created by taking the Fourier transform on short, overlapping, consecutive segments of a longer musical audio signal (described as a *short-time Fourier transform*, or *SFTF*), or more specifically:

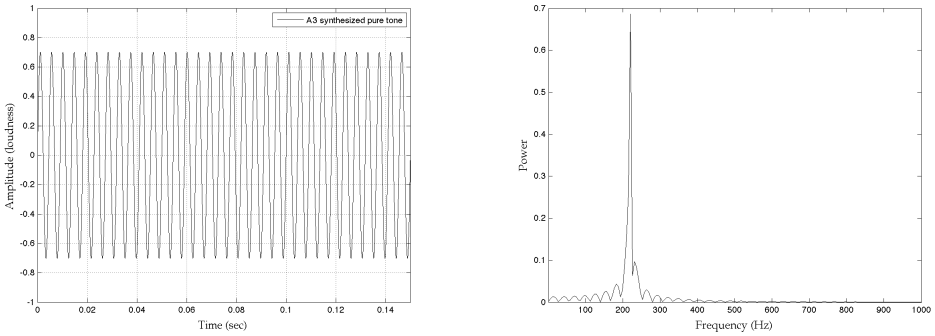
$$\text{spectrogram}(t, \omega) = |\text{STFT}(t, \omega)|^2 \tag{2.1}$$

where (t, ω) indicates a particular time and frequency. The horizontal axis on the spectrogram represents time and the vertical axis is frequency. A third dimension indicating the amplitude (or loudness/dominance) of a particular frequency at a particular time is represented by the intensity of color of each point in the image.

Fundamental frequency, overtones, and harmonics

Recall that a distinguishing characteristic of musical sounds is that its sound wave is periodic (or, as mentioned later on, can be decomposed into a weighted sum of periodic waves). An example of a simple, periodic waveform is represented by the following sinusoidal function:

$$x(t) = a \cdot \sin(2\pi f t + \phi) \tag{2.2}$$



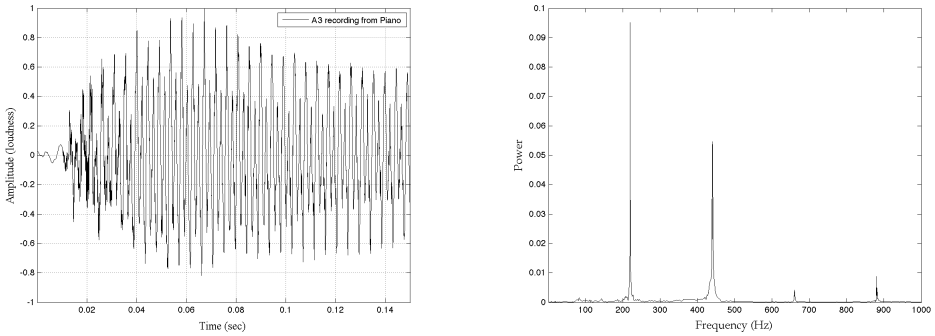
(a) (b)

Figure 2.1: A pure tone (simple sinusoidal) with frequency of 220 Hz (fundamental frequency for note $A3$) represented in three different ways: (a) original waveform, (b) output of Fourier transform

where a represents the maximum amplitude of the signal, t represents the time (in seconds), f the frequency (measured in oscillations per second, called *hertz*), and ϕ the initial phase. An example is shown in figure 2.1; its corresponding Fourier analysis shows that the waveform consists of a single frequency: 220 Hz. Though the sound produced by this sinusoidal wave will clearly correspond to the pitch labeled A_3, it is not a naturally occurring sound. In fact, it will sound unrealistic and synthesized, and can only be created using a tuning fork or a computer program.

In contrast, figure 2.2 shows the same plots for an A_3 played on a real piano. Compared to figure 2.1, we see that a) the waveform is much more complex, and b) the Fourier analysis shows the presence of many frequencies, in addition to the one at 220 Hz. The spike at 220 Hz is the largest, and is called the *fundamental frequency* or f_0. The other spikes to the right of it are called *overtones*. Intuitively, these overtones are what gives the piano note its character. In the musical world, this musical "character" is called *timbre*, and it is what distinguishes the same pitch played on a piano versus a violin, cello, flute, etc. In general, when looking at the Fourier analysis of any single note played by an instrument: the single lowest (and dominant) frequency will be the fundamental; all other higher frequencies are referred to as overtones.

More formally, the term *partials* are used to refer to the fundamental fre-



(a) (b)

Figure 2.2: An $A3$ (with fundamental frequency 220 Hz) played on the piano (as opposed to synthesized by computer) represented in three different ways: (a) original waveform, (b) output of Fourier transform (note the multiple peaks)

quency and overtones together. The term *harmonics*, more specifically refers to those frequencies that are integer multiples of the fundamental (including the fundamental which is 1 times itself). Harmonics are very important as they usually produce the largest spikes in the Fourier analysis. In fact, a fairly realistic sound wave can be produced by taking a weighted sum of sinusoidals that take on frequencies that is equivalent to the harmonics. This gives us a more realistic mathematical representation of our periodic waveform:

$$x(t) = \sum_{n=1}^{N} a_n \cdot \sin(2\pi n f_0 t + \phi_n) \tag{2.3}$$

where n indexes a sinusoidal with a frequency of the nth harmonic. We see in the next couple of sections why harmonics are particularly important as we investigate the mathematical relationship between frequency and perceived pitch.

Labeling pitches as notes

So far, we have referred to pitch simply as a subjective quality that describes a musical sound as "high" or "low". To make this more concrete, we introduce how pitches are organized and labeled using a combination of letters and numbers, called the *Helmholtz pitch notation*. Labeled pitches are referred to as *notes* and its frequencies are quantized into *semitones*, the smallest musical interval commonly

used in western tonal music. A set of 12 consecutive semitones make up a special interval called an *octave*. For reasons discussed later, each pair of notes that is exactly 12 semitones (or, an octave) apart will sound identical, the only exception being that the higher note will sound, well, "higher". In fact, notes that are an octave apart can sound so similar that even musicians can confuse them, especially when those notes are played on different instruments.

This phenomenon gives rise to the *pitch helix*, shown in Figure 2.3, which is a visualization that shows the cycle of 12 semitones repeated every octave. It also conveniently shows pitch as a composite of two dimensions: *height*, (moving vertically in terms of octaves) and *chroma* or *pitch class*, determining the horizontal/rotational position within the helix. The cyclic nature of pitches is evident in the labeling system as well. Each pitch is labeled with a letter and a number; the letter indicates one of the 12 semitones (i.e. the chroma); the number indicates which octave the semitone belongs to (i.e. the height).

To get an idea of how these notes/pitches relate to frequency, table A.1 in the appendix shows a truncated list of mappings between notes and fundamental

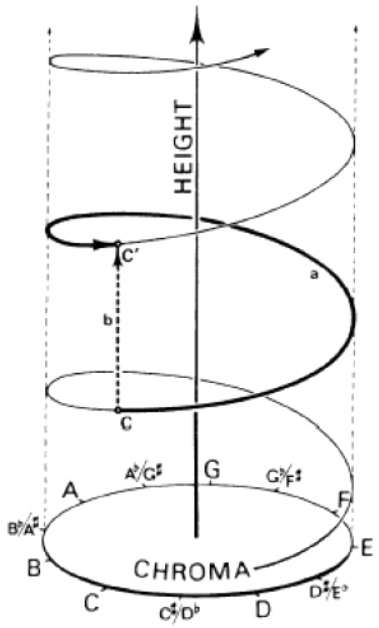


Figure 2.3: Shepard's pitch helix, which visualizes pitches in two dimensions: height and chroma.

frequencies. For example, A_3 corresponds to a frequency of 220 Hz, while A_4, the same semitone but one octave higher corresponds to the frequency 440 Hz. In the next section, we discuss how these mappings are obtained.

Equal temperament tuning: Mapping frequencies to pitches/notes

Musical tuning is the process of adjusting the frequency of pitches on musical instruments in order to establish typical/correct ratios of frequencies between pitches. Though preceded by a long and complicated history, the most common system of tuning of since the middle of the 19th century in western music is called *equal tempering*. The main characteristic of this tuning system is that the perceived "sound distance" between each pair of adjacent notes is the same. This is achieved by choosing the pitches in such a way that the ratio of frequencies for all adjacent notes is constant. This creates a logarithmic relationship between notes and frequencies.

More specifically, western classical music (which is the main subject of analysis here) uses the *12-tone equal tempering system*. This is the reason behind dividing the octave into 12 evenly spaced semitones. As mentioned before, notes that are an octave apart sound identical, except that one sounds higher than the other. Interestingly, an important natural phenomenon occurs here: Let x_2 be a note that is exactly one octave (12 semitones) above note x_1, and let f_2 and f_1 denote their respective frequencies. Then,

$$f_2 = 2 \cdot f_1 \tag{2.4}$$

This holds true for any two notes that are an octave apart. This natural phenomenon is often referred to as the "basic miracle of music" and is the basis for many modern musical systems.

In order to figure out the mapping between frequencies and pitches, we note the following constraints: 1) notes that are an octave apart have double or half the frequency of the other, and 2) the octave must be divided into 12 semitones such that the ratio, r between frequencies of any two adjacent notes is constant.

We can solve for r as follows:

$$f_1 \cdot r^{12} = f_2 \tag{2.5}$$

$$r = 2^{\frac{1}{12}} \tag{2.6}$$

This means that the ratio of frequencies between any two adjacent notes is $r = 2^{(1/12)}$, and the ratio of frequencies between notes with an interval of n semitones is similarly $r_n = 2^{(n/12)}$. As noted before, in western music, a frequency of 440 Hz is a standard reference to describe the pitch known as the note "middle A" on a piano; the frequencies of all other pitches can be computed based on this landmark and the ratio of frequencies that we just computed.

2.2.2 Rhythm, tempo, and loudness

Pitch may be the most prominent musical building block; however, there are several other really basic dimensions that must be used in combination with pitch to create music. Because they are not the focus of this thesis, we touch on each briefly. *Rhythm* is the part of music that deals with time. At a basic level, it dictates the length of time an individual note should be played for. More specifically, all musical pieces have an underlying *beat*, which is a basic unit of time. Rhythm tells us how many beats each note should last. More advanced concepts such as *metric structure*, have certain restrictions in terms of how many beats can be contained in a segment of music, thereby constraining the types of rhythms that may be created. The *tempo* of a musical piece is the speed or frequency of the beat; it is a measure of how quickly the beat flows. Music pieces with faster tempos often feel very high-energy, or grandiose. Those with slower tempos may feel relaxing, and perhaps even somber. Tempo is often measured in *beats per minute (bpm)*. For example, 60 bpm means a speed of one beat per second, a frequency of 1 Hz. *Loudness* is fairly intuitive – it is basically how loudly or softly the music is played; it is determined, for example, by the strength in which a pianist strikes a key or a violinist pushes down on the bow. The loudness and softness of music, especially in western classical music, plays a large role in the story-telling aspect of music. A *crescendo*, for example, commands a gradual increase of sound, which often causes

a building of tension, while a *decrescendo* – a gradual decrease in sound – may resolve tension or convey a more peaceful mood in the music.

2.2.3 Melody, harmony, and chords

The mid-level musical components we refer to here build on top of the low-level elements discussed before. A *melody* is a sequence of pitched notes with durations (i.e. rhythm). Together, these things create a clearly defined musical "shape" and sound. A melody can be as simple as "Twinkle Twinkle Little Star" but can also be a complex orchestral theme that is backed up by dozens of instruments. *Harmony* is a term that denotes the simultaneous combination of notes, called *chords*, and a sequence of chords, called *chord progressions*. It is also a term that is used to describe the system of structural principles that govern the combination of notes that can form pleasing-sounding chords (which is dependent on culture, of course). Having described both harmony and melody, we see that both refer to a combination of pitches – melody is the sequential combination of pitches, while a chord is the simultaneous combination of pitches. In this sense, it is sometimes very difficult to separate melody from harmony, as they heavily influence each other.

2.3 Tonality

Tonality is a system of music that arranges hierarchical pitch relationships around a *tonic*, a tonal "center" of a musical piece. If all of the basic musical elements (from the previous section) were words on a page, then the *tonic* would be the ideas expressed through those words. More specifically, a writer will first have an idea that will then cause the writer to use a certain set of vocabulary to best convey that idea. Similarly, tonality is a system of music in which a composer will first choose a *tonic* for his or her music piece, and then arrange all of the pitches in a way that the harmonies and relationships between those pitches reflect that tonic. We discuss tonality and the concept of a tonic in more detail at two different levels: a technical discussion from a music theory standpoint, and

then a more practical and intuitive look at what tonality looks like, and how it is used in music composition.

2.3.1 Theory of Tonality

In western tonal music, tonality is concretely organized through the use of *keys* and *scales*. (Tonal systems in other cultures may not necessarily evolve around such concepts.) A *key* defines a system of relationships between a series of pitches called a *scale*; in effect, each key is associated with a scale (though there are scales that are not associated with keys). There are two basic key *modes*: major and minor. A key in minor mode will have a slightly different scale than the same key in major mode. Now, within this thesis, we will only talk about keys that are associated with *diatonic scales* – each of these scales consist of seven, ordered pitch classes. Specifically, the *diatonic* characteristic of the scale refers to the specific pattern of intervals separating the seven notes. For major scales, the pattern of interval between notes is: $T - T - S - T - T - T - S$, where S stands for an interval of a semitone, while T stands for an interval of a whole tone (twice the distance of a semitone). For minor scales, the pattern is slightly different: $T - S - T - T - S - T - T$. We note that this particular version of the minor scale is called the *harmonic minor*. Other variations (including the commonly used *melodic minor*) of the minor scale also exist. Figure 2.5 visualizes the patterns of the major and minor scales.

Figure 2.4 demonstrates this pattern for the major scales: Each column indicates a semitone, and each row indicates one of the 12 major keys. Each cell along a row will either be colored or uncolored: a colored cell indicates that that particular semitone is a part of that key's scale (the roman numeral indicates the scale degree); a non-colored cell indicates that that semitone is not a part of the key's scale. We note that each key is related to the next by simple transposition of scales. Figure 2.5 shows scale patterns across two different modes, for a single key C. As with the major scale, the scale for any minor key can be obtained by simply transposing the pattern given for C minor the appropriate number of steps.

Each pitch class within the scale has a name which indicates how important

that pitch class is with respect to that particular key. Figure 2.6 shows the names of each scale degree, as well as their roman numeral shorthand. The first note in the scale is the *tonic* degree, abbreviated as the roman numeral I. As mentioned earlier, this is the central and most important pitch class within the key. In fact, the key is also identified by the tonic. For example, as shown in figures 2.5 and 2.4, both keys C major or C minor have pitch class C as its tonic. Besides the tonic, two other important pitch classes are the *dominant* degree (fifth note on the scale) and the *subdominant* degree (fourth note on the scale). Intuitively, the scales and chords based on these two pitch classes have many common notes, making it easy to transition in and out of.

How many different keys are there? Recall that there are 12 unique pitch classes. Since each of these pitch classes can be the tonic of a key, and since there are two different modes, this gives us a total of $12 \times 2 = 24$ total keys: 12 major and 12 minor keys. There are many interesting relationships between these 24 tonalities, and one common method of visualization is the *circle of fifths*, as shown in Figure 2.7. Here, the 24 keys are arranged in a circle; minor keys are inside the circle and major keys lie outside. The major and minor keys that lie directly across each other on the circle indicate that they are *relative (major or minor) keys* of each other (i.e. C major is the relative major of A minor); this means that they share the same collection of pitch classes and *key signature* – the particular group of sharp and flat signs assigned to specific pitches in order to represent the notes in the key (shown in the middle circle within the circle of fifths in Figure 2.7). The two keys only differ by the position of the third degree in their respective scales.

Neighboring pairs of major-minor keys indicate a separation of a fifth interval. For example, moving clock-wise on the outer circle, the key of G major follows C major since the pitch class G is the fifth note on the C major scale (or, the dominant degree). Not only have we established that the dominant degree is the second most important note in the scale, it also follows that the key signature of the key based on that degree (in this case, G major) differs by only one *accidental* (i.e. a sharp or a flat). This means that C and G are very similar in the sense that their tonal properties are similar. Another relationship between tonalities

Scales for Major Keys

	C	C♯ / D♭	D	D♯ / E♭	E	F	F♯ / G♭	G	G♯ / A♭	A	A♯ / B♭	B
C Major	I		II		III	IV		V		VI		VII
C♯ / D♭ Major	VII	I		II		III	IV		V		VI	
D Major		VII	I		II		III	IV		V		VI
E♭ Major	VI		VII	I		II		III	IV		V	
E Major		VI		VII	I		II		III	IV		V
F Major	V		VI		VII	I		II		III	IV	
F♯ / G♭ Major		V		VI		VII	I		II		III	IV
G Major	IV		V		VI		VII	I		II		III
A♭ Major	III	IV		V		VI		VII	I		II	
A Major		III	IV		V		VI		VII	I		II
A♯ / B♭ Major	II		III	IV		V		VI		VII	I	
B Major		II		III	IV		V		VI		VII	I

Figure 2.4: Scales for all 12 major keys. Each column indicates a semitone, and each row indicates a key. Each cell along a row will either be colored or uncolored: a colored cell indicates that that semitone is a part of that key's scale (the roman numeral indicates the scale degree); a non-colored cell indicates that that semitone is not a part of the key's scale.

are the *parallel* major and minor of a key. These keys share the same tonic (for example, C major and C minor), but have very different diatonic collections (i.e. two notes are different in the respective scales). Finally, we note that in certain circumstances, music pieces *will* use pitch classes outside of its tonic major or minor scale. Sometimes, this can be done without weakening its sense of orientation towards the tonic (referred to as *tonal*); other times, it does the opposite, and indeed creates a sound of dissonance or instability. This is referred to as *atonal*.

2.3.2 Tonality in music

The previous section gave a fairly technical definition of tonality. Here, we discuss the more practical side of tonality – what it sounds like and how it is used in music composition. When a composer writes music, they must first decide on a key to write the music piece in. This determines the kind of melodies, harmonies, and chords that will be present in the musical piece.

For example, if a musical piece is in the key of E major, then the tonic will be the note E, and the dominant degree will be the note B. We will expect the composer to use these notes quite frequently. Moreover, there are certain chords that will sound especially pleasing in the context of this key. The most basic of such chords is the *tonic triad*, which is a three-note chord whose lowest note (called the chord's *root*) is the tonic ($E - G^{\#} - B$, in this case). *Dominant chords* are also significant as they create a sense of tension in the context of the tonic chord, and needs the tonic chord for resolution. A dominant chord can be some variation of a triad that has the dominant degree as its root ($G^{\#} - B - E$, for the key of E major). Other variations include a dominant seventh chord, in which a fourth note that is the seventh degree (of the scale of the dominant degree) is added onto the chord. The interested reader should refer to [38] for more details as there are hundreds of different chord variations.

Comparison of Major and Minor Scales

	C	$\frac{C^{\sharp}}{D^{\flat}}$	D	$\frac{D^{\sharp}}{E^{\flat}}$	E	F	$\frac{F^{\sharp}}{G^{\flat}}$	G	$\frac{G^{\sharp}}{A^{\flat}}$	A	$\frac{A^{\sharp}}{B^{\flat}}$	B
C Major	I		II		III	IV		V		VI		VII
C Minor (Harmonic)	I		II	III		IV		V	VI			VII
C Minor (Natural)	I		II	III		IV		V	VI		VII	

Figure 2.5: Comparing scales across different modes (for the same key, C). We note that Figure 2.4 can be reproduced for all minor keys by taking the C minor scale pattern shown here and transposing the pattern by the appropriate number of steps (i.e. offset by interval from C)

Scale Degree Names

tonic	super-tonic	mediant	sub-dominant	dominant	sub-mediant	leading tone
I	II	III	IV	V	VI	VII

Figure 2.6: Names of different scale degrees. I indicates the first note in the scale, II indicates the second, and so on. There are a total of seven notes in western, diatonic scales, which will be the main focus of our study.

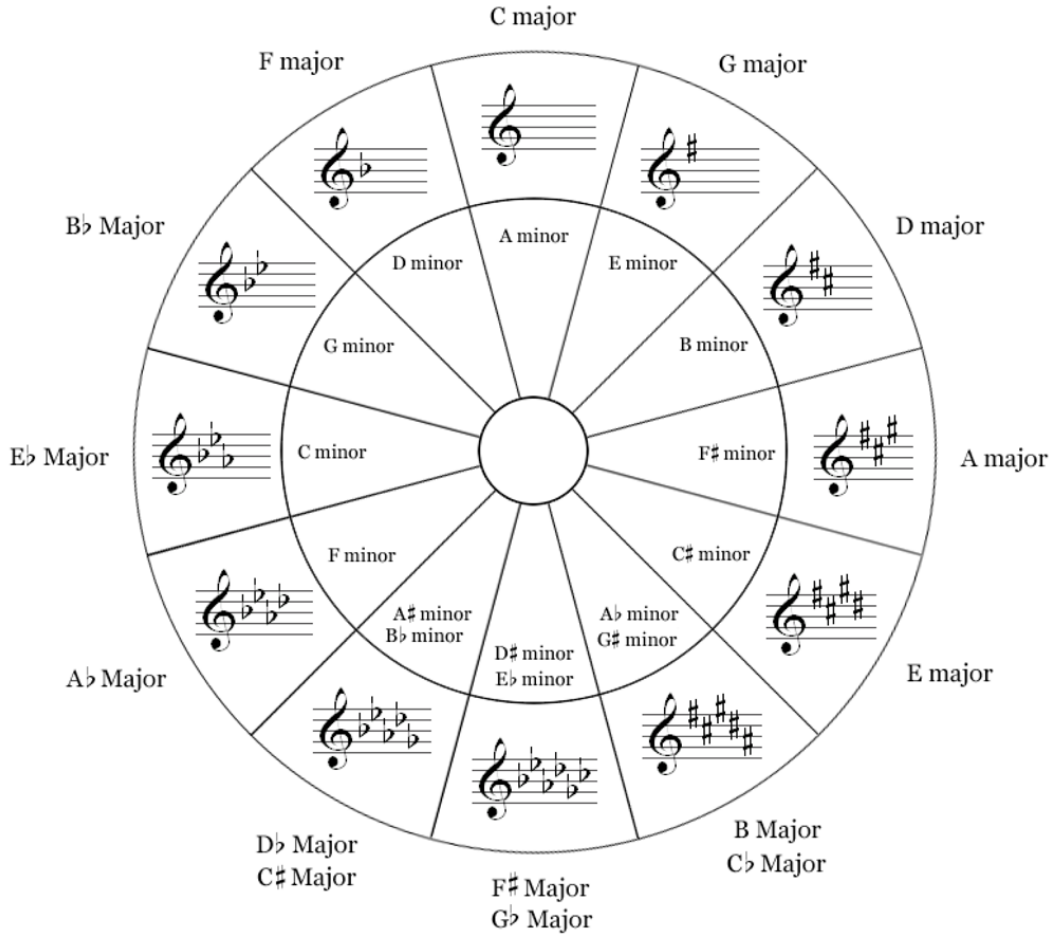


Figure 2.7: The *circle of fifths*, used to visualize the 24 major and minor keys in western tonal music. The inner circle shows minor keys and the outer circle shows major keys. Adjacent keys are separated by a fifth interval.

Though each musical piece will have a clearly defined overall key (and usually begin and end in this key), few pieces stay in this key the entire time. *Modulation* is the term for when a composition moves from one key (or one tonal center) to another within a song; it is a compositional tactic that is used to articulate or create structure, as well as add interest within the piece. The most common modulations are to closely related keys – for example, those that are neighbors on the circle of fifths (figure 2.7). The most common destination key for major keys is the dominant degree (for example, modulating from C major to G major) or even the subdominant; for minor keys, it is the mediant (III) degree (for example, from C minor to E minor). Modulations to the relative major or minor are also simple (for example, C major to A minor), as these keys share all pitches in common.

Common chord modulation is one popular method of modulation: it moves from the original key to the destination key by way of a chord both keys share. For example, the keys of G major and D major have four important chords in common: G, Bm, D, Em. Chords with G as root acts as the tonic chord for the key of G major, while also acting as the subdominant chord for the key of D major. Similarly, a chord with D as root acts as the tonic chord for D major, while also acting as the dominant chord for the key of G major. This is why it is easiest and most common to modulate between closely related keys, as this maximizes the number of pitches and chords that the two keys share.

From a cognitive standpoint, tonality plays a large role in how humans perceive the music. Tonality is what distinguishes the "sound" of western music from other sounds (such as that of Chinese or Middle-Eastern music, for example). In western tonal music, the tonality of a song can make a song sound "happy" or "sad" [34]. It can also build feelings of tension and urgency, as well as resolve them [50] by moving from dissonant-sounding chords (tension) to more stable chords (resolution). Thus, tonality not only acts as the "glue" that holds all of the basic music theory elements together, but it also largely influences people's emotional response to music.

2.4 Tonality Induction

Having discussed *tonality* in the previous section, we now review methods for how to analyze tonality in a piece of music. This process is called *tonality induction*, and its goal is to find the global tonal center of a piece, as well as how the tonal center shifts locally through the piece. Concretely, this can take on the form of several different – but highly related – tasks, which we outline below:

Key-finding. Without any modifiers, key-finding refers to finding the overall key of a piece. Almost all key-finding models classify each song into one of the 24 major/minor keys.

Local key-finding/Modulation Tracking. As a result of modulation, the tonal center of the piece will move around during the duration of a song. Local key-finding refers to finding the key of a local/short segment of music.

Chord-estimation. This may refer to literally identifying the sounds produced by multiple notes played simultaneously as a particular chord type. However, more often, chord-estimation refers to identifying the chord progressions in a piece of music, i.e. segmenting a song into uniform pieces and identifying the local key in each of those segments.

Because these tasks are so closely related, there exists a lot of overlap in concepts and approaches. For example, knowing the key of a piece allows us to predict the kind of chords that will appear, and knowing the chords and their pattern of progression will also strongly suggest a key.

The remainder of this section discusses previous work for developing algorithms that will automatically perform these tasks. Music experts are then used to evaluate the findings of these automatic algorithms. We first give an overview of the feature extraction process for both symbolic and audio-based systems (section 2.4.1). This is an important step that determines how musical input is interpreted by the computer. We then organize tonality induction models we review by the type of input the system/algorithm takes in: symbolic or acoustic. Systems that require symbolic input (section 2.4.2) tend to be tonality induction

approaches motivated from a cognitive and psychological standpoint (as they are interested in how humans perceive and understand tonality [81, 34]). Since their interest lies in the structure and organization of the musical notes, a clean and accurate symbolic representation of the musical score (by way of a file format like $MIDI$) is used for input. Systems that take acoustic signals (section 2.4.3) are generally motivated by the MIR community. Their goal is to create applications that are readily useful to the everyday listener with hundreds of thousands of .mp3 (or other audio file formats) stored on their computer. At the end of the section, Tables 2.1-2.3 provide a summary of previous studies related to symbolic/audio key/chord estimation tasks.

2.4.1 Feature Extraction

Any system that perform automatic tonality induction requires music to be inputted into a computer program somehow. The process of digitizing music input into a format that is recognizable by a computer is called *feature extraction*. Below, we describe some common methods for feature extraction for two different types of input: symbolic and audio.

Symbolic Features

Symbolic input refers to an exact transcription of a piece of music. The bare minimum of such a transcription would include all of the notes and note durations present in the piece. Depending on the file format type, other musical attributes – such as tempo, meter, and loudness information – are also encoded. There are many symbolic file formats, with MIDI being undoubtedly the most popular. This is due to the fact that MIDI is not simply a system for musical notation (as many other formats are), but also acts as an industry specification by which digital musical instruments and computers connect and communicate with each other. This enables musicians to automatically recording and digitizing their performances on MIDI instruments, making it relatively easy and quick to create MIDI formatted music files. (The other option, is to manually enter music notation into software such as Finale or Sibelius, which takes a lot more time, but results

in more accurate transcriptions.)

Symbolic MIDI files are easy to interpret. Most MIDI file readers represent each note event in a MIDI file in the form of a tuple. For example, the MIDIToolBox [20] (for Matlab) that is used in this thesis encodes each MIDI song as a list of note events. Each note event is represented by the following seven values: (1) onset time (in beats), (2) duration (in beats), (3) MIDI channel, (4) MIDI pitch, (5) velocity, (6) onset time (in seconds), (7) duration (in seconds). The fourth element – MIDI pitch – maps the 88 notes on a keyboard (from $A0$ to $C8$) to *MIDI numbers* which span from 21 to 108.

Audio Features

Audio input is differs greatly from symbolic input, for all the reasons described in section 2.2.1. As we have seen, a combination of acoustics, physics and psychology make the interpretation of audio input quite complicated and requires several pre-processing steps before arriving at a representation that is useful for our musical task at hand. Many early approaches for tonality estimation adapt the findings of the symbolic, score-based models (from the previous section) to using acoustic signals as input. Traditionally, this was a two step process: first, polyphonic transcription techniques [9, 35] were used to identify the individual notes played. Then, the same tonal models used previously would be applied to the symbolic representation in order to do whatever task at hand (global/local key-finding, chord recognition, etc.). Unfortunately, this approach suffered greatly from recognition errors at the first stage, due to the presence of noise and overlapping harmonics of notes in the spectrum of the input audio signal.

Leman [49] is often credited as the first to propose a working tonality induction system that used acoustic input. This system is based on an auditory model using a Kohonen map – a self-organizing map created using artificial neural network techniques [37] with trained tonal centers for the recognition. However, the real key to his success was avoiding the first stage of unreliable note transcription. Instead, Leman used the spectrum of the audio signal to create an intensity map of twelve semitone pitch classes, very similar in concept to the 12-dimensional input

vector used in the Krumhansl-Schmuckler and Temperley key-finding algorithm. It is observed that this procedure collapses pure tones of the same pitch class, independent of octave, to the same frequency bin, and for complex tones, the harmonics also fall into particular, related bins. Representing the audio in this way directly preserved tonal and harmonic information, and served as a more robust representation than transcribed notes. Since then, the idea of collapsing frequency bands into chroma bins has become very popular feature for audio-based tonality induction methods. Many different approaches have been proposed; most are based on one of two frequency transforms: the DFT (discrete Fourier transform) or the Constant-Q transform.

DFT-Based Features. Fujishima [24] was among the first to create a tonal system that used a DFT-based acoustic feature. He called these features *pitch class profiles* (PCP), which are computed as follows: First, the algorithm transforms a fragment of the acoustic signal to a DFT spectrum: $X(k)$ for each $k = 0, 1, \ldots, N-1$ (where N is the total number of samples in the signal) calculated as follows:

$$X(k) = \sum_{n=1}^{N-1} e^{-2\pi ikn/N} \cdot x(n) \tag{2.7}$$

Since $x(n)$ is real $X(0), \ldots, X(N/2-1)$, or just the first half, gives the entire spectrum. The PCP is then computed from the DFT spectrum:

$$PCP(p) = \sum_{\ell s.t. M(\ell)=p} ||X(\ell)||^2 \tag{2.8}$$

where $M(\ell)$ is a table that maps a spectrum bin index to the PCP index. For example:

$$M(\ell) = \begin{cases} -1 & \text{if } \ell = 0, \\ \text{round}(12\log_2((f_s \cdot \frac{1}{N})/f_{ref})) \cdot \text{mod}12 & \text{if } \ell > 0 \end{cases} \tag{2.9}$$

Here, f_{ref} is the reference frequency that falls into the first PCP bin, $PCP(0)$. The term $(f_s \cdot \frac{1}{N})$ represents the frequency of the spectrum bin $X(\ell)$. Conceptually, these PCP vectors are very similar to Leman's work, in which each of the 12 bins represent the intensity of each of the twelve semitone pitch classes. For

instance, the first PCP element shows how strong the note C (given that the reference frequency corresponds to C) is within that segment of audio. Fujishima's PCP vectors soon became the basis for others who derived DFT-based features. Gomez [26] builds on Fujishima's work. She introduces HPCP features that modify the PCP features by 1) using higher resolution bins to avoid spectral leakage, and 2) explicitly considering the presence of harmonics. Some other works of interest includes Pauws [63], who computes a spectrum that is modeled as a combination of the spectral content of the perceptual pitch, and the musical background, and Chuan [13], who looks for spectral peaks of frequencies in the DFT and determines their relative strengths.

Others have proposed variations on these DFT-based feature vectors. For example, Gomez [26] builds on Fujishima's work introducing HPCP features that modify the original PCP features in the following ways: First, HPCP bins have a higher resolution than regular PCP bins, meaning that there are often more than 12 bins. Thus, instead of contributing to a single HPCP bin, each frequency makes a weighted contribution to multiple bins in a certain window around that frequency value. The closer that frequency value is to the bin; the higher the weight of contribution. This minimizes the estimation errors that arise due to tuning differences. Second, the HPCP computation explicitly considers the presence of harmonics. Recall that when a single note is played, spectral components will appear at frequencies of the different harmonics (i.e. $f, 2f, 3f, \dots$). In order to make these extra harmonic components contribute to the HPCP bin of its fundamental frequency, a weighting procedure is used such that each frequency f_i considered will make a decaying contribution as a harmonic frequency to bins that represent:

$$f_i, \frac{f_i}{2}, \frac{f_i}{3}, \dots, \frac{f_i}{\#\text{ of harmonics}} \tag{2.10}$$

Some other works of interest includes Pauws [63], who computes a spectrum that is modeled as a combination of the spectral content of the perceptual pitch, and the musical background, and Chuan [13], who looks for spectral peaks of frequencies in the DFT and determines their relative strengths.

Constant-Q-Based Features. Another type of frequency mapping is based on a frequency transform that is similar to the DFT, called the constant-Q transform. Many others have also used the constant-Q feature successfully in their key-finding algorithms and applications [66, 84, 27, 48]. The constant-Q transform was originally introduced by Brown [7], and then Brown and Puckette [8] later proposed an algorithm for more efficient computation. Like the DFT, the constant-Q transform is also a bank of filters that separates the input signal into multiple components. Each one of these components carries a single frequency subband of the original signal. The difference lies in how the filters are spaced. The constant-Q transform has geometrically spaced filters, such that the ratio between each subsequent filter is constant.

$$f_k = f_0 \cdot 2^{\frac{k}{b}} \tag{2.11}$$

Here, f_0 is the minimal center frequency, and b is the desired number of filters per octave. The above equation may be understood more intuitively as a recurrence relation:

$$f_{k+1} = f_k \cdot 2^{\frac{1}{b}} \tag{2.12}$$

It should be clear that spacing the filters in this way makes the ratio between frequency and interval (referred to as Q) stay constant and independent of k:

$$Q = \frac{f_k}{f_{k+1} - f_k} = \frac{f_k}{f_k \cdot 2^{1/b} - f_k} = (2^{1/b} - 1)^{-1} \tag{2.13}$$

When the minimal center frequency f_0 is set, for example, to that of midinote 0 (i.e. lowest possible C on the piano, with a frequency of about 8.17 Hz), then each successive filter will have a center frequency equal to the next musical note on the chromatic scale. We note that the Fourier transform in its original form does not have this property; its filters are spaced additively instead of geometrically, and thus calls for an extra step in which frequencies must be specifically mapped to chroma bins (i.e. the table $M(\ell)$ in Fujishima's work). In the end, the constant-Q transform retains a similar 12-dimensional vector indicating the contribution of each pitch class. For a more detailed explanation of how the constant Q transform

is computed, please refer to [7] and [8]. The first reference gives a derivation for the constant Q transform based on the Fourier transform, while the second gives a derivation based on the fast Fourier transform, making the overall computation much more efficient.

This summary details the most basic prototype of fourier-based features. Additionally, later in section 4.3, we describe two additional enhancements by Harte [27] and Mauch [57] that were use in our own experiments. We note, however, that there exists a large corpus of related work involving acoustic feature extraction that is out of the scope of this thesis.

2.4.2 Tonal Models for Symbolic Input

One of the first approaches for tonal induction is attributed to Longuet-Higgins, a pioneer in developing computation models for music cognition. In his earliest work [54], Longuet-Higgins proposed that musical notes can be neatly arranged in a two-dimensional harmonic network: one axis represents notes that are five scale steps apart, and the other axis represents notes that are a major third apart. He then observed that members of the same key form distinctive shapes in this harmonic network, and proposed a key-finding method based on a Shape Matching Algorithm (SMA) that used the shapes to identify the key of subjects of Bach's "Well-Tempered Clavier." Figure 2.8 shows an example of how different keys look in the harmonic network. This key estimation algorithm analyzes the different tones of the score in order to find whether or not they are contained in the diatonic major and minor scales. It then eliminates the keys that have a tone in the sequence which is non-diatonic on this key. If all keys are eliminated or there is more than one key candidate, they use a tonic-dominant rule to estimate the key. Several other researchers have proposed networks and representations for expressing harmonic relations for automatic key-finding. Some of these include Reimann's *Tonnetz* network [15] (a re-interpretation of Leonhard Euler's original work), Shepard's pitch helix [70], Lewin and Cohn's transformational theory within the Tonnetz [52, 14], Krumhansl's torus [41], and Chew's spiral array [11].

Another early work in the field is by Winograd [82], who proposes a method

for automatic tonal analysis using ideas derived from linguistics. He eloquently discusses the complexities of such analysis by comparing it to understanding language.

> One aspect of music highly amenable to [linguistic] treatment is the harmonic structure of tonal music. First of all, it is not trivial. It takes a rather sophisticated understanding of music to perform harmonic analysis (as opposed to simple chord identification) ... Further, even experienced theorists can disagree on the "reading" (parsing) to be assigned to a particular composition ... There is of course close agreement on all but the most detailed points, and a program should strive for an "intelligent" reading. This makes the problem interesting as a problem in artificial intelligence, not only for its own sake, but because of its clear relationship to the problem of parsing and understanding structures in other languages with a semantic component.

Winograd then proposed a generative grammar (based on the works of Chomsky [12]) that attempts to provide a neat and useful way of expressing the many structural and syntactic rules governing music composition. Figure 2.9 shows a simplified example of what a grammar of tonal harmony might look like. Winograd implemented his grammar in the form of a computer program that took as input the notes representing the harmonies for each beat. Winograd's program took a first pass through the music to label harmonies, and then a second pass to determine the function of the chords. In general, the program lacked generality and mostly focused on examining the effectiveness of Winograd's experimental

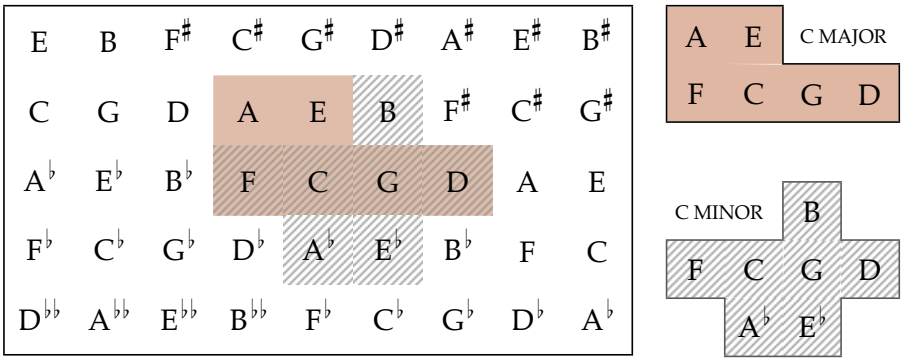


Figure 2.8: In Longuet-Higgins's key-finding algorithm, tonalities are mapped to a two dimensional structure, where different keys will take on unique shapes. Key-finding is done by recognizing these certain shapes.

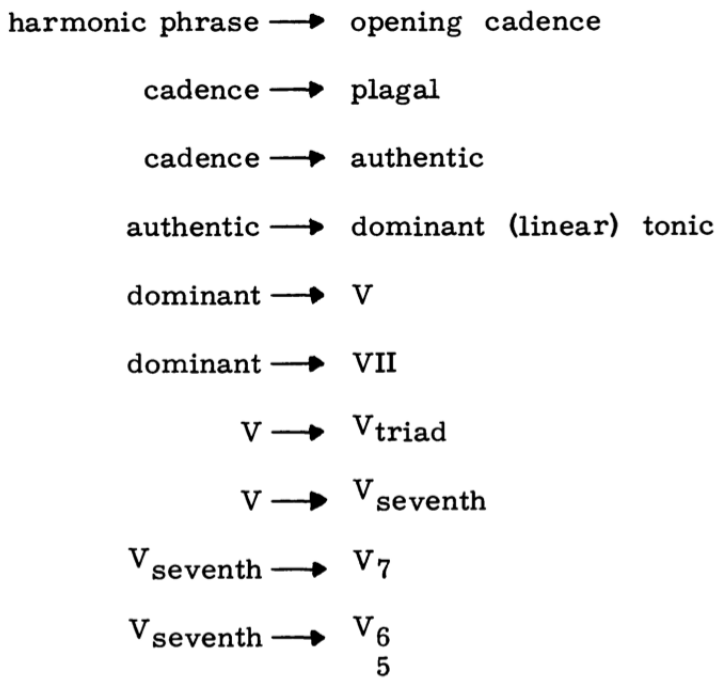


Figure 2.9: A simple generative grammar for tonal harmony by Winograd [82], who likens tonal harmonic analysis to parsing a semantic language.

grammars. It did, however, form the basis of several more complete and sophisticated harmonic analysis programs, such as BANALYZE [83] and CHORALE [19], both of which did extensive harmonic analysis (and even composed in the style) of Bach compositions.

Landmark work produced by Krumhansl and collaborators introduced the concept of "key-profiles" (used in a simple template-matching approach) that remains very popular to this day. Krumhansl's work began as a series of innovative psychology experiments called *probe-tone experiments* [43, 42, 41]. After some trial-and-error, experiments tried ensuring that both musicians and non-musicians would be on the same playing field. In short, the experiment asked subjects to listen to an incomplete scale (for example: C-D-E-F-G-A-B), and then rate (on a scale of one to seven) how well each of the 12 possible tones "complete" the scale. The definition of "completion" remains vague, however, Krumhansl's experiment revealed an innate sense in subjects (who were presumably raised listening to western tonal music) regarding what does and does not fit in a musical context. From

these experiments, a multidimensional scaling solution was used to assign precise, quantitative numbers that reflect the stability of each pitch in the context of a particular key. These numbers form a *key-profile* for that particular key. Figure 2.10 shows key-profiles for C major and C minor. Because of the cyclic relationship between different keys within each mode, key-profiles for other keys can be obtained by simple transposition.

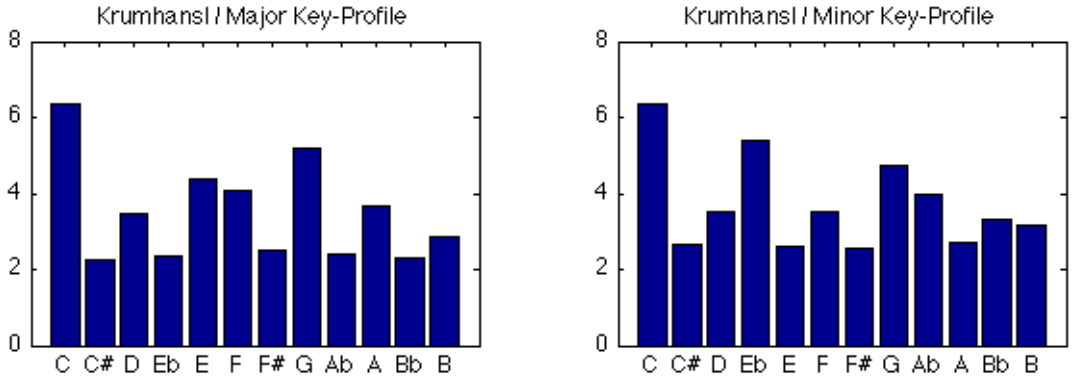


Figure 2.10: C major and C minor key-profiles derived from Krumhansl and Kessler using probe-tone studies. These profiles indicate the prevalence of each of the 12 pitch classes in a key.

The key-profile was also the point of departure for Krumhansl and Shmuckler's key-finding algorithm [41]. The premise behind the algorithm is simple: If one can determine the salience of each of the 12 semitones within a selected segment of music, then one can take the resulting 12-component vector of salience and correlate it with the key-profiles of all major and minor keys. The key-profile with the highest correlation determines the key. In most common applications of this key-finding algorithm, salience was equated with the total duration of each chromatic pitch within the musical passage. Initial tests showed that it did reasonably well on just the first four tones of Bach, Shastokovich, and Chopin preludes. Consequently, in addition to identifying the global key of a piece of music, the algorithm also worked reasonably well for modulation tracking and doing localized key-finding (achieved by simply applying the algorithm to short segments of music). This kind of simplicity and flexibility undoubtedly contributed to the algorithm's popularity as it became the key-finding algorithm of choice (for symbolic music input, or used

to adapt to audio music input) in many later applications, [65, 26, 32]. As with any successful model, it also spurred a number of researchers to approach key-finding by extending Krumhansl and Kessler's key-profiles [74, 56, 73].

Temperley [74] then built upon Krumhansl's work and contributed in several ways. First, Temperley chose to discard note duration information in the 12-dimensional input vector that represents the musical score. Instead, each pitch is given a 0 or 1 indicating if it was present in the passage of music. This was done after he observed that a repeated note in the passage would be given too much weight. Second, Temperley specifically addressed local key-finding and modulation tracking by performing the Krumhansl-Schmuckler algorithm on small segments of music (a half-measure, but no longer than one second). He further proposed that once in a certain key, most songs prefer to remain in that key unless there is strong evidence to do otherwise. Thus, to handle this, the model imposes a "change penalty" if the predicted key for one segment differs from the next. Using dynamic programming, he was able to find the optimal sequence of local keys within a song. In a later publication, Temperley also empirically derived a new set of key-profiles by averaging frequency counts of each pitch from a corpus of excerpts taken from the Kostka and Payne music theory textbook [38]. Appendix A.1 shows a sample of many new key-profiles developed by researchers, including Temperley.

2.4.3 Tonal Models for Acoustic Input

In this section, we present some models of tonality induction that take acoustic signals as input. We categorize these models into two groups, based on their underlying approach. The first group uses template-based matching approaches, where a set of key-profiles (such as the ones from Krumhansl-Kessler [41]) are pre-defined, and classification is done by assigning each acoustic feature vector to the key whose key-profile is most similar to it. The second group models the music data using machine learning methods and infers the key-profile directly from the dataset; each model has its own heuristics for performing the classification stage.

Template-based matching approaches

The techniques described in this section first use pre-defined key templates to represent each of the 24 keys, then correlate these templates with the feature vector computed from audio. The key template that yields the maximum correlation is the key that is assigned to the segment of audio represented by that feature vector. We refer to this as the *maximum key-profile correlation* (MKC) algorithm.

We begin with Fujishima [24], whose PCP vectors were used to create the first comprehensive audio chord recognition system. His system supported 27 possible chord types. Each chord type, c, was represented with a binary, 12-dimensional template vector, $CTT_c(p)$, where p is one of the 12 pitch classes. If the chord type c has the pitch class p in it, then $CTT_c(p)$ is set to 1, and 0 otherwise. For instance, the CTT for a major 7th chord with root C is $(1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1)^T$. Each segment of music was then classified as the chord type that produced the smallest euclidean distance between its CTT and the PCP. He reports very high accuracy – up to 94% on the 208 frames of the first 50 seconds from the opening theme of Smetana's "Moldau", though his experiments are restricted to monophonic music data only. Harte and Sadler [27] who derived the constant-Q based high-resolution chroma features also used a similar matching approach for a chord recognition system. They defined 48 binary chord templates (triads of major, minor, diminished and augmented for each pitch class). They used two full Beatles albums (28 songs total) for evaluation and obtained an accuracy of 62.4%.

Pauws [63] proposed a key-finding algorithm that was based on the human auditory system and how they perceived music. He computed chroma features much like Fujishima, but added several extra pre-processing stages to obtain a harmonically compressed spectrum (similar to Gomez's extended work with the HPCPs [26]). These steps included low-pass filtering, enhancing spectral components to cancel out spurious peaks, logarithmic frequency scaling, and the use of a weighting function similar to human auditory sensitivity. Using the MKC algorithm with Krumhansl-Kessler key-profiles, Pauws evaluated his system on 237 classical piano pieces that included Bach, Shastakovich, and Chopin preludes. The highest accuracy of 75.1% was obtained by evaluating the first 30 seconds of each

song, from the beginning. When adding relative, dominant, sub-dominant and parallel errors as correct matches, accuracy increased to 94.1%.

Zhu [84] proposes an interesting variation on the MKC algorithm for key-finding. His approach is based on the assumption that the scale root note plays the foremost role in tonal music (i.e. one of the 12 pitch classes), as opposed to the mode of the key (i.e. major or minor). In his approach, *pitch profile features* are first extracted. These are very similar to basic key-profiles, but with the following enhancements: First, a constant-Q transform of the input signal is taken. Second, Zhu accounts for mistuning of the recording by calibrating against the standard 440 Hz (similar to Gomez [26] and Harte's [27] approach with high-resolution chroma bins). Third, to address the presence of harmonics, partials from each note are extracted. In the last step, Zhu does something called *consonance filtering* which extracts only the prominent partials that are relevant to the music scale (i.e. all "accepted" partials within close proximity in the music must belong to a single diatonic key; otherwise, it is eliminated). One of the most novel parts of their algorithm is the key detection stage: From this pitch profile feature, only the diatonic scale root is predicted first. Once this is done, both the diatonic scale root and the pitch profile feature are used to predict the entire key (i.e. major or minor). Using 60 pieces of pop music and 12 pieces of classical music (all from Vivaldi's "The Four Seasons"), the accuracy they obtained was 90% and 83.3%, respectively. The higher accuracy in pop music than classical, they argue, is due to a smaller number of modulations.

Izmirli [32]'s algorithm not only does global key-finding, but tracks modulations as well. First, conventional chroma vectors are extracted from the input signal and an adaptive tuning algorithm is then applied to correct mis-tuned recordings. While others move to the next stage of predicting the key for the entire song based on the extracted features, Izmirli first performs a segmentation step. This segmentation aims to find the points in which the key changes; each resulting segment should then correspond to a part of the music that stays in the same local key. Segmentation is done using a variation of *non-negative matrix factorization* (NMF) [46], an algorithm that aims to decompose an $(n \times m)$ matrix V, into a

product of two smaller, rectangular matrices W and H. To accomplish segmentation, the V matrix is a $(12 \times m)$ matrix which holds averaged chroma vectors for small windows of time (i.e. first 5 seconds of chroma vectors in the song are averaged, and represent the first column of V). Upon decomposition, the W matrix will hold as many columns as there are segments in the song; each segment will correspond to a 12-dimensional column vector – similar in concept to the original chroma vector in the sense that it strongly suggests tonality information for that segment. In effect, this is similar to vector quantization, where as many similar and contiguous chroma vectors are grouped together as possible, before forming a new group (i.e. modulation to a different key). To perform modulation tracking, once segmentation is finished, a MKC algorithm using Krumhansl-Kessler key-profiles is applied to each discovered segment, and predicts a local key for that segment. For overall key-finding, the key of the first segment is taken to be the overall key of the piece.

Machine learning approaches

So far, we have looked at audio key-finding algorithms that use a pre-defined musical template in their key detection stage to estimate the overall (or local) key. While this has produced good results, this method has several disadvantages. First, there is a disconnect between the numbers obtained through cognitive experiments (i.e. the probe-tone studies) and how it actually relates to the low-level chroma features extracted from real audio signals. Second, the performance of these algorithms rely very heavily on these rigid key-profiles; they may not adapt well to diverse types of music, and do not acknowledge the specific test data at hand. Therefore, researchers have also been interested in alternative approaches that will learn musical rules from the musical audio and (sometimes) specific test data set. Machine learning algorithms provide many options here by allowing models to be built on audio training data (that is assumed to be similar to the test data), instead of heuristics or hand-crafted music theory rules.

To the best of our knowledge, Ellis and Sheh [69] were the first to use hidden Markov models (HMM) for tonality induction – specifically for local key-finding

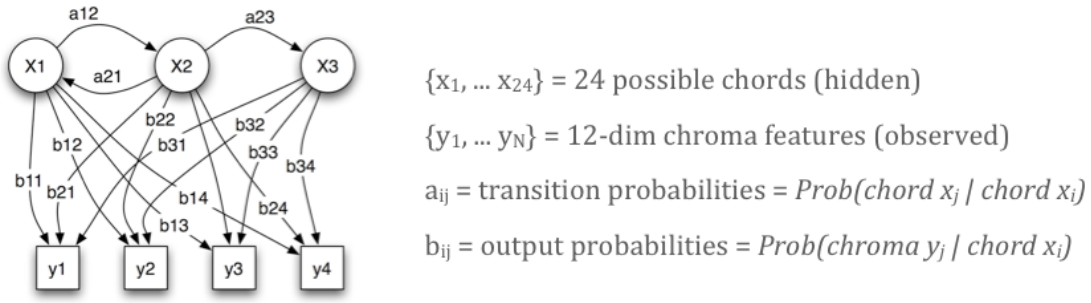


Figure 2.11: Summary of a common approach to key-finding using HMMs. Chords are modeled as hidden nodes that generate the observed chroma features from some distribution (usually Gaussian).

(i.e. chord estimation), as well as overall key detection. Since then, HMMs have become the machine learning algorithm of choice for this problem as researchers quickly recognized that music can be conveniently modeled as time-series data. The graphical model for the HMM is shown in Figure 2.11. Each node represents a time step, where time moves from left to right. The top row are "hidden" states and the bottom row are "observed" states. For the task of chord estimation, the hidden nodes usually correspond to the chord label at that particular time step (since we do not know what it is), and the observed nodes correspond to the extracted features from the audio signal that we *do* observe. Each of the hidden nodes can take on one of N possible states, where N is the total number of different chord types that the system supports. The HMM requires the following model parameters in order to be fully defined:

Initial probability distribution: An N-dimensional vector indicating the prior probability of each hidden state (i.e. chord type) occurring

Transition probabilities: An $(N \times N)$ matrix indicating probability of transitioning from one hidden state to another, for all possible pairs of states.

Observation probability distribution: The distribution by which the observed chroma features are generated from the hidden chord labels. The Gaussian distribution is frequently used and a set of N Gaussian parameters (mean and variance) are also included in the model parameters.

In Ellis and Sheh's work [69], overlapping windows of 100 ms were used to compute PCP features, which were then used to train an HMM with a total of $N = 147$ different chord types (this includes 7 different chord families: major and minor, major and minor sevenths, dominant sevenths, augmented, and diminished). The observation probabilities were assumed to be gaussian, and all model parameters were learned from the data using the *expectation maximization* (EM) algorithm [16] (a popular algorithm for maximum likelihood estimation of model parameters in the presence of hidden variables). Once model parameters are estimated, the Viterbi algorithm [80] is used to recover the chord sequence (i.e. the sequence of hidden states) with the highest likelihood given the model parameters. This best path assigns a chord to every node (representing a 100 ms segment of audio), resulting in a time-aligned chord transcription. The authors tried several different HMM configurations; the best of which used a forced alignment of known chord labels without segmentation information in order to more accurately learn better model parameters. Their system was then evaluated on 20 songs from three early Beatles albums, with 18 songs for training and 2 songs held out for test.

Chai [10] adopted a similar HMM based model for the same task of modulation tracking/chord-estimation. Her system extracts 24-dimensional PCP features from the input audio signal (for higher resolution of chroma bins). Similar to Zhu [84], the key detection step is broken into two different stages: 1) Detect the key without considering its mode – for example, both C major and A minor (which are relative keys in that they have the exact key signatures) are considered the same key, and 2) Detect the mode, major or minor. Note that this is different from Zhu's work – the first step in Zhu's system grouped together parallel keys (i.e. C minor and C major) where as Chai's work groups together relative keys (i.e. C major and A minor). Both stages in Chai's system are accomplished through HMMs. In the first stage, the HMM has 12 possible states, corresponding to the 12 keys with unique key signatures. In the second stage, an HMM with 2 possible states, corresponding to major or minor, determines the mode. We note that instead of using the conventional Gaussian for observation distributions, a cosine distance is used instead (even though it is not a proper probability distribution).

The cosine distance is measured between the observed 24-dimensional chroma vectors, and pre-defined, binary key or mode (depending on the stage) templates. The binary key template contains 1's for all notes that are in that key's diatonic scale and 0 otherwise (much like Temperley's flat key-profile), while the binary mode template contains a 1 for the dominant degree in the major template, and 1 for the tonic degree in the minor template, with 0 for every other scale degree. As a departure from Ellis and Sheh's work, all other HMM parameters (initial distributions, transition probabilities) are manually assigned and hand-tuned according to basic music theory principles. To that effect, this model is "safer" for a small data set, but also does not completely leverage the power of the learning algorithm to automatically tune those parameters.

While others used represented chord labels as hidden states in HMMs, and used the resulting model to do chord estimation, Peeters [64] proposes an HMM-based key-finding algorithm that works slightly differently: First, training is done only for a major and minor model; these HMMs contain anywhere from three to nine different states, which have no intuitive meanings (i.e. they do not correspond to chord labels as they did in Ellis and Chai since Peeters is not interested in decoding the sequence of hidden chord labels). Once model parameters are learned for both the major and minor HMMs, Peeters performs circular permutation on the learned Gaussian observation distribution parameters, which results in 24 additional major/minor HMM models. The final key estimation is done by computing the likelihood of each key model given an unknown chroma sequence, and selecting the model with the maximum likelihood. Like Ellis's work, Peeter's key-finding system is purely data-based and incorporates no prior musical knowledge. The advantages, he argues, are: 1) it does not make assumptions about the presence of harmonics of the pitch notes in chroma vectors, and 2) it does not require the choice of a specific key-profile such as Krumhansl/Temperley, or even assumption of Diatonic scales, etc., and 3) it allows for key modulations over time through transition probabilities. Unlike Ellis, however, Peeter takes on only 24 different chord types – 12 major and 12 minor – which is arguably more manageable for tests with small training sets.

Lee notes that Peeter's accuracy may have been higher had he chose to explicitly assign the states of the initial major and minor models as hidden chord labels, similar to what his predecessors had done. Thus, Lee proposes such a system in [48]. Like Peeter, Lee also trains 24 HMM models, one for each key. The likelihood of each HMM is then computed for each unknown chroma vector sequence (representing either an entire song, or a segment of a song) and the key whose HMM yields the maximum likelihood is assigned to that chroma sequence. The main difference, however is that instead of constructing states with no real meaning, Lee trains his HMMs such that each observed state (which corresponds to a chroma vector) is drawn from a hidden state that corresponds to a hidden chord label for that chroma vector. In doing so, Lee's model simultaneously performs overall key-finding *and* chord estimation at the same time. Lee argues that this is a more robust model as key-finding and chord estimation are closely related tasks. However, this is not his only contribution. In order to tackle the problem of having little or no labeled training data (as so many researchers had faced before), Lee suggested taking available symbolic music data (i.e. MIDI scores, of which many can be found online, especially for classical music), running symbolic key-finding and chord estimation algorithms, and then synthesizing time-aligned audio from the symbolic data. Since the audio and MIDI data will be perfectly time-aligned, the results of tonality induction from the MIDI data (such as global and local key predictions) can be used as training labels for the audio data. Lee argues that key-finding and chord estimation algorithms for MIDI data is accurate enough that it can be used as ground-truth labels for the audio version.

2.4.4 Summary

Having presented a sweeping overview of many different kinds of tonality induction methods, we briefly summarize our findings and try to compare performance across models. First, we note that evaluation criteria for symbolic tonality induction is much more varied than that of audio tonality induction. In general, fewer pieces of music are used for evaluation (as some models require a large amount of time for the evaluation of even a single piece), and often the aim of the work

goes beyond simply finding the correct key of the song. In that sense, it is quite difficult to compare the results reported from those models. We see that tonality induction methods based on audio, however, are much more coherent in terms of goals and applications. The fact that these algorithms are made for audio as input, strongly suggests that they are motivated by useable, everyday applications.

As we have seen, most models choose to pursue either the task of key-finding and/or chord estimation. To have a bird's eye view, we provide a summary of the previous work discussed above, as well as some others not discussed here, in Tables 2.1 through 2.3. Each previously published work indicates the main task at hand (key-finding or chord-estimation) as well as what kind of input data they focus on (symbolic or audio). We note several things about the compiled tables:

1. Some key components of each model are listed; however, it may not be comprehensive. Please refer to the author's original work for more details.

2. All accuracies are approximate, and though we tried to find the best performance from each authors' work, we cannot guarantee it to be exhaustive.

3. Some accuracies are averaged over multiple datasets (within the same genre) that an author reported.

4. MIREX scores indicate a type of composite score that rewards points to errors that predict closely related keys. This is discussed more in section 4.6.2 when we also use MIREX scores to report our accuracies

We will refer back to these tables when we discuss and evaluate our own models later on in Chapters 3 and 4.

2.5 Conclusion

In this chapter, we have presented the scientific background on which this dissertation work is largely based. We first presented some of the fundamentals of physical music making, as well as the building blocks of music composition,

Table 2.1: Summary of some notable previous studies related to symbolic/audio key/chord estimation, listed by year. See section 2.4.4 for more details about this table. This table is continued over the next two pages.

Year	Authors	Task	Model	Classes	Dataset	Evaluation
1990	Krumhansl [41]	Symbolic key	Key-profile correlation	24 keys	168 classical	Key accuracy of 75%
1999	Fujishima [24]	Audio chord	Key-profile correlation with PCP features	324 chords	1 classical	By example
2000	Temperley [76]	Symbolic local key	Key-profile correlation	24 keys	46 classical	Local key accuracy of 85%
2003	Raphael & Stoddard [67]	Symbolic key, Chord	Unsupervised HMM + Viterbi	168 chords	3 classical	By example
2003	Sheh & Ellis [69]	Audio chord	Force-aligned HMM with multivariate Gaussian emissions + Viterbi	84 chords	20 Beatles	Chord accuracy of 75%
2004	Pauws [63]	Audio key	Key-profile correlation + human auditory modeling	24 keys	237 classical	Key-prediction accuracy of 75%

Table 2.2: Summary of some notable previous studies, continued from previous page.

Year	Authors	Task	Model	Classes	Dataset	Evaluation
2005	Bello & Pickens [4]	Audio chord	HMM with hand-chosen parameters + Viterbi	24 chords	28 Beatles	Chord accuracy of 75%
2005	Harte & Sandler [27]	Audio chord	Key-profile correlation	48 Chords	28 Beatles	Chord accuracy of 62%
2005	Paiement et al. [62]	Symbolic chord	Unsupervised graphical model with binary tree structure	Many, using "Substitution Distance"	52 Jazz standards	Held-out likelihood of generated chord sequences
2005	Zhu et. al [84]	Audio key	Key-profile correlation + consonance filtering + rules	24 keys	12 classical, 60 pop	Key accuracy of 83% (classical), 90% (pop)
2005	Chai [10]	Audio key	HMM using cosine distance for emissions and hand-chosen parameters	24 keys	10 classical	Precision/recall graph by example
2006	Gomez [25]	Audio key	Key-profile correlation with enhanced HPCP features	24 keys	1000+ classical	Key accuracy (MIREX score) 86%

Table 2.3: Summary of some notable previous studies, continued from previous page.

Year	Authors	Task	Model	Classes	Dataset	Evaluation
2006	Peeters [64]	Audio Key	24 key-independent supervised HMMs with GMM emissions + Viterbi	24 keys	302 classical	Key accuracy of 85%
2007	Izmirli [32]	Audio local key	Key-profile correlation with NMF for segmentation	24 keys	169 classical, 17 pop	Local key accuracy of $\sim$ 88.3% (pop), 76.9% (classical)
2008	Lee & Slaney [48]	Audio key, chord	24 supervised, key-dependent HMMs with Gaussian emissions + Viterbi	24 keys, 36 chords	28 Beatles, 96 Classical	Accuracy 75%/97% Beatles chord/key; 85% classical key
2009	Noland [61]	Audio key	Supervised HMM with hand-chosen parameters + varying DSP parameters	24 keys	48 Baroque, 110 Beatles	Key accuracy of 98% (Baroque), 71% (Beatles)
2010	Mauch [58]	Audio chord	Supervised dynamic Bayesian network with NNLS chroma	24 chords	210 pop	Chord accuracy (MIREX score) 80%
2010	Rocher [68]	Audio local key, chord	Weighted acyclic graph + dynamic programming	24 keys, chords	174 Beatles	Accuracy of 74.9% (chord), 62.4% (key)

including elements like pitch, chords, melodies, harmonies, rhythms, tempo, loudness, and duration. We then worked our way up to higher-level concepts such as tonality, which all other musical building blocks evolve around. Some background in theoretical/cognitive studies regarding how humans understand and perceive tonality set us up for an entire field devoted to automatic tonality induction. We saw that early on, interest in tonality induction stemmed from a cognitive and theoretical standpoint, and thus largely involved symbolic music representations. More recently, interest in tonality induction has shifted, with the goal of being able to perform automatic analysis on audio music files, which is more prevalent and relevant to personal music search and organization applications.

As such, we presented an overview of tonality induction methods for acoustic input, largely focusing on two concrete tasks: key-finding and modulation tracking/chord-estimation. We looked at two broad categories of methods: template matching algorithms and machine learning based algorithms. We saw that template-matching algorithms are efficient and require no training data. However, their performance is very much dependent on the pre-defined templates or profiles, which may not be universal, and can easily vary from genre to genre. On the other hand, machine learning algorithms can learn key and chord distributions on-the-fly, based on a given data set; it can be constructed to not be specific to any particular genre of music. Models such as the HMM can also elegantly model transitions between local key changes, using a probabilistic approach, as well as how these local keys contribute to the global key of the song. The bottleneck of such algorithms, however, is the availability of training data, on which the performance is heavily dependent. This is especially problematic in supervised learning, where the training data must have ground truth that must be provided by humans, involving a great amount of time and labor.

In the next chapter, we propose a novel key-finding and chord-estimation algorithm that is quite different from all of the models that we have discussed so far. In chapter 3, we propose a model for symbolic music, and in chapter 4 we adapt the same model for audio music. Both models address the issue of lack of training data by using *unsupervised learning*. In this framework, essentially no

labeled data is needed, thus reducing the amount of manual labor required to get such a model to perform well. Our models also assume little prior music knowledge, and is flexible enough to be adapted to other genres of music, and potentially other tonal systems as well.

Chapter 3

Harmonic Analysis for Symbolic Music

3.1 Introduction

In this thesis, we show how to perform key analysis and learn musical key-profiles automatically from the statistics of large music collections. This chapter focuses on the use of symbolic music input, while the next chapter will focus on audio music input. Unlike previous studies, we take a purely data-driven approach that does not depend on extensive prior knowledge of music or supervision by domain experts. Based on a model of *unsupervised* learning, our approach bypasses the need for manually key-annotated musical pieces, a process that is both expensive and prone to error. As an additional benefit, it can also discover correlations in the data of which the designers of rule-based approaches are unaware. Since we do not rely on prior knowledge, our model can also be applied in a straightforward way to other, non-western genres of music with different tonal systems.

3.2 LDA-MIDI Model

Our model is based on Latent Dirichlet Allocation (LDA) [6], a popular probabilistic model that aims to find a short descriptions that can summarize the

contents of a large corpus. Originally proposed in the context of text document modeling, LDA posits that one way of summarizing the content of a text document quickly is to look at the set of words it uses. Because words carry very strong semantic information, documents that contain similar content will most likely use a similar set of words. As such, mining an entire corpus of text documents can expose sets of words that frequently co-occur within documents. These sets of words may be intuitively interpreted as *topics* and act as the building blocks of the short descriptions.

More formally, LDA is an unsupervised, probabilistic, generative model for discovering latent semantic topics in large collections of text data. Each discovered topic is characterized by its own particular distribution over words. Each document is then characterized as a random mixture of topics indicating the proportion of time the document spends on each topic. This random mixture of topics is essentially our "short description": It not only expresses the semantic content of a document in a concise manner, but also gives us a principled approach for describing documents quantitatively. We can now compare how similar one document is to another by looking at how similar the corresponding topic mixtures are.

Our variant of LDA is based on the premise that musical pieces in the same key use similar sets of pitches. Roughly speaking, our model treats each song as a "document" and the notes in each beat or half-measure as a "word". The goal of learning is to infer harmonic "topics" from the sets of pitches that commonly co-occur in musical pieces. These harmonic topics, which we interpret as keys, are expressed as distributions over the twelve neutral pitch-classes – very similar in nature to the key-profiles that we have talked about in previous work [41, 73]. Table 3.1 summarizes the relationship between elements of LDA and our model, LDA-MIDI. The remainder of this section describes our probabilistic topic model, first developing the form of its joint distribution, then sketching out the problems of inference and parameter estimation.

Table 3.1: Summary of analogous elements between the original LDA model, and our LDA-MIDI model, which performs harmonic analysis on music pieces.

Original LDA	LDA-MIDI
a word *contains* semantic information	**notes in segment** *contain* harmonic information
document *expressed as* a mixture of topics	**song** *expressed as* a mixture of keys
topic *defined as* a distribution over words	**key** *defined* as a distribution over pitches

3.2.1 Notation and Terminology

Below are some important notation and terminology that will be used through the thesis. We try to parallel these terms with the original LDA notation as closely as possible.

1. A *note* $u \in \{\mathrm{A}, \mathrm{A}\sharp, \mathrm{B}, \ldots, \mathrm{G}\sharp\}$ is the most basic unit of data. It is an element from the set of neutral pitch-classes. For easy reference, we map these pitch-classes to integer note values 0 through 11. We refer to $V = 12$ as the vocabulary size of our model.

2. A *segment* is a basic unit of time in a song (e.g., a measure). We denote the notes in the nth segment by $\mathbf{u}_n = \{u_{n1}, \ldots, u_{nL}\}$, where $u_{n\ell}$ is the ℓth note in the segment. Discarding the ordering of the notes, we can also describe each segment simply by the number of time each note occurs. We use x_n to denote the V-dimensional vector whose jth element x_n^j counts the number of times that the jth note appears in the nth segment.

3. A *song* s is a sequence of notes in N segments: $s = \{\mathbf{u}_1, \ldots, \mathbf{u}_N\}$. Discarding the ordering of notes within segments, we can also describe a song by the sequence of count vectors $X = (x_1, \ldots, x_N)$.

4. A music *corpus* is a collection of M songs denoted by $\mathcal{S} = \{s_1, \ldots, s_M\}$.

5. A *topic* z is a probability distribution over the vocabulary of $V = 12$ pitch-classes. Topics model particular groups of notes that frequently occur together within individual segments. In practice, these groupings should contain the principle set of pitches for a particular musical key. Thus, we interpret each topic's distribution over twelve pitch-classes as the key-profile for a musical key. We imagine that each segment in a song has its own topic (or key), and we use $\mathbf{z} = (z_1, z_2, \ldots, z_N)$ to denote the sequence of topics across all segments. In western tonal music, prior knowledge suggests to look for $K = 24$ topics corresponding to the major and minor scales in each pitch-class. Section 3.2.4 describes how we identify the topics with these traditional key-profiles.

With this terminology, we can describe our probabilistic model for songs in a musical corpus. Note that we do not attempt to model the order of note sequences within a segment or the order of segments within a song. Just as LDA for topic modeling in text treats each document as a "bag of words", our probabilistic model treats each song as a "bag of segments" and each segment as a "bag of notes".

3.2.2 Generative process

Our approach for automatic key-profiling in music is based on the generative model of LDA for discovering topics in text. However, instead of predicting words in documents, we predict notes in songs. Our model imagines a simple, stochastic procedure in which observed notes and key-profiles are generated as random variables. In addition, we model the key-profiles as latent variables whose values must be inferred by conditioning on observed notes and using Bayes rule.

We begin by describing the process for generating a song in the corpus. First, we draw a topic weight vector that determines which topics (or keys) are likely to appear in the song. The topic weight vector is modeled as a Dirichlet random variable. Next, for each segment of the song, we sample from the topic weight vector to determine the key (e.g., A minor) of that segment. Finally, we repeatedly draw notes from the key-profile until we have generated all the notes in the segment. More formally, we can describe this generative process as follows:

1. For each song in the corpus, choose a K-dimensional topic weight vector θ from the Dirichlet distribution:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \prod_i \theta^{\alpha_i - 1}. \tag{3.1}$$

   Note that α is a K-dimensional corpus-level parameter that determines which topics are likely to co-occur in individual songs. The topic weight vector satisfies $\theta_i \geq 0$ and $\sum_k \theta_k = 1$.

2. For each segment indexed by $n \in \{1, \ldots, N\}$ in a song, choose the topic $z_n \in \{1, 2, \ldots, K\}$ from the multinomial distribution $p(z_n = k|\theta) = \theta_k$.

3. For each note indexed by $\ell \in \{1, \ldots, L\}$ in the nth measure, choose a pitch-class from the multinomial distribution $p(u_{n\ell} = i|z_n = j, \beta) = \beta_{ij}$. The β parameter is a $V \times K$ matrix that encodes each topic as a distribution over $V = 12$ neutral pitch-classes. Section 3.2.4 describes how we identify these distributions as key-profiles for particular musical keys.

This generative process specifies the joint distribution over observed and latent variables for each song in the corpus. In particular, the joint distribution is given by:

$$p(\theta, \mathbf{z}, s|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) \prod_{l=1}^{L_n} p(u_{nl}|z_n, \beta). \tag{3.2}$$

Figure 3.2 (a) shows the graphical model for the joint distribution over all songs in the corpus. As in LDA [6], we use plate notation to represent independently, identically distributed random variables within the model. Whereas LDA for text describes each document as a "bag of words", we model each song as a "bag of segments", and each segment as a "bag of notes". As a result, the graphical model in Figure 3.2 (a) contains an additional plate beyond the graphical model of LDA for text. To summarize how our model works, Figure 3.1 gives a more visual overview of the parameters that are involved in this generative process (as well as their musical representations).

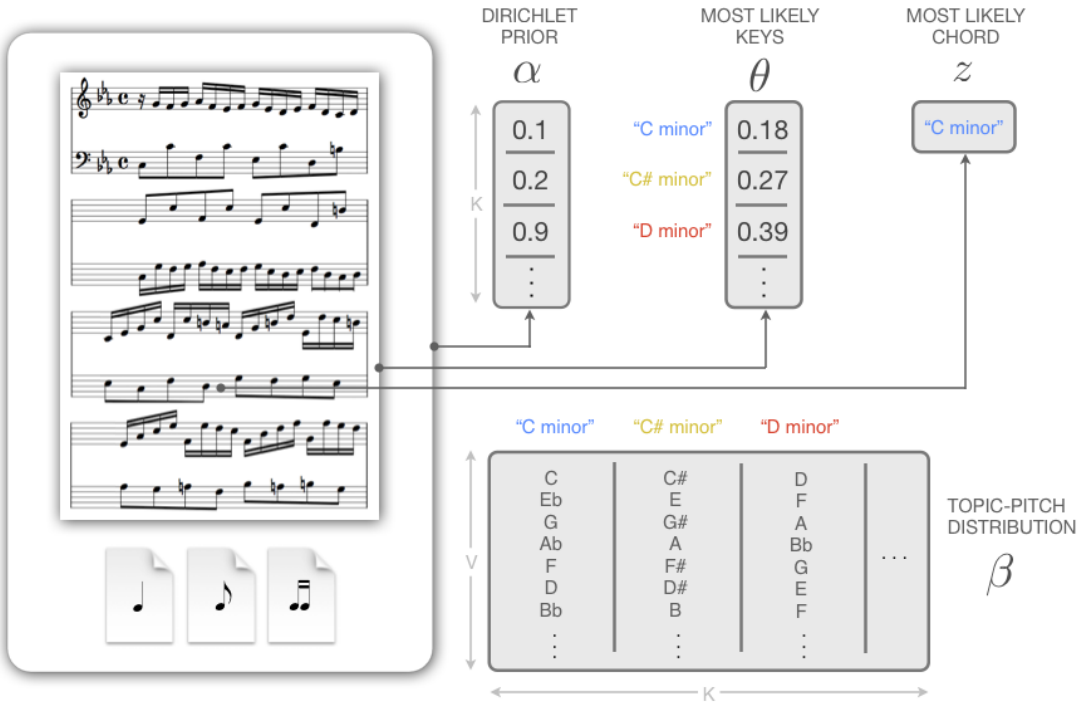


Figure 3.1: An overview of all the parameters involved in the generative process. Here, the outer box on the left hand-side represents an entire corpus of songs (with three small documents at the bottom indicating other songs in the corpus). The large musical score in the center represents the "current" song being analyzed. We see that α, the Dirichlet prior is associated with the entire corpus, as the single α parameter gives key proportions over all of the songs in the corpus. The θ parameter is learned at the song level, and gives a distribution over keys for that song (i.e. key-finding). The z parameter is learned at the segment level, and gives a distribution over keys for each segment in the song (i.e. chord-finding). Finally, the β matrix defines each of the 24 musical keys as distributions over pitches; the weights given by this distribution can be viewed as key-profiles.

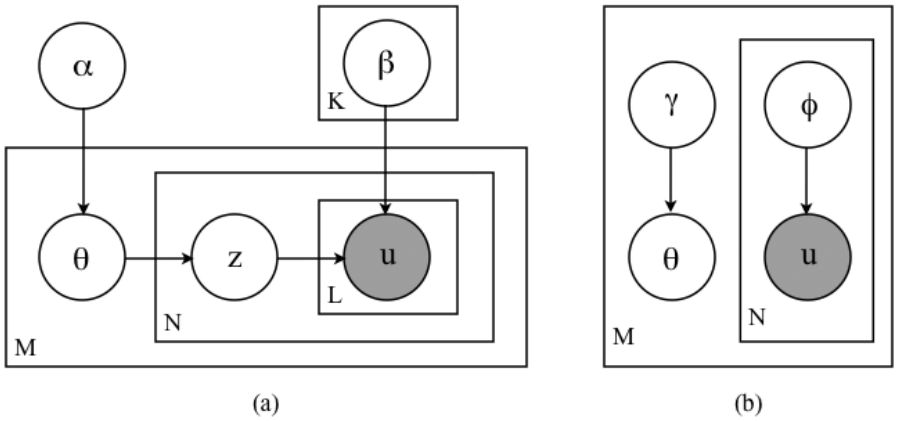


Figure 3.2: (a) Graphical representation of our model and (b) the variational approximation for the posterior distribution in eq. (3.5).

3.2.3 Inference and learning

The model in eq. (3.2) is fully specified by the Dirichlet parameter α and the musical key-profiles β. Suppose that these parameters are known. Then we can use probabilistic inference to analyze songs in terms of their observed notes. In particular, we can infer the main key-profile for each song as a whole, or for individual segments. Inferences are made by computing the posterior distribution

$$p(\theta, \mathbf{z}|s, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, s|\alpha, \beta)}{p(s|\alpha, \beta)} \tag{3.3}$$

following Bayes rule. The denominator in eq. (3.3) is the marginal distribution, or likelihood, of a song:

$$p(s|\alpha, \beta) = \int p(\theta|\alpha) \prod_{n=1}^{N} \sum_{z_n=1}^{K} p(z_n|\theta) \prod_{l=1}^{L_n} p(u_{nl}|z_n, \beta) \; d\theta. \tag{3.4}$$

The problem of learning in our model is to choose the parameters α and β that maximize the log-likelihood of all songs in the corpus, $\mathcal{L}(\alpha, \beta) = \sum_m \log p(s_m|\alpha, \beta)$. Learning is unsupervised because we require no training set with key annotations or labels.

In latent variable models such as ours, the simplest approach to learning is maximum likelihood estimation using the Expectation-Maximization (EM) algorithm [16]. The EM algorithm iteratively updates parameters by computing

expected values of the latent variables under the posterior distribution in eq. (3.3). In our case, the algorithm would iteratively alternate between an E-step, which represents each song in the corpus as a random mixture of 24 key-profiles, and an M-step, which re-estimates the weights of the pitch classes for each key-profile. Unfortunately, these expected values cannot be analytically computed; therefore, we must resort to a strategy for approximate probabilistic inference. We have developed a variational approximation for our model based on [33] that substitutes a tractable distribution for the intractable one in eq. (3.3), described below.

Variational Inference

The variational approximation for our model substitutes a tractable distribution for the intractable posterior distribution that appears in eq. (3.3). At a high level, the approximation consists of two steps. First, we constrain the tractable distribution to belong to a parameterized family of distributions whose statistics are easy to compute. Next, we attempt to select the particular member of this family that best approximates the true posterior distribution.

Figure 3.2 (b) illustrates the graphical model for the approximating family of tractable distributions. The tractable model $q(\theta, \mathbf{z}|\gamma, \phi)$ drops edges that make the original model intractable. It has the simple, factorial form:

$$q(\theta, \mathbf{z}|\gamma, \phi) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n) \tag{3.5}$$

We assume that the distribution $q(\theta|\gamma)$ is Dirichlet with variational parameter γ, while the distributions $q(z_n|\phi_n)$ are multinomial with variational parameters ϕ_n. For each song, we seek a factorial distribution of the form in eq. (3.5) to approximate the true posterior distribution in eq. (3.3). Or more specifically, for each song s_m, we seek the variational parameters γ_m and ϕ_m such that $q(\theta, \mathbf{z}|\gamma_m, \phi_m)$ best matches $p(\theta, \mathbf{z}|s_m, \alpha, \beta)$.

Though it is intractable to compute the statistics of the true posterior distribution $p(\theta, \mathbf{z}|\alpha, \beta)$ in eq. (3.3), it is possible to compute the Kullback-Leibler

(KL) divergence

$$\mathrm{KL}(q,p) = \sum_{\mathbf{z}} \int d\theta q(\theta, \mathbf{z}|\gamma, \phi) \log \frac{q(\theta, \mathbf{z}|\gamma, \phi)}{p(\theta, \mathbf{z}|s, \alpha, \beta)} \tag{3.6}$$

up to a constant term that does not depend on γ and ϕ. Note that the KL divergence measures the quality of the variational approximation. Thus, the best approximation is obtained by minimizing the KL divergence in eq. (3.6) with respect to the variational parameters γ and ϕ_n. To derive update rules for these parameters, we simply differentiate the KL divergence and set its partial derivatives equal to zero. The update rule for γ_m is analogous to the one in the LDA model for text documents [6]. The update rule for the multinomial parameters ϕ_{ni} is given by:

$$\phi_{ni} \propto \prod_{j=1}^{V} \beta_{ij}^{x_n^j} \exp[\Psi(\gamma_i)], \tag{3.7}$$

where $\Psi(\cdot)$ denotes the digamma function and x_n^j denotes the count of the jth pitch class in the nth segment of the song. We omit the details of this derivation, but refer the reader to the original work on LDA [6] for more detail.

Variational Learning

The variational approximation in eq. (3.5) can also be used to derive a lower bound on the log-likelihood $\log p(s|\alpha, \beta)$ of a song s. Summing these lower bounds over all songs in the corpus, we obtain a lower bound $\ell(\alpha, \beta, \gamma, \phi)$ on the total log-likelihood $\mathcal{L}(\alpha, \beta) = \sum_m \log p(s_m|\alpha, \beta)$. Note that the bound $\ell(\alpha, \beta, \gamma, \phi) \leq \mathcal{L}(\alpha, \beta)$ depends on the model parameters α and β as well as the variational parameters γ and ϕ across all songs in the corpus.

The variational EM algorithm for our model estimates the parameters α and β to maximize this lower bound. It alternates between two steps:

1. (E-step) Fix the current model parameters α and β, compute variational parameters $\{\gamma_m, \phi_m\}$ for each song s_m by minimizing the KL divergence in eq. (3.6).

2. (M-step) Fix the current variational parameters γ and ϕ across all songs from the E-step, maximize the lower bound $\ell(\alpha, \beta, \gamma, \phi)$ with respect to α and β.

These two steps are repeated until the lower bound on the log likelihood converges to a desired accuracy. The updates for α and β in the M-step are straightforward to derive. The update rule for β is given by:

$$\beta_{ij} \propto \sum_{m=1}^{M} \sum_{n=1}^{N} \phi_{mn}^{i} x_{mn}^{j}. \tag{3.8}$$

While the count x_{mn}^{j} in eq. (3.8) may be greater than one, this update is otherwise identical to its counterpart in the LDA model for text documents. The update rule for α also has the same form.

3.2.4 Identifying Topics as Keys

We have seen from the previous section that the estimated parameter β expresses each topic as a distribution over $V = 12$ neutral pitch-classes. While this distribution can itself be viewed as a key-profile, an additional assumption is needed for topics to be identified with particular musical keys. Specifically, we assume that key-profiles for different keys are related by simple transposition: e.g., the profile for C♯ is obtained by transposing the profile for C up by one half-step. This assumption is the full extent to which our approach incorporates prior knowledge of music.

The above assumption adds a simple constraint to our learning procedure: instead of learning $V \times K$ independent elements in the β matrix, we tie diagonal elements across different keys of the same mode (major or minor). More specifically, we add the following constraint to our algorithm: Instead of learning $V \times K$ different elements in the β matrix, we allow the algorithm to learn only V different elements for each major and minor mode. These elements are then repeated for each key-profile of the same mode but shifted by one position for each new key-profile. This gives a clean formulation of all twelve musical keys for each mode: as the set of dominant pitches in the distribution are rotated from column to column all possible 24 configurations are obtained. Figure 3.3 shows an example of what our β matrix looks like.

Enforcing this constraint, we find that the topic distributions learned by the variational EM algorithm can be unambiguously identified with the $K = 24$ major

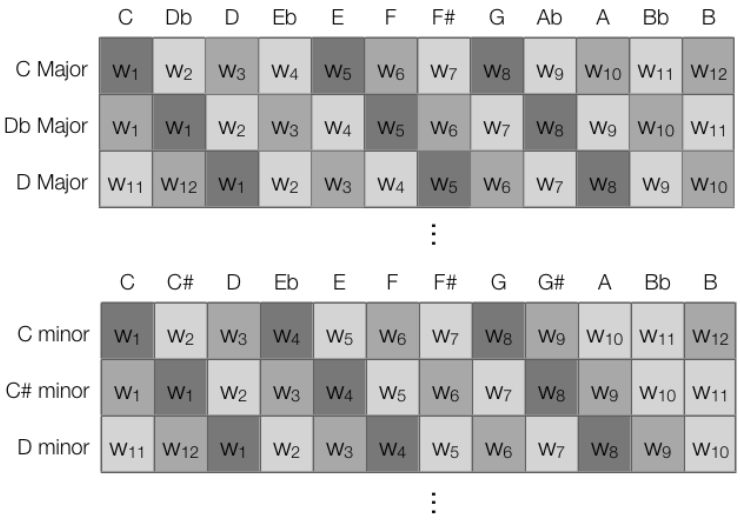


Figure 3.3: In LDA-MIDI, we force the elements of the β matrix to be tied diagonally within the major and minor modes. As such, instead of learning $(K = 24) \times (V = 12) = 288$ different parameters, we are only learn 24 (12 for major and 12 for minor). The remaining weights are achieved through circular shifting – or transposition, in musical terms.

and minor modes of classical western music. For example, one topic distribution places its highest seven weights on the pitches C, D, E, F, G, A, and B; since these are precisely the notes of the C major scale, we can unambiguously identify this topic distribution with the key-profile for C major. We see that each row of this β matrix essentially acts as a key-profile for that identified key, as we will discuss later in section 3.3.4.

3.3 Results and Evaluation

We evaluate our LDA-MIDI model in several different ways. First we talk about datasets and basic experiment set-up. Then, we evaluate how well our model does on the tasks of overall key-finding and modulation tracking by comparing it to other state-of-the-art techniques. We note that while our method requires no training or prior musical knowledge, all competitive methods either use supervised learning, or rely heavily on hand-tuned weights based on extensive music theory knowledge. In fact, our model also produces distributions that are similar to these

Table 3.2: Description of MIDI datasets used in symbolic music experiments.

ID	Size	Genre	Source
WTC-MID	96	Baroque (Bach's WTC Books I & II)	`classicalarchives.com/midi`
CLA-MID	235	Baroque, Classical	`classicalmusicmidipage.com`
ROM-MID	278	Classical, Romantic, Contemporary	`classicalarchives.com/midi`
KP-MID	46	Classical thru Contemporary	Tonal Harmony Workbook [39]
BEA-MID	174	Rock, No vocals	`davidbmidi.com`, `earlybeatles.com`

hand-tuned weights (called *key-profiles*) as a by-product, and we analyze how it compares to hand-chosen ones. Finally, we show some extensions of the basic LDA-MIDI model, as well as some simple applications of symbolic key-finding.

3.3.1 Experimental Setup

In this section, we show some results obtained by the basic LDA-MIDI model. We used four different MIDI datasets for evaluating. The first three datasets were compiled to reflect different musical genres: CL-MID contains classical piano and orchestral pieces with baroque and classical stylings; RO-MID, also contains some baroque and classical, but includes many more pieces from the romantic era, and even some from the contemporary and impressionistic eras; BE-MID, is a pop/rock dataset that contains instrumental MIDI recordings of songs from 13 albums of The Beatles. The fourth dataset, KP-MID, contains a mix of classical music, ranging from the baroque to the contemporary eras. Though this dataset is redundant in terms of genre, it is the only dataset for which chord and local key annotations were available. Table 3.2 summarizes these four datasets.

The first set of results come from using the LDA-MIDI model, with the MIDI datasets from 3.2 as input. There are a couple components of this model that are open for experimentation that have high impact on classification rates. One such component is the length of a song segment. We tried several different methods, including segmenting at the tactus level (defined as the main beat), half-

measure level, and at the full-measure level. We also tried a fixed length across all songs (for example, the equivalent length of four tactus-beats, or a fixed number of seconds). Since the MIDI data itself does not tell us where these measure boundaries occur, additional pre-processing uses beat-tracking algorithms [21] in order to make an educated guess. We report accuracies across these different methods of segmentation.

A second component determines how notes in a segment are converted to the 12-dimensional vectors inputted into the LDA-MIDI model. Octaves must be collapsed, but beyond this there are several other considerations such as whether or not to incorporate information about each note's duration, metrical position within a measure, the number of times the note appears, or the octave in which the note occurs in. We experimented with all of these options, and found that raw note duration and note count can be misleading (for example, a repeated note is given too much weight). Instead, the best option is to count the number of distinct notes that are struck. For example, the first element in 12-dimensional vector maps to the pitch class A. If the note A_4 appears six times in this segment, then pitch class A is recorded as only having a count of 1, since only one distinct note A_4 was played. However, if both A_4 and A_5 each appear once, then the pitch class A will be given a count of two, since two distinct A notes occurred. In this way, we are somewhat preserving both octave and note count information, but on the other hand, we are completely disregarding duration information.

3.3.2 Overall Key-Finding

Recall that from the posterior distribution in eq. (3.3) of the LDA-MIDI model, we can infer hidden variables θ and $\mathbf{z}$ that identify dominant keys in whole songs or segments within a song. In particular, we can identify the overall key of a song from the largest weight of the topic vector θ. Likewise, we can identify the key of particular segments from the most probable values of the topic latent variables z_n.

We first show results at the song-level, using our model to determine the overall key of the musical pieces in our datasets. We also compared our model

Table 3.3: Key-finding accuracy of our LDA model and other popular symbolic knowledge-based key-finding models. The LDA-MIDI model is abbreviated as LDA. The next five models (KK, DT, BA, HB, and CS) use the MKC key-finding model [41], with different hand-tuned key-profiles. The remaining two models (CBMS, BAYES) pair hand-tuned key-profiles with Bayes and dynamic programming to model key changes [75, 76]. (Note: We used the Melisma Analyzer software, developed by the author of this algorithm, to obtain results for the CBMS and BAYES models, but over half of the dataset for ROM-MID and BEA-MID were incompatible with their software, so we chose not to report results.)

Dataset	LDA	KK	DT	BA	HB	CS	CBMS	BAYES
WTC-MID	**0.979**	0.917	**0.979**	0.958	0.948	**0.979**	0.822	0.911
CLA-MID	**0.865**	0.793	0.844	0.823	0.823	0.848	0.626	0.781
KP-MID	**0.804**	0.674	0.696	0.717	0.698	0.652	0.696	0.761
ROM-MID	**0.795**	0.723	0.777	0.737	0.766	0.781	–	–
BEA-MID	0.678	**0.747**	0.667	0.558	0.523	0.649	–	–

to some other symbolic key-finding algorithms, as shown in Table 3.3. Most of these (denoted KK, DT, BA, HB, and CS) are knowledge-based methods that use the MKC key-finding model [41] with hand-crafted key-profile weights[1]. Appendix A.1 properly cites and describes each of these five key-profiles in details. The last two (CBMS, BAYES) use a combination of knowledge-based key-profiles and some statistical methods. For example, CBMS uses dynamic programming to find the optimal sequence of chords first [75], while BAYES[2] uses a simple application of Bayes rule and supervised learning [76]. For more details, these models are reviewed in section 2.4. From Table 3.3, we observe that though these models obtained their key-profiles using rule-based or supervised methods, our unsupervised model (with almost no prior musical knowledge) yields better results the majority of the time. We hypothesize that our unsupervised model generalizes better to multiple datasets.

The overall decrease in classification scores seem to indicate that datasets grow more challenging going down the list in Table 3.3. Since the datasets are

[1]Implemented following: http://extra.humdrum.org/man/keycor/
[2]Code from: http://www.link.cs.cmu.edu/music-analysis/

organized in rough chronological order, it is reasonable to assume that key-finding methods find earlier works (Baroque or Classical) to be less challenging than later works (Romantic and Contemporary), since the latter contains compositions that are more complex and strays further from basic music theory principles.

In further analyzing the results of our model, we see that many of the errors made by the LDA-MIDI model are due to confusion with keys that are musically very similar. To get a better understanding of these errors, we identify three specific types of errors: (1) perfect fifth errors, (2) relative major and minor errors, and (3) parallel major and minor errors, and visualize the percentage of the dataset that these errors take up. (For an overview of what these relations mean, please see section 2.3.) All other errors that do not fall into one of these three categories are considered "other", even though most of these mistakes still involve confusing fairly similar keys. Table 3.4 summarizes this information by grouping all of the errors across the five MIDI datasets by error type. We see that in general, perfect fifth and relative major/minor errors occur more frequently than parallel major/minor errors.

Relative major/minor errors are not surprising, as keys with this relationship have identical key signatures and scales. In other words, all of the pitches that are contained in the scale of one key, also occur in the scale of its relative major/minor key. The ability to distinguish between such keys requires the model to correctly weight the significance of each scale degree (i.e. A minor is the relative minor of C major – to distinguish between these two keys, we would expect the tonic pitch (C for C major, or A for A minor) to be fairly prominent. However, if note A happens to be used frequently in a C major song, then the two keys may be confused for each other. Human experts will be able to decipher the two keys by a number of external cues: by listening for the distinctive "sound" of the minor key, by looking at the chord progressions involved, and by giving more weight to bass notes or beat-synchronous notes (which will most often involve the tonic note). A downfall of our model is that, because it is given very little prior knowledge, it does not have knowledge about key-related chords, or precisely where down-beats occur. This may be a direction for future work.

We also observe that the Beatles dataset (BEA-MID) shows an especially high error rate involving perfect fourths, perfect fifths, and relative majors. The perfect fourth areas are likely due to the fact that many Beatles songs are written with a Blues influence, which makes heavy use of dominant seventh chords, instead of chords based on major triads. The dominant seventh chord is similar to a major triad, but includes a flatted seventh note, making it look more similar to the scale of the perfect fourth key, instead of the scale of its tonic (for example, the model mis-classified several E major songs as A major, because many E7 chords were present).

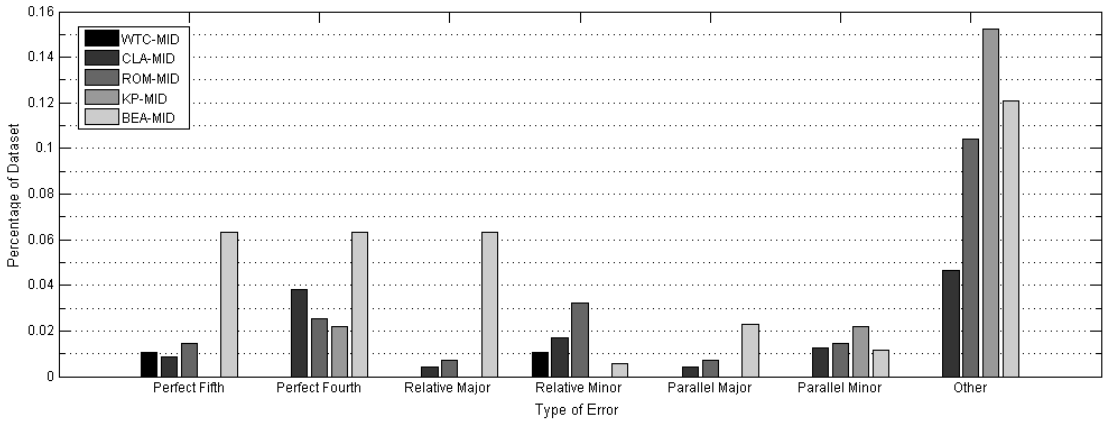


Figure 3.4: A break-down of common errors made by LDA-MIDI. The graph specifically groups errors across different datasets into respective error types.

In order to visualize errors more generally, we "superimpose" error rates onto the circle-of-fifths structure. This way, it is easy to tell the relationship between the incorrectly predicted key with the correct key. We note that in order to be more specific about cross-mode errors, we analyze major songs separately from the minor songs. Figures 3.5 and 3.6 each shows two confusion circles. The top confusion circle visualizes errors made on songs that have major key labels, and the one shown at the bottom visualizes errors for songs in minor keys. In order to make this representation key independent, we use relative scale degrees as opposed to specific key names. The key bounded by a rectangle at the 12 o' clock position (inner/outer depending on major/minor) of each circle represents the correctly labeled key as the tonic degree; all other points on the circle represent

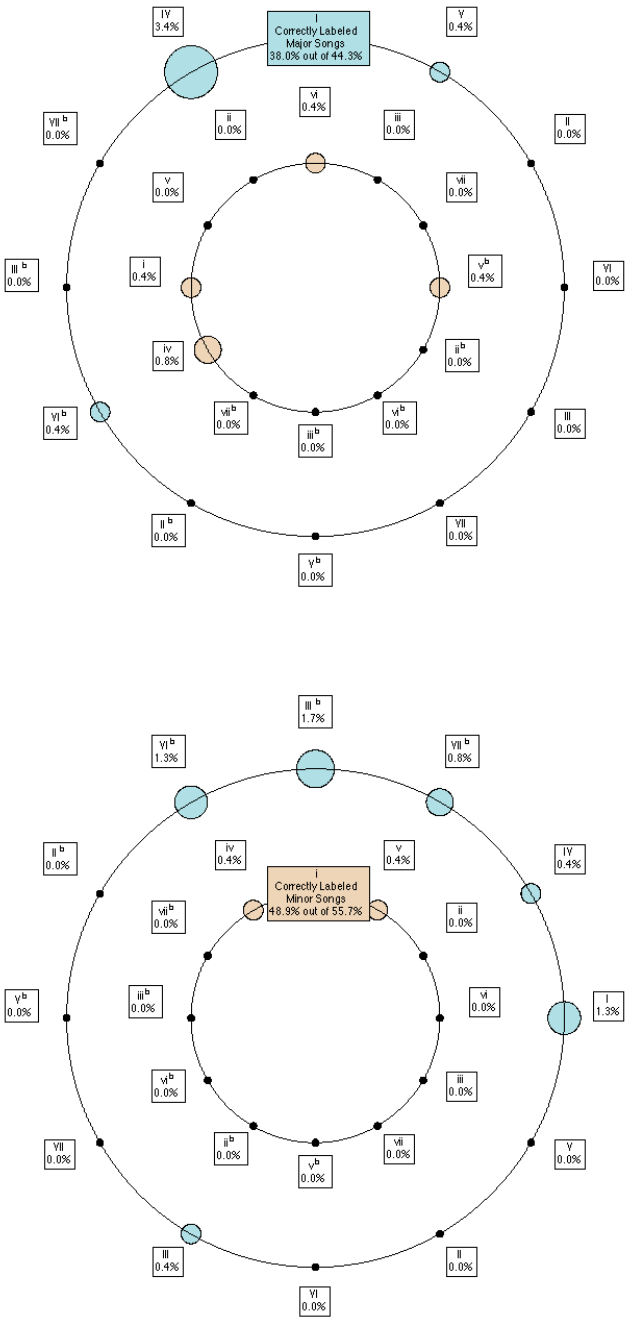


Figure 3.5: Confusion circles visualize symbolic key-finding errors made by the LDA-MIDI model. The top confusion circle breaks down the errors for ground-truth major songs from the classical MIDI dataset (CLA-MID), and the bottom circle does the same for ground-truth minor songs. All percentages shown are with respect to all songs in the dataset.

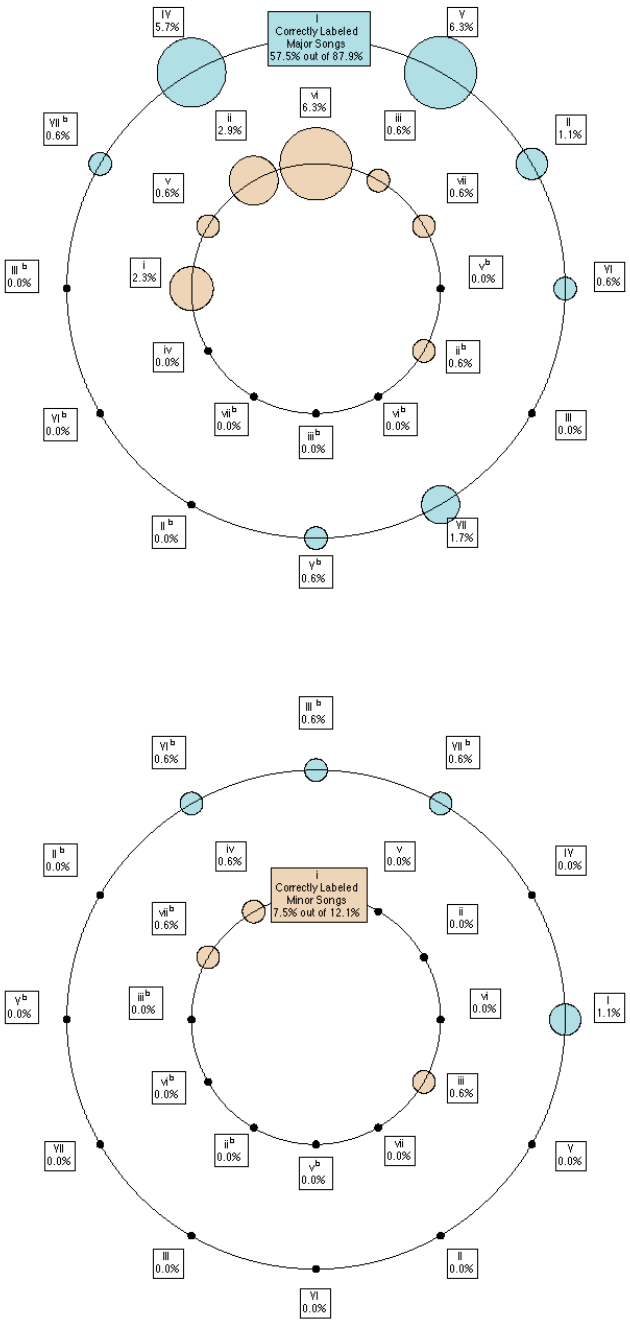


Figure 3.6: Confusion circles visualize symbolic key-finding errors made by the LDA-MIDI model. The top confusion circle breaks down the errors for ground-truth major songs from the Beatles MIDI dataset (BEA-MID), and the bottom circle does the same for ground-truth minor songs. All percentages shown are with respect to all songs in the dataset.

incorrectly labeled songs with a scale degree that is relative to that correct key. In a nutshell, points on the circle closer to the correct key are more similar. We display the confusion circles for two datasets: CLA-MID, representing classical, and BEA-MID representing pop. This gives us a bird's eye view of the type of errors that are being made. We can see that while there are a couple of outliers (which may be caused by the use of diminished, or other chord types that are not modeled), most of the errors occur close to the tonic of the circle – for example, in the major confusion circle, perfect fourth errors lie to the left of the tonic, perfect fifth errors lie to the right, and parallel minor errors are at the 9 o'clock position.

3.3.3 Chord-Estimation

Since our model naturally learns distribution over topics for each segment, we evaluate how well it does on the tasks of chord-estimation/modulation tracking (We use both terms as the line between these two tasks gets a bit hazy, depending on the genre of music we are working with. Classical datasets generally give segment labels that are more similar to local changes within key; hence, modulation tracking. But the Beatles dataset was given smaller segments as input and our model gives more fine-tuned key-labels that are more like that of chord estimation.) Recall that segment key information can be extracted from the hidden variable z that we inferred from our posterior distribution. Intuitively, z is the distribution from which the notes of each segment are drawn from and can provide information about the most likely key that each segment belongs to.

Though it is relatively straightforward to obtain these segment-level key predictions, proper evaluation remains a difficult problem, as echoed by many others working on this problem. This is because manual segment-level annotation requires particular musical expertise that is not cheap nor fast. The somewhat ambiguous nature of tonal induction sometimes makes it difficult for two experts to agree on a ground-truth label.

For our evaluation, we first show in great detail how our LDA-MIDI model analyzes the first twelve measures of Bach's Prelude in C minor from Book II of the *Well-Tempered Clavier*. Results are then compared to annotations by a music

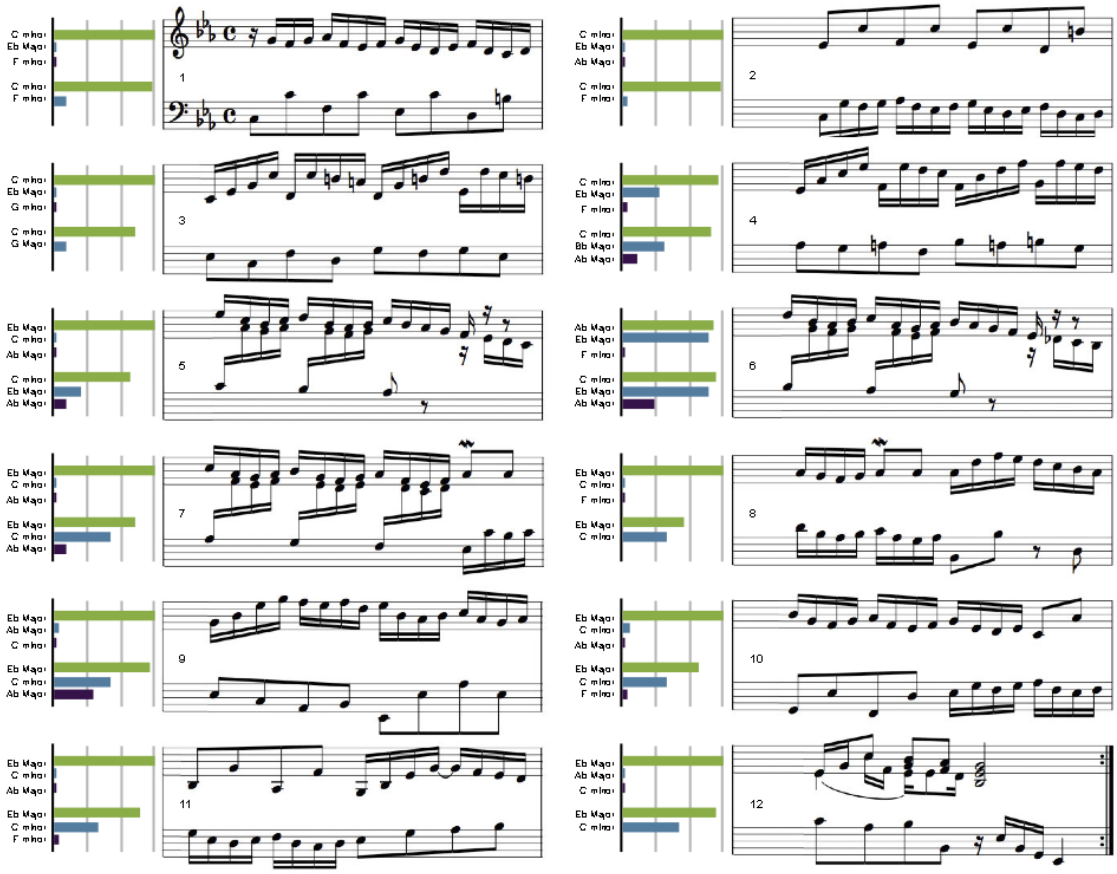


Figure 3.7: Key judgments for the first 12 measures of Bach's Prelude in C minor, WTC-II. Annotations for each measure show the top three keys (and relative strengths) chosen for each measure. The top set of three annotations are judgments from our LDA-based model; the bottom set of three are from human expert judgments [41]

theory expert, reported from [41]. This is summarized in Figure 3.7 – to the left of each measure of music shows two small bar graphs. The graph on top shows the top three keys (and relative strengths) chosen for that measure by human experts. The bottom graph shows the strengths of the three keys that were given the highest weights in the segment's key distribution. We can see that the top choice of key from our model differs from the expert judgement in only two measures (5 and 6).

For a more large-scale evaluation of modulation-tracking on a classical data set, we test on the Kostka-Payne dataset (abbreviated KP-MID in the previous section). This dataset contains musical excerpts taken from Kostka and Payne's *Tonal Harmony* workbook that includes homework exercises commonly used by beginning music theory students [38]. The ground-truth measure annotations are taken from the textbook's accompanying instructors manual. The dataset itself consists of 46 different pieces, ranging from all eras of classical music. Each piece is rather short – between 20 - 50 measures long. In total, there are 884 frames for which we must predict local keys for. We report a best accuracy score of 63.5% on this dataset. David Temperley's key-finding algorithm from his book *The Cognition of Basic Musical Structures* [75] reports a best accuracy score of 87.0% on this same dataset. However, we note that Temperley obtained this accuracy by inputting hand-corrected harmony and meter structure into his algorithm. Thus, future work for our model may also be to incorporate more structural features.

We also manually obtained measure-level chord annotations for 14 songs from the Beatle's "Please Please Me" album (which is included in dataset BEA-MID), with the help of Beatles guitar/piano score arrangements. Though annotations exist for the audio-version of this dataset we were not satisfied with results from MIDI-to-AUDIO alignment software (due to different kinds of repeats that were taken). We ended up with a total of 1115 frames, each corresponding to a measure. The results were quite encouraging: 87.4% of the measures were correctly predicted. Upon further analysis, many of the errors were extremely understandable. The most common source of error being due to chord changes within a measure – this gave the algorithm conflicting information about what chord to label the segment with. Table 3.4 shows the fraction of correctly predicted measures,

as well as whether or not the overall key was predicted correctly.

One result that may comes as a surprise is the song "Love Me Do." The algorithm correctly predicts the chords for every single measure, yet fails to predict the right key (the model chose C major, while the correct key is G major). This particular song alternates continuously between G major and C major chords (with a couple of D major chords scattered throughout). In fact, possibly more than half of the chords in this song are in C major. Unfortunately, since the LDA-MIDI model makes its decisions based on note co-occurrences, both G major and C major are given such high weights in the overall key distribution, that choosing between them is almost arbitrary (the key distribution shows that C major is given a 2% higher weighting than G major). On the other hand, a human expert would be able to look past the simple fact that C major chords appear just as often as G major chords, and see that (1) the additional D major chords hint that an overall key of C major is extremely unlikely, since the D major chord is not a part of the C major scale (but is a part of the G major scale), and (2) by looking notes on which the down-beats occur, and whether musical phrases begin with a C or G.

3.3.4 Unsupervised Learning of Key-Profiles

We have seen so far that each of the LDA parameters has corresponded to musically intuitive representations: the song-level topic proportion vector θ predicts the distribution of musical keys for the composition, and the segment-level topic proportion vector z predicts the chords present in a measure of music, or the local key that the song has modulated to. We show one more by-product of the LDA-MIDI model that gives us insight to our data. Recall that one of the learnt model parameters is the β matrix which defines each of the 24 topics as a distribution over our vocabulary of $V = 12$ pitch classes. These act very similarly as the key-profiles we have seen that have been manually hand-crafted in previous work. In our model, the LDA-MIDI model automatically learns these key-profiles in an unsupervised setting, from data.

Figure 3.8 shows the β matrix learnt from the CLA-MID dataset (which seems to look most "standard" from a music theory standpoint). We note that

Table 3.4: Chord-estimation accuracies for each song from the Beatles album "Please Please Me".

Song Title	Chord Correct	Key Correct?
I Saw Her Standing There	93.8%	Yes
Misery	97.3%	Yes
Anna Go To Him	89.9%	Yes
Chains	90.1%	Yes
Boys	92.0%	Yes
Ask Me Why	84.6%	Yes
Please Please Me	74.6%	Yes
Love Me Do	100%	**No**
P.S. I Love You	86.8%	Yes
Baby It's You	98.7%	Yes
Do You Want To Know A Secret?	64.0%	**No**
A Taste of Honey	75.6%	Yes
There's a Place	70.4%	Yes
Twist and Shout	87.2%	**No**
Average	87.4%	78.6%

these key-profiles have the same general shape as those of the KK profiles (though the actual weights for each pitch-class are not comparable since our weights denote actual probabilities). Note that in both major and minor modes, the largest weight occurs on the tonic (C), while the second and third largest weights occur on the remaining degrees of the triad (G, E for C major; G, E♭ for C minor). Our key-profiles differ only in the relatively larger weight given to the minor 7th (B♭) of C major and major 7th (B) of C minor. Otherwise, the remaining degrees of the diatonic scale (D, F, A for C major; D, F, A♭ for C minor) are given larger weights than the remaining chromatics.

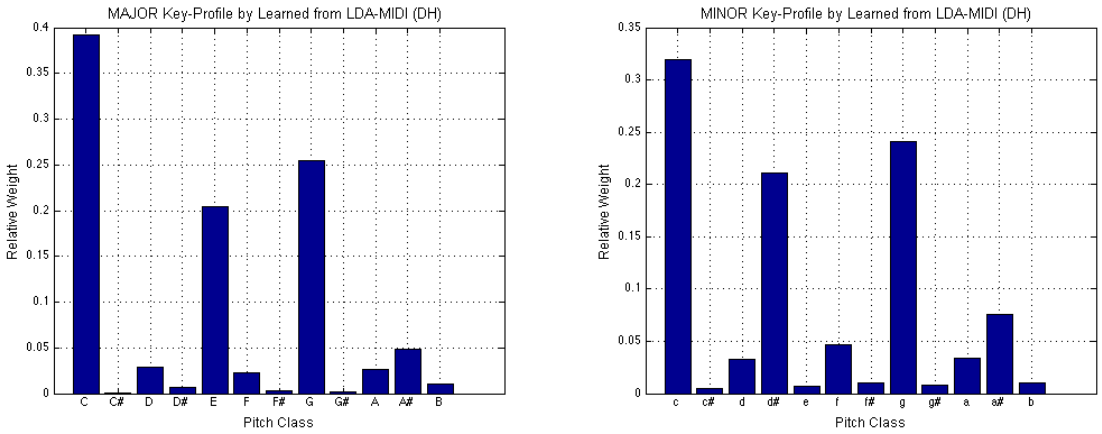


Figure 3.8: Key-profiles for C Major and C minor learned from LDA-MIDI on the CLA-MID dataset.

3.4 Extension: Learning Key Correlations

One limitation of the original LDA model for text was that the topics drawn from each document were independent of one another. For example, if we know that a text document already contains a topic regarding *education*, then there is a high probability that a topic about *schools* may also be present, as opposed to a semantically unrelated topic, like *genetics research*. However, LDA is not capable of modeling this kind of topic correlation. Under a Dirichlet, the components of the document-level topic vector are nearly independent, and leads to a strong yet unnatural assumption that the presence of one topic has no influence on the presence of another topic.

We see that this issue arises in music as well – certain groups of musical keys are more likely to be used in a song than others. For example, it is very common for a song whose overall key is C major, to also contain sounds from G major. First, there is the obvious – all three of these keys are very similar. G major is a perfect fifth away from C Major, and its key signature differs by a single accidental (the F in the G Major scale is raised half a step – or, sharped – while it is not in C Major). For this reason, the keys of G Major and C Major also share many common chords: triads rooted at the first, third, fifth, and sixth scale degrees on the C Major scale, are the same triads rooted at the fifth, seventh, first, and second

degrees on the G major scale. Since they share so many of the same scale pitches and chords, it is extremely likely to hear the "sounds" of G major baked into a C major song. On the other hand, it would be extremely unlikely to hear sounds of B major, as it is tonally very dissimilar. Learning these key correlations directly from the data may help weed out "intruding" keys that theoretically have a low probability of being in the song. We describe such a model below.

3.4.1 CTM-MIDI Model

To see if modeling key correlations improve things, we borrow concepts used in the Correlated Topic Model (CTM) [5], which can be thought of an an extension to the original LDA model for text. In order to remove the independence assumption under the Dirichlet, the CTM uses a logistic normal distribution instead. The logistic normal can be seen as a transformed multivariate Gaussian distribution on the simplex. This makes it possible for a logistic normal random variable to be used as a Multinomial parameter, which allows easy integration with the original LDA model. The logistic normal distribution is chosen in particular so that correlations between topics can be modeled through the covariance matrix of the Gaussian distribution.

Recall that in LDA, the document-level topic vector θ was drawn from a Dirichlet parameterized by a K-dimensional α. Now, θ will be drawn from a logistic normal distribution instead. The revised graphical model is shown in Figure 3.9, and the new generative process is detailed below. Note that the generative process is identical to that of LDA-MIDI except that key proportions are drawn from a logistic normal distribution rather than a Dirichlet.

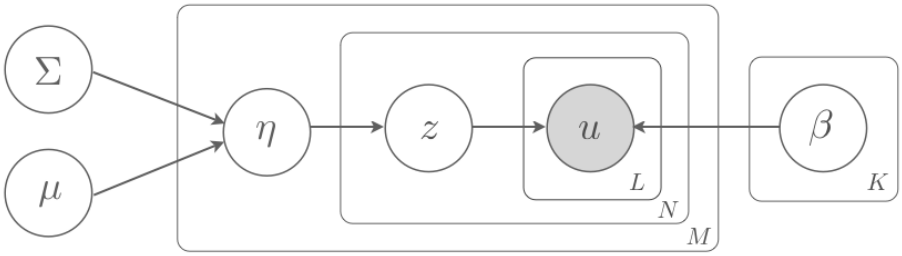


Figure 3.9: The graphical model for the Correlated Topic Model (CTM) adapted for symbolic music (CTM-MIDI). The corresponding generative process is identical to that of LDA-MIDI except that the key proportions are drawn from a logistic normal (parameterized by μ and Σ) rather than a Dirichlet. The main difference from the original CTM for text is the additional plate around the notes u – instead of each word having its own topic, each group of notes (within a segment) has its own topic.

For each song in the corpus indexed by $m \in \{1, \ldots, M\}$:

1. Draw $\eta \sim \mathcal{N}(\mu, \Sigma)$
2. Map η to the simplex: $f(\eta_i) = \exp \eta_i / \sum_j \exp \eta_j$
3. For each segment indexed by $n \in \{1, \ldots, N\}$ in a song:
   (a) Choose the topic $z_n \in \{1, 2, \ldots, K\}$ from the multinomial distribution $p(z_n = k | f(\eta)) = \theta_k$
   (b) For each note indexed by $\ell \in \{1, \ldots, L\}$ in the nth measure:
      i. Choose a pitch-class from the multinomial distribution $p(u_{n\ell} = i | z_n = j, \beta) = \beta_{ij}$.

Posterior inference is even more challenging in CTM-MIDI than it was for LDA-MIDI. Not only is the posterior distribution of the latent variables intractable to compute, but the logistic normal distribution is also not conjugate to the multinomial. In brief, the strategy employed here is to form a factorized distribution of the latent variables (similar to the one formed in LDA-MIDI), parameterized by additional variational parameters. These variational parameters are fit in such a way that the Kullback-Leibler (KL) divergence between the approximate and

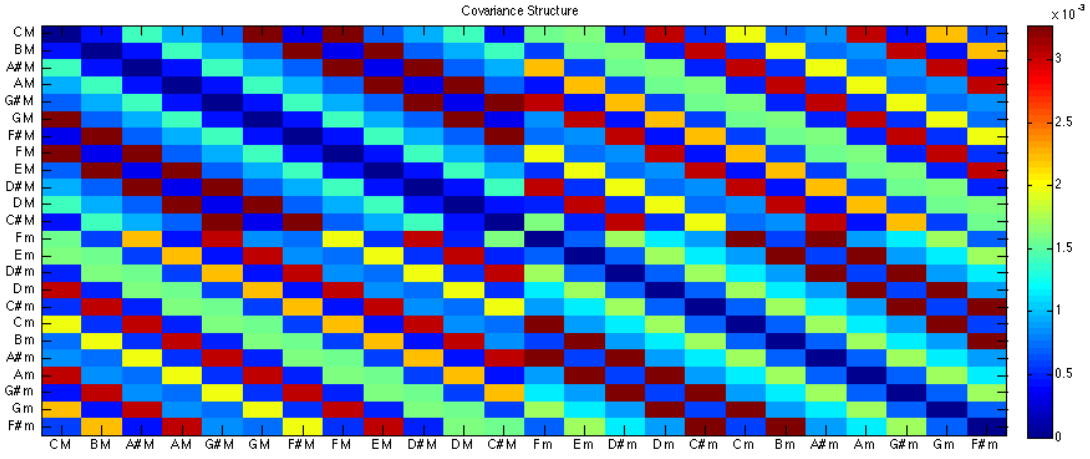


Figure 3.10: Example of the tied covariance structure learned from CTM-MIDI.

true posterior is minimized. To deal with non-conjugacy of the logistic normal distribution, iterative methods are used for optimizing the necessary variational parameters. A more thorough explanation is provided in [5]. This variational inference algorithm was modified to carry out the generative process detailed above, to work for harmonic analysis with symbolic music input. In particular three significant modifications were made to the original CTM (the first two modifications were also used in the adaptation of LDA to LDA-MIDI model):

1. An additional level of hierarchy is imposed (shown in the graphical model as an additional plate around notes u), in which topic proportions are learned for a collection of notes within the same segment.

2. Elements of the β matrix are tied diagonally such that all columns are related by transposition (i.e. circular shifting) within the confines of its own mode. Now, only 24 different elements are learned (12 for major and 12 for minor), instead of $12 \times 24 = 288$.

3. The covariance structure is also tied diagonally (similar to the β matrix). Each of the $K = 24$ rows are related by transposition (within the confines of its own mode). This assumes that correlations depend only on the intervals between certain keys, instead of the absolute key. Figure 3.10 shows an example of the learned Σ matrix, labeled with hypothesized key names.

Table 3.5: Summary of key-finding accuracies when compared to the new CTM-MIDI model for all five symbolic datasets. The first two columns show results from the CTM-MIDI and LDA-MIDI models, respectively. The last column(s) shows the competing model with the highest accuracy; the left side shows the accuracy, the right side indicates the model from which it was obtained. Bolded numbers are the highest accuracies across the row for that dataset.

Dataset	CTM	LDA	OTHER	
WTC-MID	0.958	**0.979**	**0.979**	DT, CS
CLA-MID	0.848	**0.865**	0.848	CS
KP-MID	0.783	**0.804**	0.761	BAYES
ROM-MID	**0.820**	0.795	0.781	CS
BEA-MID	**0.753**	0.678	0.747	KK

3.4.2 Results and Evaluation

To evaluate the new CTM-MIDI model, we use it to perform key-finding using the same datasets from the previous chapter. Table 3.5 summarizes these findings. It is interesting to see that during evaluation, the new CTM-MIDI model improved key-finding accuracy significantly for more contemporary datasets, while obtaining accuracies that were either the same or a little lower for other early classical datasets. It seems that learning the correlation of keys is more helpful on datasets that contain more complex chords and harmonies.

The improvement from 67.8% to 75.3% on the Beatles dataset is especially worth noting. It seems that by learning correlations between keys, the CTM-based model is able to find sets of keys that commonly occur together, even when it is not explicit in the raw MIDI notes. Figure 3.11 shows a comparison of how CTM-MIDI decreases error rates all across the board, except for relative minor errors. To further understand some of these errors, we show how the learned weights of the 24 keys are distributed for two songs in Figure 3.12 – on the left is an example of a song in which CTM correctly predicts the key, while LDA misclassifies it, and on the right is an example of the opposite: LDA is correct, but CTM is wrong. In the first example (left), LDA does not get any hint of the E Major (the correct

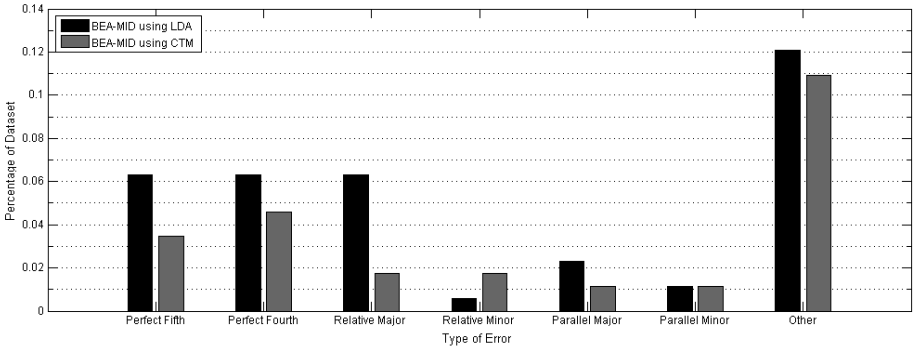


Figure 3.11: A comparison of common errors made by the CTM-MIDI and LDA-MIDI models, on the Beatles dataset.

key). This is a reasonable error for LDA because the song contains a lot of E7 chords, which will look a lot more like A major to LDA than E major. In the CTM-based model however, it seems that correlations have been learned between E Major, A Major, and B Major; we see several occurrences in which this set of three keys are given high weights. In this example, the E major is correctly given the highest weight. On the other hand, we see how learning correlations may be harmful. In the second example (right), perhaps CTM makes too strong of an assumption regarding the correlation of this same set of keys, and incorrectly gives too much weight to E major.

Interestingly, the key-profiles learned by the CTM-MIDI model also reflects the dominant seventh behavior. Figure 3.13 shows the CTM-based key-profiles learned from the Beatles dataset in the top row, and the LDA-based key-profiles from the same dataset in the bottom row. Comparing the two models, we see that in the CTM-based model, the fourth scale degree, and the seventh are emphasized more in the major profile. In the C minor profile, it is interesting to see the flatted seventh emphasized as well. All of these aspects indicate that the CTM-MIDI model has picked up on the different types of harmonies that the Beatles dataset uses, which are significantly different from other classical datasets.

We finish this section by noting that the learned covariance structure in the CTM-MIDI model provides a great visualization of how keys are related to each other in various datasets. Figure 3.14 visualizes the key correlations learned

by the model. The keys are arranged like the circle of fifths (with some slight modifications – e.g. major and minor circles are swapped, and circle is rotated). Lines connecting one key to another indicates a significant correlation; the darker (more red) the color of the line, the stronger the correlation.

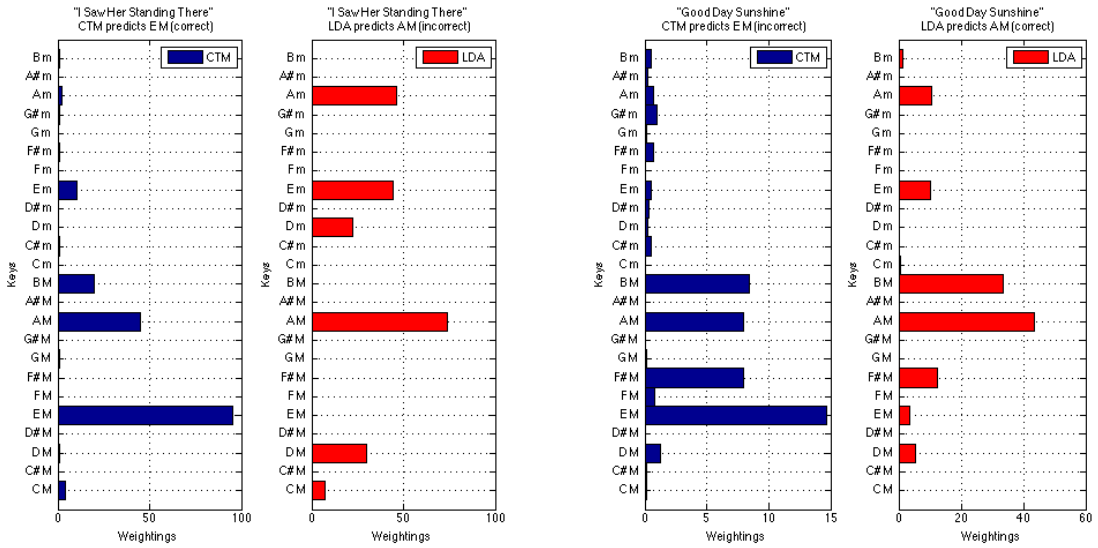


Figure 3.12: Examples of key-distributions learned by the LDA-MIDI and CTM-MIDI models, for two different Beatles songs.

3.5 Discussion & Conclusion

In this chapter, we presented two LDA-MIDI and CTM-MIDI that automatically predicts global and local keys from symbolic music input. While most previous methods either (1) use supervised frameworks that require key-annotated data, or (2) integrate extensive music theory knowledge (for example, to hand-craft musical key-profiles or use hand-chosen parameters for HMMs), our method is unsupervised, and assumes very little prior music knowledge. The extent of musical knowledge is 1) to know to look for $K = 24$ keys, and 2) the assumption that the distribution over pitches for these musical keys must be related by transposition.

The LDA-MIDI model elegantly learns three useful musical representations at once: a distribution over keys for each song to facilitate key-finding; a distri-

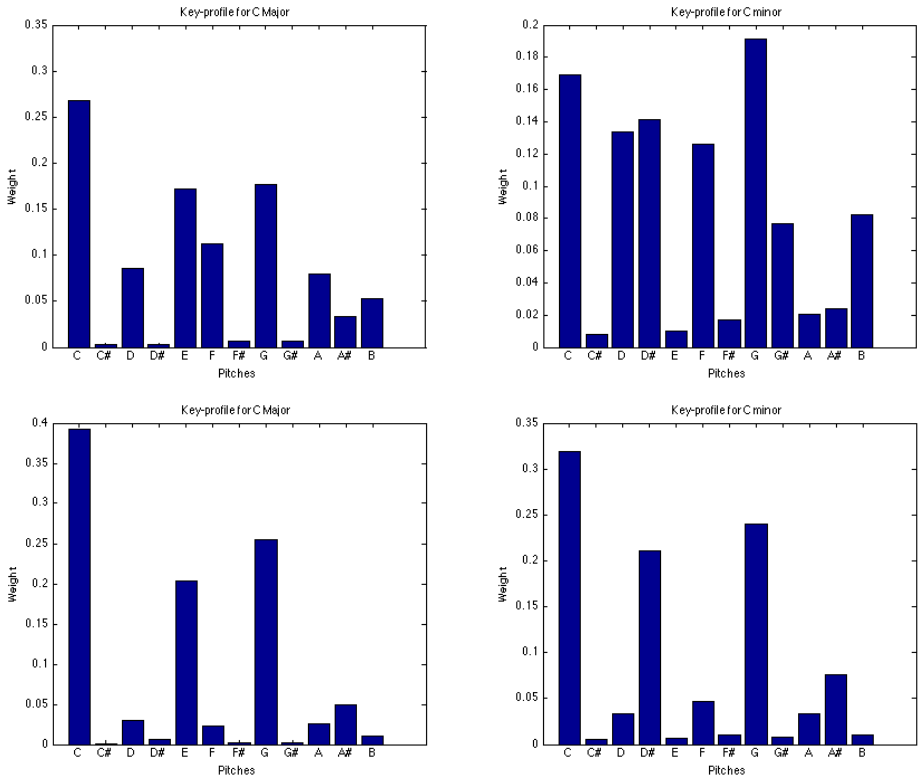


Figure 3.13: Key-profiles learned from CTM-MIDI (top row) and LDA-MIDI (bottom row) on the Beatles dataset (BEA-MID).

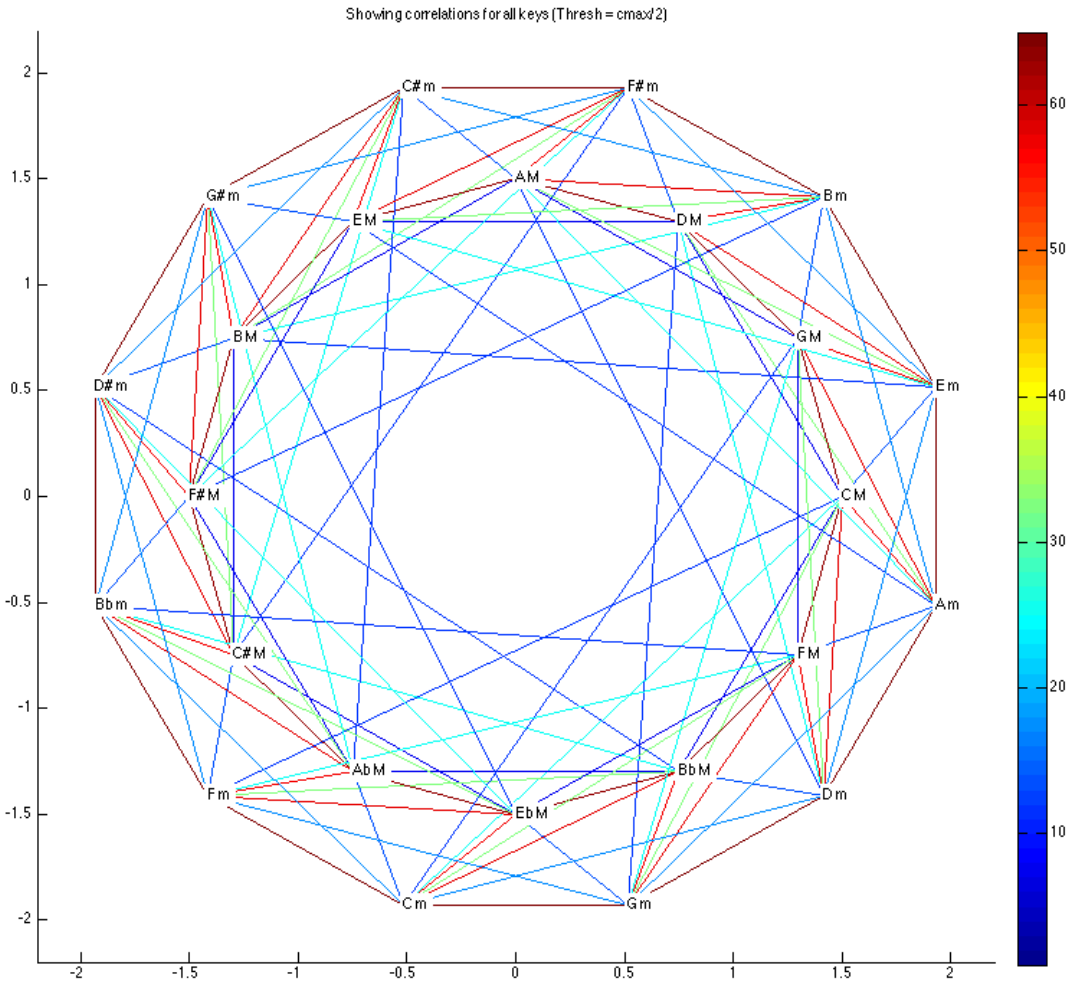


Figure 3.14: Correlations between the 24 major/minor keys, learned from CTM-MIDI. The color of the edge connecting the keys indicate correlation strength; red indicates one of the strongest correlations while blue indicates a very weak correlation. These correlations in particular improved key-finding for the symbolic Beatles dataset by almost 6% over the predictions of the LDA-MIDI model.

bution over keys for each musical segment to facilitate chord-finding/modulation tracking; and a distribution over pitches for each key to facilitate the unsupervised learning of key-profiles that help better understand the statistics of the dataset. During evaluation, the model does quite well compared to some of the state-of-the-art symbolic key-finding techniques, obtaining the highest accuracy rates across most datasets that we tested.

The CTM-MIDI model further extends the LDA-MIDI model by making the assumption that certain keys are more likely to appear together. As a result, a covariance structure between the 24 different keys are learned, and evaluations have shown that this assumption indeed improves key-finding accuracy for songs with more complex harmonic structures. In particular, since our models are key-based, chords beyond the 24 major and minor triads are not modeled by our system. When these more complex chords do occur, LDA-MIDI is most likely to confuse them with the closest major/minor triad chord (which usually has a different root). Our results show that in some instance, CTM-MIDI is able to choose the correct key based on correlation information, even when it is not explicit in the bag-of-notes MIDI representation.

So far, we have only shown models to work with symbolic input. As mentioned previously, we deem models for symbolic input important for two reasons: (1) they provide more accurate analysis for those purely interested in the analysis aspect, and (2) they provide high-accuracy annotated training data for audio-based harmonic analysis systems. However, we also acknowledge that the use of audio input is much more widespread and practical. In the next chapter, we explore similar LDA-based models that take audio input.

3.6 Acknowledgements

Chapter 3, in part, is a reprint of the material as it appears in Proceedings of the International Conference on Music Information Retrieval (ISMIR) 2009. Hu, Diane; Saul, Lawrence K. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Harmonic Analysis of Audio Data

4.1 Introduction

In the last chapter, we presented models that took symbolic MIDI files on input. These models were shown to automatically perform harmonic analysis tasks such as key-finding and modulation tracking with fairly high accuracy, while also producing several useful representations and visualizations for analysis. While we expressed the merits for investigating such a model that required symbolic input, we also recognize the more practical need for models that accept raw audio input. Doing so greatly improves the versatility of the model as it would allow for the analysis of mp3 and CD recordings, which are more readily available to the casual music listener, rather than digitized musical scores (i.e. MIDI files).

In this chapter, we describe how to adapt the models from the previous chapter to work with raw audio data. In the following, we begin with a discussion on how our audio datasets are obtained, as well as the features used in the audio-based models. We then evaluate how well the model from the previous chapter (LDA-MIDI) fares with raw audio input 4.4 (for simplicity, we refer to this combination of using audio input on the LDA-MIDI model as the LDA-QUANT model, as we will see later that audio features are first quantized), and then develop a more advanced and unified LDA-based model for audio in section 4.5 (LDA-AUDIO).

4.2 Audio Datasets

To have grounds for comparison between the MIDI and AUDIO models later on, we were intentional about the audio data used for our modeling and experimentation. Similar to Lee's approach in [48], we obtain some of our audio datasets by synthesizing our MIDI song collection using music fonts. Doing this allows us to easily compare between the MIDI and AUDIO models with the exact same songs and segment boundaries. We provide more details in our results section.

Synthesis is done using Timidity++, a free software synthesizer that can play MIDI files and then render to the sound card in real time[1]. What makes Timidity++ especially attractive is that it supports the use of SoundFonts; this is a sample-based synthesis technique that creates harmonically rich audio that sounds very similar to a real audio recording. For our experiments, we used the site `http://www.sf2midi.com/` where we found soundfont *sinfon36.cfg* which contains many orchestral instruments. Once a MIDI file is synthesized and converted into an audio (WAVE) file, the file is down-sampled to 11025 Hz (in order to speed computation).

4.3 Audio Features

Chroma vectors have been the most common choice of features for harmonic analysis tasks ever since Fujishima adopted them for his first key-estimation system [24] in 1999. In section 2.4.1, we described how each of its 12 dimensions represent the relative intensities of the 12 semitones on the (octave-independent) chromatic scale, as well as common methods for computing them. In addition, chroma vectors are efficient to compute, low in dimensionality, and (for the most part) robust to noise and harmonics. This makes them ideal for the tonality induction tasks at hand, and they very closely mirror the MIDI features (12-dimensional note counts) used for our symbolic models in the previous chapter.

While previously, in section 2.4.1, we described the basic chroma vector formulation, in our experiments we work with two additional enhancements on top

[1] `http://timidity.sourceforge.net`

of these original features. In brief, *tuned chroma vectors* [27] deals with mistunings, and *non-negative least squares (NNLS) chroma vectors* [57] uses pre-defined note templates to carry out approximate note transcription in order to reduce the presence of harmonics. We discuss both in turn below.

4.3.1 Tuned Chroma

Harte [27] proposes a tuning algorithm that is applied to a chroma vector computation that uses 36 bins per octave instead of 12. The tuning produces cleaner and stronger chroma vector signals, as well as more accurate boundaries between semitones. This is especially useful for audio recordings that contain instruments that are not tuned to concert pitch with $A4 = 440Hz$. To see the results visually, Figure 4.1 compares chroma vectors for a chromatic scale before and after the tuning algorithm. Notice that the intensities of each chroma vector bin becomes much more well-defined than before. We summarize the stages of the algorithm below for an audio sample below:

1. Compute a constant-Q transform on sliding windows of audio, using $b = 36$ instead of $b = 12$. Using 36 bins produces a higher resolution $36-$dimensional PCP that is able to distinguish between adjacent semitone frequencies regardless of the record-specific tuning.

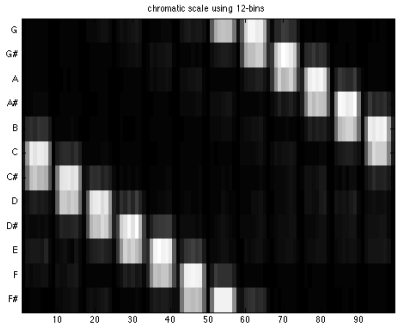
2. A 36-bin PCP is then computed from the constant-Q spectrum vector, C, mapping each semi-tone as multiple of 36 to the same bin:

$$PCP_b = \sum_{m=0}^{M} |C_{b+36m}| \qquad 1 \leq b \leq 36 \tag{4.1}$$

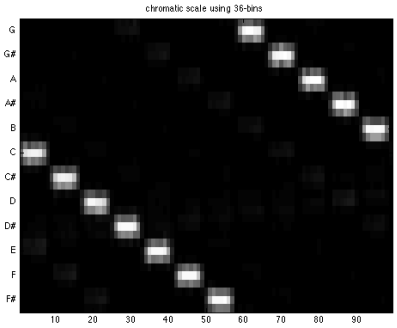
   where M is the number of octaves spanned by the constant-Q spectrum.

3. For each PCP, pick the 12 bins with the highest values (out of the 36 total bins). These bins should roughly correspond to 12 different semi-tones, and each of the high-value bins are referred to as a "peak".

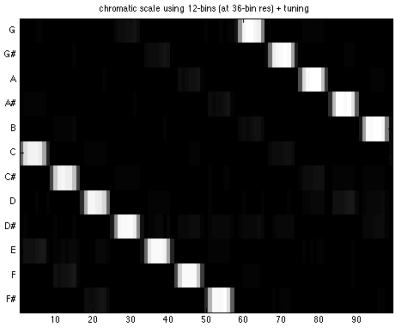
4. For each peak, look at the values of its two adjacent bins and apply quadratic interpolation to find the "true" peak in that local area (i.e. each grouping of



(a) Original 12-dimensional chroma vector



(b) Original 36-dimensional chroma vector



(c) Final, collapsed 36-dimensional chroma vector

Figure 4.1: This set of images illustrate the process by which Harte [27] quantizes chroma vectors to make them more well defined. (a) shows the original 12-dimensional chroma vector representation for a simple chromatic scale. (b) shows the 36-bin version of the same computation. (c) shows the result of finding peak intensities over all chroma vectors in the musical segment, adjusting for the offset of peak values, and then collapsing the 36-bin chroma vector back down to 12-dimensions. Comparing the final result in (c) with the original (a) shows that the quantized chroma vector more accurately defines borders between semitones.

three bins can be thought of as a local area corresponding to a semitone for which we are trying to find a true peak). The resulting peak may actually be a number between discrete bins.

5. Now, each of the 12 newly found "true peaks" can be expressed as a number between 0 and 3, relative to the starting and ending values of its two adjacent bins. This can be done by simply taking the modulo of each "true peak" computed from the previous step and 3.

6. For each PCP, we now have 12 numbers that correspond to peak values, normalized to be between the range of 0 and 3. Out of all normalized peak values across all windows in the song, we find the most frequently occuring peak value (by means of a histogram). This resulting value becomes the "center tuning value" for semitones (i.e. this center tuning value is assumed to be the *correct* peak value for all of our local three-bin groupings). This means that all peaks positioned at $+/-1.5$ (relatively) of the center tuning value, should be considered to be a part of the same semitone. Let $c =$ center tuning value. Then, the b^{th} semitone (where $b = 1, \ldots, 12$) should contain the values of peaks positioned between $3(b-1)+c-1.5$ and $3(b-1)+c+1.5$.

At the conclusion of this algorithm, we obtain a quantized and tuned 12-dimensional PCP vector with very clear semitone boundaries. Again, note the difference between images in Figure 4.1.

4.3.2 NNLS Chroma

A second variation to the conventional chroma calculation is used in our experimentation. The *NNLS chroma* was developed by Mauch [57], and is one of many that tries to remove the partials of the notes played in chords. As previously discussed, the presence of partials affects not only polyphonic music transcription (in which pitch and octave information must be recovered from audio), but makes it challenging for audio chord recognition as well. While collapsing the spectrum into 12 octave-independent bins will "fold" partials into the correct bins for some

types of chords, others will still suffer from distortion due to partials mapping into the wrong bins. Mauch explains this very well in his short example:

> The motivation for [removing partials] is the observation that the partials of the notes played in chords compromise the correct recognition of chords... The major chord (in root position) does not pose a serious problem, because the frequencies of the first six partials of the bass note (C) coincide with the chord notes: for example, a C major chord (consisting of C, E, and G) in root position has the bass note C, whose first six partials coincide with frequencies at pitches C, C, G, C, E, G. Hence, using a simple spectral mapping [i.e. conventional chroma calculations] works well for major chords. But even just considering the first inversion of the C major chord (which means that now E is the bass note), leads to a dramatically different situation: the bass note's first six partials coincide with E, E, B, E, G#, B – of which B and G# are definitely not part of the C major triad. Of course, the problem does not only apply to the bass note, but to all chord notes. This is a problem that can be eliminated by a perfect prior transcription because no partials would interfere with the signal.

Previous work in remove partials in chroma calculations have tried emphasizing the fundamental frequency component or used relatively simple forms of approximate note transcription before producing the chroma features [25, 79]. Mauch takes a more direct approach by integrating more realistic transcription methods used in polyphonic music transcription *before* collapsing the spectrum into the 12-bins that form the chroma. The note transcription algorithm he uses is based on non-negative least squares (NNLS), hence the name of the feature. The computation is briefly summarized below:

1. *Calculate the log-frequency spectrogram and applying global tuning estimation.* This is done by mapping the spectrogram to bins that are spaced a third of a semitone apart, estimating the global tuning, then adjusting the bins (based on the estimated tuning) such that the center bin corresponds to the correct frequency [64]. (This is similar to the process of obtaining quantized chroma features, described in the previous section.)

2. *Pre-process the log-frequency spectrum.* Three options are considered, but the method that is reported to work the best is subtracting the background

spectrum (using a running mean of bins that spread an octave-wide) and then dividing by its respective running standard deviation. This is similar to spectral whitening and serves to discard timbre information [36].

3. *Create a note profile for each note to be considered during transcription.* The author considers 84 notes, from A0 to G#6. Each note profile is generated in the log-frequency domain using a model with geometrically declining overtone amplitudes [25].

4. *Decompose the the individual frames of the log-frequency spectrogram Y as a linear combination of the* 84 *note profiles, E.* More formally, we seek to find x, the non-negative weights that would minimize the following function:

$$||Y - Ex|| \tag{4.2}$$

   This is framed as a non-negative least squares (NNLS) problem, to which the authors use the solution proposed by Lawson and Hanson [45]. When aggregated across all frames, this solution provides an approximate transcription of the original audio, represented by an $84 \times N$ matrix. Here, each of the N columns corresponds to an audio frame, and each of the 84 rows corresponds to a semitone.

5. *For each frame, collapse* 84 *semitone bins into* 12 *bins by summing all of the values of the respective pitches.* This results in a $12 \times N$ matrix, where each column is the NNLS chroma representation for an audio frame.

In our experimentation, we compare both quantized chroma features (described in the previous section) and NNLS chroma features. Both include a global tuning component, but only the NNLS chroma feature attempts to remove partials. Later on, we show how big of a difference this removal of partials has on our harmonic analysis tasks.

4.4 Using LDA-MIDI for Audio

In our first, most basic approach, we adapt the LDA-MIDI model (described in the previous chapter) for audio. This is done by replacing the original MIDI note count vectors with a quantized version of the chroma vectors described in the previous sections. Recall that in LDA-MIDI, each song s is represented as a sequence of notes in N segments, where each segment represents a basic unit of time in a song (e.g., a measure). Discarding all note order information, each segment x is simply described by a 12-dimensional note count vector, indicating the number of times each of the 12 pitch classes appear in that segment. As such, a song can be described by a sequence of these count vectors $s = (x_1, \ldots, x_N)$.

Below, we describe two simple methods for obtaining similar MIDI representations with acoustic input. For simplicity, we refer to the combination of using LDA-MIDI for audio as the LDA-QUANT model (since chroma features must go through a quantization step). In the first section, we first describe the basic LDA-QUANT model; in the second section, we describe an enhancement to the LDA-QUANT model that uses a dictionary-based vocabulary.

4.4.1 LDA-QUANT

Instead of a collapsed MIDI note count, LDA-QUANT takes in chroma features and then quantizes them. To compute chroma features, each song is segmented into overlapping frames with a length of 2048 samples (at a sampling rate of 11025, this is about $0.19s$) and a hop size of $2048/4 = 512$. Chroma features are computed for each frame, producing a $12 \times N'$ matrix of continuous numbers, where N' is the number of total frames produced. In order to preserve the segment boundaries from the MIDI files, all frames that map to a MIDI segment are averaged. In doing so, we obtain a $12 \times N$ representation – the same dimensionality as the original MIDI note count vectors. We also do a bit of pre-processing on these final chroma features: (1) we re-scale the values to be between 0 and 10 – this is to mimic the behavior of the audio model we will present later, and (2) we threshold very small chroma feature values and replace them with zeros. These features are

then used in place of the input MIDI vectors, and everything proceeds as normal in the LDA-MIDI model.

We emphasize again that since chroma features indicate the relative intensities of each of the 12 pitch classes within the segment, the intuitive meaning of these features are the same as that of the MIDI note count features. As touched on before though, the main challenge comes from the presence of harmonics (from the audio), as well as an ambiguous relationship between chroma intensities and note counts.

4.4.2 LDA-QUANT with Dictionary

To make the chroma features more robust to noise and harmonics in our model, we experiment with an additional step on top of the LDA-QUANT model. This is a dictionary-based approach that is similar to the feature quantization methods used in computer vision [23]. First, a dictionary is built by defining a "vocabulary" of 12-dimensional templates. Of course, there are many ways to do this. In mimicking computer vision approaches, one could first run an unsupervised clustering algorithm (such as k-means [55]), with k number of clusters, and then choose the means of those final clusters as the templates. Another approach is to manually devise templates that will reflect the major and minor keys that the model is trying to predict. The second method will most likely perform better as it contains more accurate prior musical knowledge; however, in attempts to inject as little prior musical knowledge into our model as possible, we experiment with the first method as well.

Once the vocabulary is set, we map all of the chroma vectors we observe to the most similar template from our vocabulary. (Here, similarity is measured by Euclidean distance, though other distance measures may work as well.) Then, instead of averaging all of the vectors that fall within a MIDI segment, we record the number of frames that matched to each template. In doing so, each MIDI segment will be represented by a V-dimensional vector, where V is the total size of the vocabulary (i.e. total number of templates). The ith element of this vector indicates the number of frames within that MIDI segment that map to the ith

template in the vocabulary.

For our first experiment, we chose musically informed templates that reflected the triad chord or scale of each key. A triad-based template, for example, would contain the following vectors: $[1,0,0,0,1,0,0,1,0,0,0,0]^T$ for C Major and $[1,0,0,1,0,0,0,1,0,0,0,0]^T$ for C minor, assuming that the elements correspond to $C, C^{\#}, D, \ldots$. For all other major and minor keys to be considered too, each of these two vectors are circularly shifted 12 times for triads in each of the 24 keys. We end up with a vocabulary size of $V = 24$, and $24 \times N$ feature vectors for a song with N segments. For our second experiment, we used the $k-means$ algorithm to cluster all chroma vectors into K clusters, where we experimented with $K = \{128, 256, 512, \ldots\}$. Cluster means were then chosen as templates, and the vocabulary size is set equal to K.

4.4.3 Results and Evaluation

We show the results of our experimentation, again, using five datasets. The first four datasets (WTC, CLA, KP, and ROM) are audio files synthesized from the MIDI datasets described in the previous chapter. The fifth dataset (BEA-

Table 4.1: Summary of results from the LDA-QUANT model. The columns show experimentation with the two different kinds of chroma computations discussed in section 4.3, along with whether or not the dictionary-based method discussed in section 4.4.2 is used. The model(s) that achieve the highest accuracy across each dataset is bolded. The last column shows the best accuracies achieved by MIDI-based models (from the previous chapter) for reference.

Dataset	TUNED	TUNED + DICT	NNLS	NNLS + DICT	BEST MIDI
WTC-WAV	0.229	0.875	**0.938**	**0.938**	0.979
CLA-WAV	0.570	0.764	0.763	**0.806**	0.865
KP-WAV	0.413	**0.717**	0.652	0.674	0.804
ROM-WAV	0.216	0.737	0.748	**0.791**	0.820
BEA-WAV	0.36	0.610	**0.743**	0.699	0.753

WAV) contains real audio recordings from 10 Beatles albums. Our main task for evaluation is overall key-finding, but we also show the results of chord-estimation on the two datasets for which chord labels are available.

Audio Key Finding

Table 4.1 summarizes accuracy of overall key predictions. The first four columns experiment with different types of features: *quantized* and *nnls* refer to features explained previously in this chapter. If the word *dict* follows, then the indicated feature was paired with the method of using a dictionary of triad-based templates, also described previously. For comparison, we also show the best results obtained from the corresponding MIDI datasets; this gives us an idea of how much is lost (or gained) by going from MIDI – a clean, discrete representation, to audio.

From the results table, we see that the NNLS chroma features are far superior than the quantized features alone. This tell us that the removal of harmonics before forming the chroma features is extremely important. It also seems that using the triad-based dictionary of templates helped things – in fact, the improvement of the quantized features *with* the templates over the same features *without* the templates is staggering. This suggests that the use of these triad-based templates mimics the more complex methods for removing partials before feature computation. Finally, comparing the audio models with the MIDI models also suggests that MIDI features are more suitable for tasks such as harmonic analysis. This is not surprising, as MIDI features do not have the problem of partials or noisy transcriptions. While audio features may contain more information such as timbre and melodic phrasing, those features are not particularly useful for the harmonic analysis task. In the next section, we try to improve on these basic audio models in order to close the gap between the MIDI-based models.

Audio Chord Estimation

Two of our audio datasets have chord annotations that we could evaluate our chord predictions against. The first dataset is KP-WAV – since this dataset is synthesized from MIDI, the same chord labels used in the previous chapter work

here as well. Using NNLS chroma, with no dictionary templates, we obtained an accuracy of 45.0% (out of a total of 884 frames). Comparing with the 63.5% accuracy obtained on the corresponding MIDI dataset, this is a considerable drop in accuracy.

The second dataset is BEA-WAV. While this dataset is *not* synthesized from MIDI, chord annotations were created by Harte [27] and then made available by the Centre for Digital Music at Queen Mary, University of London[2]. Using the same features and set-up as the KP-WAV dataset, we achieved a chord-estimation accuracy of 61.8%. Note that since this dataset is *not* synthesized from the MIDI Beatles dataset (BEA-MID), chord accuracy between MIDI and audio features is not directly comparable. We discuss these results in more detail later in section 4.6.

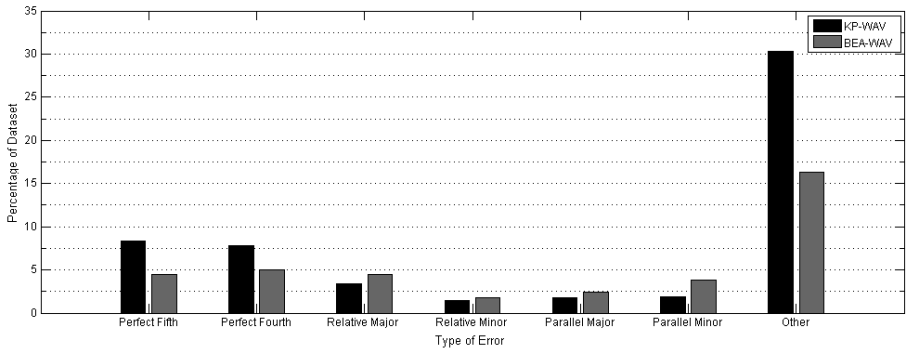


Figure 4.2: A summary of common errors made during chord-estimation. Errors are shown for two datasets: KP-WAV is synthesized from the MIDI dataset containing excerpts from the Kostka-Payne Tonal Harmony workbook [38]; BEA-WAV contains real audio from 10 Beatles albums.

Figure 4.2 shows a summary of the chord errors made on these two datasets. Compared to overall key-finding, the number of "reasonable" errors is considerably lower, and in general, chord estimation accuracy seems to be significantly lower. A possible explanation is that in these LDA-based models, chord predictions depend heavily on the overall key distribution learned for each song – when the input features are noisy, irrelevant keys are given high weights. Even though the correct overall key is given the highest weight (allowing the algorithm to predict the correct overall key), many segments will be given chord labels that are drawn from the

[2]http://www.isophonics.net

irrelevant keys. In general, we find that songs that do well at chord estimation have highly peaked key distributions (i.e. only two or three keys are given very high weights, with other keys given little or zero weights). These peaked keys generally correspond to the correct chords that are used throughout the piece. On the other hand, songs that give weights to many keys generally do poorly on the chord estimation task, as most of the keys given large weights are irrelevant keys. Future work may consider restricting the key distributions to give high weights to only a couple of keys, since most songs only contain a small set of chord changes.

4.5 LDA-AUDIO Model

In this section, we decide to further pursue a model that would specifically cater to audio input. In short, we include an extra step in the generative process that assumes that the chroma features we observe are drawn from the MIDI note counts – these note counts are now considered to be hidden variables that must be inferred. In the next section, we first highlight the differences between the previous LDA-MIDI model, and the new audio model, which we refer to as LDA-AUDIO. In the remaining sections, we flesh out the model and a new variational inference algorithm to learn and estimate the parameters of this model.

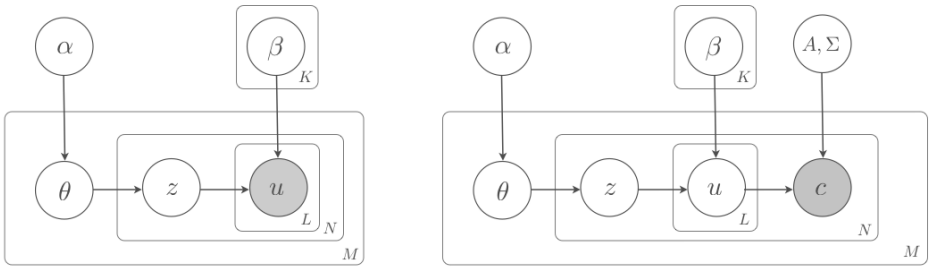


Figure 4.3: A comparison between the graphical models of the (a) LDA-MIDI model from the previous chapter, with that of the (b) LDA-AUDIO model, which will be described in this chapter.

4.5.1 Overview

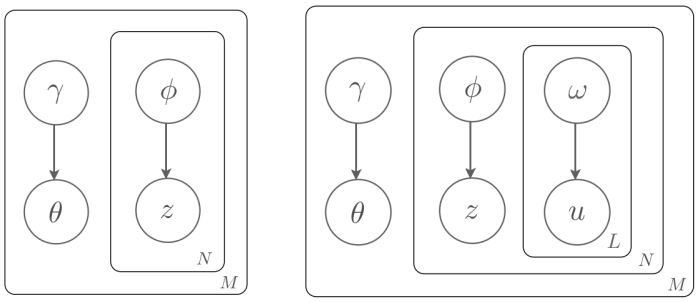
While the two models are conceptually similar, they differ substantially at a low level. To prepare the reader for the detailed discussion ahead, we first provide an overview of differences below, highlighting several important aspects of the model at a high-level. Section numbers are recorded next to each heading indicating the section in which the reader can find more details.

Generative Process (4.5.2). The differences in the generative process can be easily seen in the graphical representation of the two models. Figure 4.3 shows the two graphical models side-by-side. Previously, the MIDI notes within each musical segment was observed, and shaded. In the LDA-AUDIO model, the MIDI note must now be inferred from our new observed node: chroma vectors extracted from audio segments.

Model Parameters (4.5.4). Conceptually, the model parameters stay unchanged from LDA-MIDI (though the update rules will look different if derived purely from audio data). These include parameters α (distribution of topics over corpus) and β (topics as distribution over pitches – i.e. *key profiles*). LDA-AUDIO also introduces a couple of new model parameters A and Σ. These are gaussian parameters used for inferring the observed chroma features from the original, discrete MIDI notes (or pitch-class counts).

Variational Parameters (4.5.3). Just as in LDA-MIDI, the important posterior distribution is intractable and requires variational approximation. Figure 4.4 shows the simplified variational model for LDA-AUDIO, in comparison to that of LDA-MIDI. We see that variational parameters ϕ and γ remain unchanged; however, ω is a new variation parameter that approximates the pitch-class counts in a given musical segment.

A much more thorough discussion is presented for each of the above sections in the remainder of this chapter.



(a) LDA-MIDI (b) LDA-AUDIO

Figure 4.4: A comparison between the graphical models of the simplified variational model for (a) LDA-MIDI from the previous chapter, and (b) LDA-AUDIO, which will be described in this chapter.

4.5.2 Generative Process

The generative process of the LDA-AUDIO model builds on top of the previous LDA-MIDI model; the key difference here is that we no longer have knowledge of exact pitch information as we did before with MIDI data. Instead, our new probability model represents each input song as a collection of chroma vectors $c \in \mathcal{R}^{12}$. Previously observed notes u are now hidden random variables that must be inferred from these chroma vectors. The new generative process for each song in the corpus is outlined below:

1. Choose topic weight vector $\theta \propto Dirichlet(\alpha)$.

2. For each of n segments in a song:

   (a) Choose topic $z_n \propto Multinomial(\theta)$.

   (b) For each of ℓ notes in the nth segment:

      i. Choose a pitch-class $u_{n\ell} \propto Multinomial(\beta)$, where $\beta_{ij} = p(u_{n\ell} = i | z_n = j, \beta)$.

   (c) Generate a chroma vector $c_n \propto \mathcal{N}(Ax_n, \Sigma)$

Note that the first four steps of this generative process are identical to that of LDA-MIDI, with an additional step that occurs at the end: once all notes $\{x_{n1}, \ldots, x_{nL}\}$

have been generated for a segment x_n, a chroma vector c_n is drawn from the probability distribution

$$p(c_n|x_n, A) = \frac{1}{|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(c_n - Ax_n)^T\Sigma^{-1}(c_n - Ax_n)\right\} \tag{4.3}$$

As before, the topic weight vector θ is a K-dimensional vector, where $K = 24$, representing the 24 major and minor keys of western classical music. Topics are analogous to musical keys, and thus z_n can take on one of K keys/topics. Each note u can take on one of $V = 12$ pitches, and β encodes a profile for each key as a distribution over one of the V pitches. The chroma vectors c (described in section 4.3) are V-dimensional vectors, where each element $V_i \in (0, 1)$, indicating the strength of the ith semi-tone for the time period in that segment. Additionally, A is a $V \times V$ transformation matrix and Σ is a $V \times V$ covariance matrix that must be estimated from the data. Now, given parameters α, β, A, and Σ, and N musical segments per song, the joint distribution for a single song is given by:

$$p(\theta, \mathbf{z}, \mathbf{x}, \mathbf{c}|\alpha, \beta, A, \Sigma) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta)\ p(x_n|z_n, \beta)\ p(c_n|x_n, A, \Sigma) \tag{4.4}$$

The new graphical representation shown in Figure 4.3 reflects these changes with the additional (observed) node c, to represent chroma features, and (unobserved) node u, to represent the symbolic pitch classes that now must be inferred from the chroma features.

As in LDA-MIDI, we have two tasks at hand. First is the problem of inference. From the previous section, we have a model that is completely specified given that we know our parameters α, β, A, and Σ. Thus, we can use probabilistic inference to analyze each song in terms of the observed chroma features that we extract from audio recordings. This is done by computing the posterior distribution using Bayes rule:

$$p(\theta, \mathbf{z}, \mathbf{x}|\mathbf{c}, \alpha, \beta, A, \Sigma) = \frac{p(\theta, \mathbf{z}, \mathbf{x}, \mathbf{c}|\alpha, \beta, A, \Sigma)}{p(\mathbf{c}|\alpha, \beta, A, \Sigma)} \tag{4.5}$$

Note that we may use the notation $\mathbf{c}$ and s interchangeably, to indicate a single song, depending on the context (the former specifically denotes the set of all chroma feature vectors in a song). The denominator in Eq. 4.5 is the marginal probability

of each individual song, which can be obtained by integrating over θ (since $p(z_n|\theta)$ $= \theta_i$ for the unique i such that $z_n^i = 1$), and then summing over all possible combinations of hidden topics z and notes x:

$$p(\mathbf{c}|\alpha, \beta, A, \Sigma) = \int_\theta \sum_{\mathbf{z}} \sum_{\mathbf{x}} p(\theta, z, x, c|\alpha, \beta, A, \Sigma)\ d\theta \tag{4.6}$$

$$= \int_\theta \sum_{\mathbf{z}} \sum_{\mathbf{x}} p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta)\ p(x_n|z_n, \beta)\ p(c_n|x_n, A, \Sigma)\ d\theta \tag{4.7}$$

$$= \int_\theta p(\theta|\alpha) \prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta) \sum_{x_n} p(x_n|z_n, \beta)\ p(c_n|x_n, A, \Sigma)\ d\theta \tag{4.8}$$

The probability of the entire corpus of songs is then the product of all marginal probabilities for each song:

$$p(\mathbf{S}|\alpha, \beta, A, \Sigma) = \prod_{d=1}^{M} \int_{\theta_d} p(\theta_d|\alpha) \prod_{n=1}^{N} \sum_{z_{dn}} p(z_{dn}|\theta) \sum_{x_{dn}} p(x_{dn}|z_{dn}, \beta)\ p(c_{dn}|x_{dn}, A, \Sigma) \tag{4.9}$$

Now, we previously assumed that we know values for parameters α, β, A, and Σ in order to infer hidden variables, when in fact, we do not. Thus, the second problem is that of learning. In our model, we would like to estimate parameters that will maximize the log-likelihood of our corpus, assuming that we can compute marginal probabilities:

$$\mathcal{L}(\alpha, \beta, A, \Sigma) = \sum_m \log p(s_m|\alpha, \beta, A, \Sigma) \tag{4.10}$$

The Expectation-Maximization (EM) algorithm is typically used to find maximum likelihood solutions in a latent variable framework like this. However, the marginal probabilities in equation 4.8 are intractable. Before, in LDA and LDA-MIDI, this was due to the coupling between θ and β; now we have an additional latent variable u that exacerbates the intractability even further. This prevents a straightforward application of the EM algorithm, just as in the LDA-MIDI model, and leads us to develop a variational EM algorithm that approximates the inference.

Having introduced the generative process in the previous section, we now have two tasks at hand: to approximate the joint posterior distribution (since the true posterior is intractable) and obtain estimates for model parameters so that we can use the LDA-AUDIO model on unseen test data. We tackle the first part

in section 4.5.3 by substituting a simpler, tractable family of distributions, with new variational parameters that we must infer. Assuming that the variational parameters are known, we solve the second part by learning model parameters in a maximum likelihood fashion (section 4.5.4). Finally, we put these two parts together in section 4.5.5 with a variational expectation-maximization (EM) algorithm that alternates between inferring variational parameters and obtaining maximum likelihood estimates of model parameters.

4.5.3 Variational Inference

To approximate the joint posterior distribution, we substitute the true posterior with a simplified, factorized form instead. This effectively assuming that there are no dependencies between the variables θ, β, and u, thereby creating a new family of tractable distributions whose statistics are easy to compute. Figure 4.4 reflects the removal of these dependencies by dropping edges in the graphical model – we see that there are no longer edges between θ and z, and z and u.

This family of distributions on the latent variables is now characterized by the following variational distribution :

$$q(\theta, \mathbf{z}, x|\gamma, \phi, \omega) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n)\ q(u_n|\omega_n) \tag{4.11}$$

where γ is a Dirichlet parameter:

$$q(\theta|\gamma) = \frac{\Gamma(\sum_i \gamma_i)}{\prod_i \Gamma(\gamma_i)} \prod_{i=1}^{K} {\theta_i}^{\gamma_i - 1} \tag{4.12}$$

and both $\phi_1, \ldots, \phi_n$ and $\omega_1, \ldots, \omega_n$ are multinomial parameters:

$$q(z_n|\phi_n) = \prod_{i=1}^{K} {\phi_{ni}}^{z_{ni}} \tag{4.13}$$

$$q(u_n|\omega_n) = \prod_{j=1}^{V} \prod_{s=0}^{S} {\omega_{jns}}^{I(u_{jn}, s)} \tag{4.14}$$

Intuitively, these variational parameters that we have introduced approximate the hidden variables in our original model. For instance, γ, a Dirichlet

parameter for which each song's topic vector is drawn from, ends up approximating the distribution of topics for each song in the corpus. ϕ_n approximates the distribution of topics for the nth segment within a song. Finally, each ω_{jns} is a multinomial parameter that approximates the number of times pitch j appears in segment n of the music. Having specified this simplified family of distributions, we want to choose variational parameters γ, ϕ, and ω that will result in a probability distribution that matches the true joint distribution as closely as possible.

First, we begin by decomposing the log marginal probability as a sum of some lower-bound $\mathcal{L}$ and the KL divergence between the true and variational posteriors. This is shown below (for readability, we drop dependence parameters):

$$\ln p(\mathbf{c}|\alpha, \beta, A) \tag{4.15}$$

$$= \ln \int_\theta \sum_{\mathbf{z}} \sum_{\mathbf{x}} p(\theta, \mathbf{z}, \mathbf{x}, \mathbf{c}) \tag{4.16}$$

$$= \ln \int_\theta \sum_{\mathbf{z}} \sum_{\mathbf{x}} q(\theta, \mathbf{z}, \mathbf{x}) \frac{p(\theta, \mathbf{z}, \mathbf{x}, \mathbf{c})}{q(\theta, \mathbf{z}, \mathbf{x})} \tag{4.17}$$

$$\geq \int_\theta \sum_{\mathbf{z}} \sum_{\mathbf{x}} q(\theta, \mathbf{z}, \mathbf{x}) \ln \frac{p(\theta, \mathbf{z}, \mathbf{x}, \mathbf{c})}{q(\theta, \mathbf{z}, \mathbf{x})} \tag{4.18}$$

$$= \int_\theta \sum_{\mathbf{z}} \sum_{\mathbf{x}} q(\theta, \mathbf{z}, \mathbf{x}) \ln p(\theta, \mathbf{z}, \mathbf{x}, \mathbf{c}) - \int_\theta \sum_{\mathbf{z}} \sum_{\mathbf{x}} q(\theta, \mathbf{z}, \mathbf{x}) \ln q(\theta, \mathbf{z}, \mathbf{x}) \tag{4.19}$$

$$= E_q[\ln p(\theta, \mathbf{z}, \mathbf{x}, \mathbf{c})] - E_q[\ln q(\theta, \mathbf{z}, \mathbf{x})] \tag{4.20}$$

$$= \mathcal{L}(\gamma, \phi, \omega) \tag{4.21}$$

We note that Jensen's inequality is applied to arrive at equation (4.18). Now, in order for the factorized variational posterior to match the true posterior distribution as closely as possible, we must find the variational parameters that will minimize the difference between $\ln p(\mathbf{c}|\alpha, \beta, A)$ and the lower-bound $\mathcal{L}$. It can be verified that this difference is actually the KL divergence between these two families of functions (p and q):

$$\ln p(\mathbf{c}|\alpha, \beta, A) = \mathcal{L}(\gamma, \phi, \omega) + KL(\ q(\theta, \mathbf{z}, \mathbf{x}) \ ||\ p(\theta, \mathbf{z}, \mathbf{x}, \mathbf{c})\ ) \tag{4.22}$$

It should be clear from equation (4.22) that minimizing the KL divergence will be equivalent to maximizing $\mathcal{L}$, and thus making it the tightest lower-bound possi-

ble. As is done in the original LDA paper, we take the approach of finding the variational parameters γ^*, ϕ^*, and ω^* that directly maximize the lower-bound $\mathcal{L}$:

$$(\gamma^*, \phi^*, \omega^*) = \underset{(\gamma,\phi,\omega)}{\operatorname{argmax}}\ \mathcal{L}(\gamma, \phi, \omega) \tag{4.23}$$

This can be achieved by taking the derivative of the lower-bound, setting it to 0, and solving for γ, ϕ, and ω in succession. Section C.1 in the appendix goes into more detail about how this lower-bound is computed and how the optimal variational parameters are inferred. Assuming that we are given model parameters α, β, A, and Σ^{-1}, the following update rules are obtained for each variational parameter:

$$\gamma_i = \alpha_i + \sum_{n=1}^{N} \phi_{ni} \tag{4.24}$$

$$\phi_{ni} \propto \exp[\Psi(\gamma_i)] \prod_{j=1}^{V} \prod_{s=1}^{S} \beta_{ij}{}^{\omega_{jns} s} \tag{4.25}$$

$$\omega_{jns} = \frac{\exp[f(s)]}{\sum_{s'}^{S} \exp[f(s')]} \tag{4.26}$$

where

$$f(s) = s \sum_{k=1}^{V} \sum_{l=1}^{V} c_{nk}^T \Sigma_{kl}^{-1} A_{lj} - \frac{1}{2} s^2 M_{jj} - s^2 \sum_{k \neq j} \omega_{kns} M_{kj} + s \sum_{i=1}^{K} \phi_{ni} \ln \beta_{ij} \tag{4.27}$$

Detailed derivations for the ϕ and ω updates can be found in Appendix C.2; the update rule for γ is unchanged from the original LDA model, and thus, is not included. In passing, we do note that for the update of ω, each ω_{jns} term depends on all other ω_{kns} terms for which $j \neq k$; thus this update rule must be applied individually along that particular dimension in the matrix. Since all three update equations have terms that depend on others, all parameters must be updated repeatedly until convergence for each song.

4.5.4 Parameter Estimation

Just as in the original LDA model, we can use a maximum likelihood approach to estimate the parameters in our model. This is done by finding parameters

α, β, A, and Σ that maximize the log-likelihood of our observed data:

$$(\alpha^*, \beta^*, A^*, \Sigma^*) = \underset{(\alpha,\beta,A,\Sigma)}{\operatorname{argmax}} \sum_{m=1}^{M} \log p(\mathbf{c}_m|\alpha, \beta, A, \Sigma) \tag{4.28}$$

However, as shown before, this log-likelihood cannot be computed tractably. Instead, we can maximize the tractable lower bound obtained through variational inference with respect to these model parameters. The update equation we get for α is unchanged from the original LDA model, and is not shown here. The new update equation for β is shown below:

$$\beta = \sum_{d}^{M} Q^{(d)} \phi^{(d)} \tag{4.29}$$

$$A = cx^T(xx^T)^{-1} \tag{4.30}$$

$$\Sigma = -\frac{1}{2}cc^T + cx^T A^T - \frac{1}{2}Axx^T A^T \tag{4.31}$$

As we will discuss next in the Experimental Set-up section, the update rules for Gaussian parameters A and Σ shown here assume that pitch-class count vectors, x, are available from corresponding MIDI data during a training phase. The estimates obtained during the training phase are then used during testing. Appendix C.3 also describes the derivations for these update rules in more detail.

4.5.5 Variational EM

Both sections 4.5.3 and 4.5.4 together make up an alternating *variational* Expectation Maximization (EM) procedure that is used to find approximate Bayes estimates for our LDA-AUDIO model. Recall that the original model specified in eq. 4.4 is a latent variable model that would have traditionally utilized the Expectation-Maximization algorithm for inference and learning. However, due to the intractability of the likelihood function, we resorted to computing a tractable lower-bound instead. The variational parameter estimation acts as the expectation step as it computes the expectation of the latent variables in our model. We then use these expectation computations to maximize the lower-bound with respect to the model parameters, as shown in section 4.5.4; this acts as the maximization step. A summary of this variational EM algorithm is as follows:

E-Step: For each song, find the values of the variational parameters $\{\gamma_d^*, \phi_d^*, \omega_d^*\}$ that maximize the lower-bound using update eq. (4.24 - 4.26). This corresponds to computing the expectation of the complete-data log likelihood under the approximate lower-bound.

M-Step: Maximize the resulting lower-bound with respect to the model parameters $\{\alpha^*, \beta^*, A^*, \Sigma^*\}$, using the update eq. (4.29 - 4.31). This corresponds to finding the maximum likelihood estimates under the approximate posterior computed in the E-step.

Just as in regular EM, the expectation and maximization steps are computed iteratively until convergence. What we end up with are variational and model parameter estimates. The variational parameter α gives a rough distribution over topics in the entire corpus; γ gives a distribution over musical keys in each song, ϕ gives a distribution over musicals for each musical segment, and ω estimates the contribution of each of the 12 semi-tones in a given segment of music.

4.6 Results and Evaluation

Below, we evaluate the performance of the LDA-AUDIO model. We begin with some details about experimental set-up, and then show results on key-finding and chord-estimation tasks.

4.6.1 Experimental Set-up

In this section, we give some details about how we set up our models and experiments. For chroma features, we compare both quantized and NNLS chroma, without any sort of dictionary. The remaining details of set-up mostly concern different configurations for learning variational and model parameters A and Σ.

As shown in eq. 4.3, LDA-AUDIO model assumes that each observed chroma vector (corresponding to a segment of music) is drawn from a Gaussian distribution, parameterized by covariance Σ, and a linear transform A on the original, unobserved MIDI note count vector x. There are two ways in which the Σ and

A parameters can be estimated: (1) assuming that MIDI note counts are never available, and learning A and Σ from audio data alone, or (2) assuming that MIDI note counts for audio data can be obtained during a prior training phase.

While the first approach would be ideal (as no additional data or training phase would be necessary), in practice it performed very poorly in our experiments. This may be due to the fact that our datasets are too small to learn so many parameters accurately in unsupervised fashion. On the other hand, learning A and Σ from properly aligned MIDI and audio data provides a much more accurate estimation. While this second approach requires additional work, we note that the assumption of observing MIDI data during an initial training phase is very reasonable. Given a specific genre of music, it is not difficult to obtain a set of MIDI files and then synthesize them into audio. The parameters learned from this process should be very similar to values that would have been learned by unsynthesized audio (i.e.directly from a CD recording) songs from the same genre. We note that even if we do utilize observed MIDI data during training, our overall model is still unsupervised, as harmony (i.e. key or chord) labels are never provided during the training phase, and variational parameters (γ, ϕ, ω) are still inferred directly from the audio data. Appendix C.3.2 discusses the estimation of A and Σ in more detail.

4.6.2 Overall Key-Finding

To evaluate the LDA-AUDIO model, we again use it to perform overall key-finding. Key prediction is done in the same way as previous MIDI-based models: by looking at the key distribution for each song (from variational parameter γ), and assigning the key to the one with the highest weight. Table 4.2 summarizes these results, comparing both quantized chroma features and NNLS chroma features. In the last two columns, best results obtained from the more basic LDA-based audio models (section 4.4), as well as the best accuracy obtained from corresponding MIDI datasets (chapter 3) are also shown for reference. The highest accuracy across all audio-based models (first three columns) is bolded for each dataset.

In general, there are several trends. Again, NNLS chroma features make

a big difference to final key-finding results, for the same reasons described previously for the LDA-QUANT model: the removal of partials in the chroma features (which NNLS chroma performs) is extremely important to our key-finding task. Second, we observe that the more unified LDA-AUDIO model indeed outperforms the simpler LDA-QUANT model on the majority of datasets. This indicates that the additional assumption of drawing observed chroma features from discrete MIDI note counts is effective. In Figure 4.5, we also show a summary of the most common errors made by the LDA-AUDIO model for key-finding. Consistent with MIDI-data findings, almost half of the errors in the Beatles dataset come from perfect fourth errors. As explained before, many of the Beatles songs are written with a Blues style influence, resulting in frequent usage of a dominant seventh chord (in which the seventh degree in the scale is flatted), as opposed to chords based on the common triad. Because our model only consists of 24 major/minor chords, these dominant seventh chords are not modeled, and when mapped to the 24 major/minor chords, most closely align with the major triads in the key that is a perfect fourth above (e.g. an E7 chord is most similar to an A major triad). We saw in the previous chapter that learning correlations between the keys (using CTM-MIDI) alleviated this problem to some extent. Future work may consider

Table 4.2: Summary of results from overall key-finding for LDA-AUDIO, along with comparisons to best accuracies obtained by previous models. The first two columns show results for LDA-AUDIO using two different types of chroma features. The third column showcases the best accuracy obtained from the LDA-QUANT model (section 4.4), and the fourth column shows the best accuracy obtained from the MIDI-based models discussed in chapter 3.

	LDA-AUDIO		Previous Models	
Dataset	TUNED Chroma	NNLS Chroma	LDA-QUANT	LDA-MIDI
WTC-WAV	0.802	0.906	**0.938**	0.979
CLA-WAV	0.713	**0.819**	0.806	0.865
KP-WAV	0.565	**0.717**	**0.717**	0.804
ROM-WAV	0.709	**0.820**	0.791	0.820
BEA-WAV	0.500	**0.750**	0.743	0.753

incorporating something similar to the audio version of the model

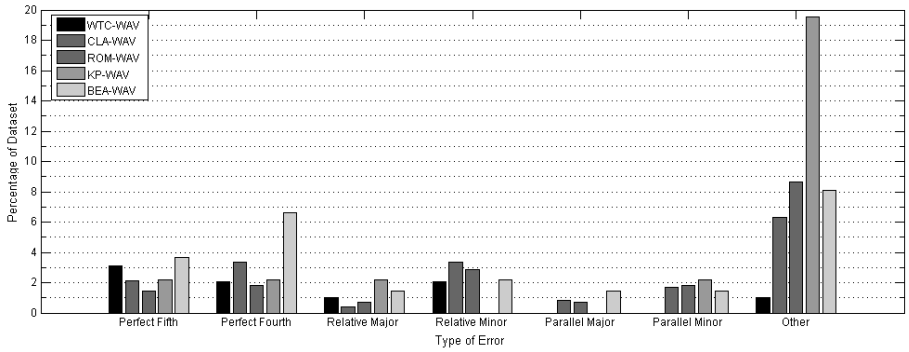


Figure 4.5: A summary of common errors made during key-finding by the LDA-AUDIO model.

While we have shown that our current audio-based model improves on previous, simpler models, it is difficult to do a fair comparison with other previously published models since there are few standard datasets. For key-finding performance on classical music, our LDA-AUDIO model gives an average of 83% across all classical datasets. According to the summary of previous work in Tables 2.1-2.3, is comparable to what others have achieved on similar classical datasets [63, 84, 64, 32] – most of which use supervised methods and/or use hand-chosen parameters. Note, also, that several previous works report using MIREX scores – this is a particular way of scoring used by the popular MIREX competition.[3] For fair comparison, our MIREX composite score is 85.0% out of a total of 657 classical songs.

Though not specifically noted in that table, some have also reported audio key-finding accuracy specifically on various subsets of Bach's Well Tempered Clavier: Lee's key and chord-finding system [48] reports an 85.4% accuracy on the 48 preludes and fugues from Book I; Noland's key-finding system [61] reports a 98% accuracy on the same dataset. Our accuracy on just Book I is 91.7% using LDA-AUDIO, and 93.8% using the basic LDA audio model. While our model does

[3]MIREX is a popular MIR competition held as a part of ISMIR. One of their tasks is audio chord-estimation, which uses the Beatles dataset as a test set. According to MIREX scoring, partial points are given when incorrectly predicting a related key. The points are rewarded as follows: Correct key: 1 point; perfect fifth: 0.5 points; relative major/minor: 0.3 points; parallel major/minor: 0.2 points. Please see http://www.music-ir.org/mirex/wiki/MIREX_HOME for more information.

not beat the state-of-the-art, we note that both reported models use supervised methods and/or include hand-derived musical information.

While many results have been reported for the audio Beatles dataset, most report on the task of chord estimation and not key-finding. A couple that do report on key-finding accuracy includes: Lee [48] who reports a 97% key-finding accuracy across 28 Beatles songs (from the albums *Please Please Me* and *Beatles for Sale*); Noland [61] who reports a 71% accuracy across 110 Beatles songs; and Rocher [68] who reports a 62.4% accuracy across 174 Beatles songs. As shown above, our highest accuracy across 136 Beatles songs is 75%. Perhaps Lee's high accuracy is due to the fact that he is the only one out of the three that trains a supervised model that includes Beatles MIDI data, while the other two models rely on hand-chosen, musically inspired parameters. Lee also uses a much smaller dataset with, arguably, some of the simpler songs. As a direct comparison, our LDA-AUDIO model achieves a much higher accuracy of 89.2% on the same dataset that Lee reports accuracy on.

4.6.3 Chord Estimation

For completeness, we also report the accuracy of the chord-estimation task using LDA-AUDIO on the same two datasets described previously in section 4.4: KP-WAV achieves a 55.9% accuracy (compared to 45.0% from the basic LDA audio model) and BEA-WAV achieves a 60.8% accuracy (compared to 67.1% from the basic LDA audio model). This is summarized in Table 4.3.

The current state-of-the-art for chord-estimation on the Beatles dataset

Table 4.3: A comparison of chord-estimation accuracies for the KP-WAV and BEA-WAV datasets. We report on both audio models described in this chapter, as well as its raw accuracy and MIREX accuracy (refer to footnote 3)

Dataset	LDA-QUANT		LDA-AUDIO	
	Raw	MIREX	Raw	MIREX
KP-WAV	0.450	0.513	**0.559**	**0.608**
BEA-WAV	**0.618**	**0.671**	0.491	0.559

(BEA-WAV) stands at around 80% and above (MIREX scoring) [57, 60]. Compared to this, our accuracies are quite low, indicating that our bag-of-segments model does not model chord progressions successfully. For this task, HMM-based models (which are used for most chord progression models) have a large advantage over our LDA-based model since chord transition probabilities are properly modeled. These make a large impact in being able to more accurately predict how likely a modulation will occur, and to which local key or chord the next segment will transition to. Our post-processing step of using mode-based filtering tries to perform temporal smoothing to address this issue, but is not nearly as powerful as transition state modeling in HMMs. Most state-of-the-art chord-estimation systems also take a unified approach in which bass notes are estimated and incorporated into chroma feature calculations, as well as beat and meter structure. For future work, we may consider incorporating temporal structure into our LDA-based models, as well as more information about beat and meter structure.

4.7 Conclusion

In this chapter, we presented several LDA-based models for harmonic analysis that takes in raw audio input. First, we experimented with using audio input into the LDA-MIDI model described in Chapter 3. Instead of using MIDI note counts as input, we use chroma features. Two methods of chroma feature calculations are described and used throughout our experiments. This basic LDA-based audio model did well, but still had lower accuracy rates than the MIDI-based models described in Chapter 3. We then presented a more unified model, called LDA-AUDIO, that improves on the previous by modeling observed chroma features as being drawn from discrete pitch-class counts. We sketch out the new generative process, as well as a new variational inference algorithm for the model. Using LDA-AUDIO, we find that key-finding accuracy generally improves, while chord-estimation accuracy improved only for the classical dataset. From these experiments, we have gained insight into where our model excels, as well as areas that could use improvement.

4.8 Acknowledgements

Chapter 4, in part, is a reprint of the material as it appears in Proceedings of the Neural Information Processing Systems (NIPS) Workshop on Topic Modeling 2009. Hu, Diane; Saul, Lawrence K. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Conclusion

5.1 Summary

In this dissertation, we have presented a novel approach for key-finding in symbolic and audio music. The main contribution of this work is developing an unsupervised framework that requires very little prior knowledge and no hand-chosen parameters. For symbolic input, our performance on key-finding tasks seems to match (if not exceed) most previously published approaches. For audio input, key-finding accuracy also seems comparable to many previously published models – even those who use supervised frameworks and require extensive prior music knowledge.

Our approach is also unique in that no time-series modeling is used. Instead, our LDA-based models represent each song as a bag-of-segments, containing notes. Segment order is not modeled, and note order within a segment is not modeled either. In doing so, we have addressed some interesting questions that we raised at the beginning of this thesis. First, we have learned that groups of notes in close proximity contain a tremendous amount of harmonic information – analogous to semantic information contained in a sentence in a book. The harmonic content is strong enough that we can successfully perform global and local key-finding with no more information than the co-occurrence information given by notes contained within a song. We also show that it is possible to learn prior music knowledge solely from data; and in doing so, provide a more flexible framework that generalizes to

varying music genres. Finally, though our model of key-finding performs with fairly high-accuracy, we find that discarding note and segment order information hinders us from successful chord estimation. In the next section, we list some ideas that we have for future directions that may improve the accuracy of harmony-related tasks, or build on top of our work.

5.2 Future Directions

There are two areas which we would like to investigate further for future work. The first is concerned with better modeling, and the second aims at novel applications that can use the harmonic descriptors that we have introduced.

5.2.1 Potential Model Enhancements

There are a couple of areas that can be improved in our LDA-based models. For both symbolic and audio input, being able to incorporate time-series information (such as note order or segment order) would most likely increase chord estimation accuracy. One possible implementation of this idea would be to add dependencies between the keys chosen at the segment level. This would involve drawing each segment-level key simultaneously from the key weight distribution as well as the key predicted from the last segment, based on a transition probability table. Learning transition probabilities in this way could greatly reduce the abrupt local key changes that we saw in our LDA-based models, as well as learn likely key transitions (which should be easily learned).

Another area of improvement that applies to both symbolic and audio-based models involves incorporating more specific information that imitates the way human experts perform key-finding and chord transcription. For example, two important considerations are notes that occur during the down-beat (often this is the first beat in a measure), or bass notes (such as the root of a chord). Usually, these notes are more prominent in the melody and sound, and thus, contain more important key information. One possible way to incorporate this into the symbolic model is to learn separate key-profiles for these particular notes and give them

higher weight in the key-finding decision process. This is a bit more tricky in audio – since we discard octave information in our current chroma representation, it is impossible to identify and distinguish bass features. One possible solution is to make use of Mauch's bass chroma features [57]. These are similar to the NNLS chroma features he proposes, but considers only low frequencies before collapsing the spectrogram into the standard 12-dimensional bins. Using these bass chroma features could lead to more accurate key-profiles and more accurate key-finding.

One last area of improvement is specific to our audio model. Recall that the generative process of LDA-AUDIO assumes that the observed chroma features are drawn from the probability distribution where c is the chroma feature, $\mathbf{u}$ are the set of MIDI notes from which the chroma features are drawn from, and A and Σ are parameters of the normal distribution. This step may be unrealistic in portraying the relationship between the chroma features, and the MIDI notes. Perhaps a more realistic method would more closely mirror polyphonic transcription methods, in which chroma features are modeled as a weighted combination of note or chord profiles. This may lead to more accurate chroma representations. Similarly, there may be better ways to define the distribution of ω, the variational parameter from which the hidden MIDI notes $\mathbf{u}$ are drawn from. Currently, it is being modeled as a 10-dimensional multinomial distribution and, unfortunately, creates a large number of parameters ($10 \times V \times N$) that must be learned for each song (where $V = 12$, the size of the vocabulary and N =the number of segments in that song).

5.2.2 Potential Applications

We have already listed many applications that have successfully used harmonic descriptors in section 1.2.2. We go into more detail regarding two of them. The first direction seeks to use song summaries generated by our model (in the form of a distribution over the 24 keys) as features in a supervised learning framework to do genre classification, or even cover song detection. It would be interesting to see how our harmonic descriptors compare to current features used for such tasks.

Second, we have begun initial studies on how harmonic descriptors can be used to model tonal tension. In music, tonal tension can be described as the

building up and resolving of conflict in music. Such work could be used for higher-level applications such as music emotion classification (where the goal is to predict the type of mood or emotion a piece of music embodies; [44] for example) or aid in segmenting scenes from a movie based on its music soundtrack (since increasing tension is often indicative of story content).

Several very complex mathematical models have been built in attempts to model musical tension one of the most most well-known works being based on Lerdahl's General Theory of Tonal Music [51]. These math/theory-based models make annotating an extremely laborious task – a task that usually only few can execute accurately. In [17], Dominquez takes a different, data-based approach. He quantitatively defines tension as the amount of conflict between local and global keys. The more difference, tonally, between a segment of music, and the overall key, the more tension the listener will feel. It is not until local and global keys are harmonious, that the listener feels as if the tension has been resolved. A potential application that builds on top of this idea is to model the global and local key and develop a similarity measure that may predict the rising and falling tension. Initial work in this direction looks promising, which caused us to begin work in collecting ground-truth labels for such a task. Since musical tension is a fairly subjective (and possibly emotive) notion, the most valuable ground-truth should come from human judgements. To collect these judgements, we have created a web interface that will play musical excerpts, and allow users to indicate the amount of musical tension they feel (while listening) buy dragging a slider left and right (right indicates increasing tension, left indicates decreasing tension). Prior work by Krumhansl [40, 22] has collected similar data and shows that such judgements are generally consistent across listeners – both musically and non-musically trained. Once these judgements are collected, evaluation of a tonal tension model will be possible at a larger scale.

5.3 Final Remarks

It is our hope that the presented framework for extracting harmonic information will make a contribution to the MIR community as well as useful MIR applications. We also hope that we have shown a novel application of the original Latent Dirichlet Allocation (LDA) model, and have made some interesting connections between the use of words in expressing semantic topics, and the use of musical notes in expressing harmonic topics. With this phase of research concluded, we hope that future work may bring more insight into the problem of automatic harmonic analysis.

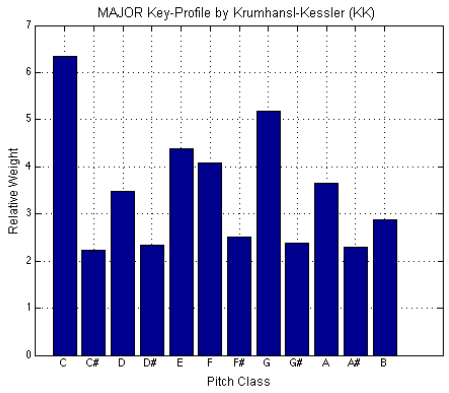
Appendix A

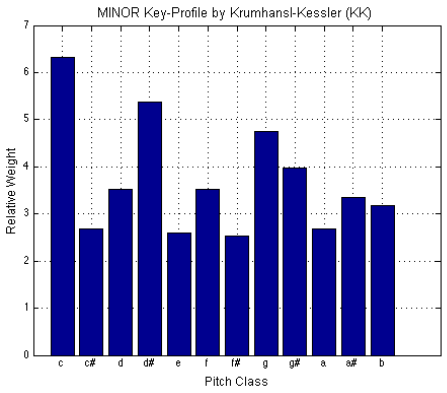
Musical References

A.1 Key-Profiles in Literature

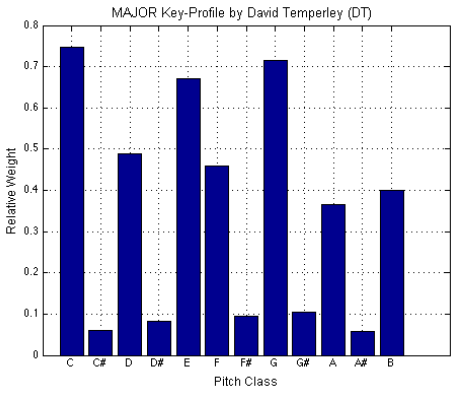
This section presents five popular key-profile weight sets developed by researchers over the years. All key-profiles are for the keys of C Major and C minor; others can be obtained through transposition. Analysis of these key-profiles are attributed to Sapp [73], and unless otherwise noted, these key-profiles are mainly used with the KS key-finding model [41], described in section 2.4.

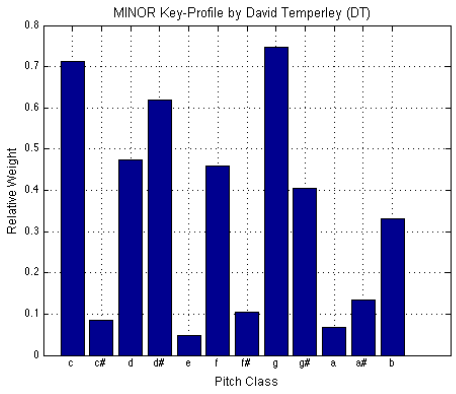
1. Original key weightings from Krumhansl and Kessler [41], obtained from several probe-tone studies. It is sometimes criticized to identify the dominant key as the tonic more often and aggressively than needed



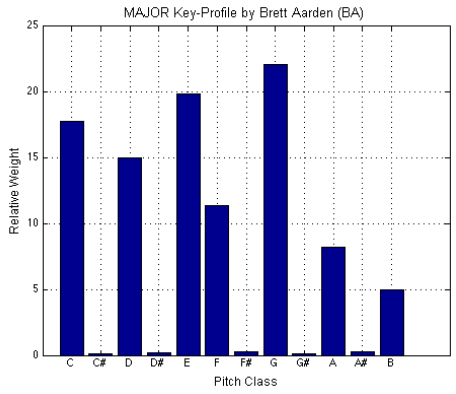


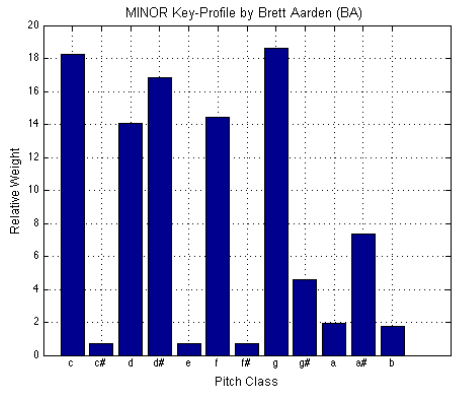
2. Weightings from Temperley [77] which seem to perform better on major songs than minor since the relative major is strongly preferred.



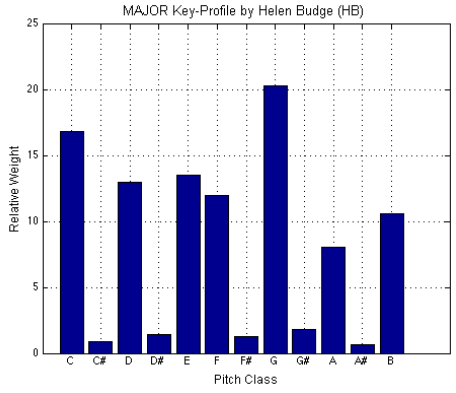


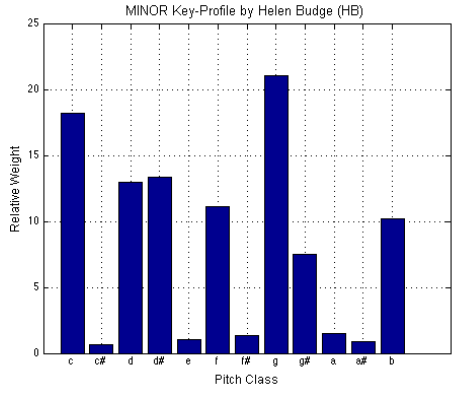
3. Weightings from Aarden, which gives slight bias toward the sub-dominant [1]



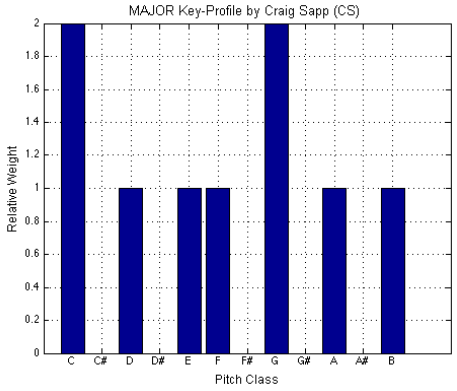


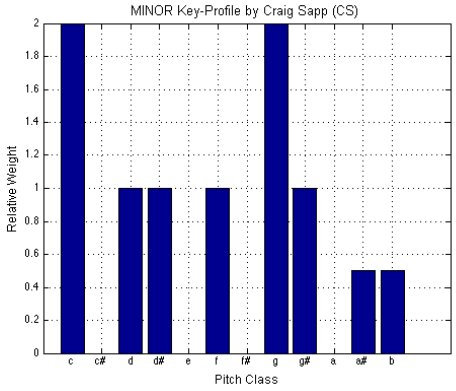
4. Weightings from Budge [28] obtained using chord statistics from a corpus



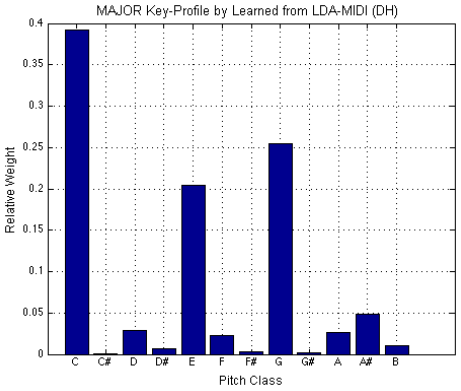


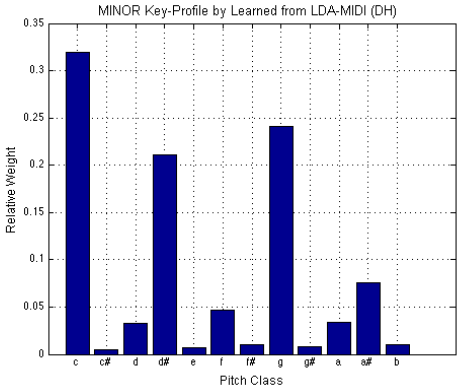
5. Weightings from Sapp [73], derived by: (1) starting with all 0 weights (2) adding 1 to pitches in scale, (3) adding 1 to tonic and dominant scale degrees (to distinguish between major/minor modes).





6. Weightings learned in an unsupervised framework from our LDA-MIDI model. Please refer to section 3.3.4 for more details.





A.2 Frequency and Pitch Mappings

Table A.1: Relationship between pitch, note names, and frequency.

Pitch Name	Alternative Spelling	Frequency
$C3$	$--$	130.81
$C^{\#}3$	$D^{b}3$	138.59
$D3$	$--$	146.83
$D^{\#}3$	$E^{b}3$	155.56
$E3$	$--$	164.81
$F3$	$--$	174.61
$F^{\#}3$	$G^{b}3$	185.00
$G3$	$--$	196.00
$G^{\#}3$	$A^{b}3$	207.65
$A3$	$--$	220.00
$A^{\#}3$	$B^{b}3$	233.08
$B3$	$--$	246.94
$C4$	$--$	261.63
$C^{\#}4$	$D^{b}4$	277.18
$D4$	$--$	293.66
$D^{\#}4$	$E^{b}4$	311.13
$E4$	$--$	329.63
$F4$	$--$	349.23
$F^{\#}4$	$G^{b}4$	369.99
$G4$	$--$	392.00
$G^{\#}4$	$A^{b}4$	415.30
$A4$	$--$	440.00
$A^{\#}4$	$B^{b}4$	466.16
$B4$	$--$	493.88
$C5$	$--$	523.25

Appendix B

Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) [6] is an unsupervised model for discovering latent semantic topics in large collections of text documents. The key insight into LDA is the premise that words contain strong semantic information about the document. Therefore, it is reasonable to assume that documents on roughly similar topics will use the same group of words. Latent topics are thus discovered by identifying groups of words in the corpus that frequently occur together within documents. Learning in this model is unsupervised because the input data is incomplete: the corpus provides only the words within documents; there is no training set with topic or subject annotations.

We will explore the LDA model in the remaining sections. In section B.2, we describe the generative procedure on which LDA is based and the joint distribution that is specified by this procedure. In section B.3, we see how the values of the latent random variables (topics) are inferred by conditioning on the observed random variables (words) using Bayes rule.

B.1 Notation and Terminology

Before moving forward, we first introduce the terminology and notation that will be used throughout the rest of this section.

1. A *word* $w \in \{1, \ldots, V\}$ is the most basic unit of discrete data. For cleaner

notation, w is a V-dimensional unit-based vector. If w takes on the ith element in the vocabulary, then $w^i = 1$ and $w^j = 0$ for all $j \neq i$.

2. A *document* is a sequence of N words denoted by $\mathbf{w} = (w_1, w_2, \ldots, w_N)$, where w_n is the nth word in the sequence.

3. A *corpus* is a collection of M documents denoted by $\mathcal{D} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_M\}$.

4. A *topic* $z \in \{1, \ldots, K\}$ is a probability distribution over the vocabulary of V words. Topics model particular groups of words that frequently occur together in documents, and thus can be interpreted as "subjects." As we will discuss in section B.2, the generative process imagines that each word within a document is generated by its own topic, and so $\mathbf{z} = (z_1, z_2, \ldots, z_N)$ denotes the sequence of topics across all words in a document.

B.2 Generative Process

LDA is an unsupervised, generative model that proposes a stochastic procedure by which words in documents are generated. Given a corpus of unlabeled text documents, the model discovers hidden topics as distributions over the words in the vocabulary. Here, words are modeled as observed random variables, while topics are observed as latent random variables. Once the generative procedure is established, we may define its joint distribution and then use statistical inference to compute the probability distribution over the latent variables, conditioned on the observed variables.

We begin by describing, at a high-level, the process for generating a document in the corpus. First, we draw a topic weight vector (modeled by a Dirichlet random variable) that determines which topics are most likely to appear in a document. For each word that is to appear in the document, we choose a single topic from the topic weight vector. To actually generate the word, we then draw from the probability distribution conditioned on the chosen topic. From this procedure, we see that each word in a document is generated by a different, randomly chosen topic. The available choice of topics for each document are, in turn, drawn from

a smooth distribution over the space of all possible topics. This process can be described more formally below.

For each document indexed by $m \in \{1 \dots M\}$ in a corpus:

1. Choose a K-dimensional topic weight vector θ_m from the distribution $p(\theta|\alpha) = \text{Dirichlet}(\alpha)$.

2. For each word indexed by $n \in \{1 \dots N\}$ in a document:

   (a) Choose a topic $z_n \in \{1 \dots K\}$ from the multinomial distribution $p(z_n = k|\theta_m) = \theta_m^k$.

   (b) Given the chosen topic z_n, draw a word w_n from the probability distribution $p(w_n = i|z_n = j, \beta) = \beta_{ij}$.

Since the generative process imagines each word to be generated by a different topic, the LDA model allows documents to exhibit multiple topics to different degrees. This overcomes the limitations of the mixture of unigrams model which only allows a single topic per document. We also note that the LDA model does not attempt to model the order of words within a document. It is precisely this "bag-of-words" concept that is central to the efficiency of the LDA model.

The parameter α is a K-dimensional parameter that stays constant over all of the documents within a corpus. By drawing topic mixture weights from α, we introduce an additional level of indirection that allows us to analyze an unseen document in relation to the rest of the existing documents in the corpus. This overcomes the limitations of the pLSI model [31] – the predecessor to LDA – which treats the topic mixture weights as a large set of individual parameters that are explicitly linked to documents in the training set. The parameter ϕ is a $V \times K$ matrix that is also estimated from data. It encodes each of the K topics as a distribution over V words. In some instances, an additional Dirichlet prior η is placed over the multinomial parameter β to account for sparsity in case of a large vocabulary. We do not discuss the learning procedure for this hyper-parameter, but the interested reader can refer to the original LDA paper for details.

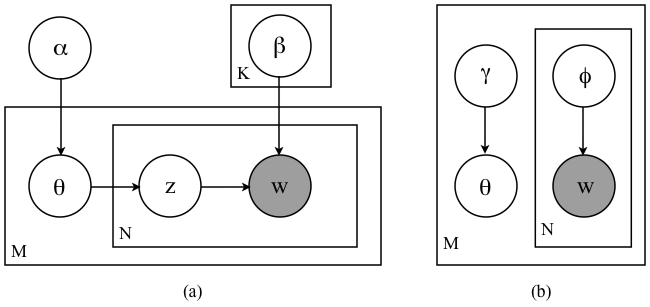


Figure B.1: (a) Graphical representation of LDA and (b) the variational approximation (right) for the posterior distribution in eq. (B.3).

The Dirichlet distribution is given by:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \prod_i \theta^{\alpha_i - 1} \tag{B.1}$$

The Dirichlet is used because it has a number of nice properties: it is in the exponential family, has finite dimensional sufficient statistics, and is a conjugate prior to the multinomial distribution. These properties will make the inference and parameter estimation algorithms for LDA simpler, as we will see in appendix B.3.1. As a Dirichlet random variable, the topic weight vector θ has the property such that $\theta^i \geq 0$ and $\sum_{i=1}^{K} \theta^i = 1$, and thus lies in the $(K-1)$-simplex. Basically, each document is characterized by its own topic weight vector, which indicates the contribution of each of the K topics to that document.

The generative process given above defines a joint distribution for each document $\mathbf{w}_m$. Assuming for now that parameters α and β are given to us, the joint distribution over the topic mixtures θ and the set of N topics $\mathbf{z}$ is:

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{B.2}$$

Figure B.1(a) shows the graphical representation of the joint distribution for the entire corpus. Plate notation is used to more concisely represent multiple independently, identically distributed random variables within the model. All variables that lie within a box (or plate) indicates that it is replicated X number of times, where X is the number in the bottom right hand corner of the box.

B.3 Inference and Learning

The model given in equation (B.2) is completely specified by the parameters β and α. Now, the central task for using LDA is to determine the posterior distribution of the latent topic variables conditioned on the words that we observe in each document. Bayes rule is used:

$$p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta)}{p(\mathbf{w}|\alpha, \beta)}$$

The normalizing factor in the denominator of eq. (B.3) is the marginal distribution, or likelihood, of a document:

$$\begin{aligned} p(\mathbf{w}|\alpha, \beta) &= \int p(\theta|\alpha) \left(\prod_{n=1}^{N} \sum_{z_n \in Z} p(z_n|\theta) p(w_n|z_n, \beta) \right) d\theta \\ &= \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \int \left(\prod_{i=1}^{K} \theta_i^{\alpha_i - 1} \right) \left(\prod_{n=1}^{N} \sum_{i=1}^{K} \prod_{j=1}^{V} (\theta_i \beta_{ij})^{w_n^j} \right) d\theta \end{aligned} \quad \text{(B.3)}$$

The optimal values of α and β are chosen to maximize the log-likelihood of all of the documents in the corpus,

$$\mathcal{L}(\alpha, \beta) = \sum_{m=1}^{M} \log p(\mathbf{w}|\alpha, \beta) \quad \text{(B.4)}$$

A maximum likelihood estimation approach is used to find the optimal parameter values. However, because the model includes latent variables $\mathbf{z}$ and θ, the data is considered incomplete, thus preventing the minimization from being able to be computed directly. As with most latent variable models, the Expectation-Maximization (EM) algorithm is used instead. The EM algorithm iteratively updates the parameters by computing expected values of the latent variables under the posterior distribution in equation (B.3).

Unfortunately, this posterior distribution is intractable due to the coupling of the θ and β parameters. Therefore, a strategy for approximate probabilistic inference must be used instead. At a high-level, the approximation consists of two steps. First, we choose a tractable family of distributions, whose statistics are easy to compute. We use this as a surrogate for the true posterior distribution:

$$q(\theta, \mathbf{z}|\gamma, \phi) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n) \quad \text{(B.5)}$$

Figure B.1(b) illustrates the corresponding graphical model for this approximating family of tractable distributions; it was obtained by dropping edges in the graph, and thus removing dependencies between variables that caused intractability. Next, we attempt to select the particular member of this family that best approximates the true posterior distribution. This is done by computing the optimal variational parameters, γ and ϕ, that minimizes the difference between the variational distribution and true posterior distribution. The following sections contain the details of variational inference and learning in this approximation.

B.3.1 Variational Inference

In our model, we substitute a surrogate variational distribution for the intractable posterior distribution $p(\theta, \mathbf{z}, x|\alpha, \beta)$, shown in eq. (B.3). This involves two steps, which we will describe in turn.

First, we must find a tractable *variational distribution*, a family of distributions parameterized by a set of *variational parameters*. This variational distribution acts as an approximation to the true distribution, but whose statistics are much easier to compute. One way to obtain a tractable family of distributions is to remove dependencies between variables that cause intractability in the true distribution. In the graphical illustration, this is equivalent to dropping some edges between nodes. Figure B.1(b) shows the new graphical illustration reflecting the simplified model. Dropping the document index m, the resulting variational distribution is as follows:

$$q(\theta, \mathbf{z}|\gamma, \phi) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n) \tag{B.6}$$

Here, the distribution $q(\theta|\gamma)$ is Dirichlet with variational parameter γ. This variational parameter exists at the document level, showing the distribution of topics in each document. The distributions $q(z_n|\phi_n)$ are multinomial with variational parameters ϕ_n. This variational parameter exists at the word level, where ϕ_{ni} is the probability that $n$the word is generated by the ith latent topic.

The second step is to select optimal parameters for the family of distributions such that the variational distributions best matches the true distribution.

More specifically, for each document $\mathbf{w}_m$, we wish to choose the variational parameters γ_m and ϕ_m such that $(\theta, \mathbf{z}|\gamma_m, \phi_m)$ is made to most closely resemble the true posterior $p(\theta, \mathbf{z}|\mathbf{w}_m, \alpha, \beta)$.

To measure the quality of resemblance, we compute a lower bound on the log-likelihood of each document $\log p(\mathbf{w}|\alpha, \beta)$ using Jensen's inequality:

$$\begin{aligned} & \log p(\mathbf{w}|\alpha, \beta) && \text{(B.7)} \\ &= \log \int \sum_{\mathbf{z} \in Z} \frac{p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) q(\theta, \mathbf{z})}{q(\theta|\mathbf{z})} \, d\theta \\ &\geq \int \sum_{\mathbf{z}} q(\theta, \mathbf{z}) \log p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) \, d\theta - \int \sum_{\mathbf{z}} q(\theta, \mathbf{z}|\gamma, \phi) \log q(\theta, \mathbf{z}) \, d\theta \\ &= L(\gamma, \phi; \alpha, \beta) && \text{(B.8)} \end{aligned}$$

It can be seen that the difference between the log likelihood $\log p(\mathbf{w}|\alpha, \beta)$ and the lower bound $\mathcal{L}(\gamma, \phi; \alpha, \beta)$ is actually the KL divergence (KL) between the variational posterior probability and the true posterior probability:

$$\mathrm{KL}(q, p) = \sum_{\mathbf{z}} \int q(\theta, \mathbf{z}|\gamma, \phi) \log \frac{q(\theta, \mathbf{z}|\gamma, \phi)}{p(\theta, \mathbf{z}|s, \alpha, \beta)} \, d\theta \qquad \text{(B.9)}$$

Thus, the KL divergence can be used to measure the quality of the variational approximation. More specifically, the best approximation is one that minimizes the KL divergence in equation (B.9) with respect to the variational parameters γ and ϕ_n for each document. To derive update rules for these parameters, we simply differentiate the KL divergence and set its partial derivatives equal to zero. The resulting update rules are as follows:

$$\phi_{ni} \propto \prod_{j=1}^{V} \beta_{ij}^{x_n^j} \exp\left(\Phi(\gamma_i) - \Phi(\textstyle\sum_{j=1}^{K} \gamma_j)\right) \qquad \text{(B.10)}$$

$$\gamma_i = \alpha_i + \sum_{n=1}^{N} \phi_{ni} \qquad \text{(B.11)}$$

We use $\Phi(\cdot)$ to denote the digamma function (the first derivative of the log Gamma function). Also, recall from section B.1 that x_n represents the nth word in the document as a V-dimensional unit-based vector. Thus, there will be exactly one value of j for which $x_n^j = 1$, and that is when j is equivalent to the vocabulary index of the nth word in the document.

B.3.2 Parameter Estimation

We wish to obtain estimates of the model parameters α and β that maximize the log likelihood of the entire corpus:

$$\mathcal{L}(\alpha, \beta) = \sum_{d=1}^{M} \log p(\mathbf{w}|\alpha, \beta) \tag{B.12}$$

However, since $p(\mathbf{w}|\alpha, \beta)$ cannot be computed tractably, we refer to a variational EM algorithm that uses the variational lower bound $L(\gamma, \phi; \alpha, \beta)$ in eq. (B.7) as a surrogate for the intractable log likelihood instead. The variational EM algorithm then estimates the parameters α and β to maximize this lower bound while fixing the variational parameters γ and ϕ to values obtained from the update rules in equations (B.11) and (B.10). The update rule for β can be written out analytically:

$$\beta_{ij} \propto \sum_{d=1}^{M} \sum_{n=1}^{N_d} \phi_{dni} w_{dn}^{j} \tag{B.13}$$

Maximizing the lower bound with respect to α is a little more complicated. Taking the derivative of $L(\gamma, \phi; \alpha, \beta)$ with respect to α_i gives:

$$\frac{\partial L}{\partial \alpha_i} = M \left(\Phi(\textstyle\sum_{j=1}^{K} \alpha_j) - \Phi(\alpha_i) \right) + \displaystyle\sum_{m=1}^{M} \left(\Phi(\gamma_{mi}) - \Phi(\textstyle\sum_{j=1}^{K} \gamma_{mj}) \right) \tag{B.14}$$

Unfortunately, this derivative, with respect to α_i, depends on α_j, for all $j \neq i$, and so we must resolve to an iterative method to find the maximum value for α. We thus invoke the linear-time Newton-Raphson algorithm in which the Hessian is inverted in linear time, to obtain α.

Given these update rules, the full variational EM algorithm is as follows:

1. (E-step) Fix the current model parameters α and β and compute the variational parameters γ_m and ϕ_m for each document $\mathbf{w}_m$ by maximizing the KL divergence in eq. (B.9) using the update rules in eq. (B.11) and (B.10).

2. (M-step) Fix the current variational parameters γ and ϕ across all songs from the E-step and compute the model parameters α and β by maximizing the lower bound in eq. (B.7) using the update rules in eq. (B.13) and (B.14).

These two steps are repeated until the lower bound on the log likelihood converges.

Appendix C

Variational Inference for LDA-AUDIO

This section contains derivation details for variational inference in the LDA-AUDIO model. The first section (C.1) provides the full expansion of the variational lower-bound described in eq. 4.23. The next three sections show how to maximize the lower-bound with respects to variational parameters γ and ω.

C.1 Variational Lower-Bound

Recall from eq. (4.22) that we can decompose the log-likelihood of each song $\mathbf{c}$ as the sum of some function $\mathcal{L}$ and the KL divergence between the variational posterior q, and true posterior p:

$$\ln p(\mathbf{c}|\alpha, \beta, A, \Sigma) \quad = \quad \mathcal{L}(\gamma, \phi, \omega) + KL(q(\theta, \mathbf{z}, \mathbf{x}||p(\theta, \mathbf{z}, \mathbf{x}, \mathbf{c})) \tag{C.1}$$

Since the KL divergence is non-negative, it should be clear that $\mathcal{L}$ is lower-bound on the true posterior $p(\mathbf{c}|\alpha, \beta, A, \Sigma)$. Eq. (4.20) also showed that the lower-bound was derived to be the difference between the expected values of the two families of

functions, which we can break down each further:

$$\mathcal{L}(\gamma, \phi, \omega) = \mathbb{E}_q[\ln p(\theta, \mathbf{z}, \mathbf{x}, \mathbf{c}|\alpha, \beta, A, \Sigma)] - \mathbb{E}_q[\ln q(\theta, \mathbf{z}, \mathbf{x})] \tag{C.2}$$

$$= (\mathbb{E}_q[\ln p(\theta|\alpha)] + \mathbb{E}_q[\ln p(\mathbf{z}|\theta)] + \mathbb{E}_q[\ln p(\mathbf{c}|\mathbf{x}, A, \Sigma)] + \mathbb{E}_q[\ln p(\mathbf{x}|\mathbf{z}, \beta)])$$

$$- (\mathbb{E}_q[\ln q(\theta|\gamma)] + \mathbb{E}_q[\ln q(\mathbf{z}|\phi)] + \mathbb{E}_q[\ln q(\mathbf{x}|\omega)]) \tag{C.3}$$

Figure C.1 shows the results of expanding each of the seven terms in eq. (C.3). The main differences between the LDA-AUDIO model and the original LDA model occur in terms (3), (4), and (7). Detailed derivations of these terms are shown below in sections C.1.1 and C.1.2.

$$E_q[\ln p(\theta|\alpha)] = \ln\Gamma(\sum_{i=1}^{K}\alpha_i) - \sum_{i=1}^{K}\ln\Gamma(\alpha_i) + \sum_{i=1}^{K}(\alpha_i - 1)(\Psi(\gamma_i) - \Psi(\sum_{i'=1}^{K}\gamma_{i'})) \tag{1}$$

$$E_q[\ln p(\mathbf{z}|\theta)] = \sum_{n=1}^{N}\sum_{i=1}^{K}\phi_{ni}(\Psi(\gamma_i) - \Psi(\sum_{i'=1}^{K}\gamma_{i'})) \tag{2}$$

$$E_q[\ln p(\mathbf{c}|\mathbf{x}, A, \Sigma)] = \ln\frac{1}{|\Sigma|^{N/2}} - \frac{1}{2}\sum_{n=1}^{N}\sum_{j=1}^{V}\sum_{k=1}^{V}c_{nk}^T\Sigma_{kj}^{-1}c_{jn} + \sum_{n=1}^{N}\sum_{j=1}^{V}\sum_{k=1}^{V}\sum_{l=1}^{V}\sum_{s=1}^{S}c_{nk}^T\Sigma_{kl}^{-1}A_{lj}\omega_{jn}s$$
$$-\frac{1}{2}\sum_{n=1}^{N}\left[\left(\sum_{j=1}^{V}\sum_{s=1}^{S}(\omega_{jns} - \omega_{jns}^2)s^2M_{jj}\right) + \left(\sum_{j=1}^{V}\sum_{k=1}^{V}\sum_{s=1}^{S}\omega_{jns}\omega_{kns}s^2M_{kj}\right)\right] \tag{3}$$

$$E_q[\ln p(\mathbf{x}|\mathbf{z}, \beta)] = \sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{V}\sum_{s=0}^{S}\phi_{ni}\ln\beta_{ij}\omega_{jns}s \tag{4}$$

$$E_z[\ln q(\theta|\gamma)] = \ln\Gamma(\sum_{i=1}^{K}\gamma_i) - \sum_{i=1}^{K}\ln\Gamma(\gamma_i) + \sum_{i=1}^{K}(\gamma_i - 1)(\Psi(\gamma_i) - \Psi(\sum_{i'=1}^{K}\gamma_{i'})) \tag{5}$$

$$E_q[\ln q(\mathbf{z}|\phi)] = \sum_{n=1}^{N}\sum_{i=1}^{K}\phi_{ni}\ln\phi_{ni} \tag{6}$$

$$E_z[\ln q(\mathbf{x}|\omega)] = \sum_{n=1}^{N}\sum_{j=1}^{V}\sum_{s=1}^{S}\omega_{jns}\ln\omega_{jns} \tag{7}$$

Figure C.1: Expanding all seven terms from the lower-bound derived in eq. (C.3). Terms (3), (4), and (7) are significantly different from the original LDA model, and are discussed in more detail in sections C.1.1 and C.1.2.

C.1.1 Computing the expectation over chroma vector c

In this section, we show how to derive the third term $E_q[p(\mathbf{c}|\mathbf{x}, A, \Sigma)]$ in the lower-bound (shown in Figure C.1). The derivation begins with:

$$E_q[\ln p(\mathbf{c}|\mathbf{x}, A)] \tag{C.4}$$

$$= E_q[\ln \prod_{n=1}^{N} p(c_n|x_n, A)] \tag{C.5}$$

$$= \ln \frac{1}{|\Sigma|^{N/2}} + E_q\left[\sum_{n=1}^{N} -\frac{1}{2}(c_n - Ax_n)^T\Sigma^{-1}(c_n - Ax_n)\right]$$

$$= \ln \frac{1}{|\Sigma|^{N/2}} - \frac{1}{2}\sum_{n=1}^{N} c_n^T\Sigma^{-1}c_n + \sum_{n=1}^{N} c_n^T\Sigma^{-1}AE_q[x_n] - \frac{1}{2}\sum_{n=1}^{N} E_q[x_n^TA^T\Sigma^{-1}Ax_n] \tag{C.6}$$

The last two terms in eq. C.6 compute different expectations over x. The first expectation term is relatively simple to resolve. In matrix form:

$$\begin{aligned}\sum_{n=1}^{N} c_n^T\Sigma^{-1}AE_q[x_n] &= \sum_{n=1}^{N} c_n^T\Sigma^{-1}A\sum_{s=1}^{S}\omega_n s \\ &= \sum_{n=1}^{N}\sum_{s=1}^{S} c_n^T\Sigma^{-1}A\omega_n s\end{aligned} \tag{C.7}$$

where ω_n is a $(V \times N)$ matrix and $\omega_n s$ is scalar multiplication between matrix ω_n and scalar s. To be complete, we also give the equivalent formulation in scalar form:

$$\sum_{n=1}^{N} c_n^T\Sigma^{-1}AE_q[x_n] = \sum_{n=1}^{N}\sum_{j=1}^{V}\sum_{k=1}^{V}\sum_{l=1}^{V}\sum_{s=1}^{S} c_{nk}^T\Sigma_{kl}^{-1}A_{lj}\omega_{jns}s \tag{C.8}$$

The second expectation term from eq. C.6 is a little less straightforward. We begin the derivation below:

$$\begin{aligned}-\frac{1}{2}\sum_{n=1}^{N} E_q[x_n^TA^T\Sigma^{-1}Ax_n] &= -\frac{1}{2}\sum_{n=1}^{N} E_q\ \mathrm{tr}[x_n^TA^T\Sigma^{-1}Ax_n] \\ &= -\frac{1}{2}\sum_{n=1}^{N} E_q\ \mathrm{tr}[x_nx_n^TA^T\Sigma^{-1}A] \\ &= -\frac{1}{2}\mathrm{tr}\left[\sum_{n=1}^{N} E_q[x_nx_n^T]A^T\Sigma^{-1}A\right]\end{aligned} \tag{C.9}$$

Given the distribution for $q(x_n|\omega_n)$, let us first compute the term $E_q[x_n x_n^T]$ found in the previous equation.

$$E_q[x_{ni}x_{nj}] = \begin{cases} E_q[x_{jn}^2] & = & \sum_{s=0}^{S} s^2 \omega_{jns} & \text{if } i = j \\ E_q[x_{in}]E_q[x_{jn}] & = & \sum_{s=0}^{S} s^2 \omega_{jns}\omega_{ins} & \text{otherwise} \end{cases}$$

In matrix form:

$$E_q[x_n {x_n}^T] = \begin{bmatrix} \sum_s s^2\omega_{1ns} & \sum_s s^2\omega_{1ns}\omega_{2ns} & \dots & \sum_s s^2\omega_{1ns}\omega_{Vns} \\ \sum_s s^2\omega_{2ns}\omega_{1ns} & \sum_s s^2\omega_{2ns} & \dots & \sum_s s^2\omega_{2ns}\omega_{Vns} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_s s^2\omega_{Vns}\omega_{1ns} & \sum_s s^2\omega_{Vns}\omega_{2ns} & \dots & \sum_s s^2\omega_{Vns} \end{bmatrix}$$

In matlab, this matrix can be constructed by doing something like this. Let:

$$U = \begin{bmatrix} 1 & 2 & \cdots & S \\ 1 & 2 & \cdots & S \\ \vdots & & & \\ 1 & 2 & \cdots & S \end{bmatrix} \quad V = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ S \end{bmatrix}^2 \quad W_n = U. * \omega_n$$

where V is a $(S \times 1)$ matrix and U is a $(V \times S)$ matrix. The off-diagonal elements can be taken from $W_n W_n^T$. The diagonal elements can be taken from $\omega_n * V$. In matrix form, this can be written as an $(V \times V)$ matrix:

$$E_q[x_n x_n^T] = \text{diag}[\omega_n V - \omega_n^2 V] + W_n W_n^T \quad \text{(C.10)}$$

$$= \text{diag}[(\omega_n - \omega_n^2)V] + W_n W_n^T \quad \text{(C.11)}$$

Continuing the derivation of eq. C.10 above, we get the following:

$$-\frac{1}{2}\sum_{n=1}^{N} E_q[x_n^T A^T \Sigma^{-1} A x_n] = -\frac{1}{2}\text{tr}\left[\sum_{n=1}^{N} E_q[x_n x_n^T] A^T \Sigma^{-1} A\right] \quad \text{(C.12)}$$

$$= -\frac{1}{2}\text{tr}\left[\sum_{n=1}^{N} (\text{diag}[(\omega_n - \omega_n^2)V] + W_n W_n^T) A^T \Sigma^{-1} A\right]$$

It will be useful later on to express eq. C.13 in scalar form, so we continue expanding. We note that since the term inside the expected value is a scalar in eq. C.10,

it is trivially equivalent to taking the trace of it as we do beginning in eq. C.13. Finally, in the following derivations, let $M = A^T \Sigma^{-1} A$.

$$
\begin{aligned}
&= -\frac{1}{2}\mathrm{tr}\left[\sum_{n=1}^{N}(\mathrm{diag}[(\omega_n - \omega_n^2)V] + W_n W_n^T)M\right] \\
&= -\frac{1}{2}\mathrm{tr}\left[\sum_{n=1}^{N}(\mathrm{diag}[(\omega_n - \omega_n^2)V])M + W_n W_n^T M\right] \\
&= -\frac{1}{2}\sum_{n=1}^{N}\mathrm{tr}[\mathrm{diag}[(\omega_n - \omega_n^2)V])M] + \mathrm{tr}[W_n W_n^T M] \\
&= -\frac{1}{2}\sum_{n=1}^{N}\left[\left(\sum_{j=1}^{V}\sum_{s=1}^{S}(\omega_{jns} - \omega_{jns}^2)s^2 M_{jj}\right) + \left(\sum_{j=1}^{V}\sum_{k=1}^{V}\sum_{s=1}^{S}\omega_{jns}\omega_{kns}s^2 M_{kj}\right)\right]
\end{aligned}
$$

Putting all of this together, we can return to our original equation from eq. C.6 at the beginning of the section and filling in the terms that we have just computed:

$$E_q[\ln p(\mathbf{c}|\mathbf{x}, A)] \tag{C.13}$$

$$= E_q[\ln \prod_{n=1}^{N} p(c_n|x_n, A)] \tag{C.14}$$

$$
\begin{aligned}
&= \ln \frac{1}{|\Sigma|^{N/2}} + E_q\left[\sum_{n=1}^{N} -\frac{1}{2}(c_n - Ax_n)^T\Sigma^{-1}(c_n - Ax_n)\right] \\
&= \ln \frac{1}{|\Sigma|^{N/2}} - \frac{1}{2}\sum_{n=1}^{N} c_n^T\Sigma^{-1}c_n + \sum_{n=1}^{N} c_n^T\Sigma^{-1}AE_q[x_n] - \frac{1}{2}\sum_{n=1}^{N} E_q[x_n^T A^T\Sigma^{-1}Ax_n] \\
&= \ln \frac{1}{|\Sigma|^{N/2}} - \frac{1}{2}\sum_{n=1}^{N} c_n^T\Sigma^{-1}c_n + \sum_{n=1}^{N}\sum_{s=1}^{S} c_n^T\Sigma^{-1}A\omega_n s \\
&\quad -\frac{1}{2}\mathrm{tr}\left[\sum_{n=1}^{N}(\mathrm{diag}[(\omega_n - \omega_n^2)V] + W_n W_n^T)A^T\Sigma^{-1}A\right]
\end{aligned} \tag{C.15}
$$

$$
\begin{aligned}
&= \ln \frac{1}{|\Sigma|^{N/2}} - \frac{1}{2}\sum_{n=1}^{N}\sum_{j=1}^{V}\sum_{k=1}^{V} c_{nk}^T\Sigma_{kj}^{-1}c_{jn} + \sum_{n=1}^{N}\sum_{j=1}^{V}\sum_{k=1}^{V}\sum_{l=1}^{V}\sum_{s=1}^{S} c_{nk}^T\Sigma_{kl}^{-1}A_{lj}\omega_{jn}s \\
&\quad -\frac{1}{2}\sum_{n=1}^{N}\left[\left(\sum_{j=1}^{V}\sum_{s=1}^{S}(\omega_{jns} - \omega_{jns}^2)s^2 M_{jj}\right) + \left(\sum_{j=1}^{V}\sum_{k=1}^{V}\sum_{s=1}^{S}\omega_{jns}\omega_{kns}s^2 M_{kj}\right)\right]
\end{aligned} \tag{C.16}
$$

C.1.2 Computing the expectation over note x

In this section, we compute two of the terms in the variational lower-bound shown in eq. C.3. The expansion of the fourth term (from Figure C.1) is shown below:

$$
\begin{aligned}
E_q[\ln p(\mathbf{x}|\mathbf{z},\beta)] &= E_q\left[\ln \prod_{n=1}^{N} p(x_n|z_n,\beta)\right] && \text{(C.17)}\\
&= E_q\left[\sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{V} \ln \beta_{ij}^{(z_{in}x_{jn})}\right] && \text{(C.18)}\\
&= \sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{V} E_q[z_{in}x_{jn}]\ln\beta_{ij} && \text{(C.19)}\\
&= \sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{V} E_q[z_{in}]E_q[x_{jn}]\ln\beta_{ij} && \text{(C.20)}\\
&= \sum_{n=1}^{N}\sum_{i=1}^{K}\sum_{j=1}^{V}\sum_{s=0}^{S} \phi_{ni}\ln\beta_{ij}\omega_{jns}s && \text{(C.21)}
\end{aligned}
$$

Note that x and z are independent of each other in eq. C.21 so we can break up that expectation term. Below, we also expand the seventh term in the lower-bound (from Figure C.1):

$$
\begin{aligned}
E_q[\ln q(\mathbf{x}|\omega)] &= E_q\left[\ln\prod_{n=1}^{N} q(x_n|\omega_n)\right] = E_q\left[\ln\prod_{n=1}^{N}\prod_{j=1}^{V}\prod_{s=0}^{S}{\omega_{jns}}^{I(x_{jn},s)}\right] && \text{(C.22)}\\
&= \sum_{n=1}^{N}\sum_{j=1}^{V}\sum_{s=0}^{S} E_q[I(x_{jn},s)]\ln\omega_{jns} = \sum_{n=1}^{N}\sum_{j=1}^{V}\sum_{s=0}^{S}\omega_{jns}\ln\omega_{jns}
\end{aligned}
$$

C.2 Inferring Variational Parameters

Recall that our goal from the previous section was to produce the tightest possible lower-bound on the log posterior. Thus, we maximize the lower-bounded obtained in eq. (C.2) with respect to each of the variational parameters γ, ϕ, and ω, where γ parameterizes a Dirichlet distribution, and both ϕ and ω parameterize multinomials. The two sections below show how to do this maximization for variational parameters ϕ and ω. The maximization for γ is very similar to the LDA-MIDI model, and is not shown here.

C.2.1 Inferring ϕ

We maximize the lower-bound $\mathcal{L}(\gamma, \phi, \omega; \alpha, \beta, A, \Sigma)$ with respect to ϕ_{ni}, the probability that the notes in the nth segment of music is generated by musical key i. Since the distribution over musical keys for each segment should be multinomial, this maximization constrains $\sum_{i=1}^{K} \phi_{ni} = 1$ for each n.

First, we write out the lower-bound with respect to relevant ϕ_{ni} terms as follows:

$$\begin{aligned} L_{[\phi_{ni}]} &= \phi_{ni} \left(\Psi(\gamma_i) - \Psi\left(\textstyle\sum_{i'=1}^{K} \gamma_{i'} \right) \right) \\ &\quad + \phi_{ni} \sum_{j=1}^{V} \sum_{s=0}^{S} \omega_{jns} s \ln \beta_{ij} - \phi_{ni} \ln \phi_{ni} + \lambda_n \left(\sum_{j=1}^{V} \phi_{ni} - 1 \right) \end{aligned} \tag{C.23}$$

Taking the derivative of $L_{[\phi_{ni}]}$ gives the following:

$$\begin{aligned} \frac{\partial L_{[\phi_{ni}]}}{\partial \phi_{ni}} &= \left(\Psi(\gamma_i) - \Psi\left(\textstyle\sum_{i'=1}^{K} \gamma_{i'} \right) \right) \\ &\quad + \sum_{j=1}^{V} \sum_{s=0}^{S} \omega_{jns} s \ln \beta_{ij} - \ln \phi_{ni} - 1 + \lambda_n \end{aligned} \tag{C.24}$$

Setting eq. C.24 equal to 0 and solving for ϕ_{ni}:

$$\phi_{ni} \propto \exp\left\{ \Psi(\gamma_i) - \Psi\left(\textstyle\sum_{i'=1}^{K} \gamma_{i'} \right) \right\} \exp\left\{ \textstyle\sum_{j=1}^{V} \sum_{s=0}^{S} \omega_{jns} s \ln \beta_{ij} \right\} \tag{C.25}$$

$$= \exp\left\{ \Psi(\gamma_i) - \Psi\left(\textstyle\sum_{i'=1}^{K} \gamma_{i'} \right) \right\} \prod_{j=1}^{V} \prod_{s=0}^{S} {\beta_{ij}}^{\omega_{jns} s} \tag{C.26}$$

C.2.2 Inferring ω

Next, we maximize the lower-bound $\mathcal{L}(\gamma, \phi, \omega; \alpha, \beta, A, \Sigma)$ with respect to ω_{njs}, the probability that pitch j in segment n has a note count of s. For convenience, let $M = A^T \Sigma^{-1} A$. Writing the lower-bound with respect to each ω_{njs} term, we get:

$$
\begin{aligned}
L_{[\omega_{njs}]} = & \sum_{k=1}^{V}\sum_{l=1}^{V} c_{nk}^{T}\Sigma_{kl}^{-1}A_{lj}\omega_{jns}s \\
& - \frac{1}{2}\left[(\omega_{njs}-\omega_{njs}^{2})s^{2}M_{jj} + \omega_{jns}^{2}s^{2}M_{jj} + 2\ \omega_{jns}s^{2}\sum_{k\neq j}^{V}\omega_{kns}M_{kj}\right] \\
& + \sum_{i=1}^{K}\phi_{ni}\ln\beta_{ij}\omega_{njs}s - \omega_{njs}\ln\omega_{njs} + \sum_{n'j'}^{N,V}\lambda_{n'j'}\left(\sum_{s'}^{S}\omega_{n'j's'}-1\right) \\
= & \sum_{k=1}^{V}\sum_{l=1}^{V} c_{nk}^{T}\Sigma_{kl}^{-1}A_{lj}\omega_{jns}s \\
& - \frac{1}{2}\left[\omega_{jns}s^{2}M_{jj} - \omega_{jns}^{2}s^{2}M_{jj} + \omega_{jns}^{2}s^{2}M_{jj} + 2\ \omega_{jns}s^{2}\sum_{k\neq j}^{V}\omega_{kns}M_{kj}\right] \\
& + \sum_{i=1}^{K}\phi_{ni}\ln\beta_{ij}\omega_{njs}s - \omega_{njs}\ln\omega_{njs} + \sum_{n'j'}^{N,V}\lambda_{n'j'}\left(\sum_{s'}^{S}\omega_{n'j's'}-1\right) \\
= & \sum_{k=1}^{V}\sum_{l=1}^{V} c_{nk}^{T}\Sigma_{kl}^{-1}A_{lj}\omega_{jns}s - \frac{1}{2}\left[\omega_{jns}s^{2}M_{jj} + 2\ \omega_{jns}s^{2}\sum_{k\neq j}^{V}\omega_{kns}M_{kj}\right] \\
& + \sum_{i=1}^{K}\phi_{ni}\ln\beta_{ij}\omega_{njs}s - \omega_{njs}\ln\omega_{njs} + \sum_{n'j'}^{N,V}\lambda_{n'j'}\left(\sum_{s'}^{S}\omega_{n'j's'}-1\right)
\end{aligned}
\tag{C.27}
$$

where

$$
\lambda_{nj} = \ln\sum_{s'}\exp\{C_{njs'}\} \tag{C.28}
$$

and C_{njs} is the expression that includes all terms that do not depend on ω_{njs}. It is used as a normalization factor while constraining the distribution of note counts in ω to be multinomial (i.e. adding up to 1). Now, taking the derivative of $L_{[\omega_{jns}]}$:

$$
\begin{aligned}
\frac{\partial L_{[\omega_{jns}]}}{\partial\omega_{jns}} \quad = \quad & s\sum_{k=1}^{V}\sum_{l=1}^{V} c_{nk}^{T}\Sigma_{kl}^{-1}A_{lj} - \frac{1}{2}s^{2}M_{jj} - s^{2}\sum_{k\neq j}\omega_{kns}M_{kj} \\
& + s\sum_{i=1}^{K}\phi_{ni}\ln\beta_{ij} - \ln\omega_{jns} - \lambda_{nj}
\end{aligned}
\tag{C.29}
$$

Finally, setting eq. (C.29) equal to 0 and solving for ω_{jns}, we get:

$$\omega_{jns} = \frac{\exp[f(s)]}{\sum_{s'}^{S} \exp[f(s')]} \tag{C.30}$$

where

$$f(s) = s \sum_{k=1}^{V} \sum_{l=1}^{V} c_{nk}^{T} \Sigma_{kl}^{-1} A_{lj} - \frac{1}{2} s^2 M_{jj} - s^2 \sum_{k \neq j} \omega_{kns} M_{kj} + s \sum_{i=1}^{K} \phi_{ni} \ln \beta_{ij} \tag{C.31}$$

C.3 Estimating Model Parameters

In this section, we show how to obtain estimates of the model parameters α, β, and A, assuming that only the raw audio data is observed. We note that in our experimentation, it is difficult for this method to obtain estimates that are as good as that obtained from the LDA-MIDI model, since the pitch information we get there is much cleaner. Nonetheless, we show this derivation as a means for comparison later on. Recall that all three model parameters exist at the corpus level. Thus, in the remainder of this section, we use $\mathcal{L}$ to represent the *overall* variational lower bound (i.e. the sum of the individual variational bounds for each song in the corpus).

C.3.1 Estimating β

We maximize the lower bound $L(\gamma, \phi, \omega; \alpha, \beta, A)$ with respect to matrix β_{ij}:

$$L_{[\beta_{ij}]} = \sum_{d=1}^{M} \sum_{n=1}^{N_d} \sum_{s=0}^{S} \phi_{ni}^{(d)} \omega_{jns}^{(d)} s \ln \beta_{ij} + \sum_{i'=1}^{K} \lambda_{i'} \left(\sum_{j'=1}^{V} \beta_{i'j'} - 1 \right) \tag{C.32}$$

Take the derivative of $L_{[\beta_{ij}]}$:

$$\frac{\partial L_{[\beta_{ij}]}}{\partial \beta_{ij}} = \frac{1}{\beta_{ij}} \left(\sum_{d=1}^{M} \sum_{n=1}^{N_d} \sum_{s=0}^{S} \phi_{ni}^{(d)} \omega_{jns}^{(d)} s \right) + \sum_{i'=1}^{K} \lambda_{i'} \tag{C.33}$$

Set eq. (C.33) to 0 and solve for β_{ij}:

$$\frac{1}{\beta_{ij}}\left(\sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{s=0}^{S}\phi_{ni}^{(d)}\omega_{jns}^{(d)}s\right) = -\sum_{i=1}^{K}\lambda_i \tag{C.34}$$

$$\beta_{ij} \propto \sum_{d=1}^{M}\sum_{n=1}^{N_d}\sum_{s=0}^{S}\phi_{ni}^{(d)}\omega_{jns}^{(d)}s \tag{C.35}$$

To see this in matrix form, let:

$$V = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ S \end{bmatrix} \quad \text{and} \quad Q^{(d)} = \begin{bmatrix} | & | & & | \\ \omega_1^{(d)}V & \omega_2^{(d)}V & \cdots & \omega_n^{(d)}V \\ | & | & & | \end{bmatrix}$$

Then β is a $(K \times V)$ matrix:

$$\beta = \sum_{d}^{M} Q^{(d)}\phi^{(d)} \tag{C.36}$$

We note that each ω_i is a $(V \times S)$ matrix and each $\omega_i^{(d)}V$ is a $(V \times 1)$ matrix. Similarly, each $Q^{(d)}$ is a $(V \times N)$ matrix and each $\phi^{(d)}$ is a $(N \times K)$ matrix. Finally, last term in eq. (C.32) is a Lagrange multiplier that restricts all rows of β to sum to 1.

C.3.2 Estimating A

A is a transform that describes how our audio chroma features correlate to our MIDI-features. More specifically, recall (from the generative process in section 4.5.2) that a Gaussian distribution models the process of drawing observed chroma vector c_n from MIDI notes in the corresponding segment of music x_n:

$$p(c_n|x_n, A) = \frac{1}{|\Sigma|^{1/2}}\exp\left\{-\frac{1}{2}(c_n - Ax_n)^T\Sigma^{-1}(c_n - Ax_n)\right\} \tag{C.37}$$

In our experimentation, we assume that MIDI notes (x_n) are observed during a training phase (but unobserved during testing), and thus estimates A from the "ground-truth" MIDI notes. While it is possible to derive the MLE for A without ground-truth MIDI notes, doing so makes it much more difficult to learn an

accurate transform, and requires a lot more data. To do so, we first expand the terms in the log posterior distribution:

$$\begin{aligned} L &= \ln \prod_{n=1}^{N} p(c_n|x_n, A, \Sigma^{-1}) \\ &= -\frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum_{n=1}^{N}(c_n - Ax_n)^T\Sigma^{-1}(c_n - Ax_n) \\ &= -\frac{N}{2}\ln|\Sigma| - \frac{1}{2}\mathrm{tr}[c^T\Sigma^{-1}c] + \mathrm{tr}[c^T\Sigma^{-1}Ax] - \frac{1}{2}\mathrm{tr}[x^TA^T\Sigma^{-1}Ax] \quad \text{(C.38)} \end{aligned}$$

Next, we maximize L with respect to A:

$$\text{Isolate terms:} \qquad L_{[A]} = \mathrm{tr}[Axc^T\Sigma^{-1}] - \frac{1}{2}\mathrm{tr}[A^T\Sigma^{-1}Axx^T] \quad \text{(C.39)}$$

$$\text{Take derivative:} \qquad \frac{\partial L_{[A]}}{\partial A} = \Sigma^{-1}cx^T - \Sigma^{-1}Axx^T \quad \text{(C.40)}$$

$$\text{Set equal to 0 and solve for A:} \qquad A = cx^T(xx^T)^{-1} \quad \text{(C.41)}$$

We can also maximize L with respect to Σ. Let $Z = \Sigma^{-1}$:

Isolate terms:

$$L_{[Z]} = -\frac{N}{2}\ln|Z^{-1}| - \frac{1}{2}\mathrm{tr}[Zcc^T] + \mathrm{tr}[ZAxc^T] - \frac{1}{2}\mathrm{tr}[(Ax)^TZAx] \quad \text{(C.42)}$$

Take derivative:

$$\frac{\partial L_{[Z]}}{\partial Z} = -Z^{-1} - \frac{1}{2}cc^T + cx^TA^T - \frac{1}{2}Axx^TA^T \quad \text{(C.43)}$$

Set equal to 0 and solve for Z:

$$Z^{-1} = \Sigma = -\frac{1}{2}cc^T + cx^TA^T - \frac{1}{2}Axx^TA^T \quad \text{(C.44)}$$

Bibliography

[1] B. Aarden. Dynamic melodic expectancy. *Ph.D. dissertation. School of Music, Ohio State University*, 2003.

[2] X. Amatriain, j. Massaguer, D. Garcia, and I. Mosquera. The clam annotator: a cross-platform audio descriptors editing tool. *International Conference on Music Information Retrieval (ISMIR)*, 2005.

[3] J. P. Bello. Audio-based cover song retrieval using approximate chord sequences: Testing shifts, gaps, swaps and beats. *International Conference on Music Information Retrieval (ISMIR)*, 2007.

[4] J. P. Bellow and J. Pickens. A robust mid-level representation for harmonic content in music signals. *International Conference on Music Information Retrieval (ISMIR)*, 2005.

[5] D. Blei and J. .D. Lafferty. Correlated topic models. *Neural Information Processing Systems (NIPS)*, 2005.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[7] J. C. Brown. Calculation of a constant-q spectral transform. *Journal of the Acoustic Society of America*, 89/1:425–434, 1991.

[8] J. C. Brown and M. S. Puckette. An efficient algorithm for the calculation of a constant-q transform. *Journal of the Acoustic Society of America*, 92:2698–2701, 1992.

[9] C. Chafe and D. Jaffe. Source separation and note identification in polyphonic music. *STAN-M-34, CCRMA, Stanford University*, 1986.

[10] W. Chai and B. Vercoe. Detection of key change in classical piano music. *International Conference on Music Information Retrieval (ISMIR)*, 2005.

[11] E. Chew. Modeling tonality: Applications to music cognition. *Proceedings of the Annual Conference of the Cognitive Science Society*, 2001.

[12] N. Chomsky. *Formal properties of grammars*. Wiley, New York, 1963.

[13] C. Chuan and E. Chew. Polyphonic audio key finding using the spiral array ceg algorithm. *In IEEE International Conference on Multimedia and Expo*, 2005.

[14] R. Cohn. Neo-riemannian operations, parsimonious trichords, and their tonnetz representations. *Journal of Music Theory*, 41/1:1–66, 1997.

[15] R. Cohn. An introduction to neo-riemannian theory: A survey and historical perspective. *Journal of Music Theory*, 42/2:167–180, 1998.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39/1:1–38, 1977.

[17] A. M. Dominguez. Tonal stability modeling from audio chroma features. *Master thesis, Universitat Pompeu Fabra, Barcelona*, 2009.

[18] S. Downie. Music information retrieval. *Annual Review of Information Science and Technology*, 37:295–340, 2003.

[19] K. Ebcioglu. An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12/3:43–51, 1988.

[20] T. Eerola and P. Toiviainen. Midi toolbox: Matlab tools for music research. *University of Jyvaskyla, Jyvaskyla, Finland*, 2004. http://www.jyu.fi/musica/miditoolbox/.

[21] D. Ellis. Beat tracking by dynamic programming. *Journal of New Music Research, Special Issue on Beat and Tempo Extraction*, 36/1:51–60, 2007. Code available at: http://labrosa.ee.columbia.edu/projects/beattrack/.

[22] M. M. Farbood. A global model of musical tension. *International Conference of Music Perception and Cognition*, pages 690–695, 2008.

[23] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. *The Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 524–531, 2005.

[24] T. Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. *International Computer Music Conference (ICMC)*, pages 464–467, 1999.

[25] E. Gomez. Tonal description of audio music signals. *Ph.D. thesis, Universitat Pompeu Fabra, Barcelona*, 2006.

[26] E. Gomez and P. Herrera. Estimating the tonality of polyphonic audio files: cognitive versus machine learning modelling strategies. *International Conference on Music Information Retrieval (ISMIR)*, 2004.

[27] C. A. Harte and M. B. Sandler. Automatic chord identiiňĄcation using a quantised chromagram. *Proceedings of Audio Engineering Society (AES)*, 2005.

[28] B. H'ector. About the determination of key of a musical excerpt. *Proceedings of Computer Music Modeling and Retrieval (CMMR)*, pages 187–203, 2005.

[29] P. Herrera. Automatic classification of percussion sounds: from acoustic features to semantic descriptions. *Doctoral dissertatin, Universitat Pompeu Fabra, Barcelona*, 2006.

[30] M. Hoffman, D. Blei, and P. Cook. Easy as cba: A simple probabilistic model for tagging music. *International Conference on Music Information Retrieval (ISMIR)*, 2009.

[31] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 2001.

[32] O. Izmirli. Localized key-finding from audio using non-negative matrix factorization for segmentation. *International Conference on Music Information Retrieval (ISMIR)*, 2007.

[33] M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. Introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

[34] P. N. Juslin. *Music and emotion. Series in affective science.* Oxford University Press, New York, 2001.

[35] K. Kashino, K. Nakadai, T. Kinoshita, and H. Tanaka. Application of bayesian probability network to music scene analysis. *Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI)*, 1995.

[36] A. P. Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. *International Conference on Music Information Retrieval (ISMIR)*, pages 216–221, 2006.

[37] T. Kohonen. *Self-organization and associative memory.* Springer-Verlag, Berlin, 1984.

[38] S. Kostka and D. Payne. *Tonal Harmony.* McGraw Hill, New York, 1995.

[39] S. M. Kostka and D. Payne. *Workbook for tonal harmony, with an introduction to twentieth-century music.* McGraw-Hill, 2000.

[40] C. Krumhansl. A perceptual analysis of mozart's piano sonata k. 282: Segmentation, tension, and musical ideas. *Music Perception*, 13/3:401–432, 1996.

[41] C. L. Krumhansl. *Cognitive foundations of musical pitch.* Oxford University Press, New York, 1990.

[42] C. L. Krumhansl and E. J. Kessler. Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. *Psychological Review*, 89:334–368, 1982.

[43] C. L. Krumhansl and R. N. Shepard. Quantification of the hierarchy of tonal functions wirhin a diatonic context. *Journal of Experimental Psychology: Human Perception and Performance*, 5:579–594, 1979.

[44] F. Kuo, M. Chiang, M. Shan, and S. Lee. Emotion-based music recommendation by association discovery from film music. *ACM International Conference on Multimedia*, pages 507–510, 2005.

[45] C. L. Lawson and R. J. Hanson. *Solving Least Squares.* Prentice-Hall, 1974.

[46] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401/6755:788–791, 1999.

[47] K. Lee. Identifying cover songs from audio using harmonic representation. *Music Information Retrieval Evaluation eXchange (MIREX) task on Audio Cover Song Identification*, 2006.

[48] K. Lee and M. Slaney. Acoustic chord transcription and key extraction from audio using key-dependent hmms trained on synthesized audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 16/2:291–301, 2008.

[49] M. Leman. *Music and schema theory: cognitive foundations of systematic musicology.* Springer-Verlag, Berlin-Heidelberg, 1995.

[50] F. Lerdahl and R. Jackendoff. *A generative theory of tonal music.* MIT Press, Cambridge, Massachusetts, 1983.

[51] F. Lerdahl and C. L. Krumhansl. Modeling tonal tension. *Music Perception*, 24/4:329–366, 2007.

[52] D. Lewin. *Generalized musical intervals and transformations.* Yale University Press, New Haven, 1987.

[53] B. Logan. Mel frequency cepstral coefficients for music modeling. *International Conference on Music Information Retrieval (ISMIR)*, 2000.

[54] H. C. Longuet-Higgins and M. J. Steedman. On interpreting bach. *Machine Intelligence*, 6:221–241, 1971.

[55] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of Fifth Berkeley Symposium on Mathematics, Statistics, and Probability*, 1, 1966.

[56] S. Madsen and G. Widmer. Key-finding with interval proiles. *International Computer Music Conference (ICMC)*, 2007.

[57] M. Mauch and S. Dixon. Approximate note transcription for the improved identification of difficult chords. *International Conference on Music Information Retrieval (ISMIR)*, 2010.

[58] M. Mauch and S. Dixon. Simultaneous estimation of chords and musical context from audio. *IEEE transactions on audio, speech, and language processing*, 2010.

[59] M. Mauch, K. C. Noland, and S. Dixon. Using musical structure to enhance automatic chord transcription. *International Conference on Music Information Retrieval (ISMIR)*, pages 231–236, 2009.

[60] Y. Ni, M. Mcvicar, R. Santos-Rodriguez, and T. De Bie. An end-to-end machine learning system for harmonic analysis of music. *IEEE Transactions on Audio, Speech and Language Processing*, 20/6:1771–1783, 2012.

[61] K. Noland and M. Sandler. Influences of signal processing, tone profiles, and chord progressions on a model for estimating the musical key from audio. *Computer Music Journal*, 33/1, 2009.

[62] J. Paiement, D. Eck, and S. Bengio. A probabilistic model for chord progressions. *International Conference on Music Information Retrieval (ISMIR)*, 2005.

[63] S. Pauws. Musical key extraction from audio. *International Conference on Music Information Retrieval (ISMIR)*, 2004.

[64] G. Peeters. Musical key estimation of audio signal based on hidden markov modeling of chroma vectors. *International Conference on Music Information Retrieval (ISMIR)*, 2006.

[65] J. Pickens, J. P. Bello, G. Monti, M. Sandler, T. Crawford, M. Dovey, and D. Byrd. Polyphonic score retrieval using polyphonic audio queries: a harmonic modeling approach. *Journal of New Music Research*, 32:223–236, 2003.

[66] H. Purwins, B. Blankertz, and K. Obermayer. A new method for tracking modulations in tonal music in audio data format. *Neural Networks - IJCNN, IEEE Computer Society*, 6:270–275, 2000.

[67] C. Raphael and J. Stoddard. Harmonic analysis with probabilistic graphical models. *International Conference on Music Information Retrieval (ISMIR)*, 2003.

[68] T. Rocher, M. Robine, P. Hanna, and L. Oudre. Concurrent estimation of chords and keys from audio. *International Conference on Music Information Retrieval (ISMIR)*, 2010.

[69] A. Sheh and D. Ellis. Chord segmentation and recognition using em-trained hidden markov models. *International Conference on Music Information Retrieval (ISMIR)*, 2003.

[70] R. N. Shepard. Structural representations of musical pitch. *The Psychology of Music*, 1982.

[71] I. Simon, D. Morris, and S. Basu. Mysong: Automatic accompaniment generation for vocal melodies. *ACM Conference on Human Factors in Computing Systems (CHI)*, 2008.

[72] A. Smeulders, W. Worring, M. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[73] C. S. Stuart. Key-profile comparisons in key-finding by correlation. *International Computer Music Conference (ICMC)*, 2008.

[74] D. Temperley. What's key for key? the krumhansl-schmuckler key-finding algorithm reconsidered. *Music Perception*, 17/1:65–100, 1999.

[75] D. Temperley. *The Cognition of Basic Musical Structures*. MIT Press, 2001. Code available at: http://www.link.cs.cmu.edu/music-analysis/cbms.html.

[76] D. Temperley. A bayesian key-finding algorithm. *Music and Artificial Intelligence*, 2002. Code available at: http://www.link.cs.cmu.edu/music-analysis/key.html.

[77] D. Temperley. *Music and Probability*. MIT Press, 2007.

[78] D. Turnbull, L. Barrington, and G. Lanckriet. Five approaches to collecting tags for music. *International Conference on Music Information Retrieval (ISMIR)*, 2008.

[79] M. Varewyck, J. Pauwels, and J .P. Martens. A novel chroma representation of polyphonic music based on multiple pitch tracking techniques. *Proceedings of ACM International Conference on Multimedia*, pages 667–670, 2008.

[80] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13/2:260–269, 1967.

[81] P. G. Vos. Tonality induction: theoretical problems and dilemmas. *Music Perception, Special Issue in Tonality Induction*, 17/4:403–416, 2000.

[82] T. Winograd. Linguistics and the computer analysis of tonal harmony. *Journal of Music Theory*, 1968.

[83] A. Winold and J. Bein. Banalyze: An artificial intelligence system for harmonic analysis. *Unpublished manuscript*, 1985.

[84] Y. Zhu and M. S. Kankanhalli. Music key detection for musical audio. *In 11th International Multimedia Modelling Conference*, 2005.